Coresets Meet EDCS: Algorithms for Matching and Vertex Cover on Massive Graphs*

Sepehr Assadi† University of Pennsylvania

MohammadHossein Bateni‡ Google Research

Aaron Bernstein§ Rutgers University

Vahab Mirrokni¶ Google Research

Cliff Stein‖ Columbia University

Abstract

There is a rapidly growing need for scalable algorithms that solve classical graph problems, such as maximum matching and minimum vertex cover, on massive graphs. For massive inputs, several different computational models have been introduced, including the streaming model, the distributed communication model, and the massively parallel computation (MPC) model that is a common abstraction of MapReduce-style computation. In each model, algorithms are analyzed in terms of resources such as space used or rounds of communication needed, in addition to the more traditional approximation ratio.

In this paper, we give a *single unified approach* that yields better approximation algorithms for matching and vertex cover in all these models. The highlights include:

- The first one pass, significantly-better-than-2-approximation for matching in random arrival streams that uses subquadratic space, namely a $(1.5+\varepsilon)$-approximation streaming algorithm that uses $\widetilde{O}(n^{1.5})$ space for constant $\varepsilon > 0$.
- The first 2-round, better-than-2-approximation for matching in the MPC model that uses subquadratic space per machine, namely a $(1.5+\varepsilon)$-approximation algorithm with $\widetilde{O}(\sqrt{mn}+n)$ memory per machine for constant $\varepsilon > 0$.

By building on our unified approach, we further develop parallel algorithms in the MPC model that give a $(1+\epsilon)$-approximation to matching and an $O(1)$-approximation to vertex cover in only $O(\log\log n)$ MPC rounds and $O(n/\text{polylog}(n))$ memory per machine. These results settle multiple open questions posed by Czumaj et al. [STOC 2018].

We obtain our results by a novel combination of two previously disjoint set of techniques, namely randomized composable coresets and edge degree constrained subgraphs (EDCS). We significantly extend the power of these techniques and prove several new structural results. For example, we show that an EDCS is a sparse certificate for large matchings and small vertex covers that is quite robust to sampling and composition.

1 Introduction

As massive graphs become more prevalent, there is a rapidly growing need for scalable algorithms that solve classical graph problems on large datasets. When dealing with massive data, the entire input graph is orders of magnitude larger than the amount of storage on one processor and hence any algorithm needs to explicitly address this issue. For massive inputs, several different computational models have been introduced, each focusing on certain additional resources needed to solve large-scale problems. Some examples include the streaming model, the distributed communication model, and the massively parallel computation (MPC) model that is a common abstraction of MapReduce-style computation (see Section 2 for a definition of MPC). The target resources in these models are the number of rounds of communication and the local storage on each machine.

Given the variety of relevant models, there has been a lot of attention on designing general algorithmic techniques that can be applicable across a wide

*A full version of this paper is available as an online preprint on arXiv [11].

†sassadi@cis.upenn.edu. Supported in part by NSF grant CCF-1617851. Research done in part while the author was a summer intern at Google Research, NYC.

‡bateni@google.com.

§bernstei@gmail.com.

¶mirrokni@google.com.

‖cliff@ieor.columbia.edu. Research supported in part by NSF grants CCF-1421161 and CCF-1714818. Some research done while visiting Google.

range of settings. We focus on this task for two prominent graph optimization problems: maximum matching and minimum vertex cover. Our main result (Section 1.2) presents a ***single unified algorithm*** that immediately implies significantly improved results (in some or all of the parameters involved) for both problems in all three models discussed above. For example, in random arrival order streams, our algorithm computes a $(1.5+\varepsilon)$-approximate matching in a single pass with $\widetilde{O}(n^{1.5})$ space; this significantly improves upon the approximation ratio of previous single-pass algorithms using subquadratic space, and is the first result to present strong evidence of a separation between random and adversarial order for matching. Another example is in the MPC model: Given $\widetilde{O}(n^{1.5})$ space per machine, our algorithm computes an a $(1.5+\varepsilon)$-approximate matching in only 2 MPC rounds; this significantly improves upon all previous results with a small constant number of rounds.

Our algorithm is built on the framework of randomized composable coreset, which was recently suggested by Assadi and Khanna [12] as a means to unify different models for processing massive graphs (see Section 1.1). A common drawback of unified approaches is that although they have the advantage of versatility, the results they yield are often not as strong as those that are tailored to one particular model. It is therefore perhaps surprising that we can design essentially a single algorithm that improves upon the state-of-the-art algorithms in all three models discussed above simultaneously. Our approach for the matching problem notably goes significantly beyond a 2-approximation, which is a notorious barrier for matching in all the models discussed above.

We also build on our techniques to achieve a second result (Section 1.3) particular to the MPC model. We show that when each machine has only $O(n)$ space (or even $O(n/\text{polylog}(n))$), $O(\log\log n)$ rounds suffice to compute a $(1+\varepsilon)$-approximate matching or a $O(1)$-approximate vertex cover. This improves significantly upon the recent breakthrough of Czumaj et al. [28], which does not extend to vertex cover, and requires $O(\log\log^2(n))$ rounds. Our results in this part settle multiple open questions posed by Czumaj et al. [28].

1.1 Randomized Composable Coresets Two examples of general techniques widely used for processing massive data sets are linear sketches (see e.g. [5, 6, 14, 23–25, 50, 51, 59]) and composable coresets (see e.g. [12, 15, 16, 18, 47, 61, 62]). Both proceed by arbitrarily partitioning the data into smaller pieces, computing a small-size summary of each piece, and then showing that these summaries can be combined into a small-size summary of the original data set. This approach has a wide range of applications, but strong impossibility results are known for both techniques for the two problems of maximum matching and minimum vertex cover that we study in this paper [14].

Recently, Assadi and Khanna [12] turned to the notion of randomized composable coresets—originally introduced in the context of submodular maximization by Mirrokni and Zadimoghadam [61] (see also [29])—to bypass these strong impossibility results. The idea is to partition the graph into *random* pieces rather than arbitrary ones. The authors in [12] designed randomized composable coresets for matching and vertex cover, but although this led to unified algorithms for many models of computation, the resulting bounds were still for the most part weaker than the state-of-the-art algorithms tailored to each particular model.

We now define randomized composable coresets in more detail; for brevity, we refer to them as randomized coresets. Given a graph $G(V,E)$, with $m=|E|$ and $n=|V|$, consider a random partition of E into k edge sets $\left\{E^{(1)},\ldots,E^{(k)}\right\}$; each edge in E is sent to exactly one of the $E^{(i)}$, picked uniformly at random, thereby partitioning graph G into k subgraphs $G^{(i)}(V,E^{(i)})$.

DEFINITION 1. (CF. [12, 61]) *Consider an algorithm* ALG *that takes as input an arbitrary graph and returns a* ***subgraph*** $\text{ALG}(G)\subseteq G$. ALG *is said to output an α-approximate randomized coreset for maximum matching if given any graph $G(V,E)$ and a random k-partition of G into $G^{(i)}(V,E^{(i)})$, the size of the maximum matching in $\text{ALG}(G^{(1)})\cup\ldots\cup\text{ALG}(G^{(k)})$ is an α-approximation to the size of the maximum matching in G with high probability. We refer to the* ***number of edges*** *in the returned subgraph by* ALG *as the* ***size*** *of the coreset. Randomized coresets are defined analogously for minimum vertex cover and other graph problems.*

It is proven in [12] that any $O(1)$-approximate randomized coreset for matching or vertex cover has size $\Omega(n)$. Thus, similarly to [12], we focus on designing randomized coresets of size $\widetilde{O}(n)$, which is optimal within logarithmic factors. The following proposition states some immediate applications of randomized coresets.

PROPOSITION 1.1. *Suppose* ALG *outputs an α-approximate randomized coreset of size $\widetilde{O}(n)$ for problem P (e.g. matching). Let $G(V,E)$ be a graph with $m=|E|$ edges. This yields:*

1. *A parallel algorithm in the MPC model that with high probability outputs an α-approximation to P in two rounds with $\widetilde{O}(\sqrt{m/n})$ machines, each with $\widetilde{O}(\sqrt{mn}+n) = \widetilde{O}(n^{1.5})$ memory.*

2. *A streaming algorithm that on* random arrival *streams outputs an α-approximation to $P(G)$ with high probability using $\widetilde{O}(\sqrt{mn}+n) = \widetilde{O}(n^{1.5})$ space.*

3. *A simultaneous communication protocol that on* randomly partitioned *inputs computes an α-approximation to $P(G)$ with high probability using $\widetilde{O}(n)$ communication per machine/player.*

A proof of Proposition 1.1 can be found in the full version of the paper [11].

1.2 First Result: Improved Algorithms via a New Randomized Coreset As our first result, we develop a new randomized composable coreset for matching and vertex cover.

RESULT 1. *There exist randomized composable coresets of size $\widetilde{O}(n)$ that for any constant $\varepsilon > 0$, give a $(3/2+\varepsilon)$-approximation for maximum matching and $(2+\varepsilon)$-approximation for minimum vertex cover with high probability.*

Our results improve upon the randomized coresets of [12] that obtained $O(1)$ and $O(\log n)$ approximation to matching and vertex cover, respectively. Our approach is entirely different, and in particular we go beyond the ubiquitous 2-approximation barrier for matching (in the full version [11], we show that the previous approach of [12] provably cannot go below 2 and present an improved analysis for that approach.). Result 1 yields a unified framework that improves upon the state-of-the art algorithms for matching and vertex cover across several computational models, for some or all the parameters involved.

First implication: streaming. We consider single-pass streaming algorithms. Computing a 2-approximation for matching (and vertex cover) in $O(n)$ space is trivial: simply maintain a maximal matching. Going beyond this barrier has remained one of the central open questions in the graph streaming literature since the introduction of the field [37]. No $o(n^2)$-space algorithm is known for this task on adversarially ordered streams and the lower bound result by Kapralov [48] (see also [40]) proves that an $\left(\frac{e}{e-1}\right)$-approximation requires $n^{1+\Omega(1/\log\log n)}$ space. To make progress on this fascinating open question, Konrad et al. [54] suggested the study of matching in random arrival streams. They presented an algorithm with approximation ratio strictly better than 2, namely $2-\delta$ for $\delta \approx 0.002$, in $O(n)$ space over random streams. A direct application of our Result 1 improves the approximation ratio of this algorithm significantly albeit at a cost of a larger space requirement.

COROLLARY 1.1. *There exists a single-pass streaming algorithm on* random arrival streams *that uses $\widetilde{O}(n^{1.5})$ space and with high probability (over the randomness of the stream) achieves a $(3/2+\varepsilon)$-approximation to the maximum matching problem for constant $\varepsilon > 0$.*

Our results provide the first strong evidence of a separation between random-order and adversarial-order streams for matching, as it is the first algorithm that beats the ratio of $\left(\frac{e}{e-1}\right)$, which is known to be "hard" on adversarial streams [48]. Although the lower bound of [48] does not preclude achieving the bounds of Corollary 1.1 in an adversarial order (because our space is $\widetilde{O}(n^{1.5})$ rather than $\widetilde{O}(n)$), the proof in [48] (see also [40]) suggests that achieving such bounds is ultimately connected to further understanding of Ruzsa-Szemerédi graphs, a notoriously hard problem in additive combinatorics (see e.g. [8, 38, 42]). From a different perspective, most (but not all) streaming lower bounds are proven by bounding the (per-player) communication complexity of the problem in the blackboard communication model, including the $\left(\frac{e}{e-1}\right)$ lower bound of [48]. Our algorithm in Result 1 can be implemented with $\widetilde{O}(n)$ (per-player) communication in this model which goes strictly below the lower bound of [48], thus establishing the first *provable separation* between adversarial- and random-partitioned inputs in the blackboard communication model for computing a matching.

Second implication: MPC. Maximum matching and minimum vertex cover are among the most studied graph optimization problems in the MPC and other MapReduce-style computation models [4, 5, 12, 20, 28, 44, 55]. As an application of Result 1, we obtain efficient MPC algorithms for matching and vertex cover in only two rounds of computation.

COROLLARY 1.2. *There exist algorithms that with high probability achieve a $(3/2+\varepsilon)$-approximation to matching and a $(2+\varepsilon)$-approximation to vertex cover in two MPC rounds and $\widetilde{O}(\sqrt{mn}+n)$ memory per machine for constant $\varepsilon > 0$.*

It follows from the results of [14] that sub-quadratic memory is not possible with one MPC round, so two

rounds is optimal. Furthermore, our implementation only requires one round if the input is distributed randomly in the first place; see [61] for details on when this assumption applies.

Our algorithms outperform the previous algorithms of [12] for matching and vertex cover in terms of approximation ratio (3/2 vs. $O(1)$ and 2 vs. $O(\log n)$), while memory and round complexity are the same. Our matching algorithm outperforms the 2-approximate maximum matching algorithm of Lattanzi et al. [55] in terms of both the approximation ratio (3/2 vs. 2) and round complexity (2 vs. 6) within the same memory. Our result for the matching problem is particularly interesting as all other MPC algorithms [4, 5, 20] that can achieve a better than two approximation (which is also a natural barrier for matching algorithms across different models) require a large (unspecified) constant number of rounds. Achieving the optimal 2 rounds is significant in this context, since the round complexity of MPC algorithms determines the dominant cost of the computation (see, e.g. [19,55]), and hence minimizing the number of rounds is the primary goal in this model.

Third implication: distributed. Maximum matching (and to a lesser degree vertex cover) has been studied previously in the simultaneous communication model owing to many applications of this model, including in achieving round-optimal distributed algorithms [12], proving lower bounds for dynamic graph streams [7,13,14,53], and applications to mechanism design [9,31,32]. As an application of Result 1, we obtain the following corollary.

COROLLARY 1.3. *There exist simultaneous communication protocols on* randomly partitioned inputs *that achieve* $(3/2+\varepsilon)$*-approximation to matching and* $(2+\varepsilon)$*-approximation to vertex cover with high probability (over the randomness of the input partitioning) with only* $\widetilde{O}(n)$ *communication per machine/player for constant* $\varepsilon > 0$.

This result improves upon the $O(1)$ and $O(\log n)$ approximation of [12] (on randomly partitioned inputs) for matching and vertex cover that were also designed by using randomized coresets. Our protocols achieve optimal communication complexity (up to polylog(n) factors) [12].

1.3 Second Result: MPC with Low Space Per Machine

Our second result concerns the MPC model with per-machine memory $O(n)$ or even $O(n/\text{polylog}(n))$. This is achieved by extending our Result 1 from random edge-partitioned subgraphs (as in randomized coresets) to random vertex-partitioned subgraphs (which we explain further below).

RESULT 2. *There exists an MPC algorithm that for any constant* $\varepsilon > 0$*, with high probability, gives a* $(1+\varepsilon)$*-approximation to maximum matching and* $O(1)$*-approximation to minimum vertex cover in* $O(\log\log n)$ *MPC rounds using only* $O(n/\text{polylog}(n))$ *memory per machine.*

Given an existing black-box reduction [56] (see also [28]), our Result 2 immediately implies a $(2+\varepsilon)$-approximation algorithm for maximum weighted matching in the same $O(\log\log(n))$ rounds, though with the memory per machine increased to $O(n\log(n))$.

Prior to [28], all MPC algorithms for matching and vertex cover [4,5,55] required $\Omega\left(\frac{\log n}{\log\log n}\right)$ rounds to achieve $O(1)$ approximation when the memory per machine was restricted to $\widetilde{O}(n)$ (which is arguably the most natural choice of parameter, similar-in-spirit to the semi-streaming restriction [37, 58]). Recently, Czumaj et al. [28] presented an (almost) 2-approximation algorithm for maximum matching that requires $O(n)$ (even $n/(\log n)^{O(\log\log n)}$) memory per machine and only $O\left((\log\log n)^2\right)$ MPC rounds. Result 2 improves upon this result on several fronts: (i) we improve the round complexity of the matching algorithm to $O(\log\log n)$, resolving a conjecture of [28] in the affirmative, (ii) we obtain an $O(1)$ approximation to vertex cover, answering another open question of [28], and (iii) we achieve all these using a considerably simpler algorithm and analysis than [28].

Comparison to results published after the appearance of our paper. After an earlier version of our paper was shared on arXiv [11], Ghaffari et al. [39] presented a result very similar to our Result 2: their bounds are exactly the same for matching, while for vertex cover they achieve a better approximation in the same asymptotic number of rounds: $(2+\varepsilon)$-approximation vs. our $O(1)$ approximation. Techniques-wise, our approaches are entirely different: the algorithms in [39] are based on an earlier round-compression technique of [28], and require an intricate local algorithm and analysis to ensure consistency between machines; see Section 1.4 below for more details.

Note that only our Result 2 is shared with the later paper of Ghaffari et al. [39]: Result 1 appears only in our paper, and is entirely specific to the particular techniques that we use.

1.4 Our Techniques Both of our results are based on a novel application of *edge degree constrained subgraphs (EDCS)* that were previously introduced by Bernstein and Stein [21] for maintaining large matchings in dynamic graphs. Previous work on EDCS [21, 22] focused on how large a matching an EDCS contains and how it can be maintained in a dynamic graph. For the two results of this paper, we instead focus on the structural properties of the EDCS, and prove several new facts in this regard.

For Result 1, we identify the EDCS as a sparse certificate for large matchings and small vertex covers which are quite robust to sampling and composition: an ideal combination for a randomized coreset. For Result 2, we use the following recursive procedure, which crucially relies upon on the robustness properties of the EDCS proved in Result 1: we repeatedly compute an EDCS of the underlying graph in a distributed fashion, redistribute it again amongst multiple machines, and recursively solve the problem on this EDCS to compute an $O(1)$-approximation to matching and vertex cover. We therefore limit the memory on each machine to only $O(n)$ (even $O(n/\text{polylog}(n))$) at the cost of increasing the number of rounds from $O(1)$ to $O(\log \log n)$. Additional ideas are needed to ensure that the approximation ratio of the algorithm does not increase beyond a fixed constant as a result of repeatedly computing an EDCS of the current graph in $O(\log \log n)$ iterations.

Comparison of techniques. Result 1 uses the definition of EDCS from [21, 22] but uses it in an entirely different setting, and hence we prove and use novel properties of EDCS in this work.

Result 2 relies on the high-level technique of vertex sampling from Czumaj et al. [28]: instead of partitioning the edges of the graph, each machine receives a random sample of the vertices, and works on the resulting induced subgraph. Other than this starting point, our approach proceeds along entirely different lines from [28], in terms of both the local algorithm computed on each subgraph and in the analysis. The main approach in [28] is *round compression*, which corresponds to compressing multiple rounds of a particular distributed algorithm into smaller number of MPC rounds by maintaining a consistent state across the local algorithms computed on each subgraph (using a highly non-trivial local algorithm and analysis). Our results, on the other hand, do *not* correspond to a round compression approach at all and we do not require any consistency in the local algorithm on each machine. Instead, we rely on structural properties of the EDCS that we prove in this paper, *independent of the algorithms that compute these subgraphs*. This allows us to bypass many of the technical difficulties arising in maintaining a consistent state across different machines which in turn results in improved bounds and a considerably simpler algorithm and analysis.

1.5 Related Work Maximum matching and minimum vertex cover are among the most studied problems in the context of massive graphs including in MPC model and MapReduce-style computation [4, 5, 12, 20, 28, 39, 44, 55], streaming algorithms [3–6, 13, 14, 25–27, 33–37, 40, 43, 48, 49, 53, 54, 57, 58, 60, 69], simultaneous communication model and similar distributed models [9, 12–14, 32, 43, 46], dynamic graphs [17, 21, 22, 64, 67, 70], and sub-linear time algorithms [45, 65, 66, 68, 71]. Beside the results mentioned already, most relevant to our work are the polylog(n)-space polylog(n)-approximation algorithm of [49] for estimating the *size* of a maximum matching in random stream, and the (3/2)-approximation communication protocol of [40] when the input is (adversarially) partitioned between two parties and the communication is from one party to the other one (as opposed to simultaneous which we studied). However, the techniques in these results and ours are completely disjoint.

Coresets, composable coresets, and randomized composable coresets are respectively introduced in [2], [47], and [61]. Composable coresets have been studied previously in nearest neighbor search [1], diversity maximization [47, 72], clustering [16, 18], and submodular maximization [15, 29, 30, 47, 61]. Moreover, while not particularly termed a composable coreset, the "merge and reduce" technique in graph streaming literature (see [58], Section 2.2) is identical to composable coresets.

2 Preliminaries

Notation. For a graph $G(V, E)$, we use $\mathsf{MM}(G)$ to denote the maximum matching size in G and $\mathsf{VC}(G)$ to denote the minimum vertex cover size. For any subset of vertices $V' \subseteq V$ and any subset of edges $E' \subseteq E$, we use $V'(E')$ to denote the set of vertices in V' that are incident on edges of E' and $E'(V')$ to denote the set of edges in E' that are incident on vertices of V'. For any vertex $v \in V$, we use $d_G(v)$ to denote the degree of v in the graph G.

Sampled Subgraphs. Throughout the paper, we work with two different notion of sampling a graph $G(V, E)$. For a parameter $p \in (0, 1)$,

- A graph $G_p^{\mathsf{E}}(V, E_p)$ is an *edge sampled subgraph* of G iff the vertex set of G_p^{E} and G are the same and every edge in E is picked independently and with

probability p in E_p.

- A graph $G_p^{\mathsf{V}}(V_p, E_p)$ is a *vertex sampled (induced) subgraph* of G iff every vertex in V is sampled in V_p independently and with probability p and G_p^{V} is the induced subgraph of G on V_p

2.1 The Massively Parallel Computation (MPC) Model We adopt the most stringent model of modern parallel computation among [10,19,41,52], the so-called *Massively Parallel Computation (MPC)* model of [19]. Let $G(V,E)$ with $n := |V|$ and $m := |E|$ be the input graph. In this model, there are p machines, each with a memory of size s and one typically requires that both $p, s = m^{1-\Omega(1)}$ i.e., polynomially smaller than the input size [10,52]. Computation proceeds in synchronous rounds: in each round, each machine performs some local computation and at the end of the round machines exchange messages to guide the computation for the next round. All messages sent and received by each machine in each round have to fit into the local memory of the machine. This in particular means that the length of the messages on each machine is bounded by s in each round. At the end, the machines collectively output the solution.

2.2 Basic Graph Theory Facts

FACT 2.1. *For any graph $G(V,E)$, $\mathsf{MM}(G) \leq \mathsf{VC}(G) \leq 2 \cdot \mathsf{MM}(G)$.*

The following propositions are well-known (see the full version [11] for proofs).

PROPOSITION 2.1. *Suppose M and V' are respectively, a matching and a vertex cover of a graph G such that $\alpha \cdot |M| \geq |V'|$; then, both M and V' are α-approximation to their respective problems.*

PROPOSITION 2.2. *Suppose $G(V,E)$ is a graph with maximum degree Δ and V_{HIGH} is the set of all vertices with degree at least $\gamma \cdot \Delta$ in G for $\gamma \in (0,1)$. Then, $\mathsf{MM}(G) \geq \frac{\gamma \cdot \Delta}{2 \cdot (\Delta+1)} \cdot |V_{\text{HIGH}}|$.*

2.3 Edge Degree Constrained Subgraph (EDCS) We introduce edge degree constrained subgraphs (EDCS) in this section and present several of their properties which are proven in previous work. We emphasize that all other properties of EDCS proven in the subsequent sections are new to this paper. An EDCS is defined formally as follows.

DEFINITION 2. ([21]) *For any graph $G(V,E)$ and integers $\beta \geq \beta^- \geq 0$, an* edge degree constraint subgraph (EDCS) *(G, β, β^-) is a subgraph $H := (V, E_H)$ of G with the following two properties:*

(P1) For any edge $(u,v) \in E_H$: $d_H(u) + d_H(v) \leq \beta$.

(P2) For any edge $(u,v) \in E \setminus E_H$: $d_H(u) + d_H(v) \geq \beta^-$.

We sometimes abuse the notation and use H and E_H interchangeably.

We use the terms "Property (P1)" and "Property (P2)" of EDCS to refer to the first and second items in Definition 2 above.

One can prove the existence of an EDCS(G, β, β^-) for any graph G and parameters $\beta^- < \beta$ using the results in [22] (Theorem 3.2) which in fact shows how to maintain an EDCS efficiently in the dynamic graph setting (see also the full version [11] for a simpler and self-contained proof of existence of EDCS and a simple polynomial time algorithm for computing it).

LEMMA 2.1. *Any graph G contains an* EDCS(G, β, β^-) *for any parameters $\beta > \beta^-$.*

It was shown in [21] (bipartite graphs) and [22] (general graphs) that for appropriate parameters and EDCS always contains an (almost) 3/2-approximate maximum matching of G. Formally:

LEMMA 2.2. ([21, 22]) *Let $G(V,E)$ be any graph and $\varepsilon < 1/2$ be a parameter. For parameters $\lambda \geq \frac{\varepsilon}{100}$, $\beta \geq 32\lambda^{-3}$, and $\beta^- \geq (1-\lambda) \cdot \beta$, in any subgraph $H :=$ EDCS(G, β, β^-), $\mathsf{MM}(G) \leq \left(\frac{3}{2} + \varepsilon\right) \cdot \mathsf{MM}(H)$.*

Lemma 2.2 implies that an EDCS of a graph $G(V,E)$ preserves the maximum matching of G approximately. We also show a similar result for vertex cover. The basic idea is that in addition to computing a vertex cover for the subgraph H(to cover all the edges in H), we also add to the vertex cover all vertices that have degree at least $\geq \beta^-/2$ in H, which by Property (P2) of an EDCS covers all edges in $G \setminus H$.

LEMMA 2.3. *Let $G(V,E)$ be any graph, $\varepsilon < 1/2$ be a parameter, and $H :=$ EDCS(G, β, β^-) for parameters $\beta \geq \frac{4}{\varepsilon}$ and $\beta^- \geq \beta \cdot (1 - \varepsilon/4)$. Suppose V_{HIGH} is the set of vertices $v \in V$ with $d_H(v) \geq \beta^-/2$ and V_{VC} is a minimum vertex cover of H; then $V_{\text{HIGH}} \cup V_{\text{VC}}$ is a vertex cover of G with size at most $(2+\varepsilon) \cdot \mathsf{VC}(H)$ (note that $\mathsf{VC}(H) \leq \mathsf{VC}(G)$).*

Proof. We first argue that $V_{\text{HIGH}} \cup V_{\text{VC}}$ is indeed a feasible vertex cover of G. To see this, notice that any edge $e \in H$ is covered by V_{VC}, and moreover by Property (P2) of EDCS, any edge $e \in E \setminus H$ has at

least one endpoint with degree at least $\beta^-/2$ in H and hence is covered by V_{HIGH}. In the following, we bound the size of $V_{\text{HIGH}} \setminus V_{\text{VC}}$ by $(1+\varepsilon) \cdot |V_{\text{VC}}|$, which finalizes the proof as clearly $|V_{\text{VC}}| = \mathsf{VC}(H)$.

Define $S := V_{\text{HIGH}} \setminus V_{\text{VC}}$ and let $N(S)$ be the set of all neighbors of S in the EDCS H. Since S is not part of the vertex cover V_{VC} of H, we should have $N(S) \subseteq V_{\text{VC}}$ as otherwise some edges between S and $N(S)$ would not be covered by the vertex cover V_{VC}. Now, since any vertex in S has degree at least $\beta^-/2$, we should have that degree of any vertex in $N(S)$ is at most $\beta - \beta^-/2$ in order to satisfy Property (P1) of EDCS H. Let $E(S)$ denote the set of edges incident on S in H. As all vertices in S belong to V_{HIGH}, we have that $|E(S)| \geq |S| \cdot \beta^-/2$. On the other hand, as all edges incident on S are going into $N(S)$ by definition, and since degree of vertices in $N(S)$ are bounded by $\beta - \beta^-/2$, we have $|E(S)| \leq |N(S)| \cdot (\beta - \beta^-/2)$. As such,

$$|S| \cdot \beta^-/2 \leq |N(S)| \cdot \big(\beta - \beta^-/2\big) \leq |V_{\text{VC}}| \cdot (1+\varepsilon) \cdot \beta^-/2,$$

implying that $|S| \leq (1+\varepsilon) \cdot |V_{\text{VC}}|$, finalizing the proof.

3 New Properties of Edge Degree Constrained Subgraphs

We study further properties of EDCS in this section. Although EDCS was used prior to our work, all the properties proven in this section are entirely new to this paper and look at the EDCS from a different vantage point.

Previous work in [21,22] studied the EDCS from the perspective of how large of matching it contains and how it can be maintained efficiently in a dynamically changing graph. In this paper, we prove several new interesting structural properties of the EDCS itself. In particular, while it is easy to see that in terms of edge sets there can be many different EDCS of some fixed graph $G(V,E)$ (consider G being a complete graph), we show that the degree distributions of every EDCS (for the same parameters β and β^-) are almost the same. In other words, the degree of any vertex v is almost the same in every EDCS of $G(V,E)$. This is in sharp contrast with similar objects such as b-matchings, which can vary a lot within the same graph. This semi-uniqueness renders the EDCS extremely robust under sampling and composition as we prove next in this section.

These new structural results on EDCS are the main properties that allows their use in our coresets and parallel algorithms in the rest of the paper. In fact, our parallel algorithms in Section 5 are entirely based on these results and do not rely at all on the fact that an EDCS contains a large matching (i.e., do not depend on Lemma 2.2 at all).

3.1 Degree Distribution Lemma

LEMMA 3.1. (DEGREE DISTRIBUTION LEMMA) *Fix a graph $G(V,E)$ and parameters $\beta, \beta^- = (1-\lambda)\cdot\beta$ (for $\lambda < 1/100$). For any two subgraphs A and B that are* $\text{EDCS}(G,\beta,\beta^-)$*, and any vertex $v \in V$,*

$$|d_A(v) - d_B(v)| = O(\log n) \cdot \lambda^{1/2} \cdot \beta.$$

In the rest of this section, we fix the parameters β, β^- and the two EDCS A and B in Lemma 3.1. The general strategy of the proof is as follows. We start with a set S_1 of all vertices which has the most difference in degree between A and B. By considering the two-hop neighborhood of these vertices in A and B, we show that there exists a set S_2 of vertices in V such that the difference between the degree of vertices in A and B is almost the same as vertices in S_1, while size of S_2 is a constant factor larger than S_1. We then use this argument repeatedly to construct the next set S_3 and so on, whereby each set is larger than the previous one by a constant factor, while the gap between the degree of vertices in A and B remains almost the same as the previous set. As this geometric increase in the size of sets can only happen in a "small number" of steps (otherwise we run out of vertices), we obtain that the gap between the degree of vertices in S_1 could have not been "too large" to begin with. We now formalize this argument, starting with a technical lemma which allows us to obtain each set S_i from the set S_{i-1} in the above argument.

LEMMA 3.2. *Fix an integer $D > 2\lambda^{1/2} \cdot \beta$ and suppose $S \subseteq V$ is such that for all $v \in S$, we have $d_A(v) - d_B(v) \geq D$. Then, there exists a set of vertices $S' \supseteq S$ such that $|S'| \geq (1+2\lambda^{1/2}) \cdot |S|$ and for all $v \in S'$, $d_A(v) - d_B(v) \geq D - 2\lambda \cdot \beta$.*

Proof. We define the following two sets T and T':

- T is the set of all neighbors of vertices in S using *only* the edges in $A \setminus B$. In other words, $T := \{v \in V \mid \exists u \in S \wedge (u,v) \in A \setminus B\}$.
- T' is the set of all neighbors of vertices in T using *only* the edges in $B \setminus A$. In other words, $T' := \{w \in V \mid \exists v \in T \wedge (v,w) \in B \setminus A\}$.

We start by proving the following property on the degree of vertices in the sets T and T'.

CLAIM 3.3. *We have,*

- *for all $v \in T$, $d_B(v) - d_A(v) \geq D - \lambda \cdot \beta$.*

- *for all* $w \in T'$, $d_A(w) - d_B(w) \geq D - 2\lambda \cdot \beta$.

Proof. For the first part, since $v \in T$, it means that there exists an edge $(u, v) \in A \setminus B$ such that $u \in S$. Since (u, v) belongs to A, by Property (P1) of an EDCS we have $d_A(v) \leq \beta - d_A(u) \leq \beta - d_B(u) - D$. On the other hand, since (u, v) does not belong to B, by Property (P2) of an EDCS we have $d_B(v) \geq \beta - \lambda \cdot \beta - d_B(u)$, completing the proof for T.

For the second part, since $w \in T'$, it means that there exists an edge $(v, w) \in B \setminus A$ such that $v \in T$. Since (v, w) does not belong to A, by Property (P2) of an EDCS we have $d_A(w) \geq \beta - \lambda \cdot \beta - d_A(v)$. Moreover, since (u, v) belongs to B, by Property (P1) of an EDCS, we have, $d_B(w) \leq \beta - d_B(v)$. This means that $d_A(w) - d_B(w) \geq d_B(v) - d_A(v) - \lambda \cdot \beta$ which is at least $D - 2\lambda \cdot \beta$ by the first part. ∎

Notice that since $D > 2\lambda \cdot \beta$, by Claim 3.3, for any vertex $v \in T$, we have $d_B(v) > d_A(v)$ and hence $S \cap T = \emptyset$ (similarly, $T \cap T' = \emptyset$, but S and T' may intersect). We define the set S' in the lemma statement to be $S' := S \cup T'$. The bound on the degree of vertices in S' follows immediately from Claim 3.3 (recall that vertices in S already satisfy the degree requirement for the set S'). In the following, we show that $|T' \setminus S| \geq 2\lambda^{1/2} \cdot |S|$, which finalizes the proof.

Recall that $E_{A \setminus B}(S)$ and $E_{A \setminus B}(S, T)$ denote the set of edges in subgraph $A \setminus B$ incident on vertices S, and between vertices S and T, respectively. We have,

$$
\begin{aligned}
&\text{(edges in } B \setminus A \text{ incident on } T \text{ are going to } T') \\
&\quad \left|E_{B \setminus A}(T, T' \setminus S)\right| = \left|E_{B \setminus A}(T)\right| - \left|E_{B \setminus A}(T, S)\right| \\
&\text{(by Claim 3.3)} \qquad \geq \left|E_{A \setminus B}(T)\right| - \left|E_{B \setminus A}(T, S)\right| \\
&\text{(edges in } A \setminus B \text{ incident on } S \text{ are also incident on } T) \\
&\qquad \geq \left|E_{A \setminus B}(S)\right| - \left|E_{B \setminus A}(S)\right| \\
&\text{(assumption on the degrees in } S \text{ across } A \text{ and } B) \\
&\qquad \geq |S| \cdot D.
\end{aligned}
$$

Finally, since B is an EDCS, the maximum degree of any vertex in $T' \setminus S$ is at most β and hence there should be at least $|S| \cdot \frac{D}{\beta} \geq 2\lambda^{1/2} \cdot |S|$ vertices in $T' \setminus S$ (as $D > 2\lambda^{1/2} \cdot \beta$). ∎

Proof. [Proof of Lemma 3.1] Suppose towards a contradiction that there exists a vertex $v \in V$ s.t. $D := d_A(v) - d_B(v) \geq 3\ln(n) \cdot \lambda^{1/2} \cdot \beta$ (the other case is symmetric). Let $D_0 = D$ and $S_0 = \{v\}$ and for $i = 1$ to $t := \lambda^{-1/2} \cdot (\ln(n) + 1)$: define the set S_i and integer D_i by applying Lemma 3.2 to S_{i-1} and D_{i-1} (i.e., $S_i = S'$ and $D_i = D_{i-1} - 2\lambda \cdot \beta$). By the lower bound on the value of D, for any $i \in [t]$, we have that $D_i \geq D - i \cdot 2\lambda \cdot \beta > 2\lambda^{1/2} \cdot \beta$, and hence we can indeed apply Lemma 3.2. As a result, we have,

$$
\begin{aligned}
|S_t| &\geq \left(1 + 2\lambda^{1/2}\right) \cdot |S_{t-1}| \\
&\geq \left(1 + 2\lambda^{1/2}\right)^t \cdot |S_0| \geq \exp\left(\lambda^{1/2} \cdot t\right) \\
&> \exp\left(\ln(n)\right) = n,
\end{aligned}
$$

which is a contradiction as there are only n vertices in the graph G. Consequently, we obtain that for any vertex v, $|d_A(v) - d_B(v)| = O(\log n) \cdot \lambda^{1/2} \cdot \beta$, finalizing the proof. ∎

3.2 EDCS in Sampled Subgraphs In this section, we prove two lemmas regarding the structure of different EDCS across sampled subgraphs. The first lemma concerns edge sampled subgraphs. We show that the degree distributions of any two EDCS for two different edge sampled subgraphs of G is almost the same no matter how the two EDCS are selected or even if the choice of the two subgraphs are *not* independent. This Lemma is is used in our Result 1 on randomized coresets (see Section 4).

LEMMA 3.4. (EDGE SAMPLED EDCS) *Fix any graph $G(V, E)$ and $p \in (0, 1)$. Let G_1 and G_2 be two* edge sampled subgraphs *of G with probability p (chosen* not *necessarily independently). Let H_1 and H_2 be arbitrary EDCSs of G_1 and G_2 with parameters $(\beta, (1 - \lambda) \cdot \beta)$. Suppose $\beta \geq 750 \cdot \lambda^{-2} \cdot \ln(n)$, then, with probability $1 - 4/n^9$, simultaneously for all $v \in V$: $|d_{H_1}(v) - d_{H_2}(v)| \leq O(\log n) \cdot \lambda^{1/2} \cdot \beta$.*

We also prove a qualitatively similar lemma for vertex sampled subgraphs. This is needed in Result 2 for parallel algorithms (see Section 5). The main difference here is that there will be a huge gap between the degree of a vertex between the two EDCS if the vertex is sampled in one subgraph but not the other one. However, we show that the degree of vertices that are sampled in both subgraphs are almost the same across the two different (and arbitrarily chosen) EDCS for the subgraphs.

LEMMA 3.5. (VERTEX SAMPLED EDCS) *Fix any graph $G(V, E)$ and $p \in (0, 1)$. Let G_1 and G_2 be two* vertex sampled subgraphs *of G with probability p (chosen* not *necessarily independently). Let H_1 and H_2 be arbitrary EDCSs of G_1 and G_2 with parameters $(\beta, (1 - \lambda)\beta)$. If $\beta \geq 750 \cdot \lambda^{-2} \cdot \ln(n)$, then, with probability $1 - 4/n^9$, simultaneously for all $v \in G_1 \cap G_2$: $|d_{H_1}(v) - d_{H_2}(v)| \leq O(\log n) \cdot \lambda^{1/2} \cdot \beta$.*

The proof of both these lemmas are along the following lines. We start with an EDCS H of the original graph G with parameters (almost) $(\beta/p, \beta^-/p)$.

We then consider the set of edges from H in each of the sampled subgraphs G_1 and G_2, i.e., the two subgraphs $H'_1 := G_1 \cap H$ and $H'_2 := G_2 \cap H$. We use the randomness in the process of sampling subgraphs G_1 and G_2 to prove that with high probability both H'_1 and H'_2 form an EDCS for G_1 and G_2, respectively, with parameters (β, β^-). Finally, we use our degree distribution lemma from Section 3.1 to argue that for any arbitrary EDCS H_1 (resp. H_2) of G_1 (resp. G_2), the degree distribution of H_1 (resp. H_2) is close to H'_1 (resp. H'_2). Since the degree distributions of H'_1 and H'_2 are close to each other already (as they are both sampled subgraphs of H), this finalizes the proof.

There are some technical differences in implementing the above intuition between the edge sampled and vertex sampled subgraphs and hence we provide a separate proof for each case.

3.2.1 EDCS in Edge Sampled Subgraphs: Proof of Lemma 3.4

Proof. [Proof of Lemma 3.4] We first prove that edge sampling an EDCS results in another EDCS for the sampled subgraph.

Claim 3.6. *Let H be an* EDCS(G, β_H, β_H^-) *for parameters $\beta_H := (1-\frac{\lambda}{2})\cdot\frac{\beta}{p}$ and $\beta_H^- := \beta_H - 1$. Suppose $G_p := G_p^E(V, E_p)$ is an edge sampled subgraph of G and $H_p := H \cap G_p$; then, with probability $1 - 2/n^9$:*

1. *For any vertex $v \in V$, $\left|d_{H_p}(v) - p \cdot d_H(v)\right| \leq \frac{\lambda}{5}\cdot\beta$.*
2. *H_p is an EDCS of G_p with parameters $(\beta, (1-\lambda)\cdot\beta)$.*

Proof. For any $v \in V$, $\mathbb{E}\left[d_{H_p}(v)\right] = p \cdot d_H(v)$ and $d_H(v) \leq \beta_H$ by Property (P1) of EDCS H. Moreover, since each neighbor of v in H is sampled in H_p independently, by Chernoff bound,

$$\Pr\left(\left|d_{H_p}(v) - p\cdot d_H(v)\right| \geq \frac{\lambda}{5}\beta\right) \leq 2\exp\left(-\frac{\lambda^2\cdot\beta}{75}\right) \leq 2\cdot\exp\left(-10\ln n\right) = \frac{2}{n^{10}},$$

where the second inequality is by the lower bound on β in Lemma 3.5 statement. In the following, we condition on the event that:

$$\forall v \in V \quad \left|d_{H_p}(v) - p\cdot d_H(v)\right| \leq \frac{\lambda}{5}\cdot\beta. \tag{3.1}$$

This event happens with probability at least $1-2/n^9$ by above equation and a union bound on $|V| = n$ vertices. This finalizes the proof of the first part of the claim. We are now ready to prove that H_p is indeed an EDCS$(G_p, \beta, (1-\lambda)\cdot\beta)$ conditioned on this event.

Consider any edge $(u, v) \in H_p$. Since $H_p \subseteq H$, $(u, v) \in H$ as well. Hence, we have,

$$\begin{aligned} d_{H_p}(u) + d_{H_p}(v) &\underset{\text{Eq (3.1)}}{\leq} p\cdot\Big(d_H(u) + d_H(v)\Big) + \frac{2\lambda}{5}\cdot\beta \\ &\leq p\cdot\beta_H + \frac{2\lambda}{5}\cdot\beta \\ &= (1-\frac{\lambda}{2})\cdot\beta + \frac{2\lambda}{5}\cdot\beta < \beta, \end{aligned}$$

where the second inequality is by Property (P1) of EDCS H and the equality is by the choice of β_H. As a result, H_p satisfies Property (P1) of EDCS for parameter β.

Now consider an edge $(u, v) \in G_p \setminus H_p$. Since $H_p = G_p \cap H$, $(u, v) \notin H$ as well. Hence,

$$\begin{aligned} d_{H_p}(u) + d_{H_p}(v) &\underset{\text{Eq (3.1)}}{\geq} p\cdot\Big(d_H(u) + d_H(v)\Big) - \frac{2\lambda}{5}\cdot\beta \\ &\geq p\cdot\beta_H^- - \frac{2\lambda}{5}\cdot\beta \\ &= (1-\frac{\lambda}{2})\cdot\beta - p - \frac{2\lambda}{5}\cdot\beta \\ &> (1-\lambda)\cdot\beta, \end{aligned}$$

where the second inequality is by Property (P2) of EDCS H and the equality is by the choice of β_H^-. As such, H_p satisfies Property (P2) of EDCS for parameter $(1-\lambda)\cdot\beta$ and hence H_p is indeed an EDCS$(G_p, \beta, (1-\lambda)\cdot\beta)$. ∎

We continue with the proof of Lemma 3.4. Let H be an EDCS(G, β_H, β_H^-) for the parameters β_H, β_H^- in Claim 3.6. The existence of H follows from Lemma 2.1 as $\beta_H^- < \beta_H$. Define $\widehat{H}_1 := H \cap G_1$ and $\widehat{H}_2 := H \cap G_2$. By Claim 3.6, $\widehat{H}_1$ (resp. $\widehat{H}_2$) is an EDCS of G_1 (resp. G_2) with parameters $(\beta, (1-\lambda)\beta)$ with probability $1 - 4/n^9$. In the following, we condition on this event.

By Lemma 3.1 (Degree Distribution Lemma), since both H_1 (resp. H_2) and $\widehat{H}_1$ (resp. $\widehat{H}_2$) are EDCS for G_1 (resp. G_2), the degree of vertices in both of them should be "close" to each other. Moreover, since by Claim 3.7 the degree of each vertex in $\widehat{H}_1$ and $\widehat{H}_2$ is close to p times its degree in H, we can argue that the vertex degrees in H_1 and H_2 are close. Formally, for any $v \in V$, we have,

$$|d_{H_1}(v) - d_{H_2}(v)| \leq \left|d_{H_1}(v) - d_{\widehat{H}_1}(v)\right| + \left|d_{\widehat{H}_1}(v) - d_{\widehat{H}_2}(v)\right|$$

$$
\begin{aligned}
&+ \left|d_{\widehat{H}_2}(v) - d_{H_2}(v)\right| \underset{\text{Lemma 3.1}}{\leq} \\
&O(\log n)\cdot\lambda^{1/2}\cdot\beta + \left|d_{\widehat{H}_1}(v) - p\cdot d_H(v)\right| \\
&+ \left|d_{\widehat{H}_2}(v) - p\cdot d_H(v)\right| \underset{\text{Claim 3.6}}{\leq} \\
&O(\log n)\cdot\lambda^{1/2}\cdot\beta + O(1)\cdot\lambda\cdot\beta,
\end{aligned}
$$

finalizing the proof. ∎

3.2.2 EDCS in Vertex Sampled Subgraphs: Proof of Lemma 3.5

Proof. [Proof of Lemma 3.5]

We first prove that vertex sampling an EDCS results in another EDCS for the sampled subgraph.

Claim 3.7. *Let H be an EDCS(G,β_H,β_H^-) for parameters $\beta_H := (1-\frac{\lambda}{2})\cdot\frac{\beta}{p}$ and $\beta_H^- := \beta_H - 1$. Suppose $G_p := G_p^V(V_p,E_p)$ is a vertex sampled subgraph of G and $H_p := H\cap G_p$; then, with probability $1-2/n^9$:*

1. *For any vertex $v\in V_p$, $\left|d_{H_p}(v) - p\cdot d_H(v)\right| \leq \frac{\lambda}{5}\cdot\beta$.*
2. *H_p is an EDCS of G_p with parameters $(\beta,(1-\lambda)\cdot\beta)$.*

Proof. For any $v\in V_p$, $\mathbb{E}\left[d_{H_p}(v)\right] = p\cdot d_H(v)$ by the independent sampling of vertices and $d_H(v)\leq\beta_H$ by Property (P1) of EDCS H. Moreover, since each neighbor of v in H is sampled in H_p independently, by Chernoff bound,

$$
\begin{aligned}
\Pr\left(\left|d_{H_p}(v) - p\cdot d_H(v)\right| \geq \frac{\lambda}{5}\beta\right) &\leq 2\cdot\exp\left(-\frac{\lambda^2\cdot\beta}{75}\right) \\
&\leq 2\cdot\exp\left(-10\ln n\right) \\
&= \frac{2}{n^{10}},
\end{aligned}
$$

where the second inequality is by the lower bound on β in Lemma 3.5 statement. In the following, we condition on the event that:

$$
\forall v\in V_p \quad \left|d_{H_p}(v) - p\cdot d_H(v)\right| \leq \frac{\lambda}{5}\cdot\beta. \tag{3.2}
$$

which happens with probability at least $1-2/n^9$ by above equation and a union bound on $|V_p|\leq n$ vertices. This finalizes the proof of the first part of the claim. We are now ready to prove that H_p is indeed an EDCS$(G_p,\beta,(1-\lambda)\cdot\beta)$ conditioned on this event.

Consider any edge $(u,v)\in H_p$. Since $H_p\subseteq H$, $(u,v)\in H$ as well. Hence, we have,

$$
\begin{aligned}
d_{H_p}(u) + d_{H_p}(v) &\underset{\text{Eq (3.2)}}{\leq} p\cdot\Big(d_H(u)+d_H(v)\Big) + \frac{2\lambda}{5}\cdot\beta \\
&\leq p\cdot\beta_H + \frac{2\lambda}{5}\cdot\beta \\
&= (1-\frac{\lambda}{2})\cdot\beta + \frac{2\lambda}{5}\cdot\beta < \beta,
\end{aligned}
$$

where the second inequality is by Property (P1) of EDCS H and the equality is by the choice of β_H. As a result, H_p satisfies Property (P1) of EDCS for parameter β.

Now consider an edge $(u,v)\in G_p\setminus H_p$. Since $H_p = G_p\cap H$, $(u,v)\notin H$ as well. Hence,

$$
\begin{aligned}
d_{H_p}(u) + d_{H_p}(v) &\underset{\text{Eq (3.2)}}{\geq} p\cdot\Big(d_H(u)+d_H(v)\Big) - \frac{2\lambda}{5}\cdot\beta \\
&\geq p\cdot\beta_H^- - \frac{2\lambda}{5}\cdot\beta \\
&= (1-\frac{\lambda}{2})\cdot\beta - p - \frac{2\lambda}{5}\cdot\beta \\
&> (1-\lambda)\cdot\beta,
\end{aligned}
$$

where the second inequality is by Property (P2) of EDCS H and the equality is by the choice of β_H^-. As such, H_p satisfies Property (P2) of EDCS for parameter $(1-\lambda)\cdot\beta$ and hence H_p is indeed an EDCS$(G_p,\beta,(1-\lambda)\cdot\beta)$. ∎

We continue with the proof of Lemma 3.5. Let H be an EDCS(G,β_H,β_H^-) for the parameters β_H,β_H^- in Claim 3.7. The existence of H follows from Lemma 2.1 as $\beta_H^- < \beta_H$. Define $\widehat{H}_1 := H\cap G_1$ and $\widehat{H}_2 := H\cap G_2$. By Claim 3.7, $\widehat{H}_1$ (resp. $\widehat{H}_2$) is an EDCS of G_1 (resp. G_2) with parameters $(\beta,(1-\lambda)\beta)$ with probability $1-4/n^9$. In the following, we condition on this event.

By Lemma 3.1 (Degree Distribution Lemma), since both H_1 (resp. H_2) and $\widehat{H}_1$ (resp. $\widehat{H}_2$) are EDCS for G_1 (resp. G_2), the degree of vertices in both of them should be "close" to each other. Moreover, since by Claim 3.7 the degree of each vertex in $\widehat{H}_1$ and $\widehat{H}_2$ is close to p times its degree in H, we can argue that the degree of shared vertices in H_1 and H_2 are close. Formally, let v be a vertex in both G_1 and G_2; we have,

$$
\begin{aligned}
&|d_{H_1}(v) - d_{H_2}(v)| \leq \\
&\quad \left|d_{H_1}(v) - d_{\widehat{H}_1}(v)\right| + \left|d_{\widehat{H}_1}(v) - d_{\widehat{H}_2}(v)\right| \\
&\quad + \left|d_{\widehat{H}_2}(v) - d_{H_2}(v)\right| \underset{\text{Lemma 3.1}}{\leq} \\
&\quad O(\log n)\cdot\lambda^{1/2}\cdot\beta + \left|d_{\widehat{H}_1}(v) - p\cdot d_H(v)\right| \\
&\quad + \left|d_{\widehat{H}_2}(v) - p\cdot d_H(v)\right| \underset{\text{Claim 3.7}}{\leq} \\
&\quad O(\log n)\cdot\lambda^{1/2}\cdot\beta + O(1)\cdot\lambda\cdot\beta,
\end{aligned}
$$

finalizing the proof. ∎

4 Randomized Coresets for Matching and Vertex Cover

We introduce our randomized coresets for matching and vertex cover in this section. Both of these results are achieved by computing an EDCS of the input graph (for appropriate choice of parameters) and then applying Lemmas 2.2 and 2.3.

4.1 Computing an EDCS from Random k-Partitions Let $G(V,E)$ be any arbitrary graph and $G^{(1)},\ldots,G^{(k)}$ be a random k-partition of G. We show that if we compute an arbitrary EDCS of each graph $G^{(i)}$ (with no coordination across different graphs) and combine them together, we obtain an EDCS for the original graph G.

1. Let $G^{(1)},\ldots,G^{(k)}$ be a random k-partition of the graph G.
2. For any $i \in [k]$, compute $C^{(i)} := \text{EDCS}(G,\beta,(1-\lambda)\cdot\beta)$ for parameters
$$\lambda = \Theta\left((\frac{\varepsilon}{\log n})^2\right) \quad \text{and} \quad \beta := \Theta(\lambda^{-3}\cdot\log n).$$
3. Let $C := \bigcup_{i=1}^{k} C^{(i)}$.

LEMMA 4.1. *With probability $1-4/n^7$, the subgraph C is an $\text{EDCS}(G,\beta_C,\beta_C^-)$ for parameters:*
$$\lambda_C := O(\log n)\cdot\lambda^{1/2} \quad \beta_C := (1+\lambda_C)\cdot k\cdot\beta$$
$$\beta_C^- := (1-2\lambda_C)\cdot k\cdot\beta.$$

Proof. Recall that each graph $G^{(i)}$ is an edge sampled subgraph of G with sampling probability $p=\frac{1}{k}$. By Lemma 3.4 for graphs $G^{(i)}$ and $G^{(j)}$ (for $i\neq j\in[k]$) and their EDCSs $C^{(i)}$ and $C^{(j)}$, with probability $1-4/n^9$, for all vertices $v\in V$:

$$|d_{C^{(i)}}(v) - d_{C^{(j)}}(v)| \leq O(\log n)\cdot\lambda^{1/2}\cdot\beta = \lambda_C\cdot\beta. \tag{4.3}$$

By taking a union bound on all $\binom{k}{2}\leq n^2$ pairs of subgraphs $G^{(i)}$ and $G^{(j)}$ for $i\neq j\in[k]$, the above property holds for all $i,j\in[k]$, with probability at least $1-4/n^7$. We condition on this event.

We now prove that C is indeed an $\text{EDCS}(G,\beta_C,\beta_C^-)$. First, consider an edge $(u,v)\in C$ and let $j\in[k]$ be such that $(u,v)\in C^{(j)}$ as well. We have,

$$\begin{aligned} d_C(u)+d_C(v) &= \sum_{i=1}^{k} d_{C^{(i)}}(u) + \sum_{i=1}^{k} d_{C^{(i)}}(v) \\ &\underset{\text{Eq (4.3)}}{\leq} k\left(d_{C^{(j)}}(u)+d_{C^{(j)}}(v)\right) + k\lambda_C\beta \\ &\text{(by Property (P1) of EDCS } C^{(j)} \text{ with parameter } \beta) \\ &\leq k\cdot\beta + k\cdot\lambda_C\beta = \beta_C. \end{aligned}$$

Hence, C satisfies Property (P1) of EDCS for parameter β_C. Now consider an edge $(u,v)\in G\setminus C$ and let $j\in[k]$ be such that $(u,v)\in G^{(j)}\setminus C^{(j)}$ (recall that each edge in G is sent to exactly one graph $G^{(j)}$ in the random k-partition). We have,

$$\begin{aligned} d_C(u)+d_C(v) &= \sum_{i=1}^{k} d_{C^{(i)}}(u) + \sum_{i=1}^{k} d_{C^{(i)}}(v) \\ &\underset{\text{Eq (4.3)}}{\geq} k\left(d_{C^{(j)}}(u)+d_{C^{(j)}}(v)\right) - k\lambda_C\beta \\ &\geq k\cdot(1-\lambda)\beta - k\lambda_C\beta \\ &\text{(by Property (P2) of EDCS } C^{(j)} \text{ with parameter } (1-\lambda)\cdot\beta) \\ &\geq (1-2\lambda_C)\cdot k\cdot\beta = \beta_C^-. \end{aligned}$$

Hence, C also satisfies the second property of EDCS for parameter β_C^-, finalizing the proof. ∎

4.2 EDCS as a Coreset for Matching and Vertex Cover We are now ready to present our randomized coresets for matching and vertex cover using the EDCS as the coreset, formalizing Result 1.

THEOREM 4.1. *Let $G(V,E)$ be a graph and $G^{(1)},\ldots,G^{(k)}$ be a random k-partition of G. For any $\varepsilon\in(0,1)$, any $\text{EDCS}(G^{(i)},\beta,(1-\lambda)\cdot\beta)$ for $\lambda := \Theta\left((\frac{\varepsilon}{\log n})^2\right)$ and $\beta := \Theta(\varepsilon^{-6}\cdot\log^7 n)$ is a $(3/2+\varepsilon)$-approximation randomized composable coreset of size $O(n\cdot\beta)$ for the maximum matching problem.*

Proof. By Lemma 4.1, the union of the coresets, i.e., the k EDCSs, is itself an $\text{EDCS}(G,\beta_C,\beta_C^-)$, such that $\beta_C^- = (1-\Theta(\varepsilon))\cdot\beta_C$. Hence, by Lemma 2.2, the maximum matching in this EDCS is of size $(2/3-\varepsilon)\cdot\mathsf{MM}(G)$. The bound on the size of the coreset follows from Property (P1) of EDCS as maximum degree in the EDCS computed by each machine is at most β and hence size of coreset is $O(n\cdot\beta)=\widetilde{O}_\varepsilon(n)$. ∎

To present our coreset for the vertex cover problem, we need to slightly relax the definition of randomized coreset. Following [12], we augment the definition of randomized coresets by allowing the coresets to also contain a fixed solution (which is counted in the size of the coreset) to be directly added to the final solution of the composed coresets. In other words, the coreset contains both subsets of vertices (to be always included in the final vertex cover) and

edges (to guide the choice of additional vertices in the vertex cover). This definition is necessary for the vertex cover problem due to the hard-to-verify feasibility constraint of this problem; see [12] for details.

THEOREM 4.2. *Let $G(V,E)$ be a graph and $G^{(1)},\ldots,G^{(k)}$ be a random k-partition of G. For any $\varepsilon \in (0,1)$, any* EDCS$(G^{(i)},\beta,(1-\lambda)\cdot\beta)$ *for $\lambda := \Theta\left((\frac{\varepsilon}{\log n})^2\right)$ and $\beta := \Theta(\varepsilon^{-6}\cdot\log^7 n)$ plus the set of vertices with degree larger than $(1-\Theta(\varepsilon))\cdot\beta/2$ in the EDCS (to be added directly to the final vertex cover) is a $(2+\varepsilon)$-approximation randomized composable coreset of size $O(n\cdot\beta)$ for the minimum vertex cover problem.*

Proof. By Lemma 4.1, the union of the coresets, i.e., the k EDCSs, is itself an EDCS(G,β_C,β_C^-) C, such that $\beta_C^- = (1-\Theta(\varepsilon))\cdot\beta_C$. Suppose first that instead of each coreset fixing the set of vertices to be added to the final vertex cover, we simply add all vertices with degree more than $\beta_C^-/2$ to the vertex cover and then compute a minimum vertex cover of C. In this case, by Lemma 2.3, the returned solution is a $(2+\varepsilon)$-approximation to the minimum vertex cover of G.

To complete the argument, recall that the degree of any vertex $v \in V$ is essentially the same across all machines (up to an additive term of $\varepsilon\cdot\beta$) by Lemma 3.4, and hence the set of vertices with degree more than $\beta_C^-/2$ would be a subset of the set of fixed vertices across all machines. Moreover, any vertex added by any machine to the final vertex cover has degree at least $(1-\Theta(\varepsilon))\cdot\beta_C^-/2$ and hence we can apply Lemma 2.3, with a slightly smaller parameter ε to argue that the returned solution is still a $(2+\varepsilon)$-approximation. ∎

5 Massively Parallel Algorithms

In this section, we show that a careful adaptation of our coresets construction together with the structural results proven for EDCS in Section 3 can be used to obtain MPC algorithms with much smaller memory with a slight increase in the number of rounds.

THEOREM 5.1. *There exists an MPC algorithm that with high probability computes an $O(1)$ approximation to both maximum matching and minimum vertex cover in $O(\log\log n+\log\left(\frac{n}{s}\right))$ MPC rounds with per-machine memory $s = n^{\Omega(1)}$.*

By setting $s = O(n/\text{polylog}(n))$ in Theorem 5.1, we achieve an $O(1)$-approximation algorithm to both matching and vertex cover in $O(\log\log n)$ MPC rounds on machines of memory $O(n/\text{polylog}(n))$, formalizing Result 2.

In the following, for the sake of clarity, we mostly focus on proving Theorem 5.1 for the natural case when memory per machine is $s = \widetilde{O}(n)$, and postpone the proof for all range of parameter s to full version [11]. The overall idea of our algorithm is as follows. Instead of the edge sampled subgraphs used by our randomized coresets, we start by picking $k = O(n)$ *vertex* sampled subgraphs of G with sampling probability roughly $1/\sqrt{n}$ and send each to a separate machine. Each machine then locally computes an EDCS of its input (with parameters $\beta = \text{polylog}(n)$ and $\beta^- \approx \beta$) with no coordination across the machines. Unlike the MPC algorithm obtained by our randomized coreset approach (Corollary 1.2), where the memory per machine was as large as $\Theta(n\sqrt{n})$, here we cannot collect all these smaller EDCSes on a single machine of memory $\widetilde{O}(n)$. Instead, we repartition them across the machines again (and discard remaining edges) and repeat the previous process on this new graph. The main observation is that after each step, the maximum degree of the remaining graph (i.e., the union of all EDCSes) would drop quadratically (e.g., from potentially $\Omega(n)$ to $\widetilde{O}(\sqrt{n})$ in the first step). As such, in each subsequent step, we can pick a smaller number of vertex sampled subgraphs, each with a higher sampling probability than previous step, and still each graph fits into the memory of a single machine. Repeating this process for $O(\log\log n)$ steps reduces the maximum degree of the remaining graph to $\text{polylog}(n)$. At this point, we can store the final EDCS on a single machine and solve the problem locally.

Unfortunately this approach on its own would only yield a $(3/2)^{O(\log\log n)} = \text{polylog}(n)$ approximation to matching, since by Lemma 2.2 each recursion onto an EDCS of the graph could introduce a $3/2$-approximation. A similar problem exists for vertex cover. In the proof of Lemma 2.3, computing a vertex cover of G from its EDCS H involves two steps: we add to the vertex cover all vertices with high degree in H to cover the edges in $G\setminus H$, and then we separately compute a vertex cover for the edges in H. Since H cannot fit into a single machine, the second computation is done recursively: in each round, we find an EDCS of the current graph (which is partitioned amongst many machines), add to the vertex cover all high degree vertices in this EDCS, and then recurse onto the sparser EDCS. A straightforward analysis would only lead to an $O(\log\log n)$ approximation.

We improve the approximation factor for both vertex cover and matching by showing that they can serve as witnesses to each other. Every time we

add high-degree vertices to the vertex cover, we will also find a large matching incident to these vertices: we show that this can be done in $O(1)$ parallel rounds. We then argue that their sizes are always within a constant factor of each other, so both are a constant approximation for the respective problem (by Proposition 2.1).

The rest of this section is organized as follows. We first present our subroutine for computing the EDCS of an input graph in parallel using vertex sampled subgraphs. Next, we present a simple randomized algorithm for finding a large matching incident on high degree vertices of an input graph. Finally, we combine these two subroutines to provide our main parallel algorithm for approximating matching and vertex cover. We finish this section by specifying the MPC implementation of our parallel algorithm and finalize the proof of Theorem 5.1.

5.1 A Parallel Algorithm for EDCS We now present our parallel algorithm for computing an EDCS via vertex sampling. In the following, we use a slightly involved method of sampling the vertices using limited independence. This is due to technical reasons in the MPC implementation of this algorithm. To avoid repeating the arguments, we present our algorithm for all range of memory $s = n^{\Omega(1)}$, but encourage the reader to consider $s = n$ for more intuition.

ParallelEDCS(G, Δ, s). A parallel algorithm for EDCS of a graph G with maximum degree Δ on machines of memory $\widetilde{O}(s)$.

1. Define $p = (200 \log n) \cdot \sqrt{\frac{s}{n \cdot \Delta}}$ and $k = \frac{800 \log n}{p^2}$.
2. Create k vertex sampled subgraphs $G^{(1)}, \ldots, G^{(k)}$ on k different machines as follows:
   (a) Let $\kappa := (20 \log n)$. Each vertex v in G *independently* picks a *κ-wise independent hash function* $h_v : [k] \to [1/p]$.
   (b) The graph $G^{(i)}$ is the induced subgraph of G on vertices $v \in V$ with $h_v(i) = 0$.
3. Define parameters $\lambda := (2 \cdot \log n)^{-3}$ and $\beta := 750 \cdot \lambda^{-2} \cdot \ln(n)$.
4. For $i = 1$ to k in parallel: Compute $C^{(i)} = \text{EDCS}(G^{(i)}, \beta, (1-\lambda) \cdot \beta)$ locally on machine i.
5. Define the *multi-graph* $C(V, E_C)$ with $E_C := \bigcup_{i=1}^{k} C^{(i)}$ (allowing for multiplicities). Notice that this multi-graph is edge partitioned across the machines.

For any vertex $v \in V$, define $I(v) \subseteq k$ as the set of indices of the subgraphs that sampled vertex v. Indices in $I(v)$ are κ-wise independent random variables. Each graph $G^{(i)}$ is a vertex sampled subgraph of G with sampling probability p.

We state some simple properties of ParallelEDCS (the proof appear in full version [11]).

PROPOSITION 5.1. *For $\Delta \geq \left(\frac{n}{s}\right) \cdot \left(400 \cdot \log^{12}(n)\right)$, with probability $1 - 2/n^8$,*

1. *For any vertex $v \in V$, $|I(v)| = p \cdot k \pm \lambda \cdot p \cdot k$.*
2. *For any edge $e \in E$, there exists at least one index $i \in [k]$ such that e belongs to $G^{(i)}$.*

We now prove that the graph C defined in the last line of ParallelEDCS is also an EDCS of G with appropriate parameters. The proof is quite similar to that of Lemma 4.1 with some additional care to handle the difference between vertex sampled subgraphs and edge sampled ones (see full version [11]).

LEMMA 5.1. *For $\Delta \geq \left(\frac{n}{s}\right) \cdot \left(400 \cdot \log^{12}(n)\right)$, with probability $1 - 5/n^7$, C is an* EDCS(G, β_C, β_C^-) *for parameters:*

$$\lambda_C := \lambda^{1/2} \cdot \Theta(\log n) = o(1), \beta_C := pk \cdot (1 + \lambda_C)\beta,$$
$$\beta_C^- = pk \cdot (1 - \lambda_C)\beta.$$

Before moving on, we also show that the memory of $\widetilde{O}(s)$ per machine in ParallelEDCS is enough for storing each subgraph $G^{(i)}$ and computing $C^{(i)}$ locally.

CLAIM 5.2. *With probability $1 - 1/n^{18}$, the total number of edges in each subgraph $G^{(i)}$ of G in* ParallelEDCS(G, Δ, s) *is $O(s \cdot \log^2 n)$.*

Proof. Let v be a vertex in $G^{(i)}$. By the independent sampling of vertices in a vertex sampled subgraph, we have that $\mathbb{E}[d_{G^{(i)}}(v)] = p \cdot d_G(v) \leq p \cdot \Delta = \Theta(\sqrt{\frac{s \cdot \Delta}{n}} \cdot \log n)$. By Chernoff bound, with probability $1 - 1/n^{20}$, degree of v is $O(\sqrt{\frac{s \cdot \Delta}{n}} \cdot \log n)$. We can then take a union bound on all vertices in $G^{(i)}$ and have that with probability $1 - 1/n^{19}$, the maximum degree of $G^{(i)}$ is $O(\sqrt{\frac{s \cdot \Delta}{n}} \cdot \log n)$. At the same time, the expected number of vertices sampled in $G^{(i)}$ is at most $p \cdot n = \Theta(\sqrt{\frac{s \cdot n}{\Delta}} \cdot \log n)$. Another application of Chernoff bound ensures that the total number of vertices sampled in $G^{(i)}$ is $O(\sqrt{\frac{s \cdot n}{\Delta}} \cdot \log n)$ with probability $1 - 1/n^{19}$. As a result, the total number of edges in $G^{(i)}$ is $O(\sqrt{\frac{s \cdot \Delta}{n}} \cdot \log n) \cdot O(\sqrt{\frac{s \cdot n}{\Delta}} \cdot \log n) = O(s \cdot \log^2 n)$ with probability at least $1 - 1/n^{18}$. ∎

5.2 Random Match Algorithm We design a subroutine for finding a large matching incident on the set of "high" degree vertices of a given graph G which its edges are distributed across many machines.

$\mathsf{RandomMatch}(G, S, \Delta)$. A parallel algorithm for finding a matching M incident on given vertices S in a graph G with maximum degree Δ.

1. Sample each vertex in S with probability $1/2$ independently to obtain a set S'.
2. For each vertex in S' pick one of its incident edges to $G \setminus S'$ uniformly at random. Let E_{smpl} be the set of these edges.
3. Let M be the matching in E_{smpl} consists of all edges with unique endpoints; these are edges $(u, v) \in E_{\mathsf{smpl}}$ such that neither u nor v are incident on any other edge of E_{smpl}.

We prove that if the set S consists of high degree vertices of G, then $\mathsf{RandomMatch}(G, S, \Delta)$ finds a large matching in S.

LEMMA 5.3. *Suppose $G(V, E)$ is a graph with maximum degree $\Delta \geq 100 \log n$ and $S \subseteq V$ is such that for all $v \in S$, $d_G(v) \geq \Delta/3$. The size of the matching $M := \mathsf{RandomMatch}(G, S, \Delta)$ is in expectation $\mathbb{E}\,|M| = \Theta(|S|)$.*

Proof. Fix any vertex $v \in S'$; we argue that with high probability, degree of v to vertices in $G \setminus S'$ is at least $\Delta/7$. This follows immediately as in expectation, at most half of the neighbors of v belong to S' and we can apply Chernoff bound as $\Delta \geq 100 \log n$. We apply a union bound on all vertices in S' and in the following we condition on the event that all these vertices have at least $\Delta/7$ edges to $G \setminus S'$.

By construction, any vertex in S' has degree exactly one in E_{smpl}. As such, to lower bound the size of M, we only need to lower bound the number of vertices in $G \setminus S'$ that have degree exactly one in E_{smpl}. Fix a vertex $v \in S'$ and consider the neighbor $u \in G \setminus S'$ of v in E_{smpl}. We know that u has at most $\Delta - 1$ other neighbors in S' and each of these neighbors are choosing u with probability at most $7/\Delta$ (as each of them has at least $\Delta/7$ neighbors).

$$\Pr\left(u \text{ has degree 1 in } E_{\mathsf{smpl}}\right) \geq \left(1 - \frac{7}{\Delta}\right)^{\Delta - 1} = \Theta(1).$$

As such, in expectation, $\Theta(S)$ vertices in $G \setminus S'$ also have degree exactly one in E_{smpl}, which implies $\mathbb{E}\,|M| = \Theta(|S|)$. ∎

5.3 A Parallel Algorithm for Matching and Vertex Cover We now present our main parallel algorithm. For sake of clarity, we present and analyze our algorithm here for the case when the memory allowed per each machine is $\widetilde{O}(n)$. In the full version of the paper [11], we show how to easily extend this algorithm to the case when memory per machine is $O(s)$ for any choice of $s = n^{\Omega(1)}$.

$\mathsf{ParallelAlgorithm}(G, \Delta)$. A parallel algorithm for computing a vertex cover V_{ALG} and a matching M_{ALG} of a given graph G with maximum degree at most Δ.

1. If $\Delta \leq \left(400 \cdot \log^{12} n\right)$ send G to a single machine and run the following algorithm locally: Compute a maximal matching M_{ALG} in G and let V_{ALG} be the set of vertices matched by M_{ALG}. Return V_{ALG} and M_{ALG}.
2. If $\Delta > \left(400 \cdot \log^{12} n\right)$, we run the following algorithm.
3. Compute an EDCS $C := \mathsf{ParallelEDCS}(G, \Delta, n)$ in parallel. Let β_C, β_C^- be the parameters of this EDCS (as specified in Claim 5.4 below).
4. Define $V_{\text{HIGH}} := \left\{v \in V \mid d_C(v) \geq \beta_C^-/2\right\}$ be the set of "high" degree vertices in C.
5. Compute a matching $M_{\text{HIGH}} := \mathsf{RandomMatch}(C, V_{\text{HIGH}}, \beta_C)$.
6. Define $V^- := V \setminus \left(V_{\text{HIGH}} \cup V(M_{\text{HIGH}})\right)$ as the set of vertices that are neither high degree in C nor matched by M_{HIGH}. Let C^- be the induced subgraph of C on vertices V^- with parallel edges removed.
7. Recursively compute $(V_{\text{REC}}, M_{\text{REC}}) := \mathsf{ParallelAlgorithm}(C^-, \beta_C)$.
8. Return $V_{\text{ALG}} := V_{\text{HIGH}} \cup V(M_{\text{HIGH}}) \cup V_{\text{REC}}$ and $M_{\text{ALG}} := M_{\text{HIGH}} \cup M_{\text{REC}}$.

The following claim is a corollary of Lemma 5.1.

CLAIM 5.4. *$C := \mathsf{ParallelEDCS}(G, \Delta, n)$ computed in $\mathsf{ParallelAlgorithm}(G, \Delta)$ is an $\text{EDCS}(G, \beta_C, \beta_C^-)$ for parameters:*

$$\lambda_C := o(1) \quad \beta_C := \sqrt{\Delta} \cdot O(\log^5 n) \quad \beta_C^- := (1 - \lambda_C) \cdot \beta_C$$

with probability at least $1 - 1/n^5$.

Similarly, the following follows from Lemma 5.3.

CLAIM 5.5. *Conditioned on* $C = \text{EDCS}(G, \beta_C, \beta_C^-)$, *matching* $M_{\text{HIGH}} = \mathsf{RandomMatch}(C, V_{\text{HIGH}}, \beta_C)$ *has expected size* $\mathbb{E}\,|M_{\text{HIGH}}| = \Omega(|V_{\text{HIGH}}|)$.

Let T be the number of recursive calls made by $\mathsf{ParallelAlgorithm}(G, \Delta)$. We refer to any $t \in [T]$ as a *step* of ParallelAlgorithm. We bound the total number of steps as follows. The proof is straightforward.

CLAIM 5.6. *The total number of steps made by* $\mathsf{ParallelAlgorithm}(G, \Delta)$ *is* $T = O(\log\log\Delta)$.

In each step, ParallelAlgorithm runs the subroutines ParallelEDCS and RandomMatch once. We say that a run of ParallelEDCS is *valid* in this step iff the high probability event in Claim 5.4 happens. Roughly speaking, this means that ParallelEDCS is valid when it returns the "correct" output. Additionally, we say that a step of ParallelAlgorithm is valid if ParallelEDCS subroutine in this step is valid. We define $\mathcal{E}_{\mathsf{valid}}$ as the event that all T steps of $\mathsf{ParallelAlgorithm}(G, \Delta)$ are valid. By Claims 5.4 each step of ParallelAlgorithm is valid with probability at least $1 - 1/n^5$. As there are in total $T = O(\log\log n)$ steps by Claim 5.6, $\mathcal{E}_{\mathsf{valid}}$ happens with probability at least $1 - 1/n^4$.

We are now ready to prove the correctness of ParallelAlgorithm.

LEMMA 5.7. $\mathsf{ParallelAlgorithm}(G, n)$ *with constant probability outputs an* $O(1)$*-approximate matching* M_{ALG} *and* $O(1)$*-approximate vertex* V_{ALG} *of* G.

Proof. It is clear that the second parameter in $\mathsf{ParallelAlgorithm}(G, n)$ is an upper bound on the maximum degree of G and hence G satisfies the requirement of ParallelAlgorithm. In the following, we condition on the event $\mathcal{E}_{\mathsf{valid}}$ which happens with high probability by the above discussion. As such, we also have that any recursive call to $\mathsf{ParallelAlgorithm}(C^-, \beta_C)$ is valid (i.e., β_C is indeed an upper bound on degree of C^-) simply because C^- is a subgraph of an EDCS and hence its maximum degree is bounded by β_C.

We first argue that V_{ALG} and M_{ALG} are respectively a feasible vertex cover and a feasible matching of G. The case for M_{ALG} is straightforward; the set of vertices matched by M_{HIGH} is disjoint from the vertices in M_{REC} as all vertices matched by M_{HIGH} are removed in C^-, and hence (by induction) $M_{\text{ALG}} = M_{\text{HIGH}} \cup M_{\text{REC}}$ is a valid matching in G. Now consider the set of vertices V_{ALG}. By conditioning on the event $\mathcal{E}_{\mathsf{valid}}$, C is indeed an $\text{EDCS}(G, \beta_C, \beta_C^-)$. Hence, by Property (P2) of EDCS C, any edge $e \in G \setminus C$ has at least one neighbor in V_{HIGH} and is thus covered by V_{HIGH}. Additionally, as we pick $V(M_{\text{HIGH}})$ in the vertex cover, any edge incident on these vertices are also covered. This implies that $V_{\text{HIGH}} \cup V(M_{\text{HIGH}})$ plus any vertex cover of the remaining graph C^- is a feasible vertex cover of G. As V_{REC} is a feasible vertex cover of C^- by induction, we obtain that V_{ALG} is also a feasible vertex cover of G (the base case in step 1 where a maximal matching is computed locally is trivial).

We now show that sizes of M_{ALG} and V_{ALG} are within a constant factor of each other with constant probability. By Proposition 2.1 this implies that both are an $O(1)$-approximation to their respective problem. At each step, the set of vertices added to the V_{ALG} are of size $|V(M_{\text{HIGH}})| + |V_{\text{HIGH}}| \leq 3\,|V_{\text{HIGH}}|$ (as M_{HIGH} is incident on V_{HIGH}). The set of edges added to matching M_{ALG} are of size M_{HIGH} which is in expectation equal to $\Theta(|V_{\text{HIGH}}|)$ by Claim 5.5. As such, by induction and linearity of expectation, this implies that $\mathbb{E}\,|M_{\text{ALG}}| = \Theta(|V_{\text{ALG}}|)$ (the base case is again trivial). To conclude, we can apply a Markov bound (on size of $|V_{\text{ALG}}| - |M_{\text{ALG}}|$) and obtain that with constant probability $|M_{\text{ALG}}| = \Theta(|V_{\text{ALG}}|)$, which finalizes the proof. ∎

We note that in Lemma 5.7, we only achieved a constant factor probability of success for ParallelAlgorithm. We can however run this algorithm in parallel $O(\log n)$ times and pick the best solution to achieve a high probability of success while still having $\widetilde{O}(n)$ memory per machine and $O(\log\log n)$ rounds.

5.4 MPC Implementation of the Parallel Algorithm In this section, we briefly specify the details in implementing ParallelAlgorithm in the MPC model on machines of memory $\widetilde{O}(n)$. Throughout this section, we assume that the event $\mathcal{E}_{\mathsf{valid}}$ defined in the previous section holds and hence we are implicitly conditioning on this (high probability) event.

Our implementation is based on using by now standard tools in the MPC model for sorting and search in parallel introduced originally by [41] as specified in [28]. On machines with memory $n^{\Omega(1)}$, the sort operation in [41] allows us to sort a set of key-value pairs of size $\text{polylog}(n)$ in $O(1)$ MPC rounds. We can also do a parallel search: given a set A of key-value pairs and a set of queries each containing a key of an element in A, we can annotate each query with the corresponding key-value pair from A, again in $O(1)$ MPC rounds.

We follow the approach of [28] by using these operations to broadcast information from vertices to their incident edges. We build a collection of key-value pairs, where each key is a vertex and the value is the corresponding information. Then, each edge (u, v) may issue two queries to obtain the information

associated with u and v. For more details, we refer the reader to Section 6 in [28]. The following lemma states the main properties of our implementation.

LEMMA 5.8. *For a given graph $G(V,E)$, one can implement the following algorithms in the MPC model with at most $O(n)$ machines with memory $\widetilde{O}(n)$ with probability $1-1/n^4$:*

1. *Each call to* ParallelEDCS *in* ParallelAlgorithm(G,n) *in $O(1)$ MPC rounds.*
2. *Each call to* RandomMatch *in* ParallelAlgorithm(G,n) *in $O(1)$ MPC rounds.*
3. ParallelAlgorithm(G,n) *in $O(\log\log n)$ MPC rounds.*

We prove each part of this lemma separately.

Implementation of ParallelEDCS**.** To perform the vertex sampling approach in ParallelEDCS, we need to annotate each edge with the subgraph(s) it is assigned to. To do this, each vertex v in the current graph only needs to specify which subgraphs it resides on and broadcast this to its adjacent edges. Recall that unlike in [28], in our way of vertex sampling, each vertex can resides in multiple subgraphs (up to $\widetilde{O}(\sqrt{n})$ ones). Broadcasting this amount of information directly to adjacent edges of each vertex is not possible within the memory restrictions of the MPC model. However, recall that we are using an $O(\log n)$-wise independent hash function for determining the subgraphs each vertex v is going to reside on. Hence, the vertex v only needs to broadcast this hash function to its adjacent edges which requires $\text{polylog}(n)$ bits for representation (see, e.g. [63]) and thus can be done in $O(1)$ MPC rounds on machines of memory $n^{\Omega(1)}$.

We then send all edges assigned to one subgraph to a dedicated machine. By Claim 5.2, the number of edges assigned to each machine is at most $\widetilde{O}(n)$ with high probability and hence it can fit the memory of the machine. We can then locally compute an EDCS of this subgraph and annotate the edges in this EDCS as the edges of the final multigraph C. All this can be easily done in $O(1)$ MPC rounds, hence finalizing this part of the proof.

Implementation of RandomMatch**.** Each vertex in S' simply needs to annotate one of its edges uniformly at random, and each annotated edge only needs to "mark" its other endpoint in $G \setminus S'$. Any vertex in $G \setminus S'$ which is marked exactly once then inform the edge that marked it to join the matching M. This part can again be done in only $O(1)$ rounds on machines with memory $n^{\Omega(1)}$.

Implementation of ParallelAlgorithm**.** We can now combine the results in the previous two parts to show how to implement ParallelAlgorithm in the MPC model. Consider a step of ParallelAlgorithm. We saw that ParallelEDCS and RandomMatch can both be implemented in $O(1)$ MPC models. In particular, all edges in subgraph C computed by ParallelEDCS are now annotated and hence we can ignore all remaining edges. We can also compute the degree of each vertex in this subgraph in $O(1)$ rounds using a simple sort and search technique (see Lemma 6.1 in [28]). We can hence compute the set of vertices V_{HIGH} and pass it to RandomMatch as the set S. Finally, we can mark vertices in $V_{\text{HIGH}} \cup V(M_{\text{HIGH}})$ and remove them from the graph (by broadcasting this information to all their neighbors). After this, we know which vertices belong to C^- for the next step and which edges are still active. We can hence recursively solve the problem on the graph C^- in the next steps. As each step requires $O(1)$ MPC rounds and there are $O(\log\log n)$ steps in total by Claim 5.6, this results in an MPC algorithm with $O(\log\log n)$ rounds.

This concludes the proof of Theorem 5.1 for the case of $s = \widetilde{O}(n)$. The extension to all ranges of $s = n^{\Omega(1)}$ and further optimizing the number of machines appear in the full version [11].

5.5 Further Improvements In the remainder of this section, we show that using standard techniques, one can improve the approximation ratio of our matching algorithm significantly. In particular,

COROLLARY 5.1. *There exists an MPC algorithm that given a graph G and $\varepsilon \in (0,1)$, with high probability computes a $(2+\varepsilon)$-approximation to maximum matching of G in $O(\log(1/\varepsilon)\cdot\log\log n)$ MPC rounds using only $O(n/\text{polylog}(n))$ memory per machine.*

COROLLARY 5.2. *There exists an MPC algorithm that given a graph G and $\varepsilon \in (0,1)$, with high probability computes a $(1+\varepsilon)$-approximation to the maximum matching of G in $(1/\varepsilon)^{O(1/\varepsilon)} \cdot (\log\log n)$ MPC rounds using only $O(n/\text{polylog}(n))$ memory per machine.*

We note that above corollaries hold for all range of per machine memory $s = n^{\Omega(1)}$ similar to Theorem 5.1; however, for simplicity, we only consider the most interesting case of $s = O(n/\text{polylog}(n))$.

5.5.1 Proof of Corollary 5.2 The idea is to simply run our MPC algorithm in Theorem 5.1, to compute a matching M_{ALG}, remove all vertices matched by M_{ALG} from the graph G, and repeat. Clearly, the set of all matchings computed like this is

itself a matching of G. In the following, we show that only after $O(\log 1/\varepsilon)$ repetition of this procedure, one obtains a $(2+\varepsilon)$-approximation to the maximum matching of G.

Let $\alpha = O(1)$ be the approximation ratio of the algorithm in Theorem 5.1. Suppose we repeat the above process for $T := (\alpha \cdot \log(1/\varepsilon))$ steps. For any $t \in [T]$, let M_t be the matching computed so far, i.e., the union of the all the matchings in the first t applications of our α-approximation algorithm. Also let $G_{t+1} := G \setminus V(M_t)$, i.e., the graph remained after removing vertices matched by M_t. Note that M_{t+1} is an α-approximation to the maximum matching of G_{t+1}. Moreover, $\mathsf{MM}(G_{t+1}) \geq \mathsf{MM}(G) - 2|M_t|$ as each edge in M_t can only match (and hence remove) two vertices of any maximum matching of G. This implies that $|M_{t+1}| \geq |M_t| + \frac{1}{\alpha} \cdot (\mathsf{MM}(G) - 2|M_t|)$ for all $t \in [T]$. We now have,

$$\mathsf{MM}(G) - 2|M_T| \leq \left(1 - \frac{2}{\alpha}\right) \cdot (\mathsf{MM}(G) - 2 \cdot |M_{T-1}|)$$

(by applying the second equation to M_{T-1})

$$\leq \left(1 - \frac{2}{\alpha}\right)^2 (\mathsf{MM}(G) - 2 \cdot |M_{T-2}|)$$

(by recursively applying the previous equation)

$$\leq \left(1 - \frac{2}{\alpha}\right)^T \cdot \mathsf{MM}(G)$$

$$\leq \exp\left(-\frac{2}{\alpha} \cdot \alpha \cdot \log(1/\varepsilon)\right) \cdot \mathsf{MM}(G)$$

$$\leq \varepsilon \cdot \mathsf{MM}(G).$$

Hence, after $T = O(\log 1/\varepsilon)$ steps, the matching computed by the above algorithm is of size $(2+\varepsilon) \cdot \mathsf{MM}(G)$. It is immediate to verify that the new algorithm can be implemented in the MPC model with machines of memory $O(n/\text{polylog}(n))$ and $O(\log(1/\varepsilon) \cdot \log\log n)$ MPC rounds. ∎

5.5.2 Proof of Corollary 5.2 Corollary 5.2 can be proven using Theorem 5.1 plus a simple adaption of the multi-pass streaming algorithm of McGregor [57] for maximum matching to the MPC model. The high level approach in [57] is to reduce the problem of finding a $(1+\varepsilon)$-approximate maximum matching in G to many instances of finding a maximal matching in multiple adaptively chosen subgraphs of G. It was then shown that there exists a single pass streaming algorithm that can both determine the appropriate subgraph of G needed in each step of this reduction and compute a maximal matching of this subgraph. Hence, after by this streaming algorithm in multiple passes over the stream, we obtain a $(1+\varepsilon)$-approximation to the maximum matching.

We show that essentially the same approach can also be used in the MPC model. The main difference is to switch from computing a maximal matching to finding an $O(1)$-approximate maximum matching using our Theorem 5.1. The details of this reduction are postponed to the full version of the paper [11].

Acknowledgements The first author is grateful to his advisor Sanjeev Khanna for the previous collaboration in [12] that was the starting point of this project, to Michael Kapralov for helpful discussions regarding the streaming matching problem and results in [48], and to Krzysztof Onak for helpful discussions regarding the results in [28].

References

[1] S. Abbar, S. Amer-Yahia, P. Indyk, S. Mahabadi, and K. R. Varadarajan. Diverse near neighbor problem. In *Symposuim on Computational Geometry 2013, SoCG '13, Rio de Janeiro, Brazil, June 17-20, 2013*, pages 207–214, 2013.

[2] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51(4):606–635, 2004.

[3] K. J. Ahn and S. Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. *Inf. Comput.*, 222:59–79, 2013.

[4] K. J. Ahn and S. Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. In *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2015, Portland, OR, USA, June 13-15, 2015*, pages 202–211, 2015.

[5] K. J. Ahn, S. Guha, and A. McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 459–467. SIAM, 2012.

[6] K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 5–14, 2012.

[7] Y. Ai, W. Hu, Y. Li, and D. P. Woodruff. New characterizations in turnstile streams with applications. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 20:1–20:22, 2016.

[8] N. Alon, A. Moitra, and B. Sudakov. Nearly complete graphs decomposable into large induced matchings and their applications. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 1079–1090, 2012.

[9] N. Alon, N. Nisan, R. Raz, and O. Weinstein. Welfare maximization with limited interaction. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1499–1512, 2015.

[10] A. Andoni, A. Nikolov, K. Onak, and G. Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 574–583, 2014.

[11] S. Assadi, M. Bateni, A. Bernstein, V. S. Mirrokni, and C. Stein. Coresets meet EDCS: algorithms for matching and vertex cover on massive graphs. *CoRR*, abs/1711.03076, 2017.

[12] S. Assadi and S. Khanna. Randomized composable coresets for matching and vertex cover. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017, Washington DC, USA, July 24-26, 2017*, pages 3–12, 2017.

[13] S. Assadi, S. Khanna, and Y. Li. On estimating maximum matching size in graph streams. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1723–1742, 2017.

[14] S. Assadi, S. Khanna, Y. Li, and G. Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1345–1364, 2016.

[15] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause. Streaming submodular maximization: massive data summarization on the fly. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 671–680, 2014.

[16] M. Balcan, S. Ehrlich, and Y. Liang. Distributed k-means and k-median clustering on general communication topologies. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 1995–2003, 2013.

[17] S. Baswana, M. Gupta, and S. Sen. Fully dynamic maximal matching in o(log n) update time. *SIAM J. Comput.*, 44(1):88–113, 2015.

[18] M. Bateni, A. Bhaskara, S. Lattanzi, and V. S. Mirrokni. Distributed balanced clustering via mapping coresets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2591–2599, 2014.

[19] P. Beame, P. Koutris, and D. Suciu. Communication steps for parallel query processing. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*, pages 273–284, 2013.

[20] S. Behnezhad, M. Derakhshan, H. Esfandiari, E. Tan, and H. Yami. Brief announcement: Graph matching in massive datasets. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017, Washington DC, USA, July 24-26, 2017*, pages 133–136, 2017.

[21] A. Bernstein and C. Stein. Fully dynamic matching in bipartite graphs. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 167–179, 2015.

[22] A. Bernstein and C. Stein. Faster fully dynamic matchings with small approximation ratios. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 692–711, 2016.

[23] S. Bhattacharya, M. Henzinger, D. Nanongkai, and C. E. Tsourakakis. Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 173–182, 2015.

[24] L. Bulteau, V. Froese, K. Kutzkov, and R. Pagh. Triangle counting in dynamic graph streams. *Algorithmica*, 76(1):259–278, 2016.

[25] R. Chitnis, G. Cormode, H. Esfandiari, M. Hajiaghayi, A. McGregor, M. Monemizadeh, and S. Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1326–1344, 2016.

[26] R. H. Chitnis, G. Cormode, M. T. Hajiaghayi, and M. Monemizadeh. Parameterized streaming: Maximal matching and vertex cover. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1234–1251, 2015.

[27] M. Crouch and D. S. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, pages 96–104, 2014.

[28] A. Czumaj, J. Lacki, A. Madry, S. Mitrovic, K. Onak, and P. Sankowski. Round compression for parallel matching algorithms. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 471–484, 2018.

[29] R. da Ponte Barbosa, A. Ene, H. L. Nguyen, and J. Ward. The power of randomization: Distributed submodular maximization on massive datasets. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1236–1244, 2015.

[30] R. da Ponte Barbosa, A. Ene, H. L. Nguyen, and J. Ward. A new framework for distributed submodular maximization. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 645–654, 2016.

[31] S. Dobzinski. Computational efficiency requires simple taxation. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 209–218, 2016.

[32] S. Dobzinski, N. Nisan, and S. Oren. Economic efficiency requires interaction. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 233–242, 2014.

[33] S. Eggert, L. Kliemann, and A. Srivastav. Bipartite graph matchings in the semi-streaming model. In *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, pages 492–503, 2009.

[34] L. Epstein, A. Levin, J. Mestre, and D. Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM J. Discrete Math.*, 25(3):1251–1265, 2011.

[35] H. Esfandiari, M. Hajiaghayi, and M. Monemizadeh. Finding large matchings in semi-streaming. In *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain.*, pages 608–614, 2016.

[36] H. Esfandiari, M. T. Hajiaghayi, V. Liaghat, M. Monemizadeh, and K. Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1217–1233, 2015.

[37] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005.

[38] J. Fox, H. Huang, and B. Sudakov. On graphs decomposable into induced matchings of linear sizes. *Bulletin of the London Mathematical Society*, 49(1):45–57, 2017.

[39] M. Ghaffari, T. Gouleakis, C. Konrad, S. Mitrovic, and R. Rubinfeld. Improved massively parallel computation algorithms for mis, matching, and vertex cover. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 129–138, 2018.

[40] A. Goel, M. Kapralov, and S. Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 468–485. SIAM, 2012.

[41] M. T. Goodrich, N. Sitchinava, and Q. Zhang. Sorting, searching, and simulation in the mapreduce framework. In *Algorithms and Computation - 22nd International Symposium, ISAAC 2011, Yokohama, Japan, December 5-8, 2011. Proceedings*, pages 374–383, 2011.

[42] W. Gowers. Some unsolved problems in additive/combinatorial number theory. *preprint*, 2001.

[43] V. Guruswami and K. Onak. Superlinear lower bounds for multipass graph processing. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 287–298, 2013.

[44] N. J. A. Harvey, C. Liaw, and P. Liu. Greedy and local ratio algorithms in the mapreduce model. *CoRR*, abs/1806.06421, 2018.

[45] A. Hassidim, J. A. Kelner, H. N. Nguyen, and K. Onak. Local graph partitions for approximation and testing. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 22–31, 2009.

[46] Z. Huang, B. Radunovic, M. Vojnovic, and Q. Zhang. Communication complexity of approximate matching in distributed graphs. In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, pages 460–473, 2015.

[47] P. Indyk, S. Mahabadi, M. Mahdian, and V. S. Mirrokni. Composable core-sets for diversity and coverage maximization. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pages 100–108, 2014.

[48] M. Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1679–1697, 2013.

[49] M. Kapralov, S. Khanna, and M. Sudan. Approximating matching size from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 734–751, 2014.

[50] M. Kapralov, Y. T. Lee, C. Musco, C. Musco, and A. Sidford. Single pass spectral sparsification in dynamic streams. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 561–570, 2014.

[51] M. Kapralov and D. Woodruff. Spanners and sparsifiers in dynamic streams. *PODC*, 2014.

[52] H. J. Karloff, S. Suri, and S. Vassilvitskii. A model of computation for mapreduce. In *Proceedings of*

the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010, pages 938–948, 2010.

[53] C. Konrad. Maximum matching in turnstile streams. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 840–852, 2015.

[54] C. Konrad, F. Magniez, and C. Mathieu. Maximum matching in semi-streaming with few passes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, pages 231–242, 2012.

[55] S. Lattanzi, B. Moseley, S. Suri, and S. Vassilvitskii. Filtering: a method for solving graph problems in mapreduce. In *SPAA 2011: Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures, San Jose, CA, USA, June 4-6, 2011 (Co-located with FCRC 2011)*, pages 85–94, 2011.

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

[57] A. McGregor. Finding graph matchings in data streams. In *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, pages 170–181, 2005.

[58] A. McGregor. Graph stream algorithms: a survey. *SIGMOD Record*, 43(1):9–20, 2014.

[59] A. McGregor, D. Tench, S. Vorotnikova, and H. T. Vu. Densest subgraph in dynamic graph streams. In *Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part II*, pages 472–482, 2015.

[60] A. McGregor and S. Vorotnikova. Planar matching in streams revisited. *In Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 19th International Workshop, APPROX 2016, and 20th International Workshop, RANDOM 2016*, 2016.

[61] V. S. Mirrokni and M. Zadimoghaddam. Randomized composable core-sets for distributed submodular maximization. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 153–162, 2015.

[62] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 2049–2057, 2013.

[63] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[64] O. Neiman and S. Solomon. Simple deterministic algorithms for fully dynamic maximal matching. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 745–754, 2013.

[65] H. N. Nguyen and K. Onak. Constant-time approximation algorithms via local improvements. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 327–336, 2008.

[66] K. Onak, D. Ron, M. Rosen, and R. Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1123–1131, 2012.

[67] K. Onak and R. Rubinfeld. Maintaining a large matching and a small vertex cover. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 457–464, 2010.

[68] M. Parnas and D. Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theor. Comput. Sci.*, 381(1-3):183–196, 2007.

[69] A. Paz and G. Schwartzman. A $(2 + \varepsilon)$-approximation for maximum weight matching in the semi-streaming model. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2153–2161, 2017.

[70] S. Solomon. Fully dynamic maximal matching in constant update time. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 325–334, 2016.

[71] Y. Yoshida, M. Yamamoto, and H. Ito. Improved constant-time approximation algorithms for maximum matchings and other optimization problems. *SIAM J. Comput.*, 41(4):1074–1093, 2012.

[72] S. A. Zadeh, M. Ghadiri, V. S. Mirrokni, and M. Zadimoghaddam. Scalable feature selection via distributed diversity maximization. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 2876–2883, 2017.