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

Sepehr Assadi*
University of Pennsylvania

MohammadHossein Bateni†
Google Research

Aaron Bernstein‡
Technical University of Berlin

Vahab Mirrokni§
Google Research

Cliff Stein¶
Columbia University

Abstract

As massive graphs become more prevalent, there is a rapidly growing need for *scalable* algorithms that solve classical graph problems, such as maximum matching and minimum vertex cover, on large datasets. 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. For example, we give:

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

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 in the recent paper of 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.

*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. Supported in part by Einstein Grant.

§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.

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 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 an (almost) 3/2-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 (almost) 3/2-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. [29], 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. [29].

1.1 Randomized Composable Coresets

Two examples of general techniques widely used for processing massive data sets are linear sketches (see e.g. [5,6,14,23,25,26,51,52,60]) and composable coresets (see e.g. [12,15,16,18,48,62,63]). 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 [62]

(see also [30])—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 (Randomized Composable Coreset [12, 62])**.** *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. 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. Any α-approximate randomized coreset of size $\widetilde{O}(n)$ for matching or vertex cover immediately yields the following algorithms for these problems on a graph $G(V,E)$ with n vertices and m edges: (see Proposition 6.1 for more details and proofs.)

- **Streaming:** An α-approximation single-pass streaming algorithm on random arrival streams using $\widetilde{O}(\sqrt{mn}+n) = \widetilde{O}(n^{1.5})$ space.
- **MPC:** An α-approximation MPC algorithm in two rounds with $O(\sqrt{m/n})$ machines, each with $\widetilde{O}(\sqrt{mn}+n) = \widetilde{O}(n^{1.5})$ memory.
- **Distributed:** An α-approximation simultaneous communication protocol on randomly partitioned inputs using $\widetilde{O}(n)$ communication per machine/player.

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 a $(2+\varepsilon)$-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 Section 8.3 of the full version, we show that the previous approach of [12] provably cannot go below 2.). 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 [38]. No $o(n^2)$-space algorithm is known for this task on adversarially ordered streams and the lower bound result by Kapralov [49] (see

also [41]) 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. [55] 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. *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 an (almost) $(3/2)$-approximation to the maximum matching problem.*

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 [49]. Although the lower bound of [49] does not preclude achieving the bounds of Corollary 1 in an adversarial order (because our space is $\widetilde{O}(n^{1.5})$ rather than $\widetilde{O}(n)$), the proof in [49] (see also [41]) 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, 39, 43]). 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 [49]. 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 [49], thus establishing the first provable separation between adversarial- and random-partitioned inputs in the blackboard model for approximating matchings.

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, 29, 45, 56]. As an application of Result 1, we obtain efficient MPC algorithms for matching and vertex cover in only two rounds of computation.

Corollary 2. *There exist MPC algorithms that with high probability achieve an (almost) $(3/2)$-approximation to matching and an (almost) 2-approximation to vertex cover in two MPC rounds and $\widetilde{O}(\sqrt{mn}+n)$ memory per machine*[1].

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 [62] 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. [56] 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, 56]), and hence minimizing the number of rounds is the primary goal in this model.

Third implication: distributed simultaneous communication. Maximum matching (and to a lesser degree vertex cover) has been studied previously in the simultaneous communication model

[1] The approximation factor for vertex cover degrades to 4 if one requires local computation on each machine to be polynomial time; see Remarks A.1 and 10.2 in the full version.

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, 54], and applications to mechanism design [9, 32, 33]. As an application of Result 1, we obtain the following corollary.

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

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 $\text{polylog}(n)$ 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 [57] (see also [29]), 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 [29], all MPC algorithms for matching and vertex cover [4, 5, 56] 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 [38, 59]). In a very recent breakthrough, Czumaj et al. [29] 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 [29] in the affirmative, (ii) we obtain an $O(1)$ approximation to vertex cover, answering another open question of [29], and (iii) we achieve all these using a considerably simpler algorithm and analysis than [29].

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. [40] 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 [40] are based on an earlier round-compression technique of [29], 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. [40]: 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 efficiently 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. [29]: 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 [29], in terms of both the local algorithm computed on each subgraph and in the analysis. The main approach in [29] 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.

Organization. We present our notation and definitions of EDCS and the MPC model in Section 2. We then prove our structural results on EDCS in Section 3. Section 4 briefly describes how to prove Result 1 using these results. Finally, Section 5 contains the proof sketch of Result 2. A full version of the paper containing omitted proofs and details appears at the end of the paper. In particular, we entirely postpone our new analysis of the randomized coreset of [12] as well as a lower bound on its performance to Section 8 of the full version but refer the reader to this result as a warm-up to our main results. Further related work are discussed in Section 6.5 of the full version.

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 vertex $v \in V$, $d_G(v)$ denotes the degree of v in the graph G.

Sampled Subgraphs. We use two different ways of sampling a graph $G(V, E)$. For $p \in (0, 1)$,

- 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

The Massively Parallel Computation (MPC) Model. We adopt the most stringent model of modern parallel computation among [10,19,42,53], 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. One typically requires that both $p, s = m^{1-\Omega(1)}$ i.e., polynomially smaller than the input size [10,53]. Computation proceeds in synchronous rounds: in each round, each machine performs some local computation and at the end of the round machines exchange messages. 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.

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

In the remainder of the paper, we use the terms "Property (P1)" and "Property (P2)" of EDCS to refer to the first and second items in Definition 2 above.

Previous work in [21,22] has shown that for any choice of $\beta^- < \beta$, an EDCS(G,β,β^-) always exists (see Lemma 7.4 in the full version). Moreover, it was shown in [21] (bipartite graphs) and [22] (general graphs) that an EDCS accurately preserves the maximum matching in G. It is easy to show a qualitatively similar result for vertex cover. We summarize these results in the lemma below (see Lemmas 7.5 and 7.6 in the full version for a more thorough discussion).

Lemma 2.1 ([21,22]). *Consider any parameters $\varepsilon > 0$, $\beta = \Omega(\varepsilon^{-3})$ and $\beta^- = (1-\Theta(\varepsilon)) \cdot \beta$. If H is an* EDCS(G,β,β^-) *of G then:* **(1)** $\mathsf{MM}(G) \leq (3/2+\varepsilon) \cdot \mathsf{MM}(H)$ *and* **(2)** *given only the graph H, one can recover a $(2+\varepsilon)$-approximation to $\mathsf{VC}(G)$ without further access to the original graph G.*

Proof Sketch. The proof that $\mathsf{MM}(G) \leq (3/2+\varepsilon) \cdot \mathsf{MM}(H)$ is presented in [21,22] and requires a fairly complicated argument. The proof for vertex cover however is simpler. The basic idea is that we start with $\mathsf{VC}(H)$ to cover the edges of H, and then by Property (P2) of EDCS H we can cover edges in $G \setminus H$ by adding to the vertex cover all vertices v with $d_H(v) \geq \beta^-/2$. ∎

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 dynamic graph. In this section, we prove several new structural properties of the EDCS itself. In particular, while it easy to see that in terms of edge-sets there can be many different EDCSes of some fixed graph $G(V,E)$ (consider G being a complete graph), we show that an EDCS is still "semi-unique" in that the degree distributions of every EDCS (for the same 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. This is in sharp contrast with similar objects such as maximum matchings or b-matchings, which can vary a lot within the same graph.

We now formally state the semi-uniqueness property (Lemma 3.1), which is used it to show that the EDCS is extremely robust under sampling and composition (Lemmas 3.2 and 3.3).

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* EDCS(G,β,β^-)*, and any vertex $v \in V$, $|d_A(v) - d_B(v)| = O(\log n)\cdot\lambda^{1/2}\cdot\beta$.*

Proof Sketch. We start with a set S_1 of vertices that have the most difference in degree between A and B. For simplicity, assume vertex degrees are all larger by an additive factor of Δ in A compared to B. We look at all neighbors of S_1 in $A \setminus B$, denoted by T_1, and then all neighbors of T_1 in $B \setminus A$ plus the original set S_1 which we denote by S_2. We then use the two properties of EDCS in Definition 2 to prove that the vertices in S_2 still have a larger degree in A compared to B by an additive factor which is only slightly smaller than Δ, while the size of S_2 is a constant factor larger than S_1. The main observation behind this claim is that since vertices in S have a "large" degree in A, their neighbors in $A \setminus B$ in T_1 should have a "small" degree to satisfy Property (P1) of EDCS A. Similarly, since vertices in S_1 have a "small" degree in B, and since the edges in $A \setminus B$ are missing from the EDCS B, by Property (P2) of EDCS B the vertices in T_1 should have a "large" degree in B. Applying this idea one more time to vertices in T_1 in order to obtain the set S_2, we can see that vertices in S_2 have a "large" degree in A and a "small" degree in B, and that the decrease in the original gap Δ between the degree of vertices in A vs B is only $O(\beta - \beta^-) = O(\lambda\cdot\beta)$.

We use this argument repeatedly to construct the next set S_3 and so on. With each iteration the new set S_i is larger than the previous one S_{i-1} by a constant *multiplicative* factor, while the A-vs.-B degree gap only decreases by a small *additive* factor from S_{i-1} to S_i. Now on one hand we can keep iterating this process as long as the A-vs.-B degree gap remains $\geq \Delta/2$. On the other hand, a multiplicative increase in set size can only happen for $O(\log(n))$ steps before we run out of vertices. Thus, the gap must decrease from Δ to $\Delta/2$ after only $O(\log(n))$ steps of a small additive decrease, which gives us an upper bound on the original gap Δ.

The formal proof is rather technical and is presented in Section 9.1 of the full version. ∎

EDCS in Sampled Subgraphs. We use the Degree Distribution Lemma to prove two lemmas regarding EDCS in sampled subgraphs. The first 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 3.2 (EDCS in Edge Sampled Subgraphs)**.** *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 from edge sampling is that there will be a huge gap between the degree of a vertex between the two EDCS if it is sampled in one subgraph but not the other. However, we show that the degree of vertices that are sampled in *both* subgraphs are still almost the same.

Lemma 3.3 (EDCS in Vertex Sampled Subgraphs)**.** *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 proofs of both these lemmas proceed along the following lines. We start with any 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, H'_1 and H'_2 are both an EDCS of G_1 and G_2, respectively, with parameters (β, β^-). Moreover, since they are sampled from the same underlying EDCS H, H'_1 and H'_2 end up having similar degree distributions (denoted by $H'_1 \sim H'_2$). Finally, we use our degree distribution lemma (Lemma 3.1) to argue that for any arbitrary EDCS H_1 of G_1 and H_2 of G_2, degree distribution of H_1 (resp. H_2) is close to H'_1 (resp. H'_2), namely $H_1 \sim H'_1$ and $H_2 \sim H'_2$. We thus have $H_1 \sim H'_1 \sim H'_2 \sim H_2$, completing the proof. The formal proofs of these two lemmas are deferred to Section 9 of the full version.

4 Randomized Coresets for Matching and Vertex Cover

In this short section, we introduce our randomized coresets for matching and vertex cover and sketch the proof of Result 1. We show that one can compute an EDCS of the input graph G by simply taking the union of the EDCSes computed on each subgraph in the random k-partition. We then invoke Lemma 2.1 to obtain a large matching and a small vertex cover of G from this EDCS.

Let G be any arbitrary graph and $G^{(1)}, \ldots, G^{(k)}$ be a random k-partition of G. Our randomized coreset for both matching and vertex cover is simply any arbitrarily chosen EDCS of each $G^{(i)}$ with parameters $\beta = \text{polylog}(n)$ and $\beta^- = \big(1 - \Theta((\varepsilon/\log n)^2)\big) \cdot \beta$. Since the maximum degree of any EDCS is at most β by Property (P1) of EDCS, size of this coreset is only $O(n \cdot \beta) = \widetilde{O}(n)$. It thus only remains to prove the correctness of this coreset.

Let $H^{(i)}$ be the EDCS computed on each graph $G^{(i)}$. The first part of the proof shows that $H := \bigcup_{i=1}^{k} H^{(i)}$ is an EDCS for the original graph G with parameters (roughly) equal to $(k \cdot \beta, k \cdot \beta^-)$ with high probability. This is intuitively true because by Lemma 3.2, the degree of every vertex v across all subgraphs $H^{(i)}$ is essentially the same, so for any $i \in [k]$, we have $d_H(v) \approx k \cdot d_{H^{(i)}}(v)$. As a result, for any edge (u, v) in $H^{(i)}$ (for any $i \in [k]$), Property (P1) of EDCS $H^{(i)}$ ensures that $d_H(u) + d_H(v) \approx k \cdot (d_{H^{(i)}}(u) + d_{H^{(i)}}(v)) \lesssim k \cdot \beta$, so (u, v) satisfies Property (P1) of H as well. Similarly, every $(u, v) \in G \setminus H$ must be in $G^{(i)} \setminus H^{(i)}$ for some $i \in [k]$, so by Property (P2) of EDCS $H^{(i)}$ we have $d_H(u) + d_H(v) \approx k \cdot (d_{H^{(i)}}(u) + d_{H^{(i)}}(v)) \gtrsim k \cdot \beta^-$. So (u, v) satisfies Property (P2) of EDCS H as well. See Lemma 10.1 in the full version for the formal proof.

We can now easily conclude the proof of Result 1: since $H := \bigcup_{i=1}^{k} H^{(i)}$ is an EDCS of the original graph G with high probability, Lemma 2.1 guarantees that we can recover an (almost) $(3/2)$-approximate matching and an (almost) $(2+\varepsilon)$-approximate vertex cover for G using only H. see Theorems 9 and 10 in Section 10.2 of the full version for formal proofs

5 MPC Algorithms for Matching and Vertex Cover

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 while increasing the number of required rounds to only $O(\log \log n)$.

Theorem 4. *There exists an MPC algorithm that given a graph $G(V, E)$ with high probability computes an $O(1)$ approximation to both maximum matching and minimum vertex cover of G in $O(\log \log n + \log \left(\frac{n}{s}\right))$ MPC rounds on machines of memory $s = n^{\Omega(1)}$.*

By setting $s = O(n/\text{polylog}(n))$ in Theorem 4, 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 4 (reducing the approximation ratio to $(1+\varepsilon)$ for matching can be done by adapting the reduction of [58] from streaming to MPC model; see Section 11.6.2 in full version for details).

In the following, for the sake of clarity, we focus on sketching the proof of Theorem 4 for the natural case when memory per machine is $s = \widetilde{O}(n)$. (See Section 11.5 of full version for the additional (standard) ideas needed to reduce the memory to any $s = n^{\Omega(1)}$ at a cost of increasing the number of the rounds by an additive $O(\log\left(\frac{n}{s}\right))$ factor); The overall idea of the algorithm in this result 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 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 $\Omega(n)$ 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 2.1 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 2.1, 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. Whenever we add high-degree vertices to the vertex cover, we will also find a large matching incident to these vertices; this can be done in $O(1)$ parallel rounds. We then argue that their sizes are always within a constant factor of each other, which is well known to imply that both are a constant approximation for the respective problem (see Proposition 7.2).

The rest of this section is organized as follows. We first describe the subroutines for computing an EDCS from vertex sampled subgraphs and for obtaining a large matching incident on high degree vertices. We then combine these subroutines to provide our main parallel algorithm. The MPC implementation details of our algorithm as well as extension of the maximum matching algorithm to $(2+\varepsilon)$ and $(1+\varepsilon)$-approximation is deferred to Section 11.4 and Section 11.6 of the full version.

Main Subroutines. In the following two subroutines, both the input graph and the output edges are distributed across multiple machines. The formal description of these subroutines and the proofs of their correctness are presented in Section 11.1 and Section 11.2 of the full version.

ParallelEDCS(G, Δ). A parallel algorithm for computing an EDCS of a given graph G with maximum degree Δ on machines of memory $\widetilde{O}(n)$. With high probability, the computed EDCS has parameters $\beta = O(\sqrt{\Delta}) \cdot \text{polylog}(n)$ and $\beta^- \geq 0.99 \cdot \beta$ and is edge-partitioned across multiple machines.

RandomMatch(G, S, Δ). Let G be a graph with maximum degree Δ and let S be a set of vertices with degree at least $\Delta/3$ in G. RandomMatch(G, S, Δ) is a parallel algorithm running on machines of memory $n^{\Omega(1)}$ that finds a matching M such that every edge in M is incident to S, and $\mathbb{E}\,|M| = \Theta(|S|)$. (The existence of such a large matching in G follows from basic facts in graph theory; see Proposition 7.3.)

We briefly describe the main ideas behind each of these subroutines. ParallelEDCS(G, Δ) works by creating $O(\Delta)$ vertex sampled subgraphs of G with sampling probability roughly $1/\sqrt{\Delta}$, each on a separate machine. It is easy to see that the total number of edges in each such subgraph is at most $\widetilde{O}(n)$ with high probability and hence this graph fits the memory of a single machine. Each machine then computes an EDCS of its input with parameters $\beta = \text{polylog}(n)$ and $\beta^- \approx \beta$. Similar to Section 4, we can prove that the union of these edges is an EDCS of the original graph with the aforementioned parameters (this time using Lemma 3.3 for vertex sampled subgraphs).

RandomMatch(G, S, Δ) is based on a simple randomized procedure. The machines collectively sample each edge incident to S in G with probability $1/\Delta$ independently, and then add all edges with *unique endpoints* in this process to the final matching M (these edges can be found in parallel using a parallel sort subroutine). Since degree of vertices in G is at most Δ, while degree of vertices in S is at least $\Delta/3$, one can show that the expected size of the matching returned is $\Omega(|S|)$.

Parallel Algorithm for Matching and Vertex Cover. We now present our main algorithm.

ParallelAlgorithm(G, Δ). 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; 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}. Otherwise, continue the following algorithm.
2. Compute an EDCS $C :=$ ParallelEDCS(G, Δ) in parallel. Let $\beta_C = O(\sqrt{\Delta}) \cdot \text{polylog}(n)$, and $\beta_C^- = 0.99 \cdot \beta_C$ be the parameters of this EDCS.
3. 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.
4. Compute a matching $M_{\text{HIGH}} :=$ RandomMatch$(C, V_{\text{HIGH}}, \beta_C)$.
5. 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^-.
6. Recursively compute $(V_{\text{REC}}, M_{\text{REC}}) :=$ ParallelAlgorithm(C^-, β_C).
7. 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 total number of recursive calls made by ParallelAlgorithm(G, Δ) is $O(\log\log\Delta)$ as the maximum degree of the underlying graph drops from Δ to $\widetilde{O}(\sqrt{\Delta})$ after each call (see Claim 11.8 in the full version). The detailed proof of the following lemma appears in Section 11.3 of the full version.

Lemma 5.1. *For any graph $G(V, E)$,* ParallelAlgorithm(G, n)*, with constant probability, outputs a matching M_{ALG} which is an $O(1)$-approximation to the maximum matching of G and a vertex cover V_{ALG} which is an $O(1)$-approximation to the minimum vertex cover of G.*

Proof Sketch. In the following, we assume that all the calls made to ParallelEDCS and RandomMatch in ParallelAlgorithm return a correct answer which happens with high probability by a union bound.

We first argue that V_{ALG} and M_{ALG} are respectively a feasible vertex cover and a feasible matching of G with high probability. 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}. Since C is an EDCS(G, β_C, β_C^-), 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 is also covered. Thus $V_{\text{HIGH}} \cup V(M_{\text{HIGH}})$ covers all edges in $G \setminus C^-$, and (by induction) V_{REC} covers all the edges in C^-

We now show that the sizes of M_{ALG} and V_{ALG} are within a constant factor of each other in expectation, which is well known to imply that both are an $O(1)$-approximation to their respective problem (see Proposition 7.2). By correctness of RandomMatch, V_{HIGH} and M_{HIGH} are within a constant factor of each other in expectation. Moreover, clearly $|V(M_{\text{HIGH}})| \leq 2\,|V_{\text{HIGH}}|$, so the vertex cover $V_{\text{HIGH}} \cup V(M_{\text{HIGH}})$ and the matching M_{HIGH} chosen at each step are within a constant factor of each other. By induction, the final outputs V_{ALG} and M_{ALG} are also within a constant factor of each other in expectation. This, together with a simple application of Markov bound ensures that with probability $\Omega(1)$, V_{ALG} and M_{ALG} are an $O(1)$-approximation to their respective problem. ∎

We note that in Lemma 5.1, 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. In Sections 11.4.1 and 11.5 of full version, we further discuss optimizing the number of machines and the per-machine memory of this algorithm, respectively.

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 [49], and to Krzysztof Onak for helpful discussions regarding the results in [29].

FULL VERSION OF THE PAPER

6 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 7 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 range of settings. We focus on this task for two prominent graph optimization problems: maximum matching and minimum vertex cover. Our main result (Section 6.2) 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 an (almost) 3/2-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 (almost) 3/2-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 6.1). 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 6.3) 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. [29], 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. [29].

6.1 Randomized Composable Coresets

Two examples of general techniques widely used for processing massive data sets are linear sketches (see e.g. [5,6,14,23,25,26,51,52,60]) and composable coresets (see e.g. [12,15,16,18,48,62,63]). 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 [62] (see also [30])—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 3 (Randomized Composable Coreset [12, 62])**.** *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. 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 6.1. *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 6.1 can be found in Appendix A.

6.2 First Result: Improved Algorithms via a New Randomized Coreset

We start by studying the previous randomized coreset of [12] for matching, which was simply to pick a maximum matching of each machine's subgraph as its coreset. This is arguably the most natural approach to the problem and results in truly sparse subgraphs (maximum degree one). As a warm-up to our main results, we present a simpler and improved analysis (compared to that in [12]), which shows that this coreset achieves a 3-approximation (vs. 9-approximation proven in [12]). We also show that there exist graphs on which the approximation ratio of this coreset is at least 2. This suggests that to achieve a better than 2 approximation, fundamentally different ideas are needed which brings us to our first main result.

Result 3. *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. We notably go beyond the ubiquitous 2-approximation barrier for matching (in Section 8.3, we show that the previous approach of [12] provably cannot go below 2). *Result 3 yields a unified framework that improves upon the state-of-the art algorithms for matching and vertex cover across several computational models in one or all 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 [38]. No $o(n^2)$-space algorithm is known for this task on *adversarially ordered* streams and the lower bound result by Kapralov [49] (see also [41]) 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. [55] 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 direction application of our Result 3 improves the approximation ratio of this algorithm significantly albeit at a cost of a larger space requirement.

Corollary 5. *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 an (almost)* $(3/2)$*-approximation to the maximum matching problem.*

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 [49]. Although the lower bound of [49] does not preclude achieving the bounds of Corollary 5 in an adversarial order (because our space is $\widetilde{O}(n^{1.5})$ rather than $\widetilde{O}(n)$), the proof in [49] (see also [41]) 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, 39, 43]). 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 [49]. Our algorithm in Result 3 can be implemented with $\widetilde{O}(n)$ (per-player) communication in this model which goes strictly below the lower bound of [49], 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, 29, 45, 56]. As an application of Result 3, we obtain efficient MPC algorithms for matching and vertex cover in only *two* rounds of computation.

Corollary 6. *There exist MPC algorithms that with high probability achieve an (almost)* $(3/2)$*-approximation to matching and an (almost)* 2*-approximation to vertex cover in* two *MPC rounds and* $\widetilde{O}(\sqrt{mn}+n)$ *memory per machine*[2].

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 [62] for details on when this assumption applies.

[2]The approximation factor for vertex cover degrades to 4 if one requires local computation on each machine to be polynomial time; see Remarks A.1 and 10.2.

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. [56] 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, 56]), and hence minimizing the number of rounds is the primary goal in this model.

Third implication: distributed simultaneous communication. 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, 54], and applications to mechanism design [9, 32, 33]. As an application of Result 3, we obtain the following corollary.

Corollary 7. *There exist simultaneous communication protocols on* randomly partitioned inputs *that achieve (almost)* (3/2)*-approximation to matching and (almost)* 2*-approximation to vertex cover with high probability (over the randomness of the input partitioning) with only* $\widetilde{O}(n)$ *communication per machine/player.*

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].

6.3 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 4. *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 [57] (see also [29]), our Result 4 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 [29], all MPC algorithms for matching and vertex cover [4, 5, 56] 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 [38, 59]). In a very recent breakthrough, Czumaj et al. [29] 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 4 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 [29] in the affirmative, (ii) we obtain an $O(1)$ approximation to vertex cover, answering another open question of [29], and (iii) we achieve all these using a considerably simpler algorithm and analysis than [29].

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. [40] 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 [40] are based on an earlier round-compression technique of [29], 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 4 is shared with the later paper of Ghaffari et al. [40]: Result 3 appears only in our paper, and is entirely specific to the particular techniques that we use.

6.4 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 efficiently 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 3, 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 4, we use the following recursive procedure, which crucially relies upon on the robustness properties of the EDCS proved in Result 3: 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 3 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 4 relies on the high-level technique of vertex sampling from Czumaj et al. [29]: 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 [29], in terms of both the local algorithm computed on each subgraph and in the analysis. The main approach in [29] 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.

6.5 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, 29, 40, 45, 56], streaming algorithms [3–6, 13, 14, 26–28, 34–38, 41, 44, 49, 50, 54, 55, 58, 59, 61, 70], simultaneous communication model and similar distributed models [9, 12–14, 33, 44, 47], dynamic graphs [17, 21, 22, 65, 68, 72], and sub-linear time algorithms [46, 66, 67, 69, 74]. Beside the results mentioned already, most relevant to our work are the polylog(n)-space polylog(n)-approximation algorithm of [50]

for estimating the *size* of a maximum matching in random stream, and the (3/2)-approximation communication protocol of [41] 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], [48], and [62]. Composable coresets have been studied previously in nearest neighbor search [1], diversity maximization [48, 75], clustering [16, 18], and submodular maximization [15, 30, 31, 48, 62]. Moreover, while not particularly termed a composable coreset, the "merge and reduce" technique in graph streaming literature (see [59], Section 2.2) is identical to composable coresets.

7 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.

We use capital letters to denote random variables. Let $\{X_i\}_{i=1}^n$ and $\{Y_i\}_{i=1}^n$ be a sequence of random variables on a common probability space such that $\mathbb{E}[X_i \mid Y_1, \ldots, Y_{i-1}] = X_{i-1}$ for all i. The sequence $\{X_i\}$ is referred to as a *martingale* with respect to $\{Y_i\}$. A summary of concentration bounds we use in this paper appears in Appendix B.1.

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

7.1 The Massively Parallel Computation (MPC) Model

We adopt the most stringent model of modern parallel computation among [10, 19, 42, 53], 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, 53]. 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.

7.2 Basic Graph Theory Facts

Fact 7.1. *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.

Proposition 7.2. *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.*

Proof. $\mathsf{VC}(G) \underset{\text{Fact 7.1}}{\geq} \mathsf{MM}(G) \geq |M| \geq \frac{1}{\alpha} \cdot |V'| \geq \frac{1}{\alpha} \cdot \mathsf{VC}(G) \underset{\text{Fact 7.1}}{\geq} \frac{1}{\alpha} \cdot \mathsf{MM}(G)$. ∎

Proposition 7.3. *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, $\mathsf{MM}(G) \geq \frac{\gamma \cdot \Delta}{2 \cdot (\Delta+1)} \cdot |V_{\text{HIGH}}|$.*

Proof. By Vizing's theorem [73], G can be edge colored by at most $\Delta+1$ colors. As each color class forms a matching, this means that there exists a matching M in G with $|M| \geq \frac{|E|}{\Delta+1}$. Moreover, we have $|E| \geq \frac{1}{2} \cdot |V_{\text{HIGH}}| \cdot \gamma \cdot \Delta$, finalizing the proof. ∎

7.3 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 4 ([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.

In the remainder of the paper, we use the terms "Property (P1)" and "Property (P2)" of EDCS to refer to the first and second items in Definition 4 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. As we are only interested in existence of EDCS in this paper, we provide a simpler and self-contained proof of this fact in Appendix B.2, which also implies a simple polynomial time algorithm for computing any EDCS of a given graph G.

Lemma 7.4. *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 7.5 ([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 7.5 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 7.6. *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,

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

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

8 Warmup: A 3-Approximation Coreset for Matching

A natural randomized coreset for the matching problem was previously proposed by [12]: simply compute a *maximum matching* of each graph $G^{(i)}$. We refer to this randomized coreset as the $\mathsf{MaxMatching}$ coreset. It was shown in [12] that $\mathsf{MaxMatching}$ is an $O(1)$-approximation randomized coreset for the matching problem (the hidden constant in the O-notation was bounded by 9 in [12]). As a warm up, we propose a better analysis of this randomized coreset in this section.

Theorem 8. *Let $G(V,E)$ be a graph with $\mathsf{MM}(G) = \omega(k \log n)$ and $G^{(1)}, \ldots, G^{(k)}$ be a random k-partition of G. Any maximum matching of the graph $G^{(i)}$ is a $(3+o(1))$-approximation randomized composable coreset of size $O(n)$ for the maximum matching problem.*

Assumption on $\mathsf{MM}(G)$. In this section, we follow [12] in assuming that $\mathsf{MM}(G) = \omega(k \log n)$ since otherwise we can immediately obtain a (non-randomized) composable coreset with approximation ratio one (an exact maximum matching) and size $\widetilde{O}(k^2)$ for the matching problem using the results in [26].

A crucial building block in our proof of Theorem 8 is a new concentration result for the size of maximum matching in edge sampled subgraphs that we prove in the next section. This result is quite general and can be of independent interest.

8.1 Concentration of Maximum Matching Size under Edge Sampling

Let $G(V,E)$ be any arbitrary graph and $p \in (0,1)$ be a parameter (possibly depending on size of the graph G). Define $G_p^{\mathsf{E}}(V, E_p)$ as a subgraph of G obtained by sampling each edge in E independently and with probability p, i.e., an edge sampled subgraph of G. We show that $\mathsf{MM}(G_p)$ is concentrated around its expected value.

Lemma 8.1. *Let $G(V,E)$ be any arbitrary graph, $p \in (0,1)$ be a parameter, and $\mathbb{E}\left[\mathsf{MM}(G_p^{\mathsf{E}})\right] \leq \mu$. For any $\lambda > 0$,*

$$\Pr\left(\left|\mathsf{MM}(G_p^{\mathsf{E}}) - \mathbb{E}\left[\mathsf{MM}(G_p^{\mathsf{E}})\right]\right| \geq \lambda\right) \leq 2 \cdot \exp\left(-\frac{\lambda^2 \cdot p}{2 \cdot \mu}\right).$$

Proof. For simplicity, define $G_p := G_p^{\mathsf{E}}$. Let C be any minimum vertex cover in the graph G. We use *vertex exposure* martingales over vertices in C to prove this result. Fix an arbitrary ordering of vertices in C and for any $v \in C$, define $C^{<v}$ as the set of vertices in C that appear before v in this ordering. For each $v \in C$, we define a random variable $Y_v \in \{0,1\}^{V \setminus C^{<v}}$ as a vector of indicators whether a possible edge (i.e., an edge already in G) between the vertices v and $u \in V \setminus C^{<v}$ appears

in G_p or not. Since C is a vertex cover of G, every edge in G is incident on some vertex of C. As a result, the graph G_p is uniquely determined by the vectors $Y_1, \ldots, Y_{|C|}$. Define a sequence of random variables $\{X_i\}_{i=1}^{|C|}$, whereby $X_i = \mathbb{E}\left[\mathsf{MM}(G_p) \mid Y_1, \ldots, Y_i\right]$. The following claim is standard.

Claim 8.2. *The sequence $\{X_i\}_{i=1}^{|C|}$ is a martingale with respect to the sequence $\{Y_i\}_{i=1}^{|C|}$.*

Proof. For any $i \leq |C|$,

$$\begin{aligned}
\mathbb{E}\left[X_i \mid Y_1, \ldots, Y_{i-1}\right] &= \mathop{\mathbb{E}}_{(y_1,\ldots,y_{i-1})}\Big[\mathbb{E}\left[\mathsf{MM}(G_p) \mid Y_1 = y_1, \ldots, Y_{i-1} = y_{i-1}, Y_i\right] \mid Y_1 = y_1, \ldots, Y_{i-1} = y_{i-1}\Big] \\
&= \mathop{\mathbb{E}}_{(y_1,\ldots,y_{i-1})}\Big[\mathsf{MM}(G_p) \mid Y_1 = y_1, \ldots, Y_{i-1} = y_{i-1}\Big] \\
&\qquad\qquad \text{(as we are "averaging out" } Y_i \text{ in the outer expectation)} \\
&= \mathbb{E}\Big[\mathsf{MM}(G_p) \mid Y_1, \ldots, Y_{i-1}\Big] = X_{i-1}. \qquad \blacksquare
\end{aligned}$$

Notice that $X_0 := \mathbb{E}\left[\mathsf{MM}(G_p)\right]$ and $X_{|C|} = \mathsf{MM}(G_p)$ as fixing $Y_1, \ldots, Y_{|C|}$ uniquely determines the graph G_p. Hence, we can use Azuma's inequality to show that value of $X_{|C|}$ is close to X_0 with high probability. To do this, we need a bound on $|C|$, as well as each term $|X_i - X_{i-1}|$. Bounding each $|X_i - X_{i-1}|$ term is quite easy; the set of edges incident on the vertex i can only change the maximum matching in G_p by 1 (as i can only be matched once), and hence $|X_i - X_{i-1}| \leq 1$. In the following, we also bound the value of $|C|$.

Claim 8.3. $|C| \leq 2 \cdot \mu/p$.

Proof. Since size of a minimum vertex cover of a graph G is at most twice the size of its maximum matching (Fact 7.1), we have that $|C| \leq 2 \cdot \mathsf{MM}(G)$. It is also straightforward to verify that $p \cdot \mathsf{MM}(G) \leq \mathbb{E}\Big[\mathsf{MM}(G_p)\Big] \leq \mu$, since p fraction of the edges of any maximum matching of G appear in G_p in expectation; hence $|C| \leq 2 \cdot \mu/p$. $\blacksquare$

We are now ready to finalize the proof. By setting $c_i = 1$ for all $i \leq |C|$, we can use Azuma's inequality (Proposition B.1) with parameters λ and c_i for the martingales $\{X_i\}$, and obtain that,

$$\begin{aligned}
\Pr\Big(\left|\mathsf{MM}(G_p) - \mathbb{E}\left[\mathsf{MM}(G_p)\right]\right| \geq \lambda\Big) = \Pr\Big(\left|X_{|C|} - X_0\right| \geq \lambda\Big) &\underset{\text{Proposition B.1}}{\leq} 2 \cdot \exp\left(-\frac{\lambda^2}{\sum_{i \in C} c_i^2}\right) \\
= 2 \cdot \exp\left(-\frac{\lambda^2}{|C|}\right) &\underset{\text{Claim 8.3}}{\leq} 2 \cdot \exp\left(-\frac{\lambda^2 \cdot p}{2 \cdot \mu}\right)
\end{aligned}$$

finalizing the proof. $\blacksquare$

8.2 Proof of Theorem 8

Let $G(V, E)$ be any arbitrary graph and $G^{(1)}, \ldots, G^{(k)}$ be a random k-partition of G. Recall that MaxMatching coreset simply computes a maximum matching M_i on each graph $G^{(i)}$ for $i \in [k]$; hence, we only need to show that the graph $H(V, M_1 \cup \ldots \cup M_k)$ has a large matching compared to the graph G.

Let $M^\star$ be any fixed maximum matching in G, and let $\mu := |M^\star| = \mathsf{MM}(G)$. Our approach is to show that either each graph $G^{(i)}$ has a large matching already, i.e., $|M_i| \geq \mu/3$, or many edges of $M^\star$ are picked in M_i as well. In the latter case, the union of edges in M_i for $i \in [k]$ has a large intersection with $M^\star$ and hence contains a large matching.

Define $G^-(V, E^-)$ whereby $E^- := E \setminus M^\star$. Let $G_i^- := G^- \cap G^{(i)}$ be the intersection of the graph $G^{(i)}$ and G^-. Finally, define μ_i^- as the *maximum matching size* in G_i^-. Using our concentration result from the previous section, we can show that,

Claim 8.4. *Let $\varepsilon \in (0,1)$ be a parameter. Suppose $\mu \geq 4 \cdot \varepsilon^{-2} \cdot k \log n$; then, there exists an integer $\mu^- \in [n]$ such that with probability $1 - o(1)$ (over the random k-partition), $\mu_i^- = \mu^- \pm \varepsilon \cdot \mu$ simultaneously for all $i \in [k]$.*

Proof. Let $p = 1/k$; the graph G_i^- is a subgraph of G^- obtained by picking each edge in G^- independently and with probability p. Let $\mu^- := \mathbb{E}\left[\mathsf{MM}(G_i^-)\right] \leq \mu$ (notice that the marginal distribution of G_i^- graphs for all $i \in [k]$ are identical). By setting $\lambda = \varepsilon \cdot \mu$ in Lemma 8.1, we have that,

$$\Pr\left(\left|\mu_i^- - \mu^-\right| \geq \lambda\right) \underset{\text{Lemma 8.1}}{\leq} 2 \cdot \exp\left(-\frac{\varepsilon^2 \cdot \mu^2 \cdot p}{2 \cdot \mu}\right) \leq 2 \cdot \exp\left(-2 \log n\right) \leq \frac{1}{n^2}$$

where the second inequality is by the assumption on the value of μ. Taking a union bound over all $k \leq n$ subgraphs G_i^- for $i \in [k]$ finalizes the proof. ∎

In the following, we condition on the event in Claim 8.4. We now have,

Lemma 8.5. *Let μ, μ^- and ε be as in Claim 8.4. If $\mu^- \leq \mu/3$, then $\left|\bigcup_{i=1}^k M_i \cap M^\star\right| \geq \mu/3 - 3\varepsilon \cdot \mu$ w.p. $1 - o(1)$.*

Proof. Fix an index $i \in [k]$ and notice that conditioning on the event in Claim 8.4, only fixes the set of edges in G_i^-. Let M_i^- be any maximum matching in G_i^-; by definition, $\mu_i^- = \left|M_i^-\right|$. By conditioning on the event in Claim 8.4, we have $\mu_i^- \leq \mu^- + 2\varepsilon \cdot \mu$. It is straightforward to verify that there are at least $|M^\star| - 2 \cdot \left|M_i^-\right| = \mu - \mu_i^- \geq \mu/3 - 2\varepsilon \cdot \mu$ edges e in $M^\star$ such that neither of endpoints of e are matched by M_i^-. We refer to these edges as *free* edges and use $M_f^\star \subseteq M^\star$ to denote them.

Note that even after conditioning on G_i^-, the edges in $M^\star$, and consequently $M_f^\star$, appear in the graph $G^{(i)}$ independently and with probability $1/k$. As such, using a Chernoff bound (by assumption on the value of μ), w.p. $1 - 1/n^2$, $\left|M_f^\star\right|/k - \varepsilon \cdot \mu/k$ edges of $M_f^\star$ appear in $G^{(i)}$. Since these edges can be directly added to the matching M_i^- (as neither endpoints of them are matched in M_i^-), this implies that there exists a matching of size $\mu_i^- + \frac{1}{k} \cdot (\mu/3 - 3\varepsilon \cdot \mu)$ in $G^{(i)}$ w.p. $1 - 1/n^2$.

Now let M_i be the maximum matching computed by $\mathsf{MaxMatching}$; the above argument implies that $|M_i| \geq \mu_i^- + \frac{1}{k} \cdot (\mu/3 - 3\varepsilon \cdot \mu)$. On the other hand, notice that $\left|M_i \cap G_i^-\right| \leq \mu_i^-$ as $M_i \cap G_i^-$ forms a matching in the graph G_i^- and μ_i^- denotes the maximum matching size in this graph. This means that $|M_i \cap M^\star| = \left|M_i \setminus G_i^-\right| \geq \frac{1}{k} \cdot (\mu/3 - 3\varepsilon \cdot \mu)$. To finalize the proof, notice that by a union bound over all k matchings M_i, we have that with probability $1 - 1/n$,

$$\left|\bigcup_{i=1}^k M_i \cap M^\star\right| = \sum_{i=1}^k |M_i \cap M^\star| \geq k \cdot \frac{1}{k} \cdot (\mu/3 - 3\varepsilon \cdot \mu) = \mu/3 - 3\varepsilon \cdot \mu. \qquad \blacksquare$$

We can now easily prove Theorem 8.

Proof of Theorem 8. By our assumption that $\mu = \mathsf{MM}(G) = \omega(k \log n)$, we can take ε in Claim 8.4 and Lemma 8.5 to be some arbitrary small constant, say $\varepsilon = o(1)$. Define μ^- as in Lemma 8.5. If $\mu^- > \mu/3$, we are already done as by Claim 8.4, for any $i \in [k]$, $|M_i| \geq \mu^- - o(\mu) \geq \mu/3 - o(\mu)$ and hence the union of matchings $M_1, \ldots, M_k$ surely has a $(3 + o(1))$ approximate matching. On the other hand, if $\mu^- \leq \mu/3$, we can apply Lemma 8.5, and argue that $\mu/3 - o(\mu)$ edges of the matching $M^\star$ appear in the union of matchings $M_1, \ldots, M_k$, which finalizes the proof. ∎

8.3 Lower Bound on Approximation Ratio of MaxMatching

We also show that there exists a graph for which the approximation ratio of MaxMatching is arbitrarily close to 2. This implies that we cannot improve the analysis of MaxMatching much further and in particular beat the approximation ratio of 2.

Lemma 8.6. *There exists a graph $G(V,E)$ such that for any random k-partition of G ($k \leq n^{1-\delta}$ for any constant $\delta > 0$), the* MaxMatching *coreset can only find a matching of size at most $\left(\frac{1}{2}+\frac{1}{k}\right)\cdot \mathsf{MM}(G)$ with high probability.*

Proof. The vertex set of the graph G consists of four sets of vertices L_1, L_2, R_1, R_2 with $|L_1| = \frac{n}{2}+\frac{n}{k}$ and $|L_2| = |R_1| = |R_2| = \frac{n}{2}$. G is a bipartite graph with $L_1 \cup L_2$ on one side of the bipartition and $R_1 \cup R_2$ on the other side. There is a complete bipartite graph between L_1 and R_2, a perfect matching between L_2 and R_2 and a matching of size $|R_1|$ between L_1 and R_1.

It is easy to verify that there exists a matching of size $|R_1|+|R_2| = n$ in G and hence $\mathsf{MM}(G) \geq n$. Suppose we create a random k-partition $G^{(1)}, \ldots, G^{(k)}$ of G and each machine $i \in [k]$ computes an arbitrary maximum matching M_i of its input graph (i.e., compute the MaxMatching coreset). In the following, we argue that the maximum matching in the graph $H(V, M_1 \cup \ldots, M_k)$ is of size $(\frac{1}{2}+\frac{1}{k})\cdot n$ with high probability, which concludes the proof.

To prove the lemma, we need the following simple claim about the maximum matching in the edge sampled subgraphs of G.

Claim 8.7. *Suppose $G_p^{\mathsf{E}}(V,E_p)$ is an edge sampled subgraph of G with probability $p = 1/k$; then, w.p. $1-1/n^5$, there exists a matching M_p in G such that:*

1. *M_p is a maximum matching in G_p, i.e., $|M_p| = \mathsf{MM}(G)$.*
2. *No edges between L_2 and R_2 belong to M_p.*

Proof. A simple application of Chernoff bound ensures that the total number of edges between L_1 and R_1 in G_p is at most $2p \cdot n = n/k$ with probability at least $1-1/n^{10}$. In the following, we condition on this event. Define $M_{1,1}$ as the matching consisting of the edges between L_1 and R_1 in G_p and let $L_1^- := L_1 \setminus L_1(M_{1,1})$ be the set of vertices in L_1 that are *not* incident on $M_{1,1}$.

Consider the graph between L_1^- and R_2. Note that since $|M_{1,1}| \leq n/k$, we have $\left|L_1^-\right| \geq |R_2|$. By the independence in the sampling of edges and the fact that in G, L_1^- and R_1 forms a bipartite clique, the set of edges between L_1^- and R_2 in G_p form a random bipartite graph with probability of having each edge equal to $p = 1/k = \omega(\log n)$. Using standard facts about random graphs (see, e.g., [24], Chapter 7), this implies that there exists a matching of size $|R_2|$ between L_1^- and R_2 in G_p with probability $1-1/n^{10}$. Let M_p be the union of this matching and $M_{1,1}$.

It is clear that M_p does not have any edges between L_2 and R_2. To see that M_p is indeed a maximum matching of G_p, notice that all vertices in $R_1 \cup R_2$ in G_p that have non-zero degree are matched by M_p and hence there cannot be any larger matching in G. ∎

We are now ready to finalize the proof of Lemma 8.6. Recall that each graph $G^{(i)}$ is an edge sampled subgraph of G with probability $1/k$. We can apply Claim 8.7 to each graph $G^{(i)}$ and by a union bound, w.p. $1-1/n^4$, there exists a suitable maximum matching $M_p^{(i)}$ in each graph $G^{(i)}$. Since we are choosing an *arbitrary* maximum matching of $G^{(i)}$ as its coreset, we can assume that $M_p^{(i)}$ would be chosen from each graph $G^{(i)}$, i.e., $M_i = M_p^{(i)}$ for all $i \in [k]$. This implies that no edge incident to vertices in L_2 are chosen among all coresets $M_1 \cup \ldots \cup M_k$. As a result, the maximum matching in the graph $H(V, M_1 \cup \ldots \cup M_k)$ can have size at most $|L_1| = \left(\frac{1}{2}+\frac{1}{k}\right)\cdot n$, finalizing the proof as $\mathsf{MM}(G) = n$. ∎

9 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 maximum matchings or 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 11 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 7.5 at all).

9.1 Degree Distribution Lemma

Lemma 9.1 (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* EDCS(G, β, β^-)*, 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 9.1. 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 9.2. *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 9.3. *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 vertices in 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 9.3, 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 9.3 (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,

$$\begin{aligned}
\left|E_{B \setminus A}(T, T' \setminus S)\right| &= \left|E_{B \setminus A}(T)\right| - \left|E_{B \setminus A}(T, S)\right| \\
&\text{(as all the edges in } B \setminus A \text{ that are incident on } T \text{ are going to } T') \\
&\geq \left|E_{A \setminus B}(T)\right| - \left|E_{B \setminus A}(T, S)\right| \\
&\text{(as by Claim 9.3, the degree of vertices in } T \text{ is larger in } B \setminus A \text{ compared to } A \setminus B) \\
&\geq \left|E_{A \setminus B}(S)\right| - \left|E_{B \setminus A}(S)\right| \\
&\text{(as all edges in } A \setminus B \text{ incident on } S \text{ are also incident on } T) \\
&\geq |S| \cdot D \\
&\text{(by the assumption on the degree of vertices in } S \text{ in subgraphs } A \text{ and } B)
\end{aligned}$$

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 of Lemma 9.1. 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 9.2 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 9.2. As a result, we have,

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

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. ∎

9.2 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 3 on randomized coresets (see Section 10).

Lemma 9.4 (EDCS in Edge Sampled Subgraphs)**.** *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 4 for parallel algorithms (see Section 11). 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 9.5 (EDCS in Vertex Sampled Subgraphs)**.** *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 9.1 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.

9.2.1 EDCS in Edge Sampled Subgraphs: Proof of Lemma 9.4

Proof of Lemma 9.4. We first prove that edge sampling an EDCS results in another EDCS for the sampled subgraph.

Claim 9.6. *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 vertex $v \in V$, $\mathbb{E}\left[d_{H_p}(v)\right] = p \cdot d_H(v)$ 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 (Proposition B.2), we have,

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

where the second inequality is by the lower bound on β in Lemma 9.5 statement. In the following, we condition on the event that:

$$\forall v \in V \quad \left|d_{H_p}(v) - p \cdot d_H(v)\right| \leq \frac{\lambda}{5} \cdot \beta. \tag{1}$$

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,

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

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,

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

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 9.4. Let H be an EDCS(G, β_H, β_H^-) for the parameters β_H, β_H^- in Claim 9.6. The existence of H follows from Lemma 7.4 as $\beta_H^- < \beta_H$. Define $\widehat{H}_1 := H \cap G_1$ and $\widehat{H}_2 := H \cap G_2$. By Claim 9.6, $\widehat{H}_1$ (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 9.1 (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 9.7 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,

$$\begin{aligned} |d_{H_1}(v) - d_{H_2}(v)| &\leq \left|d_{H_1}(v) - d_{\widehat{H}_1}(v)\right| + \left|d_{\widehat{H}_1}(v) - d_{\widehat{H}_2}(v)\right| + \left|d_{\widehat{H}_2}(v) - d_{H_2}(v)\right| \\ &\underset{\text{Lemma 9.1}}{\leq} O(\log n) \cdot \lambda^{1/2} \cdot \beta + \left|d_{\widehat{H}_1}(v) - p \cdot d_H(v)\right| + \left|d_{\widehat{H}_2}(v) - p \cdot d_H(v)\right| \\ &\underset{\text{Claim 9.6}}{\leq} O(\log n) \cdot \lambda^{1/2} \cdot \beta + O(1) \cdot \lambda \cdot \beta, \end{aligned}$$

finalizing the proof. ∎

9.2.2 EDCS in Vertex Sampled Subgraphs: Proof of Lemma 9.5

Proof of Lemma 9.5. We first prove that vertex sampling an EDCS results in another EDCS for the sampled subgraph.

Claim 9.7. *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 vertex $v \in V_p$, $\mathbb{E}\left[d_{H_p}(v)\right] = p \cdot d_H(v)$ 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 (Proposition B.2), we have,

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

where the second inequality is by the lower bound on β in Lemma 9.5 statement. In the following, we condition on the event that:

$$\forall v \in V_p \quad \left|d_{H_p}(v) - p \cdot d_H(v)\right| \leq \frac{\lambda}{5} \cdot \beta. \tag{2}$$

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,

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

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,

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

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 9.5. Let H be an EDCS(G, β_H, β_H^-) for the parameters β_H, β_H^- in Claim 9.7. The existence of H follows from Lemma 7.4 as $\beta_H^- < \beta_H$. Define $\widehat{H}_1 := H \cap G_1$ and $\widehat{H}_2 := H \cap G_2$. By Claim 9.7, $\widehat{H}_1$ (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 9.1 (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 9.7 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,

$$\begin{aligned}|d_{H_1}(v) - d_{H_2}(v)| &\leq \left|d_{H_1}(v) - d_{\widehat{H}_1}(v)\right| + \left|d_{\widehat{H}_1}(v) - d_{\widehat{H}_2}(v)\right| + \left|d_{\widehat{H}_2}(v) - d_{H_2}(v)\right| \\ &\underset{\text{Lemma 9.1}}{\leq} O(\log n) \cdot \lambda^{1/2} \cdot \beta + \left|d_{\widehat{H}_1}(v) - p \cdot d_H(v)\right| + \left|d_{\widehat{H}_2}(v) - p \cdot d_H(v)\right| \\ &\underset{\text{Claim 9.7}}{\leq} O(\log n) \cdot \lambda^{1/2} \cdot \beta + O(1) \cdot \lambda \cdot \beta,\end{aligned}$$

finalizing the proof. ∎

10 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 7.5 and 7.6.

10.1 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
$$\lambda = \Theta\left((\frac{\varepsilon}{\log n})^2\right) \qquad \text{and} \qquad \beta := \Theta(\lambda^{-3} \cdot \log n).$$
3. Let $C := \bigcup_{i=1}^{k} C^{(i)}$.

Lemma 10.1. *With probability $1 - 4/n^7$, the subgraph C is an $\text{EDCS}(G, \beta_C, \beta_C^-)$ for parameters:*

$$\lambda_C := O(\log n) \cdot \lambda^{1/2} \quad , \quad \beta_C := (1+\lambda_C)\cdot k \cdot \beta \quad \text{and} \quad \beta_C^- := (1-2\lambda_C)\cdot k \cdot \beta.$$

Proof. Recall that each graph $G^{(i)}$ is an edge sampled subgraph of G with sampling probability $p = \frac{1}{k}$. By Lemma 9.4 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 $v \in V$:

$$|d_{C^{(i)}}(v) - d_{C^{(j)}}(v)| \leq O(\log n) \cdot \lambda^{1/2} \cdot \beta = \lambda_C \cdot \beta. \tag{3}$$

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 $1 - 4/n^7$. In the following, 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,

$$\begin{aligned}d_C(u) + d_C(v) &= \sum_{i=1}^{k} d_{C^{(i)}}(u) + \sum_{i=1}^{k} d_{C^{(i)}}(v) \underset{\text{Eq (3)}}{\leq} k \cdot (d_{C^{(j)}}(u) + d_{C^{(j)}}(v)) + k \cdot \lambda_C \cdot \beta \\ &\leq k \cdot \beta + k \cdot \lambda_C \beta = \beta_C. \qquad \text{(by Property (P1) of EDCS } C^{(j)} \text{ with parameter } \beta)\end{aligned}$$

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,

$$\begin{aligned} d_C(u) + d_C(v) &= \sum_{i=1}^{k} d_{C^{(i)}}(u) + \sum_{i=1}^{k} d_{C^{(i)}}(v) \underset{\text{Eq (3)}}{\geq} k \cdot (d_{C^{(j)}}(u) + d_{C^{(j)}}(v)) - k \cdot \lambda_C \cdot \beta \\ &\geq k \cdot (1-\lambda) \cdot \beta - k \cdot \lambda_C \cdot \beta \geq (1 - 2\lambda_C) \cdot k \cdot \beta = \beta_C^-. \\ &\qquad \text{(by Property (P2) of EDCS } C^{(j)} \text{ with parameter } (1-\lambda)\cdot\beta) \end{aligned}$$

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

10.2 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 3.

Theorem 9. *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 10.1, 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 7.5, 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 each 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 more details.

Theorem 10. *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 $\beta := \Theta(\varepsilon^{-6} \cdot \log^7 n)$ 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 10.1, the union of the coresets, i.e., the k EDCSs, is itself an $\text{EDCS}(G, \beta_C, \beta_C^-)$ 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 7.6, 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 9.4, 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 7.6, with a slightly smaller parameter ε to argue that the returned solution is still a $(2+\varepsilon)$-approximation. ∎

Remark 10.2. *In the proof of Theorem 10, we neglected the time necessary to compute a vertex cover in the union of the coresets (as is consistent with the definition of randomized coresets). In case we require this algorithm to run in polynomial time, we need to approximate the final vertex cover in the union of the coresets as opposed to recover it exactly. In particular, by picking a 2-approximation vertex cover in the union of coreset in the proof of Lemma 7.6, we obtain an (almost) 4-approximation to the vertex cover of G.*

11 MPC Algorithms for Matching and Vertex Cover

In this section, we show that a careful adaptation of our coresets construction together with the structural results proven for EDCS in Section 9 can be used to obtain MPC algorithms with much smaller memory while increasing the number of required rounds to only $O(\log\log n)$.

Theorem 11. *There exists an MPC algorithm that given a graph $G(V,E)$ with high probability computes an $O(1)$ approximation to both maximum matching and minimum vertex cover of G in $O(\log\log n + \log\left(\frac{n}{s}\right))$ MPC rounds on machines of memory $s = n^{\Omega(1)}$.*

By setting $s = O(n/\text{polylog}(n))$ in Theorem 11, 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 4.

In the following, for the sake of clarity, we mostly focus on proving Theorem 11 for the natural case when memory per machine is $s = \widetilde{O}(n)$, and postpone the proof for all range of parameter s to Section 11.5. 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 6), 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 7.5 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 7.6, 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 7.2).

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 11.

11.1 A Parallel Algorithm for EDCS

We now present our parallel algorithm for computing an EDCS via vertex sampling. In this algorithm, the edges of the input graph as well as the output EDCS will be partitioned across multiple machines. 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 which we explain in Remark 11.1. To avoid repeating the arguments, we present our algorithm for all range of memory $s = n^{\Omega(1)}$, but encourage the reader to consider the case of $s = n$ 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. Notice that indices in $I(v)$ are κ-wise independent random variables. Additionally, it is easy to see that each graph $G^{(i)}$ is a vertex sampled subgraph of G with sampling probability p.

Remark 11.1. As opposed to the previous vertex sampling approach of Czumaj et al. [29] that resulted in a partitioning of vertices of G across different subgraphs, our way of sampling subgraphs in ParallelEDCS results in each vertex appearing in $\Theta(p \cdot k)$ different subgraphs with high probability. This is necessary for our algorithm as we need to ensure that every edge of the input graph is sampled in this process. However, this property introduces new challenges in the MPC implementation of our algorithm as a naive implementation of this idea requires communicating $\Theta(p \cdot k)$ messages per each edge of the graph which cannot be done within the memory restrictions of the MPC model. This is the main reason that we sample these subgraphs in ParallelEDCS with limited independence as opposed to truly independently to reduce the communication necessary per each edge to $O(\log n)$.

We first prove some simple properties of ParallelEDCS.

Proposition 11.2. *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 $G^{(i)}$.*

Proof. Fix any vertex $v \in V$. Clearly, $\mathbb{E}\,|I(v)| = p \cdot k$. Moreover, $|I(v)|$ is sum of zero-one κ-wise independent random variables and hence by Chernoff bound with bounded independence (Proposition B.3)

$$\Pr\left(||I(v)| - \mathbb{E}\,|I(v)|| \geq \lambda \cdot \mathbb{E}\,|I(v)|\right) \leq 2 \cdot \exp\left(-\frac{\kappa}{2}\right) \leq 2 \cdot \exp\left(-10 \log n\right) \leq 1/n^{10}.$$

Note that $\mathbb{E}\,|I(v)| = p \cdot k = \Theta\left(\sqrt{\frac{n \cdot \Delta}{s}}\right)$, $\lambda = (2 \cdot \log n)^{-3}$ and hence by the bound on Δ, we have $\lambda^2 \cdot \mathbb{E}\,|I(v)|/2 = \Theta(\log n) = \kappa$, and hence we can indeed apply Proposition B.3 here. By a union bound on all n vertices, the first part holds w.p. $\geq 1 - 1/n^9$.

We now prove the second part. Fix an edge $e \in E$ and define the indicator random variables $X_1, \ldots, X_k$ where $X_i = 1$ iff e is contained in the graph $G^{(i)}$. Define $X := \sum_{i=1}^{k} X_i$ to denote the number of graphs the edge e belongs to. Clearly, $\mathbb{E}\,[X_i] = p^2$ for all $i \in [k]$ and hence $\mathbb{E}\,[X] = p^2 \cdot k$. Moreover, the random variables X_i's are κ-wise independent for $\kappa = 20 \log n$. Hence, by Chernoff bound with bounded independence (Proposition B.3), the probability that e belongs to no graph $G^{(i)}$, i.e., $X = 0$ is at most,

$$\Pr\left(X = 0\right) \leq \Pr\left(\,|X - \mathbb{E}\,[X]| \geq \mathbb{E}\,[X]\,\right) \leq 2 \cdot \exp\left(-\frac{\kappa}{2}\right) \leq 2 \cdot \exp\left(-10 \log n\right) \leq 1/n^{10}.$$

Again, note that $\mathbb{E}\,|X| = p^2 \cdot k = 800 \log n = 2\kappa$ and hence we could apply Proposition B.3. By a union bound on all $O(n^2)$ edges, the second part also holds w.p. at least $1 - 1/n^8$. Another union bound on this event and the event in the first part finalizes the proof. ∎

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 10.1 with some additional care to handle the difference between vertex sampled subgraphs and edge sampled ones.

Lemma 11.3. *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 $\text{EDCS}(G, \beta_C, \beta_C^-)$ for parameters:*

$$\lambda_C := \lambda^{1/2} \cdot \Theta(\log n) = o(1), \qquad \beta_C := p \cdot k \cdot (1 + \lambda_C) \cdot \beta, \quad \text{and} \quad \beta_C^- = p \cdot k \cdot (1 - \lambda_C) \cdot \beta.$$

Proof. Recall that each graph $G^{(i)}$ is a vertex sampled subgraph of G with sampling probability p. Hence, by Lemma 9.5 and a union bound, with probability $1 - 4/n^7$, for any two subgraphs $C^{(i)}$ and $C^{(j)}$ for $i, j \in [k]$, and any vertex $v \in V^{(i)} \cap V^{(j)}$, we have,

$$|d_{C^{(i)}}(v) - d_{C^{(j)}}(v)| \leq O(\log n) \cdot \lambda^{1/2} \cdot \beta. \tag{4}$$

In the following, we condition on the events in Eq (4) and Proposition 11.2 which happen together with probability at least $1 - 5/n^7$.

We now prove that C is indeed an $\text{EDCS}(G, \beta_C, \beta_C^-)$ for the given parameters. 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,

$$\begin{aligned}
d_C(u) + d_C(v) &= \sum_{i \in I(u)} d_{C^{(i)}}(u) + \sum_{i \in I(v)} d_{C^{(i)}}(v) \\
&\underset{\text{Eq (4)}}{\leq} |I(u)| \cdot d_{C^{(j)}}(u) + |I(v)| \cdot d_{C^{(j)}}(v) + (|I(u)| + |I(v)|) \cdot O(\log n) \cdot \lambda^{1/2} \cdot \beta \\
&\underset{\text{Proposition 11.2}}{\leq} p \cdot k \cdot \beta + O(\lambda) \cdot p \cdot k \cdot \beta + p \cdot k \cdot O(\log n) \cdot \lambda^{1/2} \cdot \beta \\
&\qquad \text{(by Property (P1) of EDCS } C^{(j)} \text{ with parameter } \beta)
\end{aligned}$$

$$\leq p \cdot k \cdot (1+\lambda_C) \cdot \beta = \beta_C.$$

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 $(u,v) \in G^{(j)} \setminus C^{(j)}$ (the existence of j follows from conditioning on the event in Proposition 11.2). We have,

$$\begin{aligned}
d_C(u) + d_C(v) &= \sum_{i \in I(u)} d_{C^{(i)}}(u) + \sum_{i \in I(v)} d_{C^{(i)}}(v) \\
&\underset{\text{Eq (4)}}{\geq} |I(u)| \cdot d_{C^{(j)}}(u) + |I(v)| \cdot d_{C^{(j)}}(v) - (|I(u)| + |I(v)|) \cdot O(\log n) \cdot \lambda^{1/2} \cdot \beta \\
&\underset{\text{Proposition 11.2}}{\geq} p \cdot k \cdot (1-\lambda) \cdot \beta - O(\lambda) \cdot p \cdot k \cdot \beta - p \cdot k \cdot O(\log n) \cdot \lambda^{1/2} \cdot \beta \\
&\qquad \text{(by Property (P2) of EDCS } C^{(j)} \text{ with parameter } \beta) \\
&\geq p \cdot k \cdot (1-\lambda_C) \cdot \beta = \beta_C^-.
\end{aligned}$$

Hence, C also satisfies Property (P2) of EDCS for parameter β_C^-, finalizing the proof. ∎

Before moving on, we prove a simple claim that ensures 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 11.4. *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 $\mathbb{E}\left[d_{G^{(i)}}(v)\right] = p \cdot d_G(v) \leq p \cdot \Delta = \Theta(\sqrt{\frac{s \cdot \Delta}{n}} \cdot \log n)$. 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 $1 - 1/n^{18}$. ∎

11.2 Random Match Algorithm

In our main algorithm, we need a subroutine for finding a large matching incident on the set of "high" degree vertices of a given graph G which its edges are initially partitioned across many machines. In this section, we provide such an algorithm based on a simple randomized procedure that is easily implementable in constant number of MPC rounds.

RandomMatch(G, S, Δ). 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. Formally,

Lemma 11.5. *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 $G \setminus S'$, which happens with high probability.

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). Hence,

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

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

11.3 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 Section 11.5, 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 11.6 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 $M_{\text{ALG}} := M_{\text{HIGH}} \cup M_{\text{REC}}$.

We start by proving some simple properties of ParallelAlgorithm. The following claim is a direct corollary of Lemma 11.3 by setting $s = n$.

Claim 11.6. *The subgraph* $C := \mathsf{ParallelEDCS}(G, \Delta, n)$ *computed in* $\mathsf{ParallelAlgorithm}(G, \Delta)$ *is an* $\text{EDCS}(G, \beta_C, \beta_C^-)$ *for parameters:*

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

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

Similarly, the following claim follows easily from Lemma 11.5.

Claim 11.7. *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* $\mathbb{E}\,|M_{\text{HIGH}}| = \Omega(|V_{\text{HIGH}}|)$.

Proof. With this conditioning, the maximum degree of G is at most β_C, while the degree of vertices V_{HIGH} is at least $\beta_C^-/2 \geq \beta_C/3$. Hence, we can apply Lemma 11.5 and prove the statement. ∎

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 in ParallelAlgorithm as follows.

Claim 11.8. *The total number of steps made by* $\mathsf{ParallelAlgorithm}(G, \Delta)$ *is* $T = O(\log\log\Delta)$.

Proof. Define a function $F(\Delta)$ denoting the number of recursive calls made by ParallelAlgorithm with maximum degree Δ. As $\mathsf{ParallelAlgorithm}(G, \Delta)$ runs $\mathsf{ParallelAlgorithm}(C^-, \beta_C)$ for $\beta_C < \Delta^{2/3}$, we have, $F(\Delta) \leq F(\Delta^{2/3}) + 1$ for $\Delta > \left(400 \cdot \log^{12} n\right)$ and $F(\Delta) = 1$ otherwise. It is now easy to see that $F(\Delta) = O(\log\log\Delta)$, finalizing the proof. ∎

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 11.6 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 11.6 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 11.8, $\mathcal{E}_{\mathsf{valid}}$ happens with probability at least $1 - 1/n^4$.

We are now ready to prove the correctness of ParallelAlgorithm.

Lemma 11.9. *For any graph* $G(V, E)$, $\mathsf{ParallelAlgorithm}(G, n)$ *with constant probability outputs a matching* M_{ALG} *which is an* $O(1)$*-approximation to the maximum matching of* G *and a vertex cover* V_{ALG} *which is an* $O(1)$*-approximation to the minimum vertex cover 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 analysis for the base case in step 1 where a maximal matching is compute 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 7.2 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 11.7. 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 11.9, 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.

11.4 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 $\widetilde{O}(n)$. In Section 11.5, we show how to extend this to the case when memory per machine is a given parameter s. 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 [42] as specified in [29]. On machines with memory $n^{\Omega(1)}$, the sort operation in [42] allows us to sort a set of key-value pairs of size 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 [29] 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 [29]. The following lemma states the main properties of our implementation.

Lemma 11.10. *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 [29], 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 polylog(n) bits for representation (see, e.g. [64]) 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 11.4, 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 [29]). 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 11.8, this results in an MPC algorithm with $O(\log \log n)$ rounds.

11.4.1 Optimizing the Number of Machines

We conclude this section by making a remark about optimizing the number of machines (in addition to their memory) in our results as well.

As it is, the total number of machines needed to implement ParallelAlgorithm in the MPC model is $\widetilde{O}(n)$. This means that the total memory across all machines is $\widetilde{O}(n^2)$, which is proportional to the input size (up to polylog(n) factors) whenever the input graph is completely dense, i.e., has $O(n^2)$ edges. However for sparser graphs with $n^{1+\Omega(1)}$, this can be larger than the input size by a factor of $n^{1-\Omega(1)}$. This is consistent with some definitions of MapReduce-style computation such as [53, 56] but not with the strictest definitions in [10, 19], which require that the total memory of the system for a graph with m edges to be only $\widetilde{O}(m)$, i.e., proportional to the input size.

Nevertheless, a straightforward modification of our algorithm can reduce the number of machines down to $\widetilde{O}(m/n)$ which ensures that the total memory used by our algorithm is $\widetilde{O}(m)$ which adheres to the strictest restrictions of the MPC model. The only change we need to do is to work with the *average degree* of the graph in ParallelAlgorithm as opposed to its maximum degree. Concretely, in each call to ParallelAlgorithm(G, Δ), instead of computing ParallelEDCS(G, Δ, n), we compute ParallelEDCS$(G, \widetilde{\Delta}, n)$, where $\widetilde{\Delta} := m/n$ denotes the average degree of the graph G. It is easy to see that in this case, we still only need $\widetilde{O}(n)$ memory per machine (essentially the same argument in Claim 11.4 proves this), but now the total number of machines needed is only $\widetilde{O}(m/n)$ and hence we only need $\widetilde{O}(m)$ memory in total. It is easy to verify that the arguments in the proof of correctness

of ParallelEDCS can be immediately applied to this version; we omit the details.

11.5 Extension to Smaller Memory Requirement and Proof of Theorem 11

As was shown by Lemma 11.10, ParallelAlgorithm can be implemented in the MPC model with machines of memory $\widetilde{O}(n)$ and $O(\log\log n)$ MPC rounds. Combining this with Lemma 11.9 on the correctness of ParallelAlgorithm, we immediately obtain Theorem 11 for the case of $s = \widetilde{O}(n)$, i.e., an MPC algorithm with $O(1)$ approximation to both matching and vertex cover in $O(\log\log n)$ rounds with $\widetilde{O}(n)$ memory per machine.

We now show how to extend our algorithm to the case when memory per machine is $s = n^{\Omega(1)}$. For simplicity, we assume the memory per each machine is $\widetilde{O}(s)$ as opposed to $O(s)$; rescaling the parameter s with a polylog(n) factor implies the final result. Recall that there were only two places in ParallelAlgorithm that we needed $\widetilde{O}(n)$ memory per machine: in subroutine ParallelEDCS and in step 1 of the algorithm, i.e., the base case of recursion. Consequently, we only need to make the following two changes to ParallelAlgorithm(G, Δ):

1. Firstly, in ParallelAlgorithm(G, Δ), we now run the subroutine ParallelEDCS with memory per machine equal to $O(s \cdot \text{polylog}(n))$, i.e., we run ParallelEDCS(G, Δ, s).
2. Second, we change the base case of the algorithm. Whenever $\Delta < \left(\frac{n}{s}\right) \cdot \left(400 \cdot \log^{12}(n)\right)$, instead of sending all edges to a single machine and solve the problem locally, we simply use any standard $O(\log\Delta)$-round MPC algorithm for $O(1)$-approximation to matching and vertex cover that works on machines with memory $n^{\Omega(1)}$ (for example by directly simulating the distributed peeling algorithm of [68] (see also [69]). For more details, see [29] (Lemma 6.1).

Previously with machines of memory $\widetilde{O}(n)$, the maximum degree of underlying graph in each step of ParallelAlgorithm was (see Claim 11.6):

$$\Delta_1 := n, \quad \Delta_2 := \sqrt{n} \cdot \text{polylog}(n), \quad \ldots \quad \Delta_i := n^{1/2^i} \cdot \text{polylog}(n), \quad \ldots \quad \Delta_T := \text{polylog}(n),$$

where $T = O(\log\log n)$ denotes the number of steps in ParallelAlgorithm. By switching to machines of memory $\widetilde{O}(s)$, we instead have,

$$\Delta_1 := n, \quad \Delta_2 := \frac{n}{s^{1/2}} \cdot \text{polylog}(n), \quad \ldots \quad \Delta_i := \frac{n}{s^{1-1/2^{i-1}}} \cdot \text{polylog}(n), \quad \ldots \quad \Delta_T := \left(\frac{n}{s}\right) \cdot \text{polylog}(n),$$

before we reach the stopping condition of ParallelAlgorithm (this follows directly from Lemma 11.3 exactly as in Claim 11.6). After this step, we simply compute an $O(1)$-approximate matching and vertex cover directly in the remaining graph.

The analysis of the correctness of this algorithm is exactly as before. It is also straightforward to verify that this algorithm now only needs machines of memory $O(s \cdot \log^2(n))$ by choice of ParallelEDCS. Finally, the number of rounds needed by this algorithm is $O(\log\log n)$ (for reducing the maximum degree in the graph to Δ_T) plus $O(\log\Delta_T) = O(\log\left(\frac{n}{s}\right) + \log\log n)$ (for running the distributed matching and vertex cover algorithm directly when maximum degree is at most Δ_T). This concludes the proof of Theorem 11 for all range of parameter $s = n^{\Omega(1)}$.

11.6 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 12. *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 13. *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 11; however, for simplicity, we only consider the most interesting case of $s = O(n/\text{polylog}(n))$. We prove each of the above corollaries in the next two sections.

11.6.1 Proof of Corollary 12

The idea is to simply run our MPC algorithm in Theorem 11, 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 11. 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,

$$
\begin{aligned}
\mathsf{MM}(G) - 2|M_T| &\leq \left(1 - \frac{2}{\alpha}\right) \cdot \mathsf{MM}(G) - 2 \cdot \left(1 - \frac{2}{\alpha}\right) \cdot |M_{T-1}| \\
&= \left(1 - \frac{2}{\alpha}\right) \cdot (\mathsf{MM}(G) - 2 \cdot |M_{T-1}|) \\
&\leq \left(1 - \frac{2}{\alpha}\right)^2 (\mathsf{MM}(G) - 2 \cdot |M_{T-2}|) \\
&\qquad\qquad \text{(by applying the second equation to } M_{T-1}) \\
&\leq \left(1 - \frac{2}{\alpha}\right)^T \cdot \mathsf{MM}(G) \qquad \text{(by recursively applying the previous equation)} \\
&\leq \exp\left(-\frac{2}{\alpha} \cdot \alpha \cdot \log(1/\varepsilon)\right) \cdot \mathsf{MM}(G) \leq \varepsilon \cdot \mathsf{MM}(G).
\end{aligned}
$$

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. Note that as the probability of error in the algorithm in Theorem 11 is at most $1/n^4$, by a union bound, the new algorithm also outputs the correct answer with probability at least $1 - 1/n^3$.

11.6.2 Proof of Corollary 13

Corollary 13 can be proven using Theorem 11 plus a simple adaption of the multi-pass streaming algorithm of McGregor [58] for maximum matching to the MPC model. The high level approach in [58] 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 repeating this streaming algorithm in multiple passes over the stream, one can fully implement the reduction and 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 11. In the following, we briefly describe the approach in [58] and point out the modifications needed to make it work in Corollary 13. The purpose of this section is only to convince the reader that the reduction [58] can be seamlessly implemented in the MPC model and hence we do not delve into the full details of the algorithm and analysis and instead refer the reader to [58] for more details and formal proofs.

The Streaming Algorithm of [58]. The idea behind the algorithm is to start with some maximal matching M (which is easy to compute in one pass over the stream) and then *augment* this matching further over multiple *phases* by finding a large set of vertex disjoint augmenting paths of length $O(1/\varepsilon)$ in a *layered* graph created from G and the current matching M. We first introduce the concept of the layered graph that is used to reduce the task of finding augmenting paths to multiple instances of approximate matching. Given a graph G, a matching M, and an odd integer k (which is the target length of the augmenting paths to be found), we create a graph $\mathcal{L}(G, M, k)$ using the following randomized procedure:

1. The vertices in $\mathcal{L}$ are the same as G and are partitioned into $k+1$ subsets $L_1, \ldots, L_{k+1}$ called *layers*. The layer of each vertex v is determined as follows:
   (a) For any vertex u left unmatched by M, u is assigned to either L_1 or L_{k+1} chosen uniformly at random.
   (b) For any matched edge $(u, v) \in M$, the vertex u is assigned to a uniformly at random chosen even layer $L_2, L_4, \ldots, L_{k-1}$ and v is assigned to the subsequent layer.
2. For any edge $(u, v) \in E \setminus M$, if u belongs to an odd layer and v belongs to the next (even) layer then (u, v) is added to $\mathcal{L}$ as well.
3. For any edge $(u, v) \in M$, (u, v) is added to $\mathcal{L}$ as well.

One could easily verify that the edges in $\mathcal{L}$ are only between two consecutive layers and for any edge in $\mathcal{L}$ there is a unique edge in G. The main property of the above construction is that a collection of vertex disjoint paths between vertices in L_1 and L_{k+1} corresponds to a set of vertex disjoint augmenting paths of length k for M in G. It is also relatively easy to prove that as long as $|M| < (1-\varepsilon) \cdot \mathsf{MM}(G)$, then, there exists some $k = O(1/\varepsilon)$, for which the corresponding graph $\mathcal{L}(G, M, k)$ has at least $\mathsf{MM}(G) \cdot \varepsilon^{O(1/\varepsilon)}$ vertex disjoint paths between L_1 and L_{k+1} with high probability (see Theorem 1 in [58]).

We now describe how to find a large fraction of these augmenting paths in $\mathcal{L}$ (in each phase) by finding different maximal matchings between consecutive layers of $\mathcal{L}$. We first compute an approximate matching $M_{1,2}$ between L_1 and L_2. Let $L_3' \subseteq L_3$ be the set of vertices that are neighbor to matched vertices of $M_{1,2}$ (in L_2). We then, compute a matching $M_{3,4}$ between L_3' and L_4 and we continue like this. One can see this approach as growing vertex disjoint (augmenting) paths from layer L_1 to (eventually) layer L_{k+1}. If at some point, size of the matching between L_i' and L_{i+1} (for some odd i) goes below a certain threshold, we remove all vertices in L_i' from the graph (as they are mostly "dead ends") and backtrack (similar to a DFS procedure). By choosing a relatively large threshold, one can ensure that the number of backtracks is bounded by some function of $(1/\varepsilon)$ only and hence is not too large. At the same time, we like the threshold to be small enough also so that not many actual vertex disjoint paths are marked (incorrectly) as dead ends. Once we complete some paths from L_1 to L_{k+1} we remove these paths (to be augmented

later) and recurse on the remaining graph until no vertices are left. We again emphasize that at each step of this procedure, we simply find a maximal matching between some set of nodes in $\mathcal{L}$ and the above algorithm determines which sets of vertices to choose for finding the next approximate matching. We refer the reader to Section 2.3 (in particular Fig 2) of [58] for a formal definition and a pseudo-code of this algorithm.

We now argue that essentially the same algorithm can also be implemented in the MPC model using our Theorem 11 (instead of picking maximal matchings).

From Maximal to $O(1)$-Approximate Matching. We point out that the algorithm of [58] uses a maximal matching in its construction as it is easy to compute in one pass over a stream and results in a 2-approximate matching. However, we emphasize that the analysis in [58] in no way uses the "maximality" property of this matching and only relies on its approximation ratio. We do not know how to compute a maximal matching in the MPC model efficiently, however, we can use Theorem 11 to compute an $O(1)$-approximate matching. By a simple adjustment of the thresholds used in the above algorithm, we can use any $O(1)$-approximation algorithm to maximum matching in place of a maximal matching algorithm, while blowing up the number of calls to the matching subroutine between two layers by a constant factor in total.

From Streaming to MPC Model. It is easy to see that the construction of the layered graph as well as the choices to which subgraph to compute the next matching on in the above algorithm can be easily performed in constant number of rounds in the MPC model. As argued above, in place of the maximal matching algorithm in reduction of [58], we use our algorithm in Theorem 11 which requires $O(\log\log n)$ MPC rounds (compared to one pass in [58]). As a result, number of rounds in our algorithm is $O_\varepsilon(\log\log n)$ larger than the passes in the streaming algorithm of [58]. Overall, the new algorithm requires $(1/\varepsilon)^{O(1/\varepsilon)} \cdot O(\log\log n)$ MPC rounds and $O(n/\text{polylog}(n))$ memory per machine.

A Some Applications of Randomized Composable Coresets

In the following, we provide more details on the applications of randomized coresets to different computational models and in particular prove Proposition 6.1.

MPC Algorithms. The MPC model is defined in Section 7.1. We can use any α-approximation randomized coreset ALG of size s for a problem P to obtain a parallel algorithm in only *two* MPC rounds. Suppose $G(V, E)$ is the input graph and let $k := \sqrt{m/s}$. The algorithm is as follows:

1. In the first round, create a random k-partition $G^{(1)}, \ldots, G^{(k)}$ and allocate each graph $G^{(i)}$ to the machine $i \in [k]$.
2. Each machine $i \in [k]$ creates a randomized composable coreset $C_i = \mathsf{ALG}(G^{(i)})$.
3. In the second round, collect the union of all coresets to create $H := H(V, C_1, \ldots, C_k)$ on one machine and solve P in H using any offline algorithm.

It is straightforward to verify that this MPC algorithm requires $O(k) = O(\sqrt{m/s})$ machines each with $O(\sqrt{ms} + n)$ memory. Moreover, by Definition 3, the output of this algorithm is an α-approximation to $P(G)$ with high probability.

Streaming Algorithms. In the streaming model, the edges of the input graph are presented to the algorithm one by one in a sequence and the algorithm is allowed to make a single pass (or a few passes) over this sequence.

Similar to the case for MPC algorithms, any α-approximation randomized coreset ALG of size s for a problem P also imply a single-pass streaming algorithm for P in *random arrival* streams. Suppose $G(V, E)$ is the input graph which its edges are arriving in a random order in a stream of length m. The algorithms is as follows.

1. Randomly partition the edges in the stream into $k := \sqrt{m/s}$ *consecutive* pieces such that the distribution of the graphs $G^{(1)}, \ldots, G^{(k)}$ created by edges in each piece is a random k-partition of G (using the randomness in the arrival of the stream).
2. Let H be the empty graph on vertices V. For $i = 1$ to k:
   (a) Read the graph $G^{(i)}$ completely and store it in the memory temporarily.
   (b) Compute a randomized composable coreset $C_i = \mathsf{ALG}(G^{(i)})$.
   (c) Update H by adding all edges in C_i to it and discard all edges in $G^{(i)}$ to reuse the memory in the next iteration.
3. At the end of stream, solve P in H using any offline algorithm.

It is again easy to see that the total memory required by this algorithm is $O(m/k) = O(\sqrt{ms})$ (to store each graph $G^{(i)}$ in the memory temporarily) plus $O(k \cdot s) = O(\sqrt{ms})$ (to store all coresets during the stream). The output of this algorithm is then an α-approximation to $P(G)$ with high probability by Definition 3.

Simultaneous Communication Model. In this model, the input graph is edge partitioned across k machines/players and the goal is to solve the problem on the union of these graphs. In order to this, the players *simultaneously* each send a single message to an additional party called the coordinator who then outputs the solution.

Any α-approximation randomized coreset ALG of size s for a problem P immediately implies a simultaneous protocol for P on *randomly partitioned* inputs. Suppose $G(V, E)$ is the input graph

which its edges are partitioned across k parties *randomly*. Each party only needs to compute a coreset of its input graph and communicate it with the coordinator. The communication by each party then would be $O(s)$ and the coordinator can recover an α-approximation to P with high probability by Definition 3.

Remark A.1. *In the above algorithms, we neglected the computation time needed for solving P on the union of the coresets. This is consistent with the definitions of the models considered here as they all allow unbounded computation time to the algorithm. However, if one insists on having efficient time algorithms (e.g., polynomial time) then the approximation ratio of the resulting algorithm would be $\rho \cdot \alpha$ where ρ is the approximation guarantee of any offline algorithm we use for solving P in the end (in many scenarios however this naive blow-up in the approximation ratio can be avoided by additional care, although not in a black-box way anymore).*

B Missing Details from Section 7

B.1 Useful Concentration of Measure Inequalities

Azuma's inequality proves a concentration bound for martingales.

Proposition B.1 (Azuma's inequality)**.** *Let $\{X_i\}_{i=0}^n$ be a martingale (with respect to some random variables $\{Y_i\}_{i=0}^n$). Suppose there exists a sequence of integers $\{c_i\}_{i=1}^n$ such that $|X_i - X_{i-1}| \leq c_i$; then,*

$$\Pr\Big(|X_n - X_0| \geq \lambda\Big) \leq 2 \cdot \exp\left(-\frac{\lambda^2}{\sum_{i=1}^n c_i^2}\right).$$

We also use the following standard variant of the Chernoff bound as well as its generalization for variables with bounded independence.

Proposition B.2 (Chernoff bound)**.** *Let $X_1, \ldots, X_n$ be independent random variables taking value in $[0,1]$ and $X := \sum_{i=1}^n X_i$. Then, for any $\delta \in (0,1)$*

$$\Pr\Big(|X - \mathbb{E}[X]| \geq \delta \cdot \mathbb{E}[X]\Big) \leq 2 \cdot \exp\left(-\frac{\delta^2 \cdot \mathbb{E}[X]}{3}\right).$$

Proposition B.3 (Chernoff bound with bounded independence [71])**.** *Let $X_1, \ldots, X_n$ be κ-wise independent variables taking value in $[0,1]$ and $X := \sum_{i=1}^n X_i$. For any $\delta \in (0,1)$, if $\kappa \leq \delta^2 \cdot \mathbb{E}[X]/2$, then,*

$$\Pr\Big(|X - \mathbb{E}[X]| \geq \delta \cdot \mathbb{E}[X]\Big) \leq 2 \cdot \exp\left(-\frac{\kappa}{2}\right).$$

B.2 Proof of Lemma 7.4

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

Proof. Consider the following simple procedure for creating an EDCS H of a given graph G:

> While H is not an EDCS(G, β, β^-) of G:
>
> (a) Find an edge e which violates one of the properties of EDCS.
>
> (b) *Fix* the edge e, i.e., remove it from H if it was violating Property (P1) and add it to H if it was violating Property (P2).

The output of the above procedure is clearly an EDCS of graph G. However, a-priori it is not clear that this procedure ever terminates as fixing one edge e can result in many edges violating the

EDCS properties, potentially undoing the previous changes. In the following, we use a potential function argument to show that this procedure always terminates after a finite number of steps, hence implying that an EDCS$(G, \beta, \beta - 1)$ always exists.

We define the following potential function Φ:

$$\Phi := (\beta - 1/2) \cdot \sum_{u \in V} d_H(u) - \sum_{(u,v) \in H} d_H(u) + d_H(v).$$

We argue that in any step of the procedure above, the value of Φ increases by at least 1. Since the maximum value of Φ is at most $O(n \cdot \beta^2)$, this immediately implies that this procedure terminates in $O(n \cdot \beta^2)$ steps.

Define $\Phi_1 := (\beta - 1/2) \cdot \sum_{u \in V} d_H(u)$ and $\Phi_2 := -\sum_{(u,v) \in H} d_H(u) + d_H(v)$ (note the minus sign) and hence $\Phi = \Phi_1 + \Phi_2$. Let (u, v) be the edge we choose to fix at this step, H be the subgraph before fixing the edge (u, v), and H' be the resulting subgraph.

Suppose first that the edge (u, v) was violating Property (P1) of EDCS. As the only change is in the degrees of vertices u and v, Φ_1 decreases by $(2\beta - 1)$. On the other hand, $d_H(u) + d_H(v) \geq \beta + 1$ originally (as (u, v) was violating Property (P1) of EDCS) and hence after removing (u, v) Φ_2 increases by $\beta + 1$. Additionally, for each neighbor w of u and v in H', after removing the edge (u, v), $d_{H'}(w)$ decreases by one. As there are at least $d_{H'}(u) + d_{H'}(v) = d_H(u) + d_H(v) - 2 \geq \beta - 1$ choices for w, this means that in total, Φ_2 increases by at least $(\beta + 1) + (\beta - 1) = 2\beta$. As a result, in this case Φ increases by at least 1 after fixing the edge (u, v).

Now suppose that the edge (u, v) was violating Property (P2) of EDCS instead. In this case, degree of vertices u and v both increase by one, hence Φ_1 increases by $2\beta - 1$. Additionally, not that since edge (u, v) was violating Property (P2) we have $d_H(u) + d_H(v) \leq \beta^- - 1$, so the addition of edge (u, v) decreases Φ_2 by at most $d_{H'}(u) + d_{H'}(v) = d_H(u) + d_H(v) + 2 \leq \beta^- + 1$. Moreover, for each neighbor w of u and v, after adding the edge (u, v), $d_{H'}(w)$ increases by one and since there are at most $d_H(u) + d_H(v) \leq \beta^- - 1$ choices for w, Φ_2 decreases in total by at most $(\beta^- + 1) + (\beta^- - 1) = 2\beta^-$. Since $\beta^- \leq \beta - 1$, we have that Φ increases by at least $(2\beta - 1) - (2\beta^-) \geq 1$ after fixing the edge (u, v), finalizing the proof. ∎

We remark that this proof also implies a natural polynomial time algorithm for computing any EDCS of a given graph G.

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] B. Bollobás. *Random Graphs.* Number 73. Cambridge University Press, 2001.

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

[26] 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.

[27] 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.

[28] 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.

[29] A. Czumaj, J. Łącki, A. Mądry, S. Mitrović, K. Onak, and P. Sankowski. Round compression for parallel matching algorithms. *arXiv preprint arXiv:1707.03478. To appear in STOC 2018*, 2018.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] M. Ghaffari, T. Gouleakis, S. Mitrovic, and R. Rubinfeld. Improved massively parallel computation algorithms for mis, matching, and vertex cover. *CoRR*, abs/1802.08237. To appear in PODC 2018., 2018.

[41] 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.

[42] 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.

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

[44] 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.

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

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

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

[53] 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.

[54] 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.

[55] 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.

[56] 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.

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

[58] 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.

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

[60] 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.

[61] 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.

[62] 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.

[63] 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.

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

[65] 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.

[66] 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.

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-hoeffding bounds for applications with limited independence. *SIAM J. Discrete Math.*, 8(2):223–250, 1995.

[72] 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.

[73] V. G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz*, 3(7):25–30, 1964.

[74] 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.

[75] 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.