Streaming Min-max Hypergraph Partitioning

Dan Alistarh
Microsoft Research
Cambridge, United Kingdom
dan.alistarh@microsoft.com

Jennifer Iglesias*
Carnegie Mellon University
Pittsburgh, PA
jiglesia@andrew.cmu.edu

Milan Vojnovic
Microsoft Research
Cambridge, United Kingdom
milanv@microsoft.com

Abstract

In many applications, the data is of rich structure that can be represented by a hypergraph, where the data items are represented by vertices and the associations among items are represented by hyperedges. Equivalently, we are given an input bipartite graph with two types of vertices: items, and associations (which we refer to as topics). We consider the problem of partitioning the set of items into a given number of parts such that the maximum number of topics covered by a part of the partition is minimized. This is a natural clustering problem, with various applications, e.g. partitioning of a set of information objects such as documents, images, and videos, and load balancing in the context of computation platforms.

In this paper, we focus on the *streaming* computation model for this problem, in which items arrive online one at a time and each item must be assigned irrevocably to a part of the partition at its arrival time. Motivated by scalability requirements, we focus on the class of streaming computation algorithms with memory limited to be at most linear in the number of the parts of the partition. We show that a greedy assignment strategy is able to recover a hidden co-clustering of items under a natural set of recovery conditions. We also report results of an extensive empirical evaluation, which demonstrate that this greedy strategy yields superior performance when compared with alternative approaches.

1 Introduction

In a variety of applications, one needs to process data of rich structure that can be conveniently represented by a hypergraph, where associations of the data items, represented by vertices, are represented by hyperedges, i.e. subsets of items. Such data structure can be equivalently represented by a bipartite graph that has two types of vertices: vertices that represent *items*, and vertices that represent associations among items, which we refer to as *topics*. In this bipartite graph, each item is connected to one or more topics. The input can be seen as a graph with vertices belonging to (overlapping) communities.

There has been significant work on partitioning a set of items into disjoint parts such that similar items are assigned to the same part of the partition, see, e.g., [9] for a survey. This problem arises in the context of *clustering of information objects* such as documents, images or videos. For example, the goal may be to partition given collection of documents into disjoint sub-collections such that the maximum number of distinct *topics* covered by each sub-collection is minimized, resulting in a

*Work performed in part while an intern with Microsoft Research.

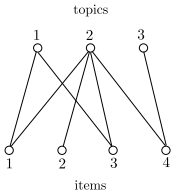


Figure 1: A simple example of a set of items with overlapping associations to topics.

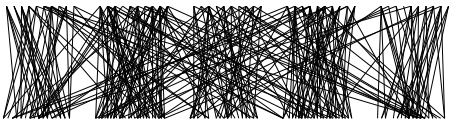


Figure 2: An example of hidden co-clustering with five hidden clusters.

parsimonious summary. The same fundamental problem also arises in *processing of complex data workloads*, including enterprise emails [11], online social networks [19], graph data processing and machine learning computation platforms [21, 22, 2], and load balancing in modern streaming query processing platforms [25]. In this context, the goal is to partition a set of data items over a given number of servers to balance the load according to some given criteria.

Problem Definition. We consider the min-max hypergraph partitioning problem defined as follows. The input to the problem is a set of *items*, a set of *topics*, a number of *parts* to partition the set of items, and a *demand matrix* that specifies which particular subset of topics is associated with each individual item. Given a partitioning of the set of items, the cost of a partition is defined as the *number of distinct topics* that are associated with items of the given partition[1]. The cost of a given partitioning is the *maximum* cost of a partition. In other words, given an input hypergraph and a partitioning of the set of vertices into a given number of disjoints parts, the cost of a partition is defined to be the number of hyperedges that have at least one vertex assigned to this partition. For example, for the simple input graph in Fig. 1, a partitioning of the set of items into two parts $\{1, 3\}$ and $\{2, 4\}$ amounts to costs of the partitions each of value 2, thus, the cost of the partitioning is value 2. The cost of a partition is a submodular function as the distinct topics associated with items of the partition correspond to a neighborhood set in the input bipartite graph.

In the streaming computation model that we consider, items arrive sequentially one at a time, and each item needs to be assigned, irrevocably, to one partition at its arrival time. This streaming computation model allows for limited memory to be used at any time during the execution whose size is restricted to be at most linear in the number of the parts of the partition. Both these assumptions arise as part of system requirements for deployment in web-scale services.

The min-max hypergraph partition problem is NP hard. The streaming computation problem is even more difficult, as less information is available to the algorithm when an item must be assigned.

Contribution. In this paper, we consider the streaming min-max hypergraph partitioning problem. We identify a greedy item placement strategy which outperforms all alternative approaches considered on real-world datasets, and can be proven to have a non-trivial recovery property: it recovers hidden co-clusters of items in probabilistic inputs subject to a set of recovery conditions.

Specifically, we show that, given a set of hidden co-clusters to be placed onto k partitions, the greedy strategy will tend to place items from the same hidden cluster onto the same partition, with high probability. In turn, this property implies that greedy will provide a constant factor approximation of the optimal partitioning on inputs satisfying the recovery property.

The probabilistic input model we consider is defined as follows. The set of topics is assumed to be partitioned into a given number $\ell \geq 1$ of disjoint hidden clusters. Each item is connected to topics according to a mixture probability distribution defined as follows. Each item first selects one of the hidden clusters as a home hidden cluster by drawing an independent sample from a uniform distribution over the hidden clusters. Then, it connects to each topic from its home hidden cluster independently with probability p, and it connects to each topic from each other hidden cluster with probability $q \leq p$. This defines a hidden co-clustering of the input bipartite graph; see Figure 2 for an example.

This model is similar in spirit to the popular stochastic block model defined as a model of an undirected graph, and it corresponds to a *hidden co-clustering* [6, 7, 18, 4] defined as a model of an undirected bipartite graph. We consider asymptotically accurate recovery of this hidden co-clustering: a hidden cluster is said to be *asymptotically recovered* if the portion of items from the given hidden

[1]From here on, we will use partition to refer to a part of a partitioning

cluster assigned to the same partition goes to one asymptotically as the number of items observed grows large. An algorithm guarantees *balanced* asymptotic recovery if, additionally, it ensures that the cost of the most loaded partition is within a constant of the average partition load.

Our main analytical result is showing that a simple greedy strategy provides *balanced asymptotic recovery* of hidden clusters (Theorem 1). We prove that a sufficient condition for the recovery of hidden clusters is that the number of hidden clusters ℓ is at least $k \log k$, where k is the number of parts of a partition of the set of items, and that the gap between the probability parameters q and p is sufficiently large: $q < \log r/(kr) < 2 \log r/r \leq p$, where r is the number of topics in a hidden cluster. Roughly speaking, this means that if the mean number of topics to which an item is associated with in its home hidden cluster of topics is at least twice as large as the mean number of topics to which an item is associated with from other hidden clusters of topics, then the simple greedy online algorithm guarantees asymptotic recovery.

The proof is based on a coupling argument, where we first show that assigning an item to a partition based on the number of topics it has in common with each partition is similar to making the assignment proportionally to the number of *items* corresponding to the same hidden cluster present on each partition. In turn, this allows us to couple the assignment strategy with a *Polya urn process* [5] with "rich-get-richer" dynamics, which implies that the policy converges to assigning each item from a hidden cluster to the same partition. Additionally, this phenomenon occurs "in parallel" for each cluster. This recovery property will imply that this strategy will ensure a constant factor approximation of the optimum assignment.

Further, we provide experimental evidence that this greedy online algorithm exhibits good performance for several real-world input bipartite graphs, outperforming more complex assignment strategies, and even some offline approaches.

2 Problem Definition and Basic Results

In this section we provide a formal problem definition, and present some basic results on the computational hardness and lower bounds.

Input. The input is defined by a set of *items* $N = \{1, 2, \ldots, n\}$, a set of *topics* $M = \{1, 2, \ldots, m\}$, and a set of k partitions. Dependencies between items and topics are given by a *demand matrix* $D = (d_{i,l}) \in \{0, 1\}^{n \times m}$ where $d_{i,l} = 1$ indicates that item i needs topic l, and $d_{i,l} = 0$, otherwise.[2]

Alternatively, we can represent the input as a *bipartite graph* $G = (N, M, E)$ where there is an edge $(i, l) \in E$ if and only if item i needs topic l or as a *hypergraph* $H = (N, E)$ where a hyperedge $e \in E$ consists of all items that use the same topic.

The Problem. An assignment of items to partitions is given by $x \in \{0, 1\}^{n \times k}$ where $x_{i,j} = 1$ if item i is assigned to partition j, and $x_{i,j} = 0$, otherwise. Given an assignment of items to partitions x, the cost of partition j is defined to be equal to the minimum number of distinct topics that are needed by this partition to cover all the items assigned to it, i.e.

$$c_j(x) = \sum_{l \in M} \min \left\{ \sum_{i \in N} d_{i,l} x_{i,j}, 1 \right\}.$$

As defined, the cost of a partition is a *submodular function* of the items assigned to it. We consider the *min-max hypergraph partitioning problem* defined as follows:

$$\begin{array}{ll} \text{minimize} & \max\{c_1(x), c_2(x), \ldots, c_k(x)\} \\ \text{subject to} & \sum_{j \in [k]} x_{i,j} = 1 \quad \forall i \in [n] \\ & x \in \{0, 1\}^{n \times k} \end{array} \tag{1}$$

We note that this problem is an instance of the submodular load balancing, as defined in [24].

Basic Results. This problem is NP-Complete, by reduction from the *subset sum* problem.

[2]The framework allows for a natural generalization to allow for real-valued demands. In this paper we focus on $\{0, 1\}$-valued demands.

Proposition 1. *The min-max hypergraph partitioning problem is NP-Complete.*

We now give a lower bound on the optimal value of the problem, using the observation that each topic needs to be made available on at least one partition.

Proposition 2. *For every partition of the set of items in k partitions, the maximum cost of a part is larger than or equal to m/k, where m is the number of topics.*

Finally, we analyze the performance of an algorithm which simply assigns items to randomly chosen partitions upon arrival. Although this is a popular strategy commonly deployed in practice for other problems, the following result shows that it does not yield a good solution for the min-max hypergraph partitioning problem.

Proposition 3. *The expected maximum load of a partition under random assignment is at least $(1 - \sum_{j=1}^{m}(1 - 1/k)^{n_j}/m) \cdot m$, where n_j is the number of items associated with topic j.*

For instance, if we assume that $n_j \geq k$ for each topic j, we obtain that the expected maximum load is of at least $(1 - 1/e)m$. This suggests that the performance of random assignment is poor: on an input where m topics form k disjoint clusters, and each item subscribes to a single cluster, the optimal solution has cost m/k, whereas, by the above claim, random assignment has approximate cost $2m/3$, yielding a competitive ratio that is linear in k.

Balanced Recovery of Hidden Co-Clusters. We relax the worst-case input requirements by defining a family of *hidden co-clustering* inputs. Our model is a generalization of the graph stochastic block model to the case of hypergraphs.

We consider a set of topics $\mathcal{R}$, partitioned into ℓ clusters $C_1, C_2, \ldots, C_\ell$, each of which contains r topics. Given these hidden clusters, each item is associated with topics as follows. Each item is first assigned a "home" cluster C_h, chosen uniformly at random among the hidden clusters. The item then connects to topics inside its home cluster by picking each topic independently with fixed probability p. Further, the item connects to topics from a fixed arbitrary "noise" set Q_h of size $\leq r/2$ outside its home cluster C_h, where the item is connected to each topic in Q_h uniformly at random, with fixed probability q. (Sampling outside topics from the set of all possible topics would in the limit lead to every partition to contain *all* possible topics, which renders the problem trivial. We do not impose this limitation in the experimental validation.)

Definition 1 (Hidden Co-Clustering)**.** *A bipartite graph is in $\mathsf{HC}(n, r, \ell, p, q)$ if it is constructed using the above process, with n items and ℓ clusters with r topics per cluster, where each item subscribes to topics inside its randomly chosen home cluster with probability p, and to topics from the noise set with probability q.*

At each time step t, a new item is presented in the input stream of items, and is immediately assigned to one of the k partitions, $S_1, S_2, \ldots, S_k$, according to some algorithm. Algorithms do not know the number of hidden clusters or their size, but can examine previous assignments.

Definition 2 (Asymptotic Balanced Recovery.)**.** *Given a hidden co-clustering $\mathsf{HC}(n, r, \ell, p, q)$, we say an algorithm* asymptotically recovers *the hidden clusters $C_1, C_2, \ldots, C_\ell$ if there exists a recovery time t_R during its execution after which, for each hidden cluster C_i, there exists a partition S_j such that each item with home cluster C_i is assigned to partition S_j with probability that goes to* 1*. Moreover, recovery is* balanced *if the ratio between the* maximum partition cost *and the* average partition cost *is upper bounded by a constant $B > 0$.*

3 Streaming Algorithm and the Recovery Guarantee

Recall that we consider the online problem, where we receive one item at a time together with all its corresponding topics. The item must be immediately and irrevocably assigned to some partition. In the following, we describe the greedy strategy, specified in Algorithm 1.

This strategy places each incoming item onto the partition whose incremental cost (after adding the item and its topics) is minimized. The immediate goal is not balancing, but rather clustering similar items. This could in theory lead to large imbalances; to prevent this, we add a *balancing constraint*

Data: Hypergraph $H = (V, E)$, received one item (vertex) at a time, k partitions, capacity bound c
Result: A partition of V into k parts

Set initial partitions $S_1, S_2, \ldots, S_k$ to be empty sets
while *there are incoming items* **do**
    Receive the next item t, and its topics R
    $I \leftarrow \{i : |S_i| \leq \min_j |S_j| + c\}$ /* partitions not exceeding capacity */
    Compute $r_i = |S_i \cap R| \; \forall i \in I$ /* size of topic intersection */
    $j \leftarrow \arg\max_{i \in I} r_i$ /* if tied, choose least loaded partition */
    $S_j \leftarrow S_j \cup R$ /* item t and its topics are assigned to S_j */
return $S_1, S_2, \ldots, S_k$

Algorithm 1: The greedy algorithm.

specifying the maximum load imbalance. If adding the item to the first candidate partition would violate the balancing constraint, then the item is assigned to the first valid partition, in decreasing order of intersection size.

3.1 Theorem Statement and Proof Outline

Our main technical result provides sufficient conditions on the cluster parameters for the greedy strategy to provide *balanced recovery of hidden clusters, with high probability.*

Theorem 1 (The Recovery Theorem)**.** *For a random input consisting of a hidden co-cluster graph G in $\mathsf{HC}(n, r, \ell, p, q)$ to be distributed across $k \geq 2$ partitions, if the number of clusters is $\ell \geq k \log k$, and the probabilities p and q satisfy $p \geq 2 \log r / r$, and $q \leq \log r / (rk)$, then greedy ensures balanced asymptotic recovery of the hidden clusters.*

Coupling and High Probability. In the following, we say that two random processes are *coupled* to mean that their random choices are the same. We say that an event occurs *with high probability (w.h.p.)* if it occurs with probability at least $1 - 1/r^c$, where $c \geq 1$ is a constant.

Proof Overview. The proof of this result can be summarized as follows. The first step will be to prove that greedy recovers a single cluster w.h.p. when assigning to just two partitions. More precisely, given a sequence of items generated from a single home cluster, and two partitions, a version of the algorithm without balancing constraints will eventually converge to assigning all incoming items to a single partition. This is a main technical step of the proof, and it is based on a coupling of greedy assignment with a "rich get richer" *Polya urn process* [5], and then using the convergence properties of such processes. Further, we extend this coupling claim from two partitions to $k > 2$ partitions, again for a single cluster, showing that, when the input consists of items from a single cluster, greedy will quickly converge to assigning all items to a single partition, w.h.p.

In the next step, we prove that the algorithm will in fact recover ℓ clusters of items in parallel, assigning each of them (i.e., most of their corresponding items) independently at random to one of the partitions, and that this convergence is not adversely affected by the fact that items also subscribe to topics from outside their home cluster. The problem of determining the maximum partition load is then reduced to showing that the maximum number of clusters that may be randomly assigned to a partition is *balanced*, as well as bounding the extra load due on a server to topics outside the home cluster and miss-assignments.

Polya Urn Processes. For reference, a Polya urn process [5] works as follows. We start each of $k \geq 2$ urns with one ball, and, at each step t, observe a new ball. We assign the new ball to urn $i \in \{1, \ldots, k\}$ with probability proportional to $(b_i)^\gamma$, where $\gamma > 0$ is a fixed real constant, and b_i is the number of balls in urn i at time t. We shall employ the following classic result.

Lemma 1 (Polya Urn Convergence [5])**.** *Consider a finite k-bin Polya urn process with exponent $\gamma > 1$, and let x_i^t be the fraction of balls in urn i at time t. Then, almost surely, the limit $X_i = \lim_{t \to \infty} x_i^t$ exists for each $1 \leq i \leq k$. Moreover, we have that there exists an urn j such that $X_j = 1$, and that $X_i = 0$, for all $i \neq j$.*

3.2 Step 1: Recovering a Single Cluster

Strategy. We first prove that, in the case of a single home cluster for all items, and two partitions ($k = 2$), with no balance constraints, the greedy algorithm with no balance constraints converges to a *monopoly*, i.e., eventually assigns all the items from the cluster onto the same partition, w.h.p. Formally, there exists some convergence time t_R and some partition S_i such that, after time t_R, all future items associated to this home cluster will be assigned to partition S_i, with probability at least $1 - 1/r^c$.

Our strategy will be to couple greedy assignment with a Polya urn process with exponent $\gamma > 1$, showing that the dynamics of the two processes are the same, w.h.p. There is one serious technical issue: while the Polya process assigns new *balls* based on the *ball* counts of urns, greedy assigns *items* (and their respective topics) based on the number of *topic intersections* between the item and the partition. It is not clear how these two measures are related.

We circumvent this issue by taking a two-tiered approach. Roughly, we first prove that, w.h.p., we can couple the number of items on a server with the number of *unique topics* assigned to the same partition. We then prove that this is enough to couple the greedy assignment with a Polya urn process with exponent $\gamma > 1$ (Lemma 4). This will imply that greedy converges to a monopoly, by Lemma 1.

Notation. Fix a time t in the execution of the greedy assignment process, corresponding to some new item being randomly generated. A topic r is *known* at time t if it has been a topic for some item up to time t. A known topic r is a *singleton* if it has been placed on one partition, but not on others. Otherwise, it is a *duplicate*. In the following, we will focus on the above quantities around the special time $t_0 = r/\log r$, which we shall prove is close to the convergence time. For simplicity, when referring to a value at time t_0, we omit the subscript.

3.2.1 Auxiliary Results

We first state some helper results characterizing the number of known and singleton topics up to some point in time. The reader may skip this sub-section, and return to it as necessary while reading the proof of Lemma 4.

Lemma 2. *The following hold.*

1. *For $0 < \epsilon < 1$ constant, the number of topics inside the cluster to which an item τ subscribes is in $[2(1-\epsilon)\log r, 2(1+\epsilon)\log r]$, w.h.p.*
2. *The expected number of known topics by time t is at least $r(1 - \exp(-2t\log r/r))$.*
3. *For any time $t \geq r/\log^2 r$, the number of known topics is at least $r/\log r$ and the number of* singleton *topics is at least $r/(2\log r)$, both w.h.p.*

Proof. The first statement follows by straightforward application of Chernoff bounds. To bound the number of known topics, notice that, at each step, a specific topic r is sampled with probability p. Therefore, the probability that r has not been sampled by time t is $(1-p)^t$. Plugging in $p = 2\log r/r$, it follows that the expected number of unknown topics up to t is $r\exp(-2t\log r/r)$, which implies the second claim.

In particular, this number of unknown topics by time $t = r/\log^2 r$ is at most $r/e^{2/\log r} \leq r(1 - 3/2\log r)$, by the Taylor expansion. Therefore, the expected number of known topics up to t is at least $3r/(2\log r)$. By a Chernoff bound, it follows that the number of known topics up to t is at most $r/\log r$, w.h.p., as required.

To lower bound the number of singleton topics, notice that it is sufficient to lower bound the number of topics that have been sampled exactly once up to and including time t. (Such topics are necessarily singletons.) The probability that a topic has been sampled *exactly once* is $tp(1-p)^{t-1}$. Since $t \geq r/\log^2 r$, we obtain that the expected number of topics that have been sampled exactly once is at least $r/(\log r e^{1/\log r}) \geq r/\log r$, for large enough r. Hence, the number of singletons up to this point is at least $r/(2\log r)$, w.h.p., which completes the proof. □

We next focus on the ratio of singleton topics between the two partitions. Define the *singleton topic ratio* as the number of singleton topics on the more loaded partition divided by the number of singleton topics on the less loaded partition. Further, define the *topic-to-item* quotient of a partition as the number of topics it contains divided by the number of items that have been assigned to it.

Lemma 3. *Assume that the singleton topic ratio at time t_0 is $\mu \leq 1/2 + \epsilon$, for fixed $\epsilon < 1$, and let $\phi(\epsilon) = \left(\frac{1/2+2\epsilon}{1/2-2\epsilon}\right)^2$. Then, the ratio between the topic-to-item quotients of the two partitions is in the interval $[1/\phi(\epsilon), \phi(\epsilon)]$, with high probability.*

Proof. Let σ_i denote the number of singleton topics on partition i. Without loss of generality, let partition 1 be the more loaded one at t_0, i.e., $\mu = \sigma_1/(\sigma_1 + \sigma_2)$. Let T_1 be the set of items assigned to the first partition between times $r/\log^2 r$ and $r/\log r$, and T_2 be the corresponding set of items for partition 2. By the Lemma statement, we have that $\sigma_1/(\sigma_1 + \sigma_2) \leq 1/2 + \epsilon$ at t_0.

Given this bound, our first claim is as follows. If $\mu \leq 1/2+\epsilon$ at time t_0, then, for all times $r/\log^2 r \leq t \leq r/\log r$, we have that $\mu_t \leq 1/2 + 2\epsilon$, w.h.p. Also, $|T_1|/(|T_1| + |T_2|) \in [1/2 - 2\epsilon, 1/2 + \epsilon]$, w.h.p.

We focus on the proof of the first statement above, and the second will follow as a corollary. Let us assume for contradiction the converse, i.e., that there exists a time step $r/\log^2 r \leq t \leq r/\log r$ for which $\mu_t > 1/2 + 2\epsilon$. We will show that, after time t, the relative gap in terms of singleton topics between the two partitions will increase, with high probability, which contradicts the bound at time t_0.

For this, consider an incoming item τ. If this item is subscribing to a known topic, which we call case 1, then it will be assigned by the intersection rule. In case 2, it will be placed uniformly at random on one of the partitions. To control this process, we split the execution from time t into blocks of $b = r/\log^4 r$ consecutive incoming items. Notice that, by Lemma 2, there are at least $r/\log r$ known topics after time $r/\log^2 r$, w.h.p. This implies that the probability that an item is *not* assigned by the intersection rule after this time is at most $(1 - 2\log r/r)^{r/\log r} \leq (1/e)^2$. Therefore, each incoming item is assigned by the intersection rule after this time, with at least constant probability.

Consider now the probability that a case-1 item gets assigned to partition 1, assuming that $\sigma_1/(\sigma_1 + \sigma_2) > 1/2 + 2\epsilon$ at the beginning of the current block. This means that the item has more topics in common with partition 1 than with partition 2. By calculation, this probability is at least

$$\frac{1/2 + 2\epsilon}{1/2 - 2\epsilon + 7b\log r/r + 1/2 + 2\epsilon} \geq 1/2 + 7\epsilon/4,$$

where we have pessimistically assumed that *all the items* in the current block get assigned to the second partition, and that each such item contains at most $7/3 \log r$ new topics. (This last fact holds w.h.p. by Lemma 2.)

For an item i during this block, let X_i be an indicator random variable for the event that the item gets assigned to partition 1, and fix $X = \sum_i X_i$. We wish to lower bound X, and will assume that these events are independent—the fact that they are positively correlated does not affect the lower bound. We apply Chernoff bounds, to get that, w.h.p., $X \geq (1 - \delta)7b\epsilon/4$, that is, the first partition gets at least $(1 - \delta)7b\epsilon/4$ *extra* items from each block, where $0 < \delta < 1$ is a constant. On the other hand, the number of case-2 items assigned is balanced, since these items are assigned randomly. In particular, it can be biased towards partition 2 by a fraction of at most $(1 - \delta)\epsilon/4$, w.h.p. We have therefore obtained that partition 1 obtains an *extra* number of items which is at least $3b\epsilon/2$ in each block, w.h.p. Summing over $\log^2 r$ blocks, we get that, over a period of $r/\log^2 r$ time steps, partition 1 gets at least $(1/2 + 3\epsilon/2)r/\log^2 r$ extra items, w.h.p.

Notice that, in turn, this item advantage also translates into a similar extra proportion of *new topics* acquired during each block. In particular, we obtain that the first partition acquires an $(1/2 + 4\epsilon/3)$ fraction of the new topics observed in a block, w.h.p. Recall that, by assumption, at the beginning of the process, partition 1 already had a fraction of $(1/2 + 2\epsilon)$ singleton topics. Therefore, the event that the singleton topic ratio is balanced by at most $1/2+\epsilon$ at t_0 has very low probability, as claimed. The proof of the second statement follows by a similar argument.

To complete the proof of Lemma 3, it is enough to notice that, by the previous claim, the ratio between the topic-to-item quotients of the two partitions is bounded as $\frac{\sigma_1+\kappa}{\sigma_2+\kappa} \cdot \frac{q_2}{q_1} \leq \left(\frac{1/2+2\epsilon}{1/2-2\epsilon}\right)^2$, which completes the proof of Lemma 3. □

3.2.2 Convergence to a Monopoly

We can now prove that one of two things must happen during the algorithm's execution: either one of the partitions gains a constant size advantage, or the algorithm can be coupled with a Polya urn process. In both cases, the algorithm will converge to a monopoly.

Lemma 4. *Given a hidden cluster input* $\mathsf{HC}(n, r, \ell, p, q)$*, with* $\ell = 1$*,* $p \geq 2\log r/r$ *and* $q = 0$*, for every* $t \geq t_0 = r/\log r$*, to be allocated onto two partitions, one of the following holds:*

1. *With high probability, the greedy algorithm with a cluster and two partitions can be coupled with a finite Polya urn process with parameter* $\gamma > 1$*, or*
2. *There exists a constant* $\rho > 0$ *such that the ratio between the number of singleton topics on the two partitions is* $> 1 + \rho$ *at time* t_0*.*

Further, in both cases, the algorithm converges to assigning all incoming items to a single partition after some time $t = O(r/\log r)$*, w.h.p.*

Proof. We proceed by induction on the time $t \geq t_0$. We will focus on time $t_0 = r/\log r$, as the argument is similar for larger values of t. Notice that we have two cases at t_0. If there exists a constant $\rho > 0$ such that the ratio between the number of singleton topics on the two partitions is $> 1 + \rho$ at time t, then we are obviously done by case 2.

Therefore, in the following, we will work in the case where the load ratio between the two partitions at time t_0 is $\leq 1 + \rho$. Without loss of generality, assume $1 \leq (\sigma_1 + \kappa)/(\sigma_2 + \kappa) \leq 1 + \rho$.

By Lemma 2, the number of singleton topics at time $t \geq t_0$ is at least a constant fraction of r, w.h.p., and it follows that there exists a constant $\epsilon > 0$ such that the singleton ratio at time t_0 is at most $1+\epsilon$. Also, the probability that an item with $3\log r/2$ distinct topics does not hit any of these known topics is at most $1/r^{3/2}$. Hence, in the following, we can assume w.h.p. that every incoming item is assigned by the intersection rule.

By Lemma 3, the ratio between the topic-to-item quotients of the two partitions at time t_0 is at most $\left(\frac{1/2+2\epsilon}{1/2-2\epsilon}\right)^2$, w.h.p. We now proceed to prove that in this case the greedy assignment process can be coupled with a Polya urn process with $\gamma > 1$, w.h.p., noting that this part of the proof is similar to the coupling argument in [20].

By Lemma 2, for $t \geq r/\log r$ steps, at least $2r/3$ topics have been observed, w.h.p. Therefore, the probability that an item with $3\log r/2$ topics does not hit any of these known topics is at most $1/r^{3/2}$. Hence, in the following, we can safely assume that every incoming item is assigned by the intersection rule.

More precisely, when a new item comes in, we check the intersection with the number of topics on each server, and assign it to the partition with which the intersection is larger. (Or randomly if the intersections are equal.) Given an item τ observed at time $t \geq r/\log r$, let A be the number of topics it has in common with partition 1, and B be the number it has in common with partition 2.

More precisely, fix $j \geq 0$ to be the size of the total intersection with either partition, and let a and b be the values of the intersections with partitions 1 and 2, respectively, conditioned on the fact that $a + b = j$. Let δ be the advantage in terms of *topics* of partition 1 versus partition 2, i.e. $(\sigma_1 + \kappa)/(\sigma_1 + \sigma_2 + 2\kappa) = 1/2 + \delta$, and $(\sigma_2 + \kappa)/(\sigma_1 + \sigma_2 + 2\kappa) = 1/2 - \delta$, where κ is the number of duplicate topics. We now analyze the probability that $a > b$.

We can see this as a one-dimensional random walk, in which we start at 0, and take j steps, going right with probability $(1/2+\delta)$, and left with probability $(1/2-\delta)$. We wish to know the probability that we have finished to the right of 0. Iterating over i, the possible value of our drift to the right, we have that

$$\Pr[a > b] = \sum_{i=[j/2]+1}^{j} \binom{j}{i} \left(\frac{1}{2}+\delta\right)^{i} \left(\frac{1}{2}-\delta\right)^{j-i} =$$
$$\left(\frac{1}{2}+\delta\right)^{[j/2]+1} \sum_{i=0}^{[j/2]} \binom{j}{i} \left(\frac{1}{2}+\delta\right)^{[j/2]-i} \left(\frac{1}{2}-\delta\right)^{i}.$$

Similarly, we obtain that

$$\Pr[a < b] = \left(\frac{1}{2}-\delta\right)^{[j/2]+1} \sum_{i=0}^{[j/2]} \binom{j}{i} \left(\frac{1}{2}+\delta\right)^{i} \left(\frac{1}{2}-\delta\right)^{[j/2]-i}.$$

Since $\delta > 0$, we have that the sum on the right-hand-side of the first equation dominates the term on the right-hand-side of the second equation. It follows that

$$\frac{\Pr[a > b]}{\Pr[a < b]} > \frac{\left(\frac{1}{2}+\delta\right)^{[j/2]+1}}{\left(\frac{1}{2}-\delta\right)^{[j/2]+1}}.$$

Since the two quantities sum up to (almost) 1, we obtain that

$$\Pr[a > b] > \frac{\left(\frac{1}{2}+\delta\right)^{[j/2]+1}}{\left(\frac{1}{2}+\delta\right)^{[j/2]+1} + \left(\frac{1}{2}-\delta\right)^{[j/2]+1}}.$$

Let δ' be the advantage that the first partition has over the second *in terms of number of items*, i.e. $1/2 + \delta' = q_1/(q_1+q_2)$. Using Lemma 3, and setting ϵ to a small constant, we obtain that $\delta \simeq \delta'$. We can therefore express the same lower bound in terms of δ'.

$$\Pr[a > b] > \frac{\left(\frac{1}{2}+\delta'\right)^{[j/2]+1}}{\left(\frac{1}{2}+\delta'\right)^{[j/2]+1} + \left(\frac{1}{2}-\delta'\right)^{[j/2]+1}}.$$

The lower bound on the right-hand-side is the probability that the ball goes in urn 1 in a Polya process with $\gamma = [j/2] + 1$. Importantly, notice that, in this process, we are assigning balls (items) with probability proportional to the number of *balls (items)* present in each bin, and have thus eliminated topics from the choice. Let β_t be the proportion of singletons at time t, i.e. $(\sigma_1 + \sigma_2)/r$. We can then eliminate the conditioning on j to obtain that

$$\Pr[A > B] \geq \sum_{j=1}^{d} \binom{d}{j} (\beta_t/r)^j (1-\beta_t/r)^{d-j} \Pr[a > b|j]. \tag{2}$$

The only case where greedy is coupled with a Polya urn process with undesirable exponent $\gamma \leq 1$ is when $j \leq 1$. However, since an item has at least $3\log r/2$ distinct topics, w.h.p., and $t \geq t_0 = r/\log r$, the probability that we hit $j \leq 1$ topics is negligible. Therefore we can indeed couple our process to a finite Polya urn process with $\gamma > 1$ at time t_0 in the case where the singleton ratio at t_0 is at most $1/2 + \epsilon$, for ϵ a small constant. We can apply the same argument by induction for all times $t \geq t_0$, noticing that, once the load ratio is larger than a fixed constant, it never falls below that constant, except with low probability. □

3.3 Step 2: k Partitions and Convergence

Multiple Partitions. Consider now greedy on $k \geq 3$ partitions, but with no load balancing constraint. We now extend the previous argument to this case.

Let $t \geq r/\log r$, and consider the state of the partitions at time t. If there exists a set of partitions which have a constant fraction more singleton topics than the others, it follows by a simple extension

of Lemma 4 (considering sets of partitions as a single partition) that these heavier partitions will attract all future items and their topics, w.h.p. The only interesting case is when the relative loads of all partitions are close to each other, say within an ϵ fraction. However, in this case, we can apply Lemma 4 to *pairs* of partitions, to obtain that some partition will gain an monopoly.

Lemma 5. *Given a single cluster instance in* $\mathsf{HC}(r, \ell, p, q)$ *with* $p \geq 2 \log r/r$ *and* $q = 0$ *to be split across* k *partitions, the greedy algorithm with no balancing constraints will recover the cluster onto a single partition w.h.p.*

Proof. Let us now fix two bins A and B. Notice that the argument of Lemma 4 applies, up to the point where we compute $\Pr[A > B]$ in Equation 2. Here, we have to condition on either A or B having the maximum number of intersections, i.e., replacing

$$\Pr[\#intersections = j] = \binom{d}{j} (\sigma_t/r)^j (1 - \sigma_t/r)^{d-j}$$

with

$$\Pr[\#intersections = j |\ A \text{ or } B \text{ in argmax }].$$

Notice that the coupling still works for $j \geq 2$. Therefore, it is sufficient to show that

$$\Pr[\#intersections \in \{0, 1\}] \geq$$
$$\Pr[\#intersections \in \{0, 1\} |\ A \text{ or } B \text{ in argmax }].$$

This holds since the event (A or B in argmax) implies that the intersection is less likely to be empty or of size 1. Therefore, the argument reduces to the two bin case. □

Speed of Convergence. Note that, by Chernoff bounds, once one of the partitions acquires a constant fraction more topics from the single cluster than the other partitions, it will acquire all future topics w.h.p. By Lemma 4, it either holds that one of the partitions dominates before time $t_0 = r/\log r$, or that we can couple greedy with a Polya urn process with $\gamma > 1$ after this time. The only remaining piece of the puzzle, before we consider the multi-cluster case, is *how fast* the Polya urn process converges to a configuration where some partition contains a constant fraction more topics than the others.

This question is addressed by Drinea et al. [8], which prove the following two facts about the two-bin and k-bin case, respectively. We state this as a single result below, combining Theorems 2.1, 2.4, and Lemma 4 from the aforementioned paper. A system of two bins is said to ϵ_0-separate if one of the bins acquires a $1/2 + \epsilon_0$ fraction of the balls. A bin B_0 is all-but-δ dominant if B_0 contains at least a $1 - \delta$ fraction of the balls thrown.

Theorem 2 (Speed of Convergence [8])**.** *The following hold.*

1. *Consider a Polya urn process with* $\gamma > 1$*, and two bins, in an arbitrary initial state with at least one ball each. Then there exist constants* ϵ_0 *and* $\lambda > 0$ *such that, after* n *steps, the probability that the two bins fail to* ϵ_0 *separate is at most* $O(n^{-\lambda})$*.*

2. *Consider a Polya urn process with* $\gamma > 1$*, and two bins. Assume that, initially, there are* n_0 *balls in the system, and that bin* B_0 *has an* ϵ_0 *advantage. We throw balls until* B_0 *is all-but-*δ *dominant, for some* $\delta > 0$*. Then, with probability* $1 - e^{\Omega(n_0)}$*,* B_0 *is all-but-*δ *dominant when the system has* $2^{x+z} n_0$ *balls, where* $x = \frac{\log(0.4/\epsilon_0)}{\log(1+(\gamma-1)/(5+4(\gamma-1)))}$*, and* $z = \frac{\log(0.1/\delta)}{2\gamma/(\gamma+1)}$*.*

3. *Suppose that when* n *balls are thrown into a pair of bins, the probability that neither is all-but-*δ *dominant is upper bounded by* $p(n, \delta)$*, non-increasing in* n*. Then when* $1 + kn/2$ *balls are thrown into* k *bins, the probability that none is all-but-*λ *dominant is at most* $\binom{k}{2} p(n, \delta)$*, for* $\lambda = \frac{\delta}{\delta + (1-\delta)(k-1)}$*.*

Convergence argument. We can apply the previous result to bound the convergence time of the algorithm as follows.

Theorem 3. *Given a hidden co-cluster graph in* $\mathsf{HC}(n, r, \ell, p, q)$*, with parameters* $p \geq 2 \log r/r$*,* $q = 0$*, and a single hidden cluster, i.e.,* $\ell = 1$*, to be split across* k *partitions, the following holds. There exists a partition* j *such that, after* $2r/\log r$ *items have been observed, each additional generated item is assigned to partition* j*, w.h.p.*

3.4 Final Step: The General Case

We now complete the proof of Theorem 1 in the general case with $\ell \geq 2$ clusters and $q > 0$. We proceed in three steps. We first show the recovery claim for general $\ell \geq 2$, but $q = 0$ and no balance constraints, then extend it for any $q \leq \log r/(rk)$, and finally show that the balance constraints are practically never violated for this type of input.

Generalizing to $\ell \geq 2$. A first observation is that, even if $\ell \geq 2$, the topics must be disjoint across clusters if $q = 0$. Also, since we assume no balance constraints, the clusters and their respective topics are independent. The assignment problem for clusters then reduces to throwing ℓ balls (the *clusters*) into k bins (the *partitions*). We use concentration bounds on the result bin loads to understand the maximum number of clusters per partition, which in turn bounds the maximum load.

Lemma 6. *Assume a clustered bipartite graph G with parameters $\ell \geq k \log k$, $p \geq 2 \log r/r$, and $q = 0$, to be split onto k partitions with no balance constraints. Then, w.h.p., greedy ensures balanced recovery of G. Moreover, the maximum number of topics per partition is upper bounded by $(1 + \beta)r\ell/k$, w.h.p., where $\beta < 1$ is a constant.*

Proof. Notice that, since the clusters are disjoint and $q = 0$, their corresponding topics must be disjoint. Also, since there is no balance constraint, the clusters and their respective topics are independent. Fix an arbitrary cluster C_i. Let t_i be the first time in the execution when we have observed $2r/\log r$ items from C_i. By Theorem 3, after time t_i there exists a partition P_j such that all future items associated to this hidden cluster will be assigned to P_j, w.h.p. Also, note that, by Lemma 2, the expected number of topics from this cluster that may have been assigned to other partitions by time t_i is at most $r(1 - 1/e^2)$, which implies that at most $8m/9$ total topics may have been assigned to other partitions by this time, w.h.p.

To examine the maximum partition load, we model this process as a balls-into-bins game in which $\ell = k \log k$ balls (the clusters) are distributed randomly across k bins (the partitions). The expected distribution per bin is of ℓ/k clusters, and, by Chernoff bounds, the maximum load per bin is $(1 + \alpha)\ell/k$, with high probability in k, where $0 < \alpha < 1$ is a small constant. This means that a partition may receive number of topics of $(1 + \alpha)r\ell/k$ from the clusters assigned to it. To upper bound the extra load due to duplicates, first recall that at most $8m/9$ total topics from each cluster may be duplicated, w.h.p. In total, since clusters are distinct, we obtain that $8r\ell/9$ total topics will be duplicated, w.h.p. Since these duplicates are distributed uniformly at random, a partition may receive an extra load of $(1 + \alpha)8r\ell/9k$ topics, w.h.p. Choosing small α, we get that the maximum load per partition is bounded by $(1 + \alpha)r\ell/k + (1 + \alpha)8r\ell/9k \leq 1.9r\ell/k$. It is interesting to contrast this to the factor obtained by random assignment of items to partitions. □

Generalizing to $q > 0$. The next step is to show that, as long as $q < \log r/(rk)$, the greedy process is not adversely affected by the existence of out-of-cluster (noise) topics, since out-of-cluster topics have a very low probability of changing the algorithm's assignment decisions.

Lemma 7. *Given $q < \log r/(rk)$, then w.h.p. the greedy process can be coupled with a greedy process on the same input with $q = 0$, where $r/\log r$ topics have been observed for each cluster of topics.*

Proof. We couple the two processes in the following way. We consider a hidden cluster input G built with $q = 0$, and a copy of that input G' where $q = \log r/(rk)$, running the algorithm in parallel on the two graphs. Notice that we can view this process on an item-by-item basis, where in the $q = 0$ copy the algorithm gets presented with an item τ, while in G' the algorithm gets presented with a variant τ' of τ from the same home cluster, which also has out-of-cluster topics, chosen uniformly at random from an arbitrary set Q_h of at most $r/2$ topics.

The key question is whether the greedy assignments are the same for items τ and τ'. We prove that this is indeed the case, with high probability. In particular, we need to show that, w.h.p., the outside topics are not enough to change the decision based on the intersection argmax.

Given an item τ' in G' which belongs to cluster C_i, notice that, by Lemma 2, it has at least $3 \log r/2$ distinct topics in C_i, w.h.p. Let t_i be the first time when at least $r/\log r$ items from C_i have

Dataset	Items	Topics	# of Items	# of Topics	# edges
Book Ratings	Readers	Books	107,549	105,283	965,949
Facebook App Data	Users	Apps	173,502	13,604	5,115,433
Retail Data	Customers	Items bought	74,333	16,470	947,940
Zune Podcast Data	Listeners	Podcasts	80,633	7928	1,037,999

Figure 3: A table showing the data sets and information about the items and topics.

been observed. After time t_i, using Chernoff bounds and the pigeonhole principle, the size of the intersection of τ with one of the k partitions must be of at least $(1-\alpha)(1-1/e)3\log r/2k$, w.h.p., where $\alpha > 0$ is a constant.

We now bound the number of topics that τ has outside C_i. Since $q < \log r/rk$, it follows that τ may have at most $(1+\beta)\log r/k$ topics outside C_i, w.h.p., where β is a constant. For small α and β, we get that the number of home cluster topics of τ exceeds the number of outside topics, w.h.p. In turn, this implies that the two random processes can be coupled for each cluster starting with time t_i, as claimed. □

We can combine Lemma 7 and Theorem 3 to obtain that greedy converges after $2r/\log r$ items have been observed out of each hidden cluster.

The Capacity Constraint. Finally, we extend the argument to show that the partition capacity constraints do not cause the algorithm to change its decisions, with high probability. The proof follows by noticing that the load distributions are balanced across servers as the algorithm progresses, as items are either distributed randomly (before convergence), or to specific partitions chosen uniformly at random (after convergence).

Lemma 8. *On a hidden co-cluster input, greedy without capacity constraints can be coupled with a version of the algorithm with a constant capacity constraint, w.h.p.*

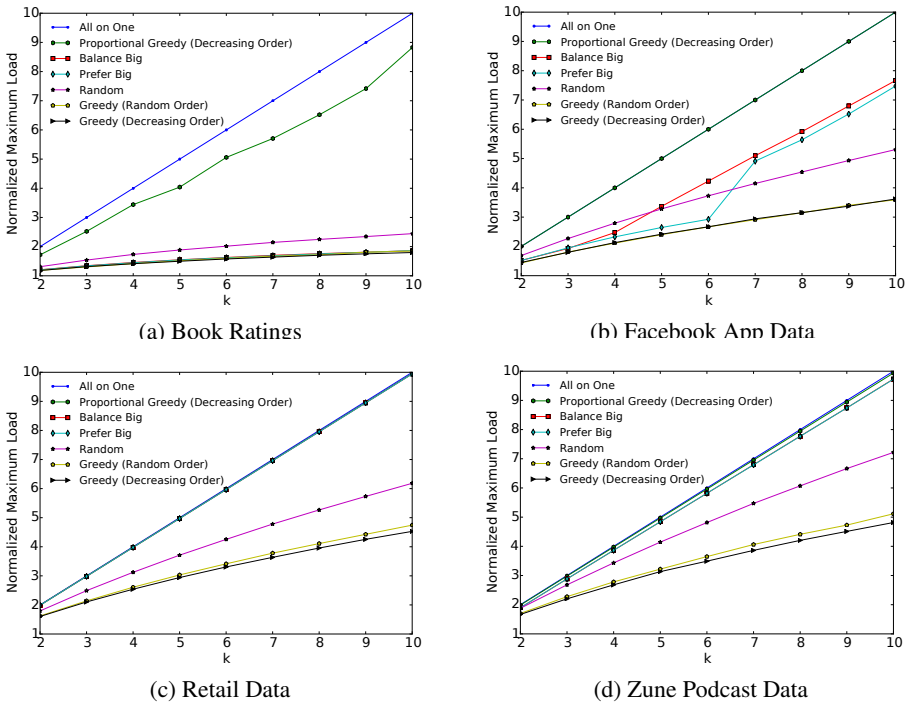
Proof. We can model the assignment process as follows: during the execution, each of the ℓ clusters has its items assigned randomly (at the beginning of the execution), then converges to assigning items to a single server. If we regard this from the point of view of each partition i at some time t, there is a contribution R_i of topics which comes from items in clusters that are still randomly assigned at t, and a contribution F_i of topics coming from items in clusters that have converged. Notice that both these contributions are *balanced across partitions*: each partition has the same probability of being assigned a random cluster; also, since clusters are assigned independently and $\ell \geq k\log k$, the weight coming from converged clusters is also balanced across partitions. Using concentration bounds for each contribution in turn, it follows that the maximally loaded partition is at most a constant fraction more loaded then the minimally loaded one, w.h.p. □

Final Argument. Putting together Lemmas 6, 7 and 8, we obtain that greedy ensures balanced recovery for general hidden cluster inputs in $\mathsf{HC}(n, r, \ell, p, q)$, for parameter values $\ell \geq k\log k$, $p \geq 2\log r/r$, and $q \leq \log r/(rk)$. This completes the proof of Theorem 1.

Moreover, the fact that each cluster is recovered can be used to bound the maximum load of a partition. More precisely, by careful accounting of the cost incurred, we obtain that the maximum load is $2.4r\ell/k$, with high probability, where the extra cost comes from initial random assignments, and from the imperfect balancing of clusters between partitions.

4 Experimental Results

Datasets and Evaluation. We first consider a set of real-world bipartite graph instances with a summary provided in Table 3. All these datasets are available online, except for Zune podcast subscriptions. We chose the consumer to be the item and the resource to be the topic. We provide an experimental validation of the analysis on synthetic co-cluster inputs in the full version of our paper.



(a) Book Ratings (b) Facebook App Data

(c) Retail Data (d) Zune Podcast Data

Figure 4: The normalized maximum load for various online assignment algorithms under different input bipartite graphs versus the numbers of partitions.

In our experiments, we considered partitioning of items onto k partitions for a range of values going from two to ten partitions. We report the maximum number of topics in a partition normalized by the cost of a perfectly balanced solution m/k, where m is the total number of topics.

Online Assignment Algorithms. We compared greedy to the following other online algorithms:

• *All-on-One* trivially assigns all items and topics to one partition.

• *Random* randomly assigns each item to a partition.

• *Balance Big* receives the items in a random order. It assigns the large items to the least loaded partition, and the small items according to greedy. An item is considered large if it subscribes to more than 100 topics, and small otherwise.

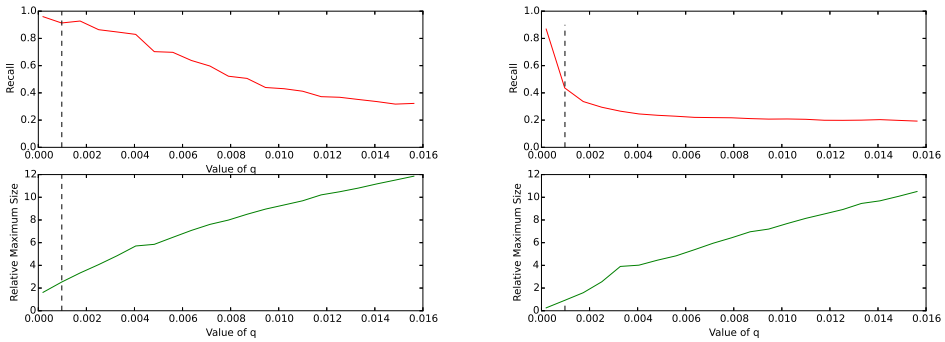
• *Prefer Big* receives the items in a random order. It keeps a buffer of up to 100 small items. When it receives a large item it puts it on the least loaded partition. When the buffer is full, it places all the small items according to greedy.

• *Greedy* is the algorithm we analyzed, which assigns the items to the partition they have the most topics in common with. We considered two variants: items arrive in *random order*, and items arrive in *decreasing order* of the number of topics. We allowed a slack (parameter c) of up to 100 topics in all experiments.

• *Proportional Allocation* receives the items in decreasing order of the number of topics. The probability an item is assigned to a partition is proportional to the number of common topics with the partition.

In addition to these online heuristics, we also compared against offline methods, such as spectral partitioning, label propagation, hMETIS [12], and PARSA [14].

Results. We found that greedy generally outperforms other online heuristics (see Figure 4). Also, the performance of greedy is improved if items arrive in decreasing order of number of topics. This observation seems intuitive: the items with larger number of topics would provide more information



(a) Testing the sufficient condition

(b) Experiments for non-uniform sources.

Figure 5: Experiments for hidden cluster bipartite graphs. The dotted line upper bounds the analytic recovery threshold.

about the underlying structure of the bipartite graph than the items with smaller number of topics. Interestingly, adding randomness to the greedy assignment made it perform far worse; most times Proportional Assignment approached the worst case scenario. The naive random assignment outperformed Proportional Assignment and regularly outperformed Prefer Big and Balance Big item assignment strategies.

Comparison with Offline Techniques. We also tested the streaming algorithm for a wide range of synthetic input bipartite graphs according to the model defined in this paper, and several offline approaches for the problem including hMetis [12], label propagation, basic spectral methods, and PARSA [14]. We found that label propagation and spectral methods are extremely time and memory intensive on our inputs, due to the large number of topics and item-topic edges. (The algorithms took more than a day to return, and often ran out of memory on a 16GB machine.) hMetis returns within seconds, however the assignments were not competitive—we note however that hMetis provides balanced hypergraph cuts, which are not necessarily a good solution to our problem. Compared to PARSA on bipartite graph inputs, greedy provides assignments with up to 3x higher max partition load. On social graphs, the performance difference can be as high as 5x. This discrepancy is natural since PARSA has the advantage of performing multiple passes through the input.

Synthetic Co-Cluster Inputs. We also considered generated hidden co-cluster inputs. In particular, we generated hidden co-cluster graphs for various values of parameters r, ℓ, p, q, and $m = r\ell$. We focus on two measures. The first is *recall*, which is defined as follows: for each cluster, we consider the partition that gets the *highest fraction* of topics from this cluster. We then average these fractions for all clusters, to obtain the recall. The second measure is the *maximum partition size*, i.e., the maximum number of topics on a partition after $m \log m$ items have been observed, normalized by m/k, which is a lower bound on the optimum. We expect these two measures to be correlated, however neither one in isolation would be sufficient to ensure that greedy provides balanced recovery.

When generating the random inputs, we select a random home cluster, then subscribe to topics from the home cluster with probability p. When subscribing to topics from outside the home cluster, we pick *every* topic from outside the cluster independently with probability q (so the noise set Q contains all topics).

Testing the Sufficient Conditions. Our first experiment, presented in Figure 5a, fixes the value of p to $2 \log r/r$, and increases the value of q from $p/(10\ell)$ (below the analytic recovery threshold) to $8p/\ell$ (above the recovery threshold). The dotted line represents an upper bound on the recovery threshold $q = p/(4\ell)$. The experiment shown is for $r = 64, \ell = 64$, and $k = 20$. The results are stable for variations of these parameters.

The experiments validate the analysis, as, below the chosen threshold, we obtain both recall over 90%, and partition size within two of optimal. We note that the threshold value we chose is actually *higher* than the value $q = \log r/(rk)$ required for the analysis.

Non-Uniform Clusters. We repeated the experiment choosing home clusters with *non-uniform probability*. In particular, we select a small set of clusters which have significantly more probability weight than the others. The experimental results are practically identical to the ones in Figure 5a, and therefore omitted. These empirical results suggest that non-uniform cluster probabilities do not affect the algorithm's behavior.

Non-Uniform Topics. Finally, in Figure 5b, we analyze the algorithm's behavior if topics have non-uniform probability weights. More precisely, we pick a small set of topics in each cluster which have disproportionately high weight. (In the experiment shown, four sources out of 64 have $.1$ probability of being chosen.) We observe that this affects the performance of the algorithm, as recall drops at a higher rate with increasing q.

The intuitive reason for this behavior is that the initial miss-classifications, before the algorithm converges, have a high impact on recall: topics with high probability weight will be duplicated on all partitions, and therefore their are no longer useful when making assignment decisions.

5 Related Work

The related problem of min-max multi-way graph cut problem, originally introduced in [24], is defined as follows: given an input graph, the objective is to partition the set of vertices such that the maximum *number of edges* adjacent to a partition is minimized. A similar problem was recently studied, e.g. [1], with respect to *expansion*, defined as the ratio of the sum of weights of edges adjacent to a partition and the minimum between the sum of the weights of vertices within and outside the given partition. The *balanced graph partition* problem is a bi-criteria optimization problem where the goal is to find a balanced partition of the set of vertices that minimizes the total number of edges cut. The best known approximation ratio for this problem is poly-logarithmic in the number of vertices [13]. The balanced graph partition problem was also considered for the set of *edges* of a graph [2]. The related problem of community detection in an input graph data has been commonly studied for the *planted partition model*, also well known as *stochastic block model*. Tight conditions for recovery of hidden clusters are known from the recent work in [17] and [15], as well as various approximation algorithms, e.g. see [3]. Some variants of hypergraph partition problems were studied by the machine learning research community, including balanced cuts studied by [10] using relaxations based on the concept of total variation, and the maximum likelihood identification of hidden clusters [18]. The difference is that we consider the min-max multi-way cut problem for a hypergraph in the streaming computation model.

Streaming computation with limited memory was considered for various canonical problems such as principal component analysis [16], community detection [23], balanced graph partition [21, 22], and query placement [25]. For the class of (hyper)graph partition problems, most of the work is restricted to studying various streaming heuristics using empirical evaluations with a few notable exceptions. A first theoretical analysis of streaming algorithms for balanced graph partitioning was presented in [20] using the framework similar to the one deployed in this paper. The paper gives sufficient conditions for a greedy streaming strategy to recover clusters of vertices for the input graph according to stochastic block model, which makes irrevocable assignments of vertices as they are observed in the input stream and uses memory limited to grow linearly with the number of clusters. As in our case, the argument uses a reduction to Polya urn processes. The two main differences with our work is that we consider a different problem (min-max hypergraph partition) and this requires a novel proof technique based on a two-step reduction to Polya urn processes. Streaming algorithms for the recovery of clusters in a stochastic block model were also studied in [23], under a weaker computation model, which does not require irrevocable assignments of vertices at instances they are presented in the input stream and allows for memory polynomial in the number of vertices.

PARSA [14] considers the same problem in an offline model, where the entire input is initially available to the algorithm, and provides an efficient distributed algorithm for optimizing multiple criteria. A key component of PARSA is a procedure for optimizing the order of examining vertices,

and its efficient implementation. By contrast, we focus on performance under arbitrary arrival order, and provide analytic guarantees under a stochastic input model.

6 Conclusion

We studied the min-max hypergraph partitioning problem in the streaming computation model with the size of memory limited to be at most linear in the number of the parts of the partition. We established first approximation guarantees for inputs according to a random bipartite graph with hidden co-clusters. The problem is re-formulated as a statistical detection problem of hidden co-clusters. We found a set of sufficient conditions for the recovery of hidden co-clusters by a simple and natural streaming algorithm, and evaluated its performance on several real-world input graphs.

There are several interesting open questions for future work. It is of interest to study the tightness of the sufficient recovery conditions, and, in general, better understand the trade-off between the memory size and the accuracy of the recovery. It is also of interest to consider the recovery problem for a wider set of random bipartite graph models. Another question of interest is to consider dynamic graph inputs with addition and deletion of items and topics.

References

[1] N. Bansal, U. Feige, R. Krauthgamer, K. Makarychev, V. Nagarajan, J. SeffiNaor, and R. Schwartz. Min-max graph partitioning and small set expansion. *SIAM J. on Computing*, 43(2):872–904, 2014.

[2] F. Bourse, M. Lelarge, and M. Vojnovic. Balanced graph edge partition. In *Proc. of ACM KDD*, 2014.

[3] Y. Chen, S. Sanghavi, and H. Xu. Clustering sparse graphs. In *Proc. of NIPS*, 2012.

[4] Y. Cheng and G. M. Church. Biclustering of expression data. In *Ismb*, volume 8, pages 93–103, 2000.

[5] F. Chung, S. Handjani, and D. Jungreis. Generalizations of Polya's urn problem. *Annals of Combinatorics*, (7):141–153, 2003.

[6] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proc. of ACM KDD*, 2001.

[7] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proc. of ACM KDD*, 2003.

[8] E. Drinea, A. M. Frieze, and M. Mitzenmacher. Balls and bins models with feedback. In *Proc. of ACM-SIAM SODA*, 2002.

[9] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(75), 2010.

[10] M. Hein, S. Setzer, L. Jost, and S. S. Rangapuram. The total variation on hypergraphs - learning hypergraphs revisited. In *Proc. of NIPS*, 2013.

[11] T. Karagiannis, C. Gkantsidis, D. Narayanan, and A. Rowstron. Hermes: clustering users in large-scale e-mail services. In *Proc. of ACM SoCC*, 2010.

[12] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. *VLSI Design*, 11(3), 2000.

[13] R. Krauthgamer, J. S. Naor, and R. Schwartz. Partitioning graphs into balanced components. 2009.

[14] M. Li, D. G. Andersen, and A. J. Smola. Graph partitioning via parallel submodular approximation to accelerate distributed machine learning. *arXiv preprint arXiv:1505.04636*, 2015.

[15] L. Massoulié. Community detection thresholds and the weak Ramanujan property. In *Proc. of ACM STOC*, 2014.

[16] I. Mitliagkas, C. Caramanis, and P. Jain. Memory limited, streaming PCA. In *Proc. of NIPS*, 2013.

[17] E. Mossel, J. Neeman, and A. Sly. Reconstruction and estimation in the planted partition model. *Probability Theory and Related Fields*, pages 1–31, 2014.

[18] L. O'Connor and S. Feizi. Biclustering using message passing. In *Proc. of NIPS*, 2014.

[19] J. M. Pujol et al. The little engine(s) that could: Scaling online social networks. *IEEE/ACM Trans. Netw.*, 20(4):1162–1175, 2012.

[20] I. Stanton. Streaming balanced graph partitioning algorithms for random graphs. In *Proc. of ACM-SIAM SODA*, 2014.

[21] I. Stanton and G. Kliot. Streaming graph partitioning for large distributed graphs. In *Proc. of ACM KDD*, 2012.

[22] C. E. Tsourakakis, C. Gkantsidis, B. Radunovic, and M. Vojnovic. FENNEL: streaming graph partitioning for massive scale graphs. In *Proc. of ACM WSDM*, 2014.

[23] S.-Y. Yun, M. Lelarge, and A. Proutiere. Streaming, memory limited algorithms for community detection. In *Proc. of NIPS*, 2014.

[24] Z. Z. Svitkina and E. Tardos. Min-max multiway cut. In K. Jansen, S. Khanna, J. Rolim, and D. Ron, editors, *Proc. of APPROX/RANDOM*, pages 207–218. 2004.

[25] B. Zong, C. Gkantsidis, and M. Vojnovic. Herding small streaming queries. In *Proc. of ACM DEBS*, 2015.