Distributed Algorithms for Matching in Hypergraphs

Oussama Hanguir* Clifford Stein*†

Abstract

We study the d-Uniform Hypergraph Matching (d-UHM) problem: given a n-vertex hypergraph G where every hyperedge is of size d, find a maximum cardinality set of disjoint hyperedges. For $d \geq 3$, the problem of finding the maximum matching is $\mathcal{NP}$-complete, and was one of Karp's 21 $\mathcal{NP}$-complete problems. In this paper we are interested in the problem of finding matchings in hypergraphs in the massively parallel computation (MPC) model that is a common abstraction of MapReduce-style computation. In this model, we present the first three parallel algorithms for d-Uniform Hypergraph Matching, and we analyse them in terms of resources such as memory usage, rounds of communication needed, and approximation ratio. The highlights include:

- A $O(\log n)$-round d-approximation algorithm that uses $O(nd)$ space per machine.
- A 3-round, $O(d^2)$-approximation algorithm that uses $\tilde{O}(\sqrt{nm})$ space per machine.
- A 3-round algorithm that computes a subgraph containing a $(d-1+\frac{1}{d})^2$-approximation, using $\tilde{O}(\sqrt{nm})$ space per machine for linear hypergraphs, and $\tilde{O}(n\sqrt{nm})$ in general.

For the third algorithm, we introduce the concept of HyperEdge Degree Constrained Subgraph (HEDCS), which can be of independent interest. We show that an HEDCS contains a fractional matching with total value at least $|M^*|/(d-1+\frac{1}{d})$, where $|M^*|$ is the size of the maximum matching in the hypergraph. Moreover, we investigate the experimental performance of these algorithms both on random input and real instances. Our results support the theoretical bounds and confirm the trade-offs between the quality of approximation and the speed of the algorithms.

1 Introduction

As massive graphs become more ubiquitous, the need for scalable parallel and distributed algorithms that solve graph problems grows as well. In recent years, we have seen progress in many graph problems (e.g. spanning trees, connectivity, shortest paths [4, 5]) and, most relevant to this work, matchings [22, 27]. A natural generalization of matchings is graphs is to matchings in *hypergraphs*. Hypergraph Matching is an important problem with many applications such as capital budgeting, crew scheduling, facility location, scheduling airline flights [53], forming a coalition structure in multi-agent systems [50] and determining the winners in combinatorial auctions [49]. (See [56] for a partial survey.). Although matching problems in graphs are one of the most well-studied problems in algorithms and optimization, the NP-hard problem of finding a maximum matching in a hypergraph is not as well understood.

In this work, we are interested in the problem of finding matchings in very large hypergraphs, large enough that we cannot solve the problem on one computer. We develop, analyze and experimentally evaluate three parallel algorithms for hypergraph matchings in the MPC model. Two of

*Columbia University, New York, NY 10027, USA

†Research partially supported by NSF grants CCF-1714818 and CCF-1822809.

the algorithms are generalizations of parallel algorithms for matchings in graphs. The third algorithm develops new machinery which we call a hyper-edge degree constrained subgraph (HEDCS), generalizing the notion of an edge-degree constrained subgraph (EDCS). The EDCS has been recently used in parallel and dynamic algorithms for graph matching problems [6, 16, 17]. We will show a range of algorithm tradeoffs between approximation ratio, rounds, memory and computation, evaluated both as worst case bounds, and via computational experiments.

More formally, a *hypergraph* G is a pair $G = (V, E)$ where V is the set of vertices and E is the set of hyperedges. A *hyperedge* $e \in E$ is a nonempty subset of the vertices. The cardinality of a hyperedge is the number of vertices it contains. When every hyperedge has the same cardinality d, the hypergraph is said to be *d-uniform*. A hypergraph is linear if the intersection of any two hyperedges has at most one vertex. A *hypergraph matching* is a subset of the hyperedges $M \subseteq E$ such that every vertex is covered at most once, i.e. the hyperedges are mutually disjoint. This notion generalises matchings in graphs. The cardinality of a matching is the number of hyperedges it contains. A matching is called maximum if it has the largest cardinality of all possible matchings, and maximal if it is not contained in any other matching. In the d-Uniform Hypergraph Matching Problem (also referred to as Set Packing or d-Set Packing when all hyperedges have cardinality d), a d-uniform hypergraph is given and one needs to find the maximum cardinality matching.

We adopt the most restrictive MapReduce-like model of modern parallel computation among [4, 11, 29, 40], the Massively Parallel Computation (MPC) model of [11]. This model is widely used to solve different graph problems such as matching, vertex cover [2, 6, 7, 27, 44], independent set [27, 34], as well as many other algorithmic problems. In this model, we have k machines (processors) each with space s. N is the size of the input and our algorithms will satisfy $k \cdot s = \tilde{O}(N)$, which means that the total space in the system is only a polylogarithmic factor more than the input size. The computation proceeds in rounds. At the beginning of each round, the data (e.g. vertices and edges) is distributed across the machines. In each round, a machine performs local computation on its data (of size s), and then sends messages to other machines for the next round. Crucially, the total amount of communication sent or received by a machine is bounded by s, its space. Each machine treats the received messages as the input for the next round. Our model limits the number of machines and the memory per machine to be substantially sublinear in the size of the input. On the other hand, no restrictions are placed on the computational power of any individual machine. The main complexity measure is therefore the memory per machine and the number of rounds R required to solve a problem, which we consider to be the "parallel time" of the algorithm. For the rest of the paper, $G(V, E)$ is a d-uniform hypergraph with n vertices and m hyperedges, and when the context is clear, we will simply refer to G as a graph and to its hyperedges as edges. $MM(G)$ denotes the maximum matching in G, and $\mu(G) = |MM(G)|$ is the size of that matching. We define $d_G(v)$ to be the degree of a vertex v (the number of hyperedges that v belongs to) in G. When the context is clear, we will omit indexing by G and simply denote it $d(v)$.

2 Our contribution and results.

We design and implement algorithms for the d-UHM in the MPC model. We will give three different algorithms, demonstrating different trade-offs between the model's parameters. Our algorithms are inspired by methods to find maximum matchings in graphs, but require developing significant new tools to address hypergraphs. We are not aware of previous algorithms for hypergraph matching in the MPC model. First we generalize the randomized coreset algorithm of [7] which finds an 3-rounds $O(1)$-approximation for matching in graphs. Our algorithm partitions the graph into random pieces across the machines, and then simply picks a maximum matching of each machine's

subgraph. We show that this natural approach results in a $O(d^2)$-approximation. While the algorithmic generalization is straightforward, the analysis requires several new ideas.

Theorem 4.2 (restated). *There exists an MPC algorithm that with high probability computes a $(3d(d-1)+3+\epsilon)$-approximation for the d-UHM problem in 3 MPC rounds on machines of memory $s = \tilde{O}(\sqrt{nm})$.*

Our second result concerns the MPC model with per-machine memory $O(d \cdot n)$. We adapt the sampling technique and a post-processing strategy of [44] to construct maximal matchings in hypergraphs, and are able to show that in d-uniform hypergraphs, this technique yields a maximal matching, and thus a d-approximation to the d-UHM problem in $O(\log n)$ rounds.

Theorem 5.1 (restated). *There exists an MPC algorithm that given a d-uniform hypergraph $G(V, E)$ with high probability computes a maximal matching in G in $O(\log n)$ MPC rounds on machines of memory $s = \Theta(d \cdot n)$.*

Our third result generalizes the edge degree constrained subgraphs (EDCS), originally introduced by Bernstein and Stein [16] for maintaining large matchings in dynamic graphs, and later used for several other problems including matching and vertex cover in the streaming and MPC model [6, 17]. We call these generalized subgraphs hyper-edge degree constrained subgraphs (HEDCS). We show that they exist for specific parameters, and that they contain a good approximation for the d-UHM problem. We prove that a HEDCS of a graph G, with well chosen parameters, contain a fractional matching with a value at least $\frac{d}{d^2-d+1}MM(G)$, and that the underlying fractional matching is special in the sense that for each hyperedge, it either assigns a value of 1 or a value less than some chosen ϵ. We call such a fractional matching an ϵ-restricted fractional matching. For Theorem 6.12, we compute a HEDCS of the graph in a distributed fashion. This procedure relies on the robustness properties that we prove for the HEDCSs under sampling.

Theorem 6.4 (restated informally). *Let G be a d-uniform hypergraph and $0 < \epsilon < 1$. There exists a HEDCS that contains an ϵ-restricted fractional matching M_f^H with total value at least $\mu(G)\left(\frac{d}{d^2-d+1} - \epsilon\right)$.*

Theorem 6.12 (restated). *There exists an MPC algorithm that given a d-uniform hypergraph $G(V, E)$, where $|V| = n$ and $|E| = m$, can construct a HEDCS of G in 2 MPC rounds on machines of memory $s = \tilde{O}(n\sqrt{nm})$ in general and $s = \tilde{O}(\sqrt{nm})$ for linear hypergraphs.*

Corollary 6.13 (restated). *There exists an MPC algorithm that with high probability achieves a $d(d-1+1/d)^2$-approximation to the d-Uniform Hypergraph Matching in 3 rounds.*

Table 1 summarizes our results.

Approximation ratio	Rounds	Memory per machine	Computation per round
$3d(d-1)+3$	3	$\tilde{O}(\sqrt{nm})$	Exponential
d	$O(\log n)$	$O(dn)$	Polynomial
$d(d-d+1/d)^2$	3	$\tilde{O}(n\sqrt{nm})$ in general $\tilde{O}(\sqrt{nm})$ for linear hypergraphs	Polynomial

Table 1: Our parallel algorithms for the d-uniform hypergraph matching problem.

Experimental results. We implement our algorithms in a simulated MPC model environment and test them both on random and real-world instances. Our experimental results are consistent with the theoretical bounds on most instances, and show that there is a trade-off between the

extent to which the algorithms use the power of parallelism and the quality of the approximations. This trade-off is illustrated by comparing the number of rounds and the performance of the algorithms on machines with the same memory size. See Section 7 for more details.

Our techniques. For Theorem 4.2 and Theorem 6.12, we use the concept of composable coresets, which has been employed in several distributed optimization models such as the streaming and MapReduce models [1, 8, 9, 10, 37, 47]. Roughly speaking, the main idea behind this technique is as follows: first partition the data into smaller parts. Then compute a representative solution, referred to as a coreset, from each part. Finally, obtain a solution by solving the optimization problem over the union of coresets for all parts. We use a randomized variant of composable coresets, first introduced in [46], where the above idea is applied on a random clustering of the data. This randomized variant has been used after for Graph Matching and Vertex Cover [6, 7], as well as Column Subset Selection [18]. Our algorithm for Theorem 4.2 is similar to previous works, but the analysis requires new techniques for handling hypergraphs. Theorem 5.1 is a relatively straightforward generalization of the corresponding result for matching.

The majority of our technical innovation is contained in Theorem 6.4 and 6.12. Our general approach is to construct, in parallel, a HEDCS that will contain a good approximation of the maximum matching in the original graph and that will fit on one machine. Then we can run an approximation algorithm on the resultant HEDCS to come up with a good approximation to the maximum matching in this HEDCS and hence in the original graph. In order to make this approach work, we need to generalize much of the known EDCS machinery [16, 17] to hypergraphs. This endeavor is quite involved, as almost all the proofs do not generalize easily and, as the results show, the resulting bounds are weaker than those for graphs. We first show that HEDCSs exist, and that they contain large fractional matchings. We then show that they are useful as a coreset, which amounts to showing that even though there can be many different HEDCS of some fixed hypergraph $G(V, E)$, the degree distributions of every HEDCS (for the same parameters) are almost identical. In other words, the degree of any vertex v is almost the same in every HEDCS of G. We show also that HEDCS are robust under edge sampling, in the sense that edge sampling from a HEDCS yields another HEDCS. These properties allow to use HEDCS in our coresets and parallel algorithm in the rest of the paper.

3 Related Work

Hypergraph Matching. The problem of finding a maximum matching in d-uniform hypergraphs is $\mathcal{NP}$-hard for any $d \geq 3$ [41, 48], and APX-hard for $d = 3$ [39]. The most natural approach for an approximation algorithm for the hypergraph matching problem is the greedy algorithm: repeatedly add an edge that doesn't intersect any edges already in the matching. This solution is clearly within a factor of d from optimal: from the edges removed in each iteration, the optimal solution can contain at most d edges (at most one for each element of the chosen edge). It is also easy to construct examples showing that this analysis is tight for the greedy approach. All the best known approximation algorithms for the Hypergraph Matching Problem in d-uniform hypergraphs are based on local search methods [13, 15, 20, 30, 36]. The first such result by Hurkens and Schrijver [36] gave a $(\frac{d}{2} + \epsilon)$-approximation algorithm. Halldorsson [30] presented a quasi-polynomial $(\frac{d+2}{3})$-approximation algorithm for the unweighted d-UHM. Sviridenko and Ward [54] established a polynomial time $(\frac{d+2}{3})$-approximation algorithm that is the first polynomial time improvement over the $(\frac{d}{2} + \epsilon)$ result from [36]. Cygan [21] and Furer and Yu [26] both provide a $(\frac{d+1}{3} + \epsilon)$ polynomial time approximation algorithm, which is the best approximation guarantee

known so far. On the other hand, Hazan, Safra and Schwartz [35] proved that it is hard to approximate d-UHM problem within a factor of $\Omega(d/\log d)$.

Matching in parallel computation models. The study of the graph maximum matching problem in parallel computation models can be traced back to PRAM algorithms of 1980s [3, 38, 45]. Since then it has been studied in the LOCAL and MPC models, and $(1+\epsilon)$- approximation can be achieved in $O(\log\log n)$ rounds using a space of $O(n)$ [6, 22, 27]. The question of finding a maximal matching in a small number of rounds has also been considered in [28, 44]. Recently, Behnezhad *et al.* [12] presented a $O(\log\log\Delta)$ round algorithm for maximal matching with $O(n)$ memory per machine. While we are not aware of previous work on hypergraph matching in the MPC model, finding a maximal hypergraph matching has been considered in the LOCAL model. Both Fischer *et al.* [24] and Harris [33] provide a deterministic distributed algorithm that computes $O(d)$-approximation to the d-UHM.

Maximum Independent Set. The d-UHM is strongly related to different variants of the Independent Set problem as well as other combinatorial problems. The maximum independent set (MIS) problem on degree bounded graphs can be mapped to d-UHM when the degree bound is d [14, 31, 32, 55]. The d-UHM problem can also be studied under a more general problem of maximum independent set on $(d+1)$-claw-free graphs [13, 20]. (See [19] of connections between d-UHM and other combinatorial optimization problems).

4 A 3-round $O(d^2)$-approximation

In this section, we generalise the randomized composable coreset algorithm of Assadi and Khanna [7]. They used a maximum matching as a coreset and obtained a $O(1)$-approximation. We use a hypergraph maximum matching and we obtain a $O(d^2)$-approximation. We first define a k-partitioning and then present our greedy approach.

Definition 4.1 (Random k-partitioning)**.** *Let E be an edge-set of a hypergraph $G(V,E)$; we say that a collection of edges $E^{(1)},\ldots,E^{(k)}$ is a random k-partition of E if the sets are constructed by assigning each edge $e\in E$ to some $E^{(i)}$ chosen uniformly at random. A random k-partition of E naturally results in partitioning the graph G into k subgraphs $G^{(1)},\ldots,G^{(k)}$ where $G^{(i)}:=G(V,E^{(i)})$ for all $i\in[k]$.*

Let $G(V,E)$ be any d-uniform hypergraph and $G^{(1)},\ldots,G^{(k)}$ be a random k-partitioning of G. We describe a simple greedy process for combining the maximum matchings of $G^{(i)}$, and prove that this process results in a $O(d^2)$-approximation of the maximum matching of G.

Algorithm 1: Greedy

```
Construct a random k-partitioning of G across the k machines. Let M^(0) := ∅.;
for i = 1 to k do
    Set MM(G^(i)) to be an arbitrary hypergraph maximum matching of G^(i).;
    Let M^(i) be a maximal matching obtained by adding to M^(i−1) the edges of MM(G^(i))
      that do not violate the matching property.;
end
return M := M^(k).;
```

Theorem 4.2. *Greedy computes, with high probability, a $(3d(d-1)+3+\epsilon)$-approximation for the d-UHM problem in 3 MPC rounds on machines of memory $s = \tilde{O}(\sqrt{nm})$.*

In the rest of the section, we present a proof sketch for Theorem 4.2. Let $c = \frac{1}{3d(d-1)+3}$, we show that $M^{(k)} \geq c \cdot \mu(G)$ w.h.p, where $M^{(k)}$ is the output of *Greedy*. The randomness stems from the fact that the matchings $M^{(i)}$ (for $i \in \{1, \ldots, k\}$) constructed by *Greedy* are random variables depending on the random k-partitioning.

We adapt the general approach in [7] for d-uniform hypergraphs, with $d \geq 3$. Suppose at the beginning of the i-th step of *Greedy*, the matching $M^{(i-1)}$ is of size $o(\mu(G)/d^2)$. One can see that in this case, there is a matching of size $\Omega(\mu(G))$ in G that is entirely incident on vertices of G that are not matched by $M^{(i-1)}$. We can show that in fact $\Omega(\mu(G)/(d^2k))$ edges of this matching are appearing in $G^{(i)}$, even when we condition on the assignment of the edges in the first $(i-1)$ graphs. Next we argue that the existence of these edges forces any maximum matching of $G^{(i)}$ to match $\Omega(\mu(G)/(d^2k))$ edges in $G^{(i)}$ between the vertices that are not matched by $M^{(i-1)}$; these edges can always be added to the matching $M^{(i-1)}$ to form $M^{(i)}$. Hence, while the maximal matching in *Greedy* is of size $o(\mu(G))$, we can increase its size by $\Omega(\mu(G)/(d^2k))$ edges in each of the first $k/3$ steps, hence obtaining a matching of size $\Omega(\mu(G)/d^2)$ at the end. The following Lemma 4.3 formalizes this argument.

Lemma 4.3. *For any $i \in [k/3]$, if $M^{(i-1)} \leq c \cdot \mu(G)$, then, w.p. $1 - O(1/n)$,*

$$M^{(i)} \geq M^{(i-1)} + \frac{1 - 3d(d-1)c - o(1)}{k} \cdot \mu(G) \ .$$

The proof of Lemma 4.3 as well as the full proof of Theorem 4.2 are in appendix A.1. We conclude this section by stating that computing a maximum matching on every machine is only required for the analysis, i.e., to show that there exists a large matching in the union of coresets. In practice, we can use an approximation algorithm to obtain a large matching from the coresets. In our experiments, we use a maximal matching instead of maximum.

5 A $O(\log n)$-rounds d-approximation algorithm

In this section, we show how to compute a maximal matching in $O(\log n)$ MPC rounds, by generalizing the algorithm in [44] to d-uniform hypergraphs. The algorithm provided by Lattanzi *el al.* [44] computes a maximal matching in graphs in $O(\log n)$ if $s = \Theta(n)$, and in at most $\lfloor c/\epsilon \rfloor$ iterations when $s = \Theta(n^{1+\epsilon})$, where $0 < \epsilon < c$ is a fixed constant. Harvey *et al.* [34] show a similar result.

The algorithm first samples $O(s)$ edges and finds a maximal matching M_1 on the resulting subgraph (we will specify a bound on the memory s later). Given this matching, we can safely remove edges that are in conflict (i.e. those incident on nodes in M_1) from the original graph G. If the resulting filtered graph H is small enough to fit onto a single machine, the algorithm augments M_1 with a matching found on H. Otherwise, we augment M_1 with the matching found by recursing on H. Note that since the size of the graph reduces from round to round, the effective sampling probability increases, resulting in a larger sample of the remaining graph.

Theorem 5.1. *Given a d-uniform hypergraph $G(V, E)$, Iterated-Sampling omputes a maximal matching in G with high probability in $O(\log n)$ MPC rounds on machines of memory $s = \Theta(d \cdot n)$.*

The proof of Theorem 5.1 is presented in appendix A.3. It first proves that after sampling edges, the size of the graph $G[I]$ induced by unmatched vertices decreases exponentially with high probability, hence the algorithm terminates in $O(\log n)$ rounds.

Algorithm 2: Iterated-Sampling

1 Set $M := \emptyset$ and $\mathcal{S} = E$.;
2 Sample every edge $e \in \mathcal{S}$ uniformly at with probability $p = \frac{s}{5|\mathcal{S}|d}$ to form E'.;
3 If $|E'| > s$ the algorithm fails. Otherwise give the graph $G(V, E')$ as input to a single machine and compute a maximal matching M' on it. Set $M = M \cup M'$.;
4 Let I be the unmatched vertices in G and $G[I]$ the induced subgraph with edges $E[I]$. If $E[I] > s$, set $\mathcal{S} := E[I]$ and return to step 2.;
5 Compute a maximal matching M'' on $G[I]$ and output $M = M \cup M''$.;

6 A 3-round $O(d^3)$-approximation using HEDCSs

In graphs, edge degree constrained subgraphs (EDCS) [16, 17] have been used as a local condition for identifying large matchings, and leading to good dynamic and parallel algorithms [6, 16, 17]. These papers showed that an EDCS exists, that it contains a large matching, that it is robust to sampling and composition, and that it can be used as a coreset.

In this section, we present a generalization of EDCS for hypergraphs. We prove that an HEDCS exists, that it contains a large matching, that it is robust to sampling and composition, and that it can be used as a coreset. The proofs and algorithms, however, require significant developments beyond the graph case. We first present definitions of a fractional matching and HEDCS.

Definition 6.1. *[**Fractional matching**]*
In a hypergraph $G(V, E)$, a fractional matching is a mapping from $y : E \mapsto [0, 1]$ such that for every edge e we have $0 \leq y_e \leq 1$ and for every vertex v : $\sum_{e \in v} y_e \leq 1$. The value of such fractional matching is equal to $\sum_{e \in E} y_e$. For $\epsilon > 0$, a fractional matching is an ϵ-restricted fractional matching if for every edge e: $y_e = 1$ or $y_e \in [0, \epsilon]$.

Definition 6.2. *For any hypergraph $G(V, E)$ and integers $\beta \geq \beta^- \geq 0$ a hyperedge degree constraint subgraph (HEDCS)(H, β, β^-) is a subgraph $H := (V, E_H)$ of G satisfying:*

- *(P1): For any hyperedge $e \in E_H$: $\sum_{v \in e} d_H(v) \leq \beta$.*
- *(P2): For any hyperedge $e \notin E_H$: $\sum_{v \in e} d_H(v) \geq \beta^-$.*

We show via a constructive proof (in Appendix A.4) that a hypergraph contains an HEDCS when the parameters of this HEDCS satisfy the inequality $\beta - \beta^- \geq d - 1$.

Lemma 6.3. *Any hypergraph G contains an HEDCS(G, β, β^-) for any parameters $\beta - \beta^- \geq d - 1$.*

We prove that an HEDCS of a graph contains a large ϵ-restricted fractional matching that approximates the maximum matching by a factor less than d. The proof is in appendix A.5.

Theorem 6.4. *Let G be a d-uniform hypergraph and $0 \leq \epsilon < 1$. Let H be a hyperedge degree constrained subgraph$(G, \beta, \beta(1-\lambda))$ with $\lambda = \frac{\epsilon}{6}$ and $\beta \geq \frac{8d^2}{d-1} \cdot \lambda^{-3}$. Then H contains an ϵ-restricted fractional matching M_f^H with total value at least $\mu(G)(\frac{d}{d^2-d+1} - \frac{\epsilon}{d-1})$.*

6.1 Sampling and constructing an HEDCS in the MPC model

Our results in this section are general and applicable to every computation model. We prove structural properties about the HEDCS that will help us construct HEDCS in the MPC model. We show that the degree distributions of every HEDCS (for the same parameters β and λ) are

almost identical. In other words, the degree of any vertex v is almost the same in every HEDCS of the same hypergraph G. We show also that HEDCS are robust under edge sampling, i.e. that edge sampling from and HEDCS yields another HEDCS, and that the degree distributions of any two HEDCS for two different edge sampled subgraphs of G is almost the same no matter how the two HEDCS are selected. In the following lemma, we argue that any two HEDCS of a graph G (for the same parameters β, β^-) are "somehow identical" in that their degree distributions are close to each other. In the rest of this section, we fix the parameters β, β^- and two subgraphs A and B that are both $HEDCS(G, \beta, \beta^-)$. The proofs are in appendix A.8.

Lemma 6.5. *(Degree Distribution Lemma). Fix a d-uniform hypergraph $G(V, E)$ and parameters β, $\beta^- = (1-\lambda)\cdot\beta$ (for λ small enough). For any two subgraphs A and B that are $HEDCS(G, \beta, \beta^-)$, and any vertex $v \in V$, then $|d_A(v) - d_B(v)| = O(\sqrt{n})\lambda^{1/2}\beta$.*

Corollary 6.6. *For d-uniform linear hypergraphs, the degree distribution is tighter, and $|d_A(v) - d_B(v)| = O(\log n)\lambda\beta$.*

Next we prove two lemmas regarding the structure of different HEDCS across sampled subgraphs. The first lemma shows that edge sampling an HEDCS results in another HEDCS for the sampled subgraph. The second lemma shows that the degree distributions of any two HEDCS for two different edge sampled subgraphs of G is almost the same.

Lemma 6.7. *Let H be a HEDCS(G, β_H, β_H^-) for parameters $\beta_H := (1-\frac{\lambda}{\alpha})\cdot\frac{\beta}{p}$ and $\beta_H^- := \beta_H - (d-1)$, and $\beta \geq 15d(\alpha d)^2 \cdot \lambda^{-2} \cdot \log n$ such that $p < 1 - \frac{2}{\alpha}$. 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 high probability:*

1. *For any vertex $v \in V$: $|d_{H_p}(v) - p \cdot d_H(v)| \leq \frac{\lambda}{\alpha d}\beta$*
2. *H_p is a HEDCS of G_p with parameters $(\beta, (1-\lambda)\cdot\beta)$.*

Lemma 6.8. *(HEDCS in Edge Sampled Subgraph). Fix any hypergraph $G(V, E)$ 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 HEDCSs of G_1 and G_2 with parameters $(\beta, (1-\lambda)\cdot\beta)$. Suppose $\beta \geq 15d(\alpha d)^2 \cdot \lambda^{-2} \cdot \log n$, then, with probability $1 - \frac{4}{n^{5d-1}}$, simultaneously for all $v \in V$: $|d_{H_1}(v) - d_{H_2}(v)| = O\big(n^{1/2}\big)\lambda^{1/2}\beta$.*

Corollary 6.9. *If G is linear, then $|d_{H_1}(v) - d_{H_2}(v)| = O\big(\log n\big)\lambda\beta$.*

We are now ready to present a parallel algorithm that will use the HEDCS subgraph. We first compute an HEDCS in parallel via edge sampling. Let $G^{(1)}, \ldots, G^{(k)}$ be a random k-partition of a graph G. We show that if we compute an arbitrary HEDCS of each graph $G^{(i)}$ (with no coordination across different graphs) and combine them together, we obtain a HEDCS for the original graph G. We then store this HEDCS in one machine and compute a maximal matching on it . We present our algorithm for all range of memory $s = n^{\Omega(1)}$. Lemma 6.10 and Corollary 6.11 serve as a proof to Theorem 6.12. Their proof is in appendix.

Lemma 6.10. *Suppose $k \leq \sqrt{m}$. Then with high probability*

1. *The subgraph C is a $HEDCS(G, \beta_C, \beta_C^-)$ for parameters: $\lambda_C = O\big(n^{1/2}\big)\lambda^{1/2}$, $\beta_C = (1 + d \cdot \lambda_C) \cdot k \cdot \beta$ and $\beta_C^- = (1 - \lambda - d \cdot \lambda_C) \cdot k \cdot \beta$.*

Algorithm 3: HEDCS-Matching(G, s): a parallel algorithm to compute a $O(d^3)$-approximation matching on a d-uniform hypergraph G with m edges on machines of memory $O(s)$

Define $k := \frac{m}{s \log n}$, $\lambda := \frac{1}{2n \log n}$ and $\beta := 500 \cdot d^3 \cdot n^2 \cdot \log^3 n$.;
$G^{(1)}, \ldots, G^{(k)}$:= random k-partition of G.;
for $i = 1$ *to* k, *in parallel* **do**
  Compute $C^{(i)} = HEDCS(G^{(i)}, \beta, (1-\lambda) \cdot \beta)$ on machine i.;
end
Define the multi-graph $C(V, E_C)$ with $E_C := \cup_{i=1}^{k} C^{(i)}$.;
Compute and output a maximal matching on C.;

2. *The total number of edges in each subgraph $G^{(i)}$ of G is $\tilde{O}(s)$.*

3. *If $s = \tilde{O}(n\sqrt{nm})$, then the graph C can fit in the memory of one machine.*

Corollary 6.11. *If G is linear, then by choosing $\lambda := \frac{1}{2\log^2 n}$ and $\beta := 500 \cdot d^3 \cdot \log^4 n$ in Algorithm 3 we have:*

1. *With high probability, the subgraph C is a $HEDCS(G, \beta_C, \beta_C^-)$ for parameters: $\lambda_C = O(\log n)\lambda$, $\beta_C = (1 + d \cdot \lambda_C) \cdot k \cdot \beta$ and $\beta_C^- = (1 - \lambda - d \cdot \lambda_C) \cdot k \cdot \beta$.*

2. *If $s = \tilde{O}(\sqrt{nm})$ then C can fit on the memory of one machine.*

Theorem 6.12. *HEDCS-Matching constructs a HEDCS of G in 2 MPC rounds on machines of memory $s = \tilde{O}(n\sqrt{nm})$ in general and $s = \tilde{O}(\sqrt{nm})$ for linear hypergraphs.*

Corollary 6.13. *HEDCS-Matching achieves a $d(d - 1 + 1/d)^2$-approximation to the d-Uniform Hypergraph Matching in 3 rounds with high probability.*

Proof. [Proof of Corollary 6.13] We show that with high probability, C verifies the assumptions of theorem 6.4. From Lemma 6.10, we get that with high probability, the subgraph C is a $HEDCS(G, \beta_C, \beta_C^-)$ for parameters: $\lambda_C = O\big(n^{1/2}\big)\lambda^{1/2}$, $\beta_C = (1 + d \cdot \lambda_C) \cdot k \cdot \beta$ and $\beta_C^- = (1 - \lambda - d \cdot \lambda_C) \cdot k \cdot \beta$. We can see that $\beta_c \geq \frac{8d^2}{d-1} \cdot \lambda_c^{-3}$. Therefore by Theorem 6.4, C contains a $(d - 1 + \frac{1}{d})$-approximate ϵ-restricted matching. Since the integrality gap of the d-UHM is at most $d - 1 + \frac{1}{d}$ (see [19] for details), then C contains a $(d - 1 + \frac{1}{d})^2$-approximate matching. Taking a maximal matching in C multiplies the approximation factor by at most d. Therefore, any maximal matching in C is a $d(d - 1 + \frac{1}{d})^2$-approximation. □

7 Computational Experiments

We conduct a wide variety of experiments on both random and real-life data [43, 51, 57]. We implement the three algorithms **Greedy**, **Iterated-Sampling** and **HEDCS-Matching** using Python, and more specifically relying on the module *pygraph* and its class *pygraph.hypergraph* to construct and perform operations on hypergraphs. We simulate the MPC model by computing a k-partitioning and splitting it into k different inputs. Parallel computations on different parts of the k-partitioning are handled through the use of the *multiprocessing* library in Python. We

compute the optimal matching through an Integer Program for small instances of random uniform hypergraphs ($d \leq 10$) as well as geometric hypergraphs. The experiments were conducted on a 2.6 GHz Intel Core i7 processor and 16 GB RAM workstation. The datasets differ in their number of vertices, hyperedges, vertex degree and hyperedge cardinality. In the following tables, n and m denote the number of vertices and number of hyperedges respectively, and d is the size of hyperedges. For Table 3, the graphs might have different number of edges and $\bar{m}$ denotes the average number of edges. k is the number of machines used to distribute the hypergraph initially. We limit the number of edges that a machine can store to $\frac{2m}{k}$. In the columns Gr, IS and HEDCS, we store the average ratio between the size of the matching computed by the algorithms **Greedy**, **Iterated-Sampling** and **HEDCS-Matching** respectively, and the size of a benchmark. This ratio is computed by the percentage $\frac{ALG}{BENCHMARK}$, where ALG is the output of the algorithm, and $BENCHMARK$ denotes the size of the benchmark. The benchmarks include the size of optimal solution when it is possible to compute, or the size of a maximal matching computed via a sequential algorithm. $\#I$ denotes the number of instances of random graphs that we generated for fixed n, m and d. β and β^- are the parameters used to construct the HEDCS subgraphs in **HEDCS-Matching**. These subgraphs are constructed using the procedure in the proof of Lemma 6.3.

7.1 Experiments with random hypergraphs

Random Uniform Hypergraphs. For a fixed n, m and d, each potential hyperedge is sampled independently and uniformly at random from the set of vertices. In Table 2, we use the size of a perfect matching $\frac{n}{d}$ as a benchmark, because a perfect matching in random graphs exists with probability $1 - o(1)$ under some conditions on m, n and d. If $d(n, m) = \frac{m \cdot d}{n}$ is the expected degree of a random uniform hypergraph, Frieze and Janson [25] showed that $\frac{d(n,m)}{n^{1/3}} \to \infty$ is a sufficient condition for the hypergraph to have a perfect matching with high probability. Kim [42] further weakened this condition to $\frac{d(n,m)}{n^{1/(5+2/(d-1))}} \to \infty$. We empirically verify, by solving the IP formulation, that for $d = 3, 5$ and for small instances of $d = 10$, our random graphs contain a perfect matching. In Table 2, the benchmark is the size of a perfect matching, while in Table 5, it is the size of a greedy maximal matching. In terms of solution quality (Tables 2 and 5) **HEDCS-Matching** performs consistently better than **Greedy**, and **Iterated-Sampling** performs significantly better than the other two. When compared to the size of a maximal matching, **Iterated-Sampling** still performs better, followed by **HEDCS-Matching**. However, the ratio is smaller when compared to a maximal matching, which is explained by the deterioration of the quality of greedy maximal matching as n and d grow. Dufosse *et al.* [23] confirm that the approximation quality of a greedy maximal matching on random graphs that contain a perfect matching degrades as a function of n and d. The performance of the algorithms decreases as d grows, which is theoretically expected since their approximations ratio are both proportional to d. The number of rounds for **Iterated-Sampling** grows slowly with n, which is consistent with $O(\log n)$ bound. Recall that the number of rounds for the other two algorithms is constant and equal to 3.

Geometric Random Hypergraphs. The second class we experimented on is random geometric hypergraphs. The vertices of a random geometric hypergraph (RGH) are randomly sampled from the uniform distribution of the space $[0, 1)^2$. A set of d different vertices $v_1, \ldots, v_d \in V$ forms a hyperedge if, and only if, the distance between any v_i and v_j is less than a previously specified parameter $r \in (0, 1)$. The parameters r and n fully characterize a RGH. We fix $d = 3$ and generate different geometric hypergraphs by varying n and r. We compare the output of the algorithms to the optimal solution that we compute through the IP formulation. Table 3 shows that the performance of our three algorithms is almost similar with **Iterated-Sampling** outperforming **Greedy**

and **HEDCS-Matching** as the size of the graphs grows. We also observe that random geometric hypergraphs do not contain perfect matchings, mainly because of the existence of some outlier vertices that do not belong to any edge. The number of rounds of **Iterated-Sampling** still grows with n, confirming the theoretical bound and the results on random uniform hypergraphs.

n	m	d	k	#I	Gr	IS	HEDCS	β	β^-	Rounds IS
15	200	3	5	500	77.6%	86.6%	82.8%	5	3	3.8
30	400		5		78.9%	88.1%	80.3%	7	4	4.56
100	3200		10		81.7%	93.4%	83.1%	5	2	5.08
300	4000		10		78.8%	88.7%	80.3%	8	6	7.05
50	800	5	6	500	66.0%	76.2%	67.0%	16	11	4.89
100	2,800		10		68.0%	79.6%	69.8%	16	11	4.74
300	4,000		10		62.2%	75.1%	65.5%	10	5	6.62
500	8,000		16		63.3%	76.4%	65.6%	10	5	7.62
500	15,000	10	16	500	44.9%	58.3%	53.9%	20	10	6.69
1,000	50,000		20		47.3%	61.3%	50.5%	20	10	8.25
2,500	100,000		20		45.6%	59.9%	48.2%	20	10	8.11
5,000	200,000		20		45.0%	59.7%	47.8%	20	10	7.89
1,000	50,000	25	25	100	27.5%	34.9%	30.8%	75	50	8.10
2,500	100,000		25		26.9%	34.0%	27.0%	75	50	8.26
5,000	250,000		30		26.7%	33.8%	28.8%	75	50	8.23
10,000	500,000		30		26.6%	34.1%	28.2%	75	50	8.46
5,000	250,000	50	30	100	22.4%	30.9%	27.9%	100	50	10.22
10,000	500,000		30		22.2%	31.0%	26.5%	100	50	10.15
15,000	750,000		30		20.9%	30.8%	26.4%	100	50	10.26
25,000	1,000,000		30		20.9%	30.8%	26.4%	100	50	10.29

Table 2: Comparison on random instances with perfect matching benchmark, of size $\frac{n}{d}$.

n	r	$\bar{m}$	d	k	#I	Gr	IS	HEDCS	β	β^-	Rounds IS
100	0.2	930.1 ± 323	3	5	100	88.3%	89.0%	89.6%	3	5	4.1
100	0.25	1329.5 ± 445		10		88.0%	89.0%	89.5%	3	5	5.2
250	0.15	13222± 3541		5		85.0%	88.6%	85.5%	4	7	8.1
300	0.15	14838 ± 4813		10		85.5%	88.0%	85.2%	4	7	11.1
300	0.2	27281 ± 8234		10		85.0%	89.0%	86.3%	4	7	13.5

Table 3: Comparison on random geometric hypergraphs with optimal matching benchmark.

7.2 Experiments with real data

PubMed and Cora Datasets. We employ two citation network datasets, namely the Cora and Pubmed datasets [51, 57]. These datasets are represented with a graph, with vertices being publications, and edges being a citation links from one article to another. We construct the hypergraphs in two steps 1) each article is a vertex; 2) each article is taken as a centroid and forms a hyperedge to connect those articles which have citation links to it (either citing it or being cited). The Cora hypergraph has an average edge size of 3.0 ± 1.1, while the average in the Pubmed hypergraph is 4.3 ± 5.7. The number of edges in both is significantly smaller than the number of vertices, therefore we allow each machine to store only $\frac{m}{k} + \frac{1}{4}\frac{m}{k}$ edges. We randomly split the edges on each machine, and because the subgraphs are small, we are able to compute the optimal matchings on each machine, as well as on the whole hypergraphs. We perform ten runs of each algorithm with different random k-partitioning and take the maximum cardinality obtained. Table 4 shows that none of the algorithms is able to retrieve the optimal matching. This behaviour can be explained

by the loss of information that using parallel machines implies. We see that **Iterated-Sampling**, like in previous experiments, outperforms the other algorithms due to highly sequential design. **HEDCS-Matching** particularly performs worse than the other algorithms, mainly because it fails to construct sufficiently large HEDCSs.

Social Network Communities. We include two larger real-world datasets, orkut-groups and LiveJournal, from the Koblenz Network Collection [43]. We use two hypergraphs that were constructed from these datasets by Shun [52]. Vertices represent individual users, and hyperedges represent communities in the network. Because membership in these communities does not require the same commitment as collaborating on academic research, these hypergraphs have different characteristics from co-citation hypergraphs, in terms of size, vertex degree and hyperedge cardinality. We use the size of a maximal matching as a benchmark. Table 4 shows that **Iterated-Sampling** still provides the best approximation. **HEDCS-Sampling** performs worse than **Greedy** on Livejournal, mainly because the ratio $\frac{m}{n}$ is not big enough to construct an HEDCS with a large matching.

Name	n	m	k	Gr	IS	HEDCS	Rounds IS
Cora	2,708	1,579	2	75.0%	83.2%	63.9%	6
PubMed	19,717	7,963	3	72.0%	86.6%	62.4%	9
Orkut	$2{,}32 \times 10^6$	$1{,}53 \times 10^7$	10	55.6%	66.1%	58.1%	11
Livejournal	$3{,}20 \times 10^6$	$7{,}49 \times 10^6$	3	44.0%	55.2%	43.3%	10

Table 4: Comparison on co-citation and social network hypergraphs.

7.3 Experimental conclusions

In the majority of our experiments, **Iterated-Sampling** provides the best approximation to the d-UHM problem, which is consistent with its theoretical superiority. On random graphs, **HEDCS-Matching** performs consistently better than **Greedy**, even-though **Greedy** has a better theoretical approximation ratio. We suspect it is because the $O(d^3)$-approximation bound on **HEDCS-Matching** is loose. We conjecture that rounding an $\epsilon-$restricted matching can be done efficiently, which would improve the approximation ratio. The performance of the three algorithms decreases as d grows. The results on the number of rounds of **Iterated-Sampling** also are consistent with the theoretical bound. However, due to its sequential design, and by centralizing the computation on one single machine while using the other machines simply to coordinate, **Iterated-Sampling** not only takes more rounds than the other two algorithms, but is also slower when we account for the absolute runtime as well as the runtime per round. We compared the runtimes of our three algorithms on a set of random uniform hypergraphs. Figure 2 and 3 (see Appendix B) show that the absolute and per round run-times of **Iterated-Sampling** grow considerably faster with the size of the hypergraphs. We can also see that **HEDCS-Sampling** is slower than **Greedy**, since the former performs heavier computation on each machine. This confirms the trade-off between the extent to which the algorithms use the power of parallelism and the quality of the approximations.

8 Conclusion

We have presented the first algorithms for the d-UHM problem in the MPC model. Our theoretical and experimental results highlight the trade-off between the approximation ratio, the necessary memory per machine and the number of rounds it takes to run the algorithm. We have also introduced the notion of HEDCS subgraphs, and have shown that an HEDCS contains a good approximation for the maximum matching and that they can be constructed in few rounds in the MPC model. We believe better approximation algorithms should be possible, especially if we can

give better rounding algorithms for a ϵ-restricted fractional hypergraph matching. For future work, it would be interesting to explore whether we can achieve better-than-d approximation in the MPC model in a polylogarithmic number of rounds. Exploring algorithms relying on vertex sampling instead of edge sampling might be a good candidate. In addition, our analysis in this paper is specific to unweighted hypergraphs, and we would like to extend this to weighted hypergraphs.

References

[1] S. Abbar, S. Amer-Yahia, P. Indyk, S. Mahabadi, and K. R. Varadarajan. Diverse near neighbor problem. In *Proceedings of the twenty-ninth annual symposium on Computational geometry*, pages 207–214. ACM, 2013.

[2] K. J. Ahn and S. Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. *ACM Transactions on Parallel Computing (TOPC)*, 4(4):17, 2018.

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

[4] A. Andoni, A. Nikolov, K. Onak, and G. Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 574–583. ACM, 2014.

[5] A. Andoni, Z. Song, C. Stein, Z. Wang, and P. Zhong. Parallel graph connectivity in log diameter rounds. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 674–685. IEEE, 2018.

[6] S. Assadi, M. Bateni, A. Bernstein, V. Mirrokni, and C. Stein. Coresets meet edcs: algorithms for matching and vertex cover on massive graphs. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1616–1635. SIAM, 2019.

[7] 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*, pages 3–12, 2017.

[8] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 671–680. ACM, 2014.

[9] M.-F. F. Balcan, S. Ehrlich, and Y. Liang. Distributed k-means and k-median clustering on general topologies. In *Advances in Neural Information Processing Systems*, pages 1995–2003, 2013.

[10] M. Bateni, A. Bhashkara, S. Lattanzi, and V. Mirrokni. Mapping core-sets for balanced clustering. *Advances in Neural Information Processing Systems*, 26:5–8, 2014.

[11] P. Beame, P. Koutris, and D. Suciu. Communication steps for parallel query processing. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI symposium on Principles of database systems*, pages 273–284. ACM, 2013.

[12] S. Behnezhad, M. Hajiaghayi, and D. G. Harris. Exponentially faster massively parallel maximal matching. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1637–1649. IEEE, 2019.

[13] P. Berman. A d/2 approximation for maximum weight independent set in d-claw free graphs. In *Scandinavian Workshop on Algorithm Theory*, pages 214–219. Springer, 2000.

[14] P. Berman and M. Fürer. Approximating maximum independent set in bounded degree graphs. In *Soda*, volume 94, pages 365–371, 1994.

[15] P. Berman and P. Krysta. Optimizing misdirection. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 192–201. Society for Industrial and Applied Mathematics, 2003.

[16] A. Bernstein and C. Stein. Fully dynamic matching in bipartite graphs. In *International Colloquium on Automata, Languages, and Programming*, pages 167–179. Springer, 2015.

[17] 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*, pages 692–711. Society for Industrial and Applied Mathematics, 2016.

[18] A. Bhaskara, A. Rostamizadeh, J. Altschuler, M. Zadimoghaddam, T. Fu, and V. Mirrokni. Greedy column subset selection: New bounds and distributed algorithms. 2016.

[19] Y. H. Chan and L. C. Lau. On linear and semidefinite programming relaxations for hypergraph matching. *Mathematical programming*, 135(1-2):123–148, 2012.

[20] B. Chandra and M. M. Halldórsson. Greedy local improvement and weighted set packing approximation. *Journal of Algorithms*, 39(2):223–240, 2001.

[21] M. Cygan. Improved approximation for 3-dimensional matching via bounded pathwidth local search. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 509–518. IEEE, 2013.

[22] A. Czumaj, J. Łacki, A. Madry, S. Mitrovic, K. Onak, and P. Sankowski. Round compression for parallel matching algorithms. *SIAM Journal on Computing*, (0):STOC18–1, 2019.

[23] F. Dufossé, K. Kaya, I. Panagiotas, and B. Uçar. Effective heuristics for matchings in hypergraphs. In *International Symposium on Experimental Algorithms*, pages 248–264. Springer, 2019.

[24] M. Fischer, M. Ghaffari, and F. Kuhn. Deterministic distributed edge-coloring via hypergraph maximal matching. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 180–191. IEEE, 2017.

[25] A. Frieze and S. Janson. Perfect matchings in random s-uniform hypergraphs. *Random Structures & Algorithms*, 7(1):41–57, 1995.

[26] M. Furer and H. Yu. Approximate the k-set packing problem by local improvements. *arXiv preprint arXiv:1307.2262*, 2013.

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

[28] M. Ghaffari and J. Uitto. Sparsifying distributed algorithms with ramifications in massively parallel computation and centralized local computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1636–1653. SIAM, 2019.

[29] M. T. Goodrich, N. Sitchinava, and Q. Zhang. Sorting, searching, and simulation in the mapreduce framework. In *International Symposium on Algorithms and Computation*, pages 374–383. Springer, 2011.

[30] M. M. Halldórsson. Approximating discrete collections via local improvements. In *SODA*, volume 95, pages 160–169, 1995.

[31] M. M. Halldórsson. Approximations of weighted independent set and hereditary subset problems. In *Graph Algorithms And Applications 2*, pages 3–18. World Scientific, 2004.

[32] M. M. Halldórsson and J. Radhakrishnan. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. *Algorithmica*, 18(1):145–163, 1997.

[33] D. G. Harris. Distributed approximation algorithms for maximum matching in graphs and hypergraphs. *arXiv preprint arXiv:1807.07645*, 2018.

[34] N. J. Harvey, C. Liaw, and P. Liu. Greedy and local ratio algorithms in the mapreduce model. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*, pages 43–52, 2018.

[35] E. Hazan, S. Safra, and O. Schwartz. On the complexity of approximating k-set packing. *computational complexity*, 15(1):20–39, 2006.

[36] C. A. J. Hurkens and A. Schrijver. On the size of systems of sets every t of which have an sdr, with an application to the worst-case ratio of heuristics for packing problems. *SIAM Journal on Discrete Mathematics*, 2(1):68–72, 1989.

[37] 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*, pages 100–108. ACM, 2014.

[38] A. Israeli and A. Itai. A fast and simple randomized parallel algorithm for maximal matching. *Information Processing Letters*, 22(2):77–80, 1986.

[39] V. Kann. Maximum bounded 3-dimensional matching in max snp-complete. *Inf. Process. Lett.*, 37(1):27–35, 1991.

[40] H. 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*, pages 938–948. SIAM, 2010.

[41] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[42] J. H. Kim. Perfect matchings in random uniform hypergraphs. *Random Structures & Algorithms*, 23(2):111–132, 2003.

[43] J. Kunegis. Konect: the koblenz network collection. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1343–1350, 2013.

[44] S. Lattanzi, B. Moseley, S. Suri, and S. Vassilvitskii. Filtering: a method for solving graph problems in mapreduce. In *Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures*, pages 85–94. ACM, 2011.

[45] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM journal on computing*, 15(4):1036–1053, 1986.

[46] V. Mirrokni and M. Zadimoghaddam. Randomized composable core-sets for distributed submodular maximization. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 153–162. ACM, 2015.

[47] 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*, pages 2049–2057, 2013.

[48] C. H. Papadimitriou. *Computational complexity*. John Wiley and Sons Ltd., 2003.

[49] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial intelligence*, 135(1-2):1–54, 2002.

[50] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1-2):209–238, 1999.

[51] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[52] J. Shun. Practical parallel hypergraph algorithms. In *Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 232–249, 2020.

[53] S. S. Skiena. *The algorithm design manual: Text*, volume 1. Springer Science & Business Media, 1998.

[54] M. Sviridenko and J. Ward. Large neighborhood local search for the maximum set packing problem. In *International Colloquium on Automata, Languages, and Programming*, pages 792–803. Springer, 2013.

[55] L. Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 453–461, 2001.

[56] R. Vemuganti. Applications of set covering, set packing and set partitioning models: A survey. In *Handbook of combinatorial optimization*, pages 573–746. Springer, 1998.

[57] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015.

A Omitted proofs

A.1 Proof of Lemma 4.3 and Theorem 4.2

Lemma 4.3. *For any $i \in [k/3]$, if $M^{(i-1)} \leq c \cdot \mu(G)$, then, w.p. $1 - O(1/n)$,*

$$M^{(i)} \geq M^{(i-1)} + \frac{1 - 3d(d-1)c - o(1)}{k} \cdot \mu(G) \ .$$

To continue we define some notation. Let M^* be an arbitrary maximum matching of G. For any $i \in [k]$, we define $M^{*<i}$ as the part of M^* assigned to the first $i-1$ graphs in the random k-partitioning, i.e., the graphs $G^{(1)}, \ldots G^{(i-1)}$. We have the following concentration result:

Claim A.1. *W.p. $1 - O(1/n)$, for any $i \in [k]$:*

$$M^{*<i} \leq \Big(\frac{i - 1 + o(i)}{k}\Big) \cdot \mu(G)$$

Proof. [of claim A.1] Fix an $i \in [k]$; each edge in M^* is assigned to $G^{(1)}, \ldots, G^{(i-1)}$, w.p. $(i-1)/k$, hence in expectation, size of $M^{*<i}$ is $\frac{i-1}{k} \cdot \mu(G)$. The claim now follows from a standard application of Chernoff bound. □

Proof. [of Lemma 4.3] Fix an $i \in [k/3]$ and the set of edges for $E^{(1)}, \ldots E^{(i-1)}$; this also fixes the matching $M^{(i-1)}$ while the set of edges in $E^{(i)}, \ldots, E^{(k)}$ together with the matching $M^{(i)}$ are still random variables. We further condition on the event that after fixing the edges in $E^{(1)}, \ldots, E^{(i-1)}, |M^{*<i}| \leq \frac{i-1+o(i)}{k} \cdot MM(G)$ which happens w.p. $1 - O(1/n)$ by claim A.1.

Let V_{old} be the set of vertices incident on $M^{(i-1)}$ and V_{new} be the remaining vertices. Let $E^{\geq i}$ be the set of edges in $E \setminus E^{(1)} \cup \ldots \cup E^{(i-1)}$. We partition $E^{\geq i}$ into two parts: (i) E_{old}: the set of edges with *at least one* endpoint in V_{old}, and (ii) E_{new}: the set of edges incident entirely on V_{new}. Our goal is to show that w.h.p. any maximum matching of $G^{(i)}$ matches $\Omega(\mu(G)/k)$ vertices in V_{new} to each other by using the edges in E_{new}; the lemma then follows easily from this.

The edges in the graph $G^{(i)}$ are chosen by independently assigning each edge in $E^{\geq i}$ to $G^{(i)}$ w.p. $1/(k-i+1)$. This independence makes it possible to treat the edges in E_{old} and E_{new} separately; we can fix the set of sampled edges of $G^{(i)}$ in E_{old} denoted by E^i_{old} without changing the distribution of edges in $G^{(i)}$ chosen from E_{new}. Let $\mu_{old} := MM(G(V, E^i_{old}))$, i.e., the maximum number of edges that can be matched in $G^{(i)}$ using only the edges in E^i_{old}. In the following, we show that w.h.p., there exists a matching of size $\mu_{old} + \Omega(\mu(G)/k)$ in $G^{(i)}$; by the definition of μ_{old}, this implies that any maximum matching of $G^{(i)}$ has to use at least $\Omega(\mu(G)/k)$ edges in E_{new}, proving the lemma.

Let M_{old} be any arbitrary maximum matching of size μ_{old} in $G(V, E^i_{old})$. Let $V_{new}(M_{old})$ be the set of vertices in V_{new} that are incident on M_{old}. We show that there is a large matching in $G(V, E_{new})$ that avoids $V_{new}(M_{old})$.

Claim A.2. $|V_{new}(M_{old})| < c \cdot d(d-1) \cdot \mu(G)$.

Proof. Since any edge in M_{old} has at least one endpoint in V_{old}, we have $|V_{new}(M_{old})| \leq (d-1)|M_{old}| \leq (d-1)|V_{old}|$. By the assertion of the lemma, $|M^{(i-1)}| < c \cdot \mu(G)$, and hence $V_{new}(M_{old})| \leq (d-1) \cdot |V_{old}| \leq d(d-1) \cdot |M^{(i-1)}| < c \cdot d(d-1) \cdot \mu(G)$. □

Claim A.3. *There exists a matching in $G(V, E_{new})$ of size $\left(\frac{k-i+1-o(i)}{k} - 2d(d-1)c\right) \cdot \mu(G)$ that avoids the vertices of $V_{new}(M_{old})$.*

Proof. By the assumption that $|M^{*<i}| \leq \frac{i-1+o(i)}{k} \cdot \mu(G)$, there is a matching of size $\frac{k-i+1-o(i)}{k} \cdot \mu(G)$ in the graph $G(V, E^{\geq i})$. By removing the edges in M that are either incident on V_{old} or $V_{new}(M_{old})$, at most $2d(d-1)c \cdot \mu(G)$ edges are removed from M. Now the remaining matching is entirely contained in E_{new} and also avoids $V_{new(Mold)}$, hence proving the claim. □

We are now ready to finalize the proof. Let M_{new} be the matching guaranteed by Claim A.3. Each edge in this matching is chosen in $G^{(i)}$ w.p. $1/(k-i+1)$ independent of the other edges; hence, by Chernoff bound, there is a matching of size

$$(1-o(1)) \cdot \left(\frac{1}{k} - \frac{o(i)}{k(k-i+1)} - \frac{2d(d-1)c}{k-i+1}\right) \cdot MM(G) \geq \frac{1-o(1)-3d(d-1)c}{k} \cdot \mu(G) \quad (i \leq k/3)$$

in the edges of M_{new} that appear in $G^{(i)}$. This matching can be directly added to the matching M_{old}, implying the existence of a matching of size $\mu_{old} + \frac{1-o(1)-3d(d-1)c}{k} \cdot \mu(G)$ in $G^{(i)}$. As we argued before, this ensures that any maximum matching of $G^{(i)}$ contains at least $\frac{1-o(1)-3d(d-1)c}{k} \cdot MM(G)$ edges in E_{new}. These edges can always be added to $M^{(i-1)}$ to form $M^{(i)}$, hence proving the lemma. □

Proof. [of Theorem 4.2] Recall that $M := M^{(k)}$ is the output matching of *Greedy*. For the first $k/3$ steps of *Greedy*, if at any step we got a matching of size at least $c \cdot \mu(G)$, then we are already done. Otherwise, at each step, by Lemma 4.3, w.p. $1 - O(1/n)$, we increase the size of the maximal matching by $\frac{1-3d(d-1)c-o(1)}{k} \cdot \mu(G)$ edges; consequently, by taking a union bound on the $k/3$ steps, w.p. $1 - o(1)$, the size of the maximal matching would be $\frac{1-3d(d-1)c-o(1)}{3} \cdot \mu(G)$. Since $c = 1/(3d(d-1)+3)$, we ensure that $\frac{1-3d(d-1)c}{3} = c$ in either case, the matching computed by *Greedy* is of size at least $\mu(G)/(3d(d-1)+3) - o(\mu(G))$, and this proves that *Greedy* is a $O(d^2)$-approximation.

All is left know is to prove that the *Greedy* Algorithm can be implemented in 3 rounds with a memory of $\tilde{O}(\sqrt{nm})$ per machine. Let $k = \sqrt{\frac{m}{n}}$ be the number of machines, each with a memory of $\tilde{O}(\sqrt{nm})$. We claim that *Greedy* can run in three rounds. In the first round, each machine randomly partitions the edges assigned to it across the k machines; this results in a random k-partitioning of the graph across the machines. In the second round, each machine sends a randomized composable coreset of its input to a designated central machine M; as there are $k = \sqrt{\frac{m}{n}}$ machines and each machine is sending $\tilde{O}(n)$ size coreset, the input received by M is of size $\tilde{O}(\sqrt{nm})$ and hence can be stored entirely on that machine. Finally, M computes the answer by combining the coresets. □

A.2 Proof of Theorem 5.1

Theorem 5.1. *Given a d-uniform hypergraph $G(V, E)$, Iterated-Sampling omputes a maximal matching in G with high probability in $O(\log n)$ MPC rounds on machines of memory $s = \Theta(d \cdot n)$.*

Proof. [of Theorem 5.1] To prove the theorem, we will need the three following lemmas:

Lemma A.1. *Let $E' \subset E$ be a set of edges chosen independently with probability p. Then with probability at least $1 - e^{-n}$, for all $I \subset V$ either $|E[I]| < 2n/p$ or $E[I] \cap E' \neq \emptyset$.*

Proof. [of Lemma A.4] Fix one such subgraph, $G[I] = (I, E[I])$ with $|E[I]| \geq 2n/p$. The probability that none of the edges in $E[I]$ were chosen to be in E' is $(1-p)^{|E[I]|} \leq (1-p)^{2n/p} \leq e^{-2n}$. Since there are at most 2^n total possible induced subgraphs $G[I]$ (because each vertex is either matched or unmatched), the probability that there exists one that does not have an edge in E' is at most $2^n e^{-2n} \leq e^{-n}$. □

Lemma A.2. *If $s \geq 20d \cdot n$ then **Iterated-Sampling** runs for at most $O(\log n)$ iterations with high probability.*

Proof. [of Lemma A.5] Fix an iteration i of **Iterated-Sampling** and let p be the sampling probability for this iteration. Let E_i be the set of edges at the beginning of this iteration, and denote by I be the set of unmatched vertices after this iteration. From Lemma A.4, if $|E[I]| \geq 2n/p$ then an edge of $E[I]$ will be sampled with high probability. Note that no edge in $E[I]$ is incident on any edge in M' . Thus, if an edge from $E[I]$ is sampled then **Iterated-Sampling** would have chosen this edge to be in the matching. This contradicts the fact that no vertex in I is matched. Hence, $|E[I]| \leq 2n/p$ with high probability.

Now consider the first iteration of the algorithm, let $G_1(V_1, E_1)$ be the induced graph on the unmatched nodes after the first step of the algorithm. The above argument implies that $|E_1| \leq \frac{10d \cdot n|E_0|}{s} \leq \frac{10d \cdot n|E|}{s} \leq \frac{|E|}{2}$. Similarly $|E_2| \leq \frac{10d \cdot n|E_1|}{s} \leq \frac{(10d \cdot n)^2|E_1|}{s^2} \leq \frac{|E|}{2^2}$. After i iterations : $|E_i| \leq \frac{|E|}{2^i}$, and the algorithm will terminate after $O(\log n)$ iterations. □

Lemma A.3. ***Iterated-Sampling** finds a maximal matching of G with high probability.*

Proof. [of Lemma A.6] First consider the case that the algorithm does not fail. Suppose there is an edge $e = \{v_1, \ldots, v_d\} \in E$ such that none of the v_i's are matched in the final matching M that the algorithm output. In the last iteration of the algorithm, since $e \in E$ and its endpoints are not matched, then $e \in E[I]$. Since this is the last run of the algorithm, a maximal matching M" of $G[I]$ is computed on one machine. Since M" is maximal, at least one of the v_i's must be matched in it. All of the edges of M'' get added to M in the last step. This yields a contradiction.

Next, consider the case that the algorithm fails. This occurs due to the set of edges E' having size larger than the memory in some iteration of the algorithm. Note that $\mathbb{E}[|E'|] = |S| \cdot p = s/5d \leq s/10$ in a given iteration. By the Chernoff Bound it follows that $|E'| \geq s$ with probability smaller than $2^s \geq 2^{-20d \cdot n}$ (since $s \geq 20d \cdot n$). By Lemma A.5 the algorithm completes in at most $O(\log n)$ rounds, thus the total failure probability is bounded by $O(\log n \cdot 2^{-20 \cdot n})$ using the union bound. □

Next we prove that **Iterated-Sampling** can be implemented in the MPC model with machines of memory $\Theta(dn)$ and $O(\log n)$ MPC rounds. Combining this with Lemma A.6 on the correctness of **Iterated-Sampling** , we immediately obtain Theorem 5.1 for the case of $s = \Theta(dn)$. Every iteration of **Iterated-Sampling** can be implemented in $O(1)$ MPC rounds. Suppose the edges are initially distributed over all machines. We can sample every edge with the probability p (in each machine) and then send the sampled edges E' to the first machine. With high probability we will have $|E'| = O(n)$, so we can fit the sampled edges in the first machine. Therefore, the sampling can be done in one MPC round. Computing the subgraph of G induced by I can be done in 2 rounds; one round to send the list of unmatched vertices I from the first machine to all the other machines, and the other round to compute the subgraph $G[I]$, and send it to a single machine if it fits, or start the sampling again. □

A.3 Proof of Theorem 5.1

Theorem 5.1. *Given a d-uniform hypergraph $G(V, E)$, Iterated-Sampling omputes a maximal matching in G with high probability in $O(\log n)$ MPC rounds on machines of memory $s = \Theta(d \cdot n)$.*

Proof. [of Theorem 5.1] To prove the theorem, we will need the three following lemmas:

Lemma A.4. *Let $E' \subset E$ be a set of edges chosen independently with probability p. Then with probability at least $1 - e^{-n}$, for all $I \subset V$ either $|E[I]| < 2n/p$ or $E[I] \cap E' \neq \emptyset$.*

Proof. [of Lemma A.4] Fix one such subgraph, $G[I] = (I, E[I])$ with $|E[I]| \geq 2n/p$. The probability that none of the edges in $E[I]$ were chosen to be in E' is $(1-p)^{|E[I]|} \leq (1-p)^{2n/p} \leq e^{-2n}$. Since there are at most 2^n total possible induced subgraphs $G[I]$ (because each vertex is either matched or unmatched), the probability that there exists one that does not have an edge in E' is at most $2^n e^{-2n} \leq e^{-n}$. □

Lemma A.5. *If $s \geq 20d \cdot n$ then **Iterated-Sampling** runs for at most $O(\log n)$ iterations with high probability.*

Proof. [of Lemma A.5] Fix an iteration i of **Iterated-Sampling** and let p be the sampling probability for this iteration. Let E_i be the set of edges at the beginning of this iteration, and denote by I be the set of unmatched vertices after this iteration. From Lemma A.4, if $|E[I]| \geq 2n/p$ then an edge of $E[I]$ will be sampled with high probability. Note that no edge in $E[I]$ is incident on any edge in M' . Thus, if an edge from $E[I]$ is sampled then **Iterated-Sampling** would have chosen this edge to be in the matching. This contradicts the fact that no vertex in I is matched. Hence, $|E[I]| \leq 2n/p$ with high probability.

Now consider the first iteration of the algorithm, let $G_1(V_1, E_1)$ be the induced graph on the unmatched nodes after the first step of the algorithm. The above argument implies that $|E_1| \leq \frac{10d \cdot n|E_0|}{s} \leq \frac{10d \cdot n|E|}{s} \leq \frac{|E|}{2}$. Similarly $|E_2| \leq \frac{10d \cdot n|E_1|}{s} \leq \frac{(10d \cdot n)^2|E_1|}{s^2} \leq \frac{|E|}{2^2}$. After i iterations : $|E_i| \leq \frac{|E|}{2^i}$, and the algorithm will terminate after $O(\log n)$ iterations. □

Lemma A.6. ***Iterated-Sampling** finds a maximal matching of G with high probability.*

Proof. [of Lemma A.6] First consider the case that the algorithm does not fail. Suppose there is an edge $e = \{v_1, \ldots, v_d\} \in E$ such that none of the v_i's are matched in the final matching M that the algorithm output. In the last iteration of the algorithm, since $e \in E$ and its endpoints are not matched, then $e \in E[I]$. Since this is the last run of the algorithm, a maximal matching M" of $G[I]$ is computed on one machine. Since M" is maximal, at least one of the v_i's must be matched in it. All of the edges of M'' get added to M in the last step. This yields a contradiction.

Next, consider the case that the algorithm fails. This occurs due to the set of edges E' having size larger than the memory in some iteration of the algorithm. Note that $\mathbb{E}[|E'|] = |S| \cdot p = s/5d \leq s/10$ in a given iteration. By the Chernoff Bound it follows that $|E'| \geq s$ with probability smaller than $2^s \geq 2^{-20d \cdot n}$ (since $s \geq 20d \cdot n$). By Lemma A.5 the algorithm completes in at most $O(\log n)$ rounds, thus the total failure probability is bounded by $O(\log n \cdot 2^{-20 \cdot n})$ using the union bound. □

Next we prove that **Iterated-Sampling** can be implemented in the MPC model with machines of memory $\Theta(dn)$ and $O(\log n)$ MPC rounds. Combining this with Lemma A.6 on the correctness of **Iterated-Sampling** , we immediately obtain Theorem 5.1 for the case of $s = \Theta(dn)$.

Every iteration of **Iterated-Sampling** can be implemented in $O(1)$ MPC rounds. Suppose the edges are initially distributed over all machines. We can sample every edge with the probability p

(in each machine) and then send the sampled edges E' to the first machine. With high probability we will have $|E'| = O(n)$, so we can fit the sampled edges in the first machine. Therefore, the sampling can be done in one MPC round. Computing the subgraph of G induced by I can be done in 2 rounds; one round to send the list of unmatched vertices I from the first machine to all the other machines, and the other round to compute the subgraph $G[I]$, and send it to a single machine if it fits, or start the sampling again. □

A.4 Proof of Lemma 6.3

Lemma 6.3. *Any hypergraph G contains an $HEDCS(G, \beta, \beta^-)$ for any parameters $\beta - \beta^- \geq d - 1$.*

Proof. Consider the following simple procedure for creating an HEDCS H of a given hypergraph G: start by initializing H to be equal to G. And while H is not an $HEDCS(G, \beta, \beta^-)$, find an edge e which violates one of the properties of $HEDCS$ and fix it. Fixing the edge e implies removing it from H if it was violating Property *(P1)* and adding it to H if it was violating Property *(P2)*. The output of the above procedure is clearly a $HEDCS$ 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 $HEDCS$ 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 a $HEDCS(G, \beta, \beta - 1)$ always exists.

We define the following potential function Φ:

$$\Phi := (\frac{2}{d}\beta - \frac{d-1}{d}) \cdot \sum_{v \in V} d_H(v) - \sum_{e \in H} \sum_{u \in e} d_H(u)$$

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 $O(\frac{2}{d}n \cdot \beta^2)$, this immediately implies that this procedure terminates in $O(\frac{2}{d}n \cdot \beta^2)$ iterations.

Define $\Phi_1 = (\frac{2}{d}\beta - \frac{d-1}{d}) \cdot \sum_{v \in V} d_H(v)$ and $\Phi_2 = \sum_{e \in H} \sum_{u \in e} d_H(u)$. Let e be the edge we choose to fix at this step, H_b be the subgraph before fixing the edge e, and H_a be the resulting subgraph. Suppose first that the edge e was violating Property *(P1)* of *HEDCS*. As the only change is in the degrees of vertices $v \in e$, Φ_1 decreases by $(2\beta - (d-1))$. On the other hand, $\sum_{v \in e} d_{H_b}(v) \geq \beta + 1$ originally (as e was violating Property (P1) of HEDCS), and hence after removing e, Φ_2 increases by $\beta + 1$. Additionally, for each edge e_u incident upon $u \in e$ in H_a , after removing the edge e, $\sum_{v \in e_u} d_{H_a}(v)$ decreases by one. As there are at least $\sum_{u \in e} d_{H_a}(u) = \sum_{u \in e} d_{H_b}(u) - d \geq \beta - (d-1)$ choices for e_u, this means that in total, Φ_2 increases by at least $2\beta + 1 - (d-1)$. As a result, in this case Φ increases by at least 1 after fixing the edge e.

Now suppose that the edge e was violating Property *(P2)* of HEDCS instead. In this case, degree of vertices $u \in e$ all increase by one, hence Φ_1 increases by $2\beta - (d-1)$. Additionally, note that since edge e was violating Property (P2) we have $\sum_{v \in e} d_{H_b}(v) \leq \beta^- - 1$, so the addition of edge e decreases Φ_2 by at most $\sum_{v \in e} d_{H_a}(v) = \sum_{v \in e} d_{H_b}(v) + d \leq \beta^- - 1 + d$. Moreover, for each edge e_u incident upon $u \in e$, after adding the edge e, $\sum_{v \in e_u} d_{H_a}(v)$ increases by one and since there are at most $\sum_{v \in e} d_{H_b}(v) \leq \beta^- - 1$ choices for e_u, Φ_2 decreases in total by at most $2\beta^- - 2 + d$. The

total variation in Φ is therefore equal to $2\beta - (d-1) - (2\beta^- - 2 + d) = 3 + 2(\beta - \beta^- - d)$. So if $\beta - \beta^- \geq d - 1$, we have that Φ increases by at least 1 after fixing edge e. □

A.5 Proof of Theorem 6.4

In order to prove Theorem 6.4, we will need the following two lemmas. The first lemma we prove is an algebraic result that will help us bound the contribution of vertices in the ϵ-restricted fractional matching. The second lemma identifies additional structure on the HEDCS, that we will use to construct the fractional matching of the theorem. For proofs of both lemmas, see Appendix A.7

Lemma A.7. *Let* $\phi(x) = \min\{1, \frac{(d-1)x}{d(\beta-x)}\}$. *If* $a_1, \ldots a_d \geq 0$ *and* $a_1 + \ldots + a_d \geq \beta(1-\lambda)$ *for some* $\lambda \geq 0$, *then* $\sum_{i=1}^{d} \phi(a_i) \geq 1 - 5 \cdot \lambda$.

Lemma A.8. *Given any* $HEDCS(G, \beta, \beta(1-\lambda))$ H, *we can find two disjoint sets of vertices* X *and* Y *that satisfy the following properties:*

1. $|X| + |Y| = d \cdot \mu(G)$.
2. *There is a perfect matching in* Y *using edges in* H.
3. *Letting* $\sigma = \frac{|Y|}{d} + \sum_{x \in X} \phi(d_H(x))$,*we have that* $\sigma \geq \mu(G)(1 - 5\lambda)$.
4. *All edges in* H *with vertices in* X *have at least one other vertex in* Y, *and have vertices only in* X *and* Y.

Proof. [of theorem 6.4] Suppose we have two sets X and Y satisfying the properties of Lemma A.8. We construct an ϵ-restricted fractional matching M_f^H using the edges in H such that $val(M_f^H) \geq (\frac{d}{d^2-d+1} - \frac{\epsilon}{d-1})\mu(G)$, where $val(M_f^H)$ is the value of the fractional matching M_f^H. Now, by Property 2 of Lemma A.8, $|Y|$ contains a perfect matching M_Y^H using edges in H. Let Y^- be a subset of Y obtained by randomly sampling exactly $1/d$ edges of M_Y^H and adding their endpoints to Y^- Let $Y^* = Y \setminus Y^-$ and observe that $|Y^-| = |Y|/d$ and $|Y^*| = \frac{d-1}{d}|Y|$.

Let H^* be the subgraph of H induced by $X \cup Y^*$ (each edge in H^* has vertices in only X and Y^*). We define a fractional matching $M_f^{H^*}$ on the edges of H^* in which all edges have value at most ϵ. We will then let our final fractional matching M_f^H be the fractional matching $M_f^{H^*}$ joined with the perfect matching in H of Y^- (so M_f^H assigns value 1 to the edges in this perfect matching). M_f^H is, by definition, a ϵ-restricted fractional matching.

We now give the details for the construction of $M_f^{H^*}$. Let $V^* = X \cup Y^*$ be the vertices of H^*, and let E^* be its edges. For any vertex $v \in V^*$, define $d_H^*(v)$ to be the degree of v in H^*. Recall that by Property 4 of Lemma A.8, if $x \in X$ then all the edges of H incident to x go to Y (but some might go to Y^-); thus, for $x \in X$, we have $E[d_H^*(x)] \geq \frac{d_H(x)(d-1)}{d}$.

We now define $M_f^{H^*}$ as follows. For every $x \in X$, we arbitrarily order the edges of H incident to x, and then we assign/add a value of $\min\left\{\frac{\epsilon}{|X \cap e|}, \frac{1}{|X \cap e|}\frac{1}{\beta - d_H(x)}\right\}$ to the edges one by one, stopping when either $val(x)$ reaches 1 or there are no more edges in H incident to x, whichever comes first. In the case that $val(x)$ reaches 1 the last edge might have added value less than $\min\left\{\frac{\epsilon}{|X \cap e|}, \frac{1}{|X \cap e|}\frac{1}{\beta - d_H(x)}\right\}$, where e is the last edge to be considered.

We now verify that $M_f^{H^*}$ is a valid fractional matching in that all vertices have value at most 1. This is clearly true of vertices $x \in X$ by construction. For a vertex $y \in Y^*$, it suffices to show that each edge incident to y receives a value of at most $1/d_H(y) \leq 1/d_H^*(y)$. To see this, first note that the only edges to which $M_f^{H^*}$ assigns non-zero values have at least two endpoints in $X \times Y^*$. Any such edge e receives value at most $\min\left\{\epsilon, \sum_{x \in X \cap e} \frac{1}{|X \cap e|} \frac{1}{\beta - d_H(x)}\right\}$, but since e is in $M_f^{H^*}$ and so in H, we have by Property *P1* of an *HEDCS* that $d_H(y) \leq \beta - d_H(x)$, and so $\sum_{x \in X \cap e} \frac{1}{|X \cap e|} \frac{1}{\beta - d_H(x)} \leq \frac{1}{|X \cap e|} \frac{|X \cap e|}{d_H(y)} \leq \frac{1}{d_H(y)}$.

By construction, for any $x \in X$, we have that the value $val(x)$ of x in $M_f^{H^*}$ satisfies :

$$\begin{aligned} val(x) &= \min\left\{1, \sum_{x \in e} \min\left\{\frac{\epsilon}{|X \cap e|}, \frac{1}{|X \cap e|} \frac{1}{\beta - d_H(x)}\right\}\right\} \\ &\geq \min\left\{1, d_H^*(x) \cdot \min\left\{\frac{\epsilon}{d-1}, \frac{1}{d-1} \frac{1}{\beta - d_H(x)}\right\}\right\} . \end{aligned}$$

Furthermore, we can bound the value of the fractional matching M_f^{H*}: as $val(M_f^{H*}) \geq \sum_{x \in X} val(x)$. For convenience, we use $val'(x) = \min\left\{1, d_H^*(x) \cdot \min\left\{\epsilon, \frac{1}{\beta - d_H(x)}\right\}\right\}$ such that

$$val(x) \geq \frac{val'(x)}{d-1} \text{ and} \tag{1}$$

$$val(M_f^{H*}) \geq \frac{1}{d-1} \sum_{x \in X} val'(x) , \tag{2}$$

Next we present a lemma, proved in Appendix A.7 that bounds $val'(x)$ for each vertex.

Lemma A.9. *For any $x \in X$, $E[val'(x)] \geq (1-\lambda)\phi(d_H(x))$.*

This last lemma, combined with (2), allows us to lower bound the value of $M_f^{H^*}$:

$$val(M_f^{H^*}) \geq \frac{1}{d-1} \sum_{x \in X} val'(x) \geq \frac{1-\lambda}{d-1} \sum_{x \in X} \phi(d_H(x)).$$

Note that we have constructed M_f^H by taking the fractional value in $M_f^{H^*}$ and adding the perfect matching on edges from Y^-. The latter matching has size $\frac{|Y^-|}{d} = \frac{|Y|}{d^2}$, and the value of $M_f^{H^*}$ is bounded by:

$$\begin{aligned} val(M_f^H) &\geq \frac{1}{d-1}(1-\lambda) \sum_{x \in X} \phi(d_H(x)) + \frac{|Y|}{d^2} \\ &= \frac{1}{d-1}\big((1-\lambda) \sum_{x \in X} \phi(d_H(x)) + \frac{|Y|}{d}\big) - \frac{|Y|}{d^2(d-1)} \\ &\geq \frac{1}{d-1}(1-\lambda)(1-5\lambda)\mu(G) - \frac{|Y|}{d^2(d-1)} \\ &\geq \frac{1}{d-1}(1-6\lambda)\mu(G) - \frac{|Y|}{d^2(d-1)} . \end{aligned}$$

To complete the proof, recall that Y contains a perfect matching in H of $|Y|/d$ edges, so if $\frac{|Y|}{d} \geq \frac{d}{d(d-1)+1}\mu(G)$ then there already exists a matching in H of size at least $\frac{d}{d(d-1)+1}\mu(G)$, so the main lemma we are trying to prove is trivially true. We can thus assume that $|Y|/d < \big(\frac{d}{d(d-1)+1}\big)\mu(G)$, in which case the previous equation yields that:

$$\begin{aligned} val(M_f^H) &\geq \frac{1}{d-1}(1-6\lambda)\mu(G) - \frac{|Y|}{d^2(d-1)} \\ &\geq (\frac{1-6\lambda}{d-1})\mu(G) - \frac{\mu(G)}{(d-1)(d(d-1)+1)} \\ &= \big(\frac{d}{d(d-1)+1} - \frac{6\lambda}{d-1}\big)\mu(G) \ . \end{aligned}$$

In both cases we get that

$$val(M_f^H) \geq (\frac{d}{d^2-d+1} - \frac{6\lambda}{d-1})\mu(G).$$

□

A.6 Proof of Lemma A.9

Lemma A.9. *For any $x \in X$, $E[val'(x)] \geq (1-\lambda)\phi(d_H(x))$.*

Proof.

- Consider the case in which $d_H(x) \leq \frac{\beta}{d}$, in this case $\frac{1}{\beta - d_H(x)} \leq \frac{d}{(d-1)\beta} < \epsilon$ and $d_{H^*}(x) \leq d_H(x) \leq \beta - d_H(x)$. This implies that $val'(x) = \frac{d_{H^*}(x)}{\beta - d_H(x)}$, so that :

$$E[val'(x)] \geq \frac{d-1}{d} \cdot \frac{d_H(x)}{\beta - d_H(x)} = \phi(d_H(x)) \ .$$

  Now consider the case in which $d_H(x) > \frac{\beta}{d}$. Then

$$E[d_{H^*}(x)] \geq \frac{(d-1)d_H(x)}{d} > \frac{(d-1)}{d^2} \cdot \beta \geq 8 \cdot \lambda^{-3} >> \epsilon^{-1} \ ,$$

  because $\beta \geq \frac{8d^2}{d-1} \cdot \lambda^{-3}$, and by a Chernoff Bound :

$$\begin{aligned} P\Big[d_{H^*}(x) < (1-\frac{\lambda}{2})\frac{(d-1)d_H(x)}{d}\Big] &\leq \exp\left(-E[d_{H^*}(x)](\frac{\lambda}{2})^2\frac{1}{2}\right) \\ &\leq \exp\left(-\lambda^{-1}\right) \\ &\leq \lambda/2 \ . \end{aligned} \tag{3}$$

- Let us now consider the case $d_H(x) > \frac{\beta}{d}$ and $\min\Big\{\epsilon, \frac{1}{\beta - d_H(x)}\Big\} = \epsilon$. With probability at least $(1-\frac{\lambda}{2})$, we have that:

$$d_{H^*}(x) \geq (\beta - \frac{1}{\epsilon})(1-\frac{\lambda}{2})\frac{d-1}{d} >> \epsilon^{-1}.$$

  Thus with probability at least $(1-\frac{\lambda}{2})$, we have that $d_{H^*}(x)\epsilon > 1$ and:

$$E[val'(x)] \geq (1-\frac{\lambda}{2}) \geq (1-\frac{\lambda}{2})\phi(x).$$

- The only case that we need to check is $d_H(x) \geq \frac{\beta}{d}$ and $\min\left\{\epsilon, \frac{1}{\beta - d_H(x)}\right\} = \frac{1}{\beta - d_H(x)}$, so that $val'(x) = \min\left\{1, \frac{d_{H^*}(x)}{\beta - d_H(x)}\right\}$. Again we have that with probability at least $(1 - \frac{\lambda}{2})$:

$$\frac{d_{H^*}(x)}{\beta - d_H(x)} \geq \frac{d-1}{d}\frac{d_H(x)}{\beta - d_H(x)}(1 - \frac{\lambda}{2}) \geq (1 - \frac{\lambda}{2})\phi(d_H(x)).$$

In other words, with probability at least $(1 - \frac{\lambda}{2})$, we have $val(x) \geq (1 - \frac{\lambda}{2})\phi(d_H(x))$, so that $E[val(x)] \geq (1 - \frac{\lambda}{2})^2\phi(d_H(x)) > (1 - \lambda)\phi(d_H(x))$. We just showed that in all cases $E[val'(x)] \geq (1 - \lambda)\phi(d_H(x))$ □

A.7 Proof of Lemma A.7 and Lemma A.8

Lemma A.7. *Let* $\phi(x) = \min\{1, \frac{(d-1)x}{d(\beta - x)}\}$. *If* $a_1, \ldots a_d \geq 0$ *and* $a_1 + \ldots + a_d \geq \beta(1 - \lambda)$ *for some* $\lambda \geq 0$, *then* $\sum_{i=1}^{d} \phi(a_i) \geq 1 - 5 \cdot \lambda$.

Proof. We will provide a proof for $d = 3$ that is easy to generalize. We first show that if $a + b + c \geq \beta$, then $\phi(a) + \phi(b) + \phi(c) \geq 1$. The claim is true if $\phi(a) \geq 1$ or $\phi(b) \geq 1$ or $\phi(c) \geq 1$. Suppose that $\phi(a) < 1$ and $\phi(b) < 1$ and $\phi(c) < 1$. Then :

$$\phi(a) + \phi(b) + \phi(c) = \frac{d-1}{d}\Big(\frac{a}{\beta - a} + \frac{b}{\beta - b} + \frac{c}{\beta - c}\Big) \geq \frac{d-1}{d}\Big(\frac{a}{b+c} + \frac{b}{a+c} + \frac{c}{a+b}\Big) .$$

By Nesbitt's Inequality we know that

$$\frac{a}{b+c} + \frac{b}{a+c} + \frac{c}{a+b} \geq \frac{d}{d-1} = \frac{3}{2},$$

and therefore $\phi(a) + \phi(b) + \phi(c) \geq 1$.

By the general Nesbitt's Inequality (See appendix D), we know that for $d > 3$

$$\sum_{i=1}^{d} \frac{a_i}{\sum_{j \neq i} a_j} \geq \frac{d}{d-1} .$$

So if $\sum_{i=1}^{d} a_i \geq \beta$, then $\sum_{i=1}^{d} \phi(a_i) \geq 1$. Now, let $\phi'(x) = \frac{d}{dx}\phi(x)$. To complete the proof, it is sufficient to show that we always have $\phi'(x) \leq \frac{5}{\beta}$. To prove this inequality, note that if $x \geq \frac{d}{2d-1}\beta$ then $\phi(x) = 1$ and thus $\phi'(x) = 0$. Now, if $x \leq \frac{d}{2d-1}\beta$ then:

$$\phi'(x) = \frac{d-1}{d}\frac{d}{dx}\frac{x}{\beta - x} = \frac{d-1}{d}\frac{\beta}{(\beta - x)^2} ,$$

which is increasing in x and maximized at $x = \frac{d}{2d-1}\beta$, in which case $\phi'(x) = \frac{(2d-1)^2}{d(d-1)}\frac{1}{\beta} \leq \frac{5}{\beta}$. In the end we get:

$$\sum_{i=1}^{d} \phi(a_i) \geq 1 - 5 \cdot \lambda .$$

□

Lemma A.8. *Given any $HEDCS(G, \beta, \beta(1-\lambda))$ H, we can find two disjoint sets of vertices X and Y that satisfy the following properties:*

1. $|X| + |Y| = d \cdot \mu(G)$.
2. *There is a perfect matching in Y using edges in H.*
3. *Letting $\sigma = \frac{|Y|}{d} + \sum_{x \in X} \phi(d_H(x))$,we have that $\sigma \geq \mu(G)(1-5\lambda)$.*
4. *All edges in H with vertices in X have at least one other vertex in Y, and have vertices only in X and Y.*

Proof.

Let M^G be some maximum integral matching in G. Some of the edges in M^G are in H, while others are in $G \setminus H$. Let X_0 contain all vertices incident to edges in $M_G \cap (G \setminus H)$, and let Y_0 contain all vertices incident to edges in $M_G \cap H$. We now show that X_0 and Y_0 satisfy the first three properties of the lemma. Property 1 is satisfied because $X_0 \cup Y_0$ consists of all matched vertices in M_G. Property 2 is satisfied by definition of Y_0. To see that Property 3 is satisfied, remark that the vertices of Y_0 each contribute exactly $\frac{1}{d}$. Now, X_0 consists of $|X_0|/d$ disjoint edge in $G \setminus H$, and by Property P2 of a HEDCS, for each such edge e : $\sum_{x \in e} d_H(x) \geq \beta(1-\lambda)$ and by Lemma A.7, we have $\sum_{x \in e} \phi(d_H(x)) \geq (1-5\lambda)$ and each one of these vertices contributes in average at least $\frac{1-5\lambda}{d}$ to σ, just as desired. Loosely speaking, $\phi(d_H(x))$ will end up corresponding to the profit gained by vertex x in the fractional matching M_f^H.

Consider Y_0 and X_0 from above. These sets might not satisfy the Property 4 (that all edges in H with an endpoint in X have at least one other endpoint in Y). Can we transform these into sets X_1 and Y_1, such that the first three properties still hold and there are no hyperedges with endpoints in X_1 and $V \setminus (X_1 \cup Y_1)$; at this stage, however, there will be possibly edges in H with different endpoints in X_1. To construct X_1,Y_1, we start with $X = X_0$ and $Y = Y_0$, and present a transformation that terminates with $X = X_1$ and $Y = Y_1$. Recall that X_0 has a perfect matching using edges in $G \setminus H$. The set X will maintain this property throughout the transformation, and each vertex $x \in X$ has always a unique *mate* e'. The construction does the following : as long as there exists an edge e in H containing x and only endpoints in X and $V \setminus (X \cup Y)$, let e' be the mate of x, we then remove the endpoints of e' from X and add the endpoints of e to Y. Property 1 is maintained because we have removed d vertices from X and added d to Y. Property 2 is maintained because the vertices we added to Y were connected by an edge in H. Property 3 is maintained because X clearly still has a perfect matching in $G \setminus H$, and for the vertices $\{x_1, x_2, \ldots, x_d\} = e'$, the average contribution is still at least $\frac{1-5\lambda}{d}$, as above. We continue this process while there is an edge with endpoints in X and $V \setminus (X \cup Y)$. The process terminates because each time we are removing d vertices from X and adding d vertices to Y. We end up with two sets X_1 and Y_1 such that the first three properties of the lemma are satisfied and there are no edges with endpoints in X_1 and $V \setminus (X_1 \cup Y_1)$. This means that for any edges in H incident to X, this edge is either incident to Y as well or incident to only points in X.

We now set $X = X_1$ and $Y = Y_1$ and show how to transform X and Y into two sets that satisfy all four properties of the lemma. Recall that X_1 still contains a perfect matching using edges in $G \setminus H$; denote this matching M_X^G. Our final set, however, will not guarantee such a perfect matching. Let M_X^H be a maximal matching in X using edges in H (with edges

not incident to Y, because they already satisfy Property 4). Consider the edge set $E_X^* = M_X^G \cup M_X^H$.

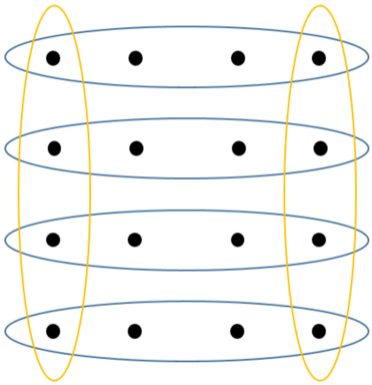

Figure 1: Example with $d = 4$. In blue the edges of M_X^G and in yellow the edges of M_X^H

We now perform the following simple transformation, we remove the endpoints of the edges in M_X^H from X and add them directly to Y. Property 1 is preserved because we are deleting and adding the same number of vertices from X and to Y respectively. Property 2 is preserved because the endpoints we add to Y are matched in M_X^H and thus in H. We will see later for Property 3.

Let's check if Property 4 is preserved. To see this, note that we took M_H^X to be maximal among edges of H that contain only endpoints in X, and moved all the matched vertices in M_X^H to Y. Thus all vertices that remain in X are free in M_H^X, and so there are no edges between only endpoints in X after the transformation. There are also no edges in H containing endpoints from X and $V \setminus (X \cup Y)$, because originally they don't exist for $X = X_1$

Let's check if the Property 3 is preserved. As before, this involves showing that after the transformation, the average contribution of a vertex in $X \cup Y$ to σ is at least $\frac{1-5\lambda}{d}$. (Because every vertex in X is incident to an edge in E_X^*, each vertex is accounted for in the transformation.) Now, all vertices that were in Y_1 remain in Y , so their average contribution remains at $1/d$. We thus need to show that the average contribution to σ among vertices in X_1 remains at least $\frac{1-5\lambda}{d}$ after the transformation.

Let $n = |M_X^G|$ and $k = |M_X^H|$. Since M_X^G is a perfect matching on X_1, we always have $k \leq n$. If $n = k$, then all vertices of X_1 are transferred to Y and clearly their average contribution to σ is $\frac{1}{d} \geq \frac{1-5\lambda}{d}$. Now consider when $k \leq n-1$. Let the edges of M_X^H be $\{e_1, \dots e_k\}$ and these of M_X^G be $\{e'_1, \dots e'_n\}$. Let X' be the set of vertices that remain in X_1. Because the edges of M_X^G are not H, the by two properties of HEDCS :

$$\begin{aligned} \sum_{1 \leq i \leq n,\ x \in e'_i} d_H(x) &\geq n\beta(1-\lambda) \ , \text{ and} \\ \sum_{1 \leq i \leq k,\ x \in e_i} d_H(x) &\leq k\beta \ . \end{aligned}$$

The sum of the degrees of vertices in X' can be written as the following difference:

$$\begin{aligned}\sum_{x\in X'} d_H(x) &= \sum_{1\le i\le n,\ x\in e'_i} d_H(x) - \sum_{1\le i\le k,\ x\in e_i} d_H(x) \\ &\ge n\beta(1-\lambda) - k\beta \\ &= (n-k)\cdot\beta - n\beta\lambda\ .\end{aligned} \tag{4}$$

Now we prove that (5) implies that the contribution of vertices from X' on average at least $\frac{1-5\lambda}{d}$.

Claim A.4. $\sum_{x\in X'} \phi(d_H(x)) \ge (n-k) - 5n\cdot\lambda$.

By claim A.4, the average contribution among the vertices that are considered is

$$\frac{(n-k) - 5n\cdot\lambda + k}{nd} = \frac{1-5\lambda}{d}\ ,$$

which proves Property 3.

Proof. [of claim A.4]

Let's denote $m := n-k$. Recall that the number of vertices in X' is equal to md and $\phi(x) = \frac{d-1}{d}\frac{x}{\beta-x}$. Here we will prove it for $\lambda = 0$. This means that we will prove that

$$\sum_{x\in X'} d_H(x) \ge m\cdot\beta \Rightarrow \sum_{x\in X'} \phi(d_H(x)) \ge m\ .$$

If $m = n-k = 1$, then the result clearly holds by Lemma A.7. Let's suppose $k < n-1$ and thus $m \ge 2$. By Lemma A.7, we know that:

$$\frac{md-1}{md}\sum_{x\in X'} \frac{d_H(x)}{m\cdot\beta - d_H(x)} \ge 1\ . \tag{5}$$

We also know that :

$$\frac{m\beta - a}{\beta - a} = m + \frac{(m-1)a}{\beta-a}\ ,\ \text{and} \tag{6}$$

$$\frac{a}{\beta-a} = m\cdot\frac{a}{m\beta-a} + \frac{(m-1)a^2}{(m\beta-a)(\beta-a)}\ . \tag{7}$$

Combining (5) and (7), we get:

$$\sum_{x\in X'} \frac{d_H(x)}{\beta - d_H(x)} \ge \frac{m^2 d}{md-1} + \sum_{x\in X'} \frac{(m-1)d_H(x)^2}{(m\beta - d_H(x))(\beta - d_H(x))}\ .$$

which leads to:

$$\begin{aligned}\sum_{x\in X'} \phi(d_H(x)) &\ge \frac{d-1}{d}\sum_{x\in X'} \frac{d_H(x)}{\beta - d_H(x)} \\ &\ge m\cdot\frac{m(d-1)}{md-1} + \frac{d-1}{d}\sum_{x\in X'} \frac{(m-1)d_H(x)^2}{(m\beta - d_H(x))(\beta - d_H(x))} \\ &\ge m + \frac{d-1}{d}\sum_{x\in X'} \frac{(m-1)d_H(x)^2}{(m\beta - d_H(x))(\beta - d_H(x))} - \frac{m-1}{md-1}\ .\end{aligned} \tag{8}$$

By convexity of the function $x \mapsto \frac{x^2}{(m\beta - x)(\beta - x)}$:

$$\begin{aligned}\sum_{x \in X'} \frac{d_H(x)^2}{(m\beta - d_H(x))(\beta - d_H(x))} &\geq md \frac{(\frac{\sum d_H(x))}{md})^2}{(m\beta - \frac{\sum d_H(x))}{md}(\beta - \frac{\sum d_H(x))}{md})} \\ &\geq \frac{m\beta}{(md-1)(d-1)} \ .\end{aligned} \tag{9}$$

Where the last inequality is due to $\sum_{x \in X'} d_H(x) \geq m \cdot \beta$

Therefore, when $\beta \geq \frac{d}{2}$ the right hand side of (8) becomes:

$$\begin{aligned} m + \frac{d-1}{d} \sum_{x \in X'} \frac{d_H(x)^2}{(m\beta - d_H(x))(\beta - d_H(x))} - \frac{1}{md-1} &\geq m + \frac{1}{md-1}\left(\frac{m\beta}{d} - 1\right) \\ &\geq m. \end{aligned}$$

□ □

A.8 Proofs from Section 5.1

Lemma 6.5. *(Degree Distribution Lemma). Fix a d-uniform hypergraph $G(V, E)$ and parameters β, $\beta^- = (1-\lambda)\cdot\beta$ (for λ small enough). For any two subgraphs A and B that are $HEDCS(G, \beta, \beta^-)$, and any vertex $v \in V$, then $|d_A(v) - d_B(v)| = O(\sqrt{n})\lambda^{1/2}\beta$.*

Proof. Suppose that we have $d_A(v) = k\lambda\beta$ for some k and that $d_B(v) = 0$. We will show that if the $k = \Omega(\frac{\sqrt{n}}{\lambda^{1/2}})$, then this will lead to a contradiction. Let e be one of the $k\lambda\beta$ edges that are incident to v in A. $e \notin B$ so $\sum_{u \neq v} d_B(u) \geq (1-\lambda)\beta$. From these $(1-\lambda)\beta$ edges, at most $(1-k\lambda)\beta$ can be in A in order to respect *(P1)*, so at least we will have $(k-1)\lambda\beta$ edges in $B \setminus A$, thus we have now covered $k\lambda\beta + (k-1)\lambda\beta$ edges in both A and B. Let's keep focusing on the edge e, and especially on one of its $(k-1)\lambda\beta$ incident edges in $B \setminus A$. Let e_1 be such an edge. $e_1 \in B \setminus A$, therefore $\sum_{v' \in e_1} d_A(v') \geq (1-\lambda)\beta$. The edges incident to e_1 in A that we have covered so far are at most $(1-k\lambda)\beta$, therefore we still need at least $(k-1)\lambda$ new edges in A to respect *P(1)*. Out of these $(k-1)\lambda$ edges, at most $\lambda\beta$ can be in B (because e_1 has already $(1-\lambda)\beta$ covered edges incident to it in B). Therefore at least $(k-2)\lambda\beta$ are in $A \setminus B$. Thus, we have so far covered at least $k\lambda\beta + (k-1)\lambda\beta + (k-2)\lambda\beta$. One can see that we can keep doing this until we cover at least $\frac{k(k+1)}{2}\lambda\beta$ edges in both A and B. The number of edges in each of A and B cannot exceed $n \cdot \beta$ (each vertex has degree $\leq \beta$), therefore we will get a contradiction if $\frac{k(k+1)}{2}\lambda\beta > 2n\beta$, which holds if $k > \frac{2\sqrt{n}}{\lambda^{1/2}}$.

Therefore $k \leq \frac{2\sqrt{n}}{\lambda^{1/2}}$, and $d_A(v) = k\lambda\beta \leq 2\sqrt{n}\lambda^{1/2}\beta$. □

Corollary 6.6. *For d-uniform linear hypergraphs, the degree distribution is tighter, and $|d_A(v) - d_B(v)| = O(\log n)\lambda\beta$.*

Proof. For linear hypergraphs, assume that $d_A(v) = k\lambda\beta$ with $k = \Omega(\log n)$ and $d_B(v) = 0$. The difference in the analysis is that, for every edge e belonging to the $k\lambda\beta$ edges that are incident to v in A, we can find a new set of at least $(k - (d+1))\lambda\beta$ edges in $B \setminus A$. In fact, for such an edge e, every one of the $(1-\lambda)\beta$ edges that verify $\sum_{u \neq v} d_B(u) \geq (1-\lambda)\beta$ intersect e in exactly on vertex

that is not v. The same goes for the subset of at least $(k-1)\lambda\beta$ that are in $B\setminus$ A, these edges already intersect e, and can at most have one intersection in between them. At most $d(d-1)$ of these edges can be considered simultaneously for different $e_1, \ldots, e_d$ from the $k\lambda\beta$ edges incident to v. Therefore, for every edge e, we can find at least a set of $(k-1)\lambda\beta d(d-1) \geq (k-(d+1))\lambda\beta$ new edges that in $B \setminus A$. This means that at this point we have already covered $k\lambda\beta(k-(d+1))\lambda\beta$ edges in both A and B. One can see that we can continue covering new edges just like in the previous lemma, such that at iteration l, the number of covered edges is at least

$$k(k-(d+1))(k-2(d+1))\ldots(k-l(d+1))(\lambda\beta)^l,$$

for $l \leq \frac{k-1}{d+1}$. It is easy to see that for $l = \frac{k-1}{d+1}$, we will have $k(k-(d+1))(k-2(d+1))\ldots(k-l(d+1))(\lambda\beta)^l > 2n\beta$ if $k = \Omega(\log n)$, which will be a contradiction. □

Lemma 6.7. *Let H be a HEDCS(G, β_H, β_H^-) for parameters $\beta_H := (1-\frac{\lambda}{\alpha})\cdot\frac{\beta}{p}$ and $\beta_H^- := \beta_H-(d-1)$, and $\beta \geq 15d(\alpha d)^2 \cdot \lambda^{-2} \cdot \log n$ such that $p < 1 - \frac{2}{\alpha}$. 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 high probability:*

1. *For any vertex $v \in V$: $|d_{H_p}(v) - p \cdot d_H(v)| \leq \frac{\lambda}{\alpha d}\beta$*
2. *H_p is a HEDCS of G_p with parameters $(\beta, (1-\lambda)\cdot\beta)$.*

Proof. Let $\alpha' := \alpha d$. For any vertex $v \in V$, $E[d_{H_p}(v)] = p \cdot d_H(v)$ and $d_H(v) \leq \beta_H$ by Property (P1) of HEDCS H. Moreover, since each edge incident upon v in H is sampled in H_p independently, by a Chernoff bound:

$$P\Big(|d_{H_p}(v) - p \cdot d_H(v)| \geq \frac{\lambda}{\alpha'}\beta\Big) \leq 2 \cdot \exp(-\frac{\lambda^2\beta}{3 \cdot \alpha'^2}) \leq \frac{2}{n^{5d}} .$$

In the following, we condition on the event that:

$$|d_{H_p}(v) - p \cdot d_H(v)| \leq \frac{\lambda}{\alpha'}\beta .$$

This event happens with probability at least $1 - \frac{2}{n^{5d-1}}$ 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 a $HEDCS(G_p, \beta, (1-\lambda)\cdot\beta)$ conditioned on this event.

Consider any edge $e \in H_p$. Since $H_p \subset H$, $e \in H$ as well. Hence, we have,

$$\sum_{v\in e} d_{H_p}(v) \leq p \cdot \beta_H + \frac{d\lambda}{\alpha'}\beta = (1 - \frac{\lambda}{\alpha} + \frac{d\lambda}{\alpha'})\beta = \beta ,$$

because $\frac{\alpha}{\alpha'} = \frac{1}{d}$, where the second inequality is by Property (P1) of HEDCS H and the equality is by the choice of β_H . As a result, H_p satisfies Property (P1) of HEDCS for parameter β.

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

$$\begin{aligned}
\sum_{v\in e} d_{H_p}(v) \geq p \cdot \beta_H^- - \frac{d\lambda}{\alpha'}\beta &= (1 - \frac{\lambda}{\alpha} - \frac{d\lambda}{\alpha'})\beta - p \cdot (d-1) \\
&= (1 - \frac{2\lambda}{\alpha})\beta - p \cdot (d-1) \\
&> (1-\lambda)\cdot\beta .
\end{aligned}$$

□

Lemma 6.8. *(HEDCS in Edge Sampled Subgraph). Fix any hypergraph $G(V,E)$ 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 HEDCSs of G_1 and G_2 with parameters $(\beta, (1-\lambda)\cdot\beta)$. Suppose $\beta \geq 15d(\alpha d)^2 \cdot \lambda^{-2} \cdot \log n$, then, with probability $1 - \frac{4}{n^{5d-1}}$, simultaneously for all $v \in V$: $|d_{H_1}(v) - d_{H_2}(v)| = O\big(n^{1/2}\big)\lambda^{1/2}\beta$.*

Proof. Let H be an HEDCS(G, β_H, β_H^-) for the parameters β_H, β_H^-. The existence of H follows since $\beta_H - (d-1) \geq \beta_H^-$. Define $\hat{H}_1 := H \cap G_1$ and $\hat{H}_2 := H \cap G_2$. By Lemma 6.7, $\hat{H}_1$ (resp. $\hat{H}_2$) is a HEDCS of G_1 (resp. G_2) with parameters $(\beta, (1-\lambda)\beta)$ with probability $1 - \frac{4}{n^{5d-1}}$. In the following, we condition on this event.

By Lemma 6.5 (Degree Distribution Lemma), since both H_1 (resp. H_2) and $\hat{H}_1$ (resp. $\hat{H}_2$) are HEDCS for G_1 (resp. G_2), the degree of vertices in both of them should be "close" to each other. Moreover, since by Lemma 6.7 the degree of each vertex in $\hat{H}_1$ and $\hat{H}_2$ 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

$$\begin{aligned} |d_{H_1}(v) - d_{H_2}(v)| &\leq |d_{H_1}(v) - d_{\hat{H}_1}(v)| + |d_{\hat{H}_1}(v) - d_{\hat{H}_2}(v)| + |d_{\hat{H}_2}(v) - d_{H_2}(v)| \\ &\leq O\big(n^{1/2}\big)\lambda^{1/2}\beta + |d_{\hat{H}_1}(v) - pd_H(v)| + |d_{\hat{H}_2}(v) - pd_H(v)| \\ &\leq O\big(n^{1/2}\big)\lambda^{1/2}\beta + O(1)\cdot\lambda\cdot\beta \ . \end{aligned}$$

□

Lemma 6.10. *Suppose $k \leq \sqrt{m}$. Then with high probability*

1. *The subgraph C is a $HEDCS(G, \beta_C, \beta_C^-)$ for parameters: $\lambda_C = O\big(n^{1/2}\big)\lambda^{1/2}$, $\beta_C = (1 + d\cdot\lambda_C)\cdot k\cdot\beta$ and $\beta_C^- = (1-\lambda-d\cdot\lambda_C)\cdot k\cdot\beta$.*
2. *The total number of edges in each subgraph $G^{(i)}$ of G is $\tilde{O}(s)$.*
3. *If $s = \tilde{O}(n\sqrt{nm})$, then the graph C can fit in the memory of one machine.*

Proof. [of Lemma 6.10]

1. Recall that each graph $G^{(i)}$ is an edge sampled subgraph of G with sampling probability $p = \frac{1}{k}$. By Lemma 6.8 for graphs $G^{(i)}$ and $G^{(j)}$ (for $i \neq j \in [k]$) and their HEDCSs $C^{(i)}$ and $C^{(j)}$, with probability $1 - \frac{4}{n^{5d-1}}$, for all vertices $v \in V$:

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

   By taking a union bound on all $\binom{k}{2} \leq n^d$ 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 $1 - \frac{4}{n^{4d-1}}$. In the following, we condition on this event.

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

$$\begin{aligned} \sum_{v\in e} d_C(v) &= \sum_{v\in e}\sum_{i=1}^{k} d_{C^{(i)}}(v) \\ &\leq k\cdot\sum_{v\in e} d_{C^{(j)}}(v) + d\cdot k\cdot\lambda_C\cdot\beta \\ &\leq k\cdot\beta + d\cdot k\cdot\lambda_C\cdot\beta \\ &= \beta_C \ . \end{aligned}$$

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

$$\begin{aligned}\sum_{v \in e} d_C(v) &= \sum_{v \in e} \sum_{i=1}^{k} d_{C^{(i)}}(v) \\ &\geq k \cdot \sum_{v \in e} d_{C^{(j)}} - d \cdot k\lambda_C\beta \\ &\geq k \cdot (1-\lambda) \cdot \beta - d \cdot k\lambda_C\beta \ .\end{aligned}$$

2. Let $E^{(i)}$ be the edges of $G^{(i)}$. By the independent sampling of edges in an edge sampled subgraph, we have that $E\big[|E^{(i)}|\big] = \frac{m}{k} = \tilde{O}(s)$. By Chernoff bound, with probability $1 - \frac{1}{k \cdot n^{20}}$, the size of $E^{(i)}$ is $\tilde{O}(s)$. We can then take a union bound on all k machines in $G^{(i)}$ and have that with probability $1 - 1/n^{20}$, each graph $G^{(i)}$ is of size $\tilde{O}(s)$.

3. The number of edges in C is bounded by $n \cdot \beta_c = O(n \cdot k \cdot \beta) = \tilde{O}(\frac{n^3 m}{s}) = \tilde{O}(s)$.

□

Corollary 6.11. *If G is linear, then by choosing $\lambda := \frac{1}{2\log^2 n}$ and $\beta := 500 \cdot d^3 \cdot \log^4 n$ in Algorithm 3 we have:*

1. *With high probability, the subgraph C is a $HEDCS(G, \beta_C, \beta_C^-)$ for parameters: $\lambda_C = O(\log n)\lambda$, $\beta_C = (1 + d \cdot \lambda_C) \cdot k \cdot \beta$ and $\beta_C^- = (1 - \lambda - d \cdot \lambda_C) \cdot k \cdot \beta$.*

2. *If $s = \tilde{O}(\sqrt{nm})$ then C can fit on the memory of one machine.*

Proof. [of Corollary 6.11]

1. Similarly to Lemma 6.10 and by using corollary 6.9, we know that for graphs $G^{(i)}$ and $G^{(j)}$ (for $i \neq j \in [k]$) and their HEDCSs $C^{(i)}$ and $C^{(j)}$, with high probability , for all vertices $v \in V$:

$$|d_{C^{(i)}}(v) - d_{C^{(j)}}(v)| \leq O\big(\log n\big)\lambda\beta.$$

   By taking a union bound on all $\binom{k}{2} \leq n^d$ 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 $1 - \frac{4}{n^{4d-1}}$. In the following, we condition on this event. Showing that C is indeed a $HEDCS(G, \beta_C, \beta_C^-)$ follows by the same analysis from the proof of Lemma 6.10.

2. The number of edges in C is bounded by $n \cdot \beta_c = O(n \cdot k \cdot \beta) = O(n \cdot k) = \tilde{O}(\frac{nm}{s}) = \tilde{O}(s)$.

□

B Figures and Tables

n	m	d	k	#I	Gr	IS	HEDCS	β	λ
15	200	3	5	500	79.1%	87.5%	82.3%	5	3
30	400		5		82.6%	91.3%	84.4%	7	4
100	3200		10		83.9%	96.2%	88.2%	5	2
300	4000		10		81.1%	92.0%	86.3%	4	2
50	800	5	6	500	76.0%	89.1%	78.5%	16	11
100	2,800		10		77.9%	92.1%	81.9%	16	11
300	4,000		10		77.8%	93.9%	87.1%	10	5
500	8,000		16		79.2%	94.3%	85.9%	10	5
500	15,000	10	16	500	71.8%	90.6%	79.2%	20	10
1,000	50,000		20		73.8%	92.3%	81.5%	20	10
2,500	100,000		20		72.2%	91.5%	80.7%	20	10
5,000	200,000		20		72.5%	90.7%	79.8%	20	10
1,000	5,0000	25	20	100	68.2%	87.5%	75.6%	75	50
2,500	100,000		25		69.0%	87.9%	74.3%	75	50
5,000	250,000		30		67.8%	87.3%	75.1%	75	50
10,000	500,000		30		67.2%	86.9%	73.7%	75	50
5,000	250,000	50	30	100	67.4%	86.6%	74.0%	100	50
10,000	500,000		30		68.1%	87.1%	73.4%	100	50
15,000	750,000		30		66.9%	86.2%	73.2%	100	50
25,000	1,000,000		30		67.3%	86.0%	72.8%	100	50

Table 5: Comparison on random uniform instances with maximal matching benchmark.

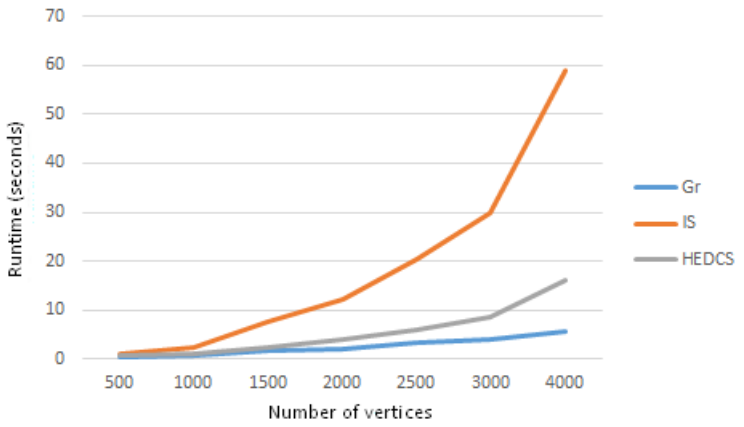

Figure 2: Runtime of the three algorithms when $d = 3$ and $m = 20 \cdot d \cdot n$

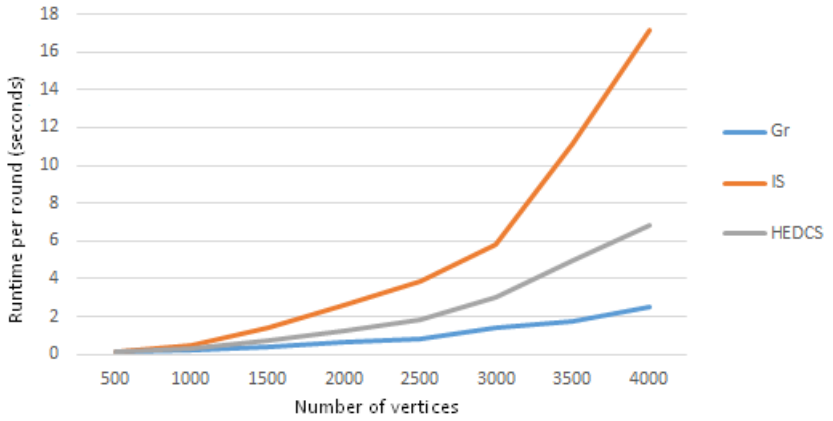

Figure 3: Runtime per round (maximum over all machines in every round) of the three algorithms when $d = 3$ and $m = 20 \cdot d \cdot n$

C 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 $\delta \in (0, 1)$

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

D Nesbitt's Inequality

Nesbitt's inequality states that for positive real numbers a, b and c,

$$\frac{a}{b+c} + \frac{b}{a+c} + \frac{c}{a+b} \geq \frac{3}{2},$$

with equality when all the variables are equal. And generally, if $a_1, \ldots, a_n$ are positive real numbers and $s = \sum_{i=1}^{n}$, then:

$$\sum_{i=1}^{n} \frac{a_i}{s - a_i} \geq \frac{n}{n-1},$$

with equality when all the a_i are equal.