Sampling for Approximate Bipartite Network Projection

Nesreen K. Ahmed[1], **Nick Duffield**[2], **Liangzhen Xia**[2]

[1] Intel Labs, CA

[2] Texas A&M University

nesreen.k.ahmed@intel.com, duffieldng@tamu.edu, xialiangzhen1226@gmail.com

Abstract

Bipartite networks manifest as a stream of edges that represent transactions, e.g., purchases by retail customers. Many machine learning applications employ neighborhood-based measures to characterize the similarity among the nodes, such as the pairwise number of common neighbors (CN) and related metrics. While the number of node pairs that share neighbors is potentially enormous, only a relatively small proportion of them have many common neighbors. This motivates finding a weighted sampling approach to preferentially sample these node pairs. This paper presents a new sampling algorithm that provides a fixed size unbiased estimate of the similarity matrix resulting from a bipartite graph stream projection. The algorithm has two components. First, it maintains a reservoir of sampled bipartite edges with sampling weights that favor selection of high similarity nodes. Second, arriving edges generate a stream of *similarity updates* based on their adjacency with the current sample. These updates are aggregated in a second reservoir sample-based stream aggregator to yield the final unbiased estimate. Experiments on real world graphs show that a 10% sample at each stage yields estimates of high similarity edges with weighted relative errors of about 10^{-2}.

1 Introduction

Networks arise as a natural representation for data, where nodes represent people/objects and edges represent the relationships among them. The recent years have witnessed a tremendous amount of research devoted to the analysis and modeling of complex networks [Liben-Nowell and Kleinberg, 2007]. Bipartite networks are a special class of networks represented as a graph $G = (U, V, K)$, whose nodes divide into two sets U and V, with edges allowed only between two nodes that belong to different sets, *i.e.*, $(u, v) \in K$ is an edge, only if $u \in U$ and $v \in V$. Thus, bipartite networks represent relationships between two different types of nodes. Bipartite networks are a natural model for many systems and applications. For example, bipartite networks are used to model the relationships between users/customers and the products/services they consume. General examples include collaboration networks in which actors are connected by a common collaboration act (*e.g.*, author-paper, actor-movie) and opinion networks in which users are connected by shared objects (*e.g.*, user-product, user-movie, reader-book). Clearly, a bipartite network manifests as a *stream of edges* representing the transactions between two types of nodes over time, *e.g.*, retail customers purchasing products daily. Moreover, these dynamic bipartite networks are usually large, due to the prolific amount of activity carrying a wealth of useful behavioral data for business analytics.

While the bipartite representation is indeed useful by itself, many applications focus on analyzing the relationships among a particular set of nodes [Zhou *et al.*, 2007]. For the convenience of these applications, a bipartite network is usually compressed by using a one-mode projection (*i.e.*, projection on one set of the nodes), this is called *bipartite network projection*. For example, for a one-mode projection on U, the projected graph will contain only U-nodes and two nodes $u, u' \in U$ are connected if there is at least one common neighbor $v \in V$, such that $(u, v) \in K$ and $(u', v) \in K$. This results in the U-projection graph $G_U = (U, K_U, C)$ which is a weighted graph characterized by the set of nodes U, and the edges among them K_U. The matrix $C = \{C(u, u')\}_{U \times U}$ represents the weighted adjacency matrix for the U-projection graph, where the weight $C(u, u')$ represents the strength of the similarity between the two nodes $u, u' \in U$.

How to weight the edges has been a key question in one-mode projections and their applications. Several weighting functions were proposed. For example, neighborhood-based methods [Ning *et al.*, 2015; Zhou *et al.*, 2007] measure the similarity between two nodes proportional to the overlap of their neighbor sets. Another example in [Fouss *et al.*, 2007] uses random walks to measure the similarity between nodes. Finding similar nodes (*e.g.*, users, objects, items) in a graph is a fundamental problem with applications in recommender systems [Koren, 2008], collaborative filtering [Herlocker *et al.*, 2004], social link prediction [Liben-Nowell and Kleinberg, 2007], text analysis [Salton *et al.*, 1993], among others.

Motivated by these applications, we study the bipartite network projection problem in the *streaming* computational model [Muthukrishnan, 2005]. Given a bipartite network whose edges arrive as a stream in some arbitrary order, we

compute the projection graph (*i.e.*, weighted matrix C) as the stream is progressing. We focus on *common neighbors* as the weight function. The common neighbors weight is defined for any two nodes $u, u' \in U$, as the size of the intersection of their neighborhood sets $\Gamma(u)$ and $\Gamma(u')$, where $\Gamma(u) \subset V$ is the set of neighbors of u. Thus, their projected weight is $C(u, u') = |\Gamma(u) \cap \Gamma(u')|$. It is convenient to think of a bipartite network as a (binary) matrix $A \in \mathbb{R}^{U \times V}$, where the rows represent the node set U, and the columns represent the node set V. In this case, computing the U-projection matrix using common neighbors is equivalent to $C = AA^\intercal$, where $C \in \mathbb{R}^{U \times U}$. In addition, the common neighbors is a fundamental component in many weighing functions (*e.g.*, cosine similarity), such as those used in collaborative filtering.

The naive solution for this problem is to compute $C = AA^\intercal$ exhaustively with $\mathcal{O}(|U|^2)$ for space and time complexity. However, this is unfeasible for streaming/large bipartite networks [Muthukrishnan, 2005; Ahmed *et al.*, 2014b]. Instead, given a streaming bipartite network (whose edges arrive over time), our goal is to compute a sample of the projection graph that contains an unbiased estimate of the *largest entries* in the projection matrix C.

Contributions. Our main contribution is a novel single-pass, adaptive, weighted sampling scheme in fixed storage for approximate bipartite projection in streaming bipartite networks. Our approach has three steps. First, we maintain a weighted edge sample from the streaming bipartite graph. Second, we observe that the number of common neighbors $C(u, u')$ between two vertices u and $u' \in U$ is equal to the number of wedges (u, v, u') connecting them, where $(u, v) \in K$ and $(u', v) \in K$ for some $v \in V$. Thus, each bipartite edge arriving to the sample generates unbiased estimators of updates to the similarity matrix through the wedges it creates. Third, a further sample-based aggregation accumulates estimates of the projection graph in fixed-size storage.

2 Framework

Problem Definition and Key Intuition. Let $G = (U, V, K)$ be a bipartite graph, and $\Gamma(u) = \{v : (u, v) \in K\}$ denote the set of neighbors of $u \in U$. We study the problem of *bipartite network projection* in data streams, where G is compressed by using a one-mode projection. Thus, for a one-mode projection on U, the *projected graph* will contain only U-nodes and two nodes $u, u' \in U$ are connected if there is at least one common neighbor $v \in V$, such that $(u, v) \in K$ and $(u', v) \in K$. This results in the U-projection graph $G_U = (U, K_U, C)$ which is a weighted graph characterized by the set of nodes U, and the edges among them K_U. The matrix $C = \{C(u, u')\}_{U \times U}$ represents the weighted adjacency matrix for the U-projection graph, where the weight $C(u, u')$ represents the strength of the similarity between any two nodes $u, u' \in U$. In this paper, we propose a novel approximation framework based on sampling to avoid the direct computation of all pairs in C.

Definition 1 (APPROXIMATE BIPARTITE PROJECTION). *Given a bipartite network $G = (U, V, K)$ with (binary) adjacency matrix $A \in \mathbb{R}^{U \times V}$: find the vertex pair $(u, u') \in U \times U$ that maximizes $C = AA^\intercal$. More generally, assume a given*

parameter k, find the k vertex pairs $\{(u_1, u_1'), \ldots, (u_k, u_k')\}$ corresponding to the k largest entries in C.

Definition 1 corresponds to finding the pairs with largest number of common neighbors. Intuitively, the number of common neighbors $C(u, u')$ between two vertices $u, u' \in U$, is equivalent to the number of wedges (u, v, u') connecting them, where $(u, v) \in K$ and $(u', v) \in K$ for some $v \in V$.

Streaming Bipartite Network Projection. Bipartite networks are used to model dynamically evolving transactions represented as a stream of edges between two types of nodes over time. In the *streaming bipartite graph* model, edges K arrive in some arbitrary order $\{e_i : i \in [|K|]\}$. Let $K_t = \{e_i : i \in [t]\}$ denote the first t arriving edges, $G_t = (U_t, V_t, K_t)$ the bipartite graph induced by the first t arriving edges, and C_t the corresponding similarity matrix. We aim to estimate the largest entries of C_t for any t.

2.1 Adaptive Bipartite Graph Sampling

We construct a weighted fixed-size reservoir sample of bipartite edges in which edge weights dynamically adapt to their topological importance (*i.e.*, priority). For a reservoir of size m, we admit the first m edges, while for $t > m$, the sample set comprises a subset $\widehat{K}_t \leq K_t$ of the first t arriving edges, with fixed size $|\widehat{K}_t| = m$ for each $t \geq m$. This is achieved by provisionally admitting the arriving edge at each $t > m$ to the reservoir, then discarding one of the resulting $m+1$ edges by the random mechanism that we now describe.

Since edges are assumed unique, each edge e_i is identified with its the arrival order $i \in [|K|]$. All sampling outcomes are determined by independent random variables β_i, uniformly distributed in $(0, 1]$, assigned to each edge i on arrival. Any edge present in the sample at time $t \geq i$ possess a weight $w_{i,t}$ whose form is described in Section 2.2. The **priority** of i at time t is defined as $r_{i,t} = w_{i,t}/\beta_i$. Edge i is provisionally admitted to the reservoir forming the set $\widehat{K}_i' = \widehat{K}_{i-1} \cup \{i\}$, from which we then discard the edge with minimal priority, whose value is called the **threshold**.

Theorem 1 below establishes unbiased estimators of edge counts. Define the **edge indicator** $S_{i,t}$ taking the value 1 if $t \geq i$ and 0 otherwise. We will construct inverse probability **edge estimators** $\widehat{S}_{i,t} = I(i \in \widehat{K}_t)/q_{i,t}$ of $S_{i,t}$ and prove they are unbiased. This entails showing that $q_{i,t} = \min\{1, \min_{i \leq s \leq t} w_{i,s}/z_s\}$ is the probability that $i \in K_t$, conditional on the set $\mathcal{Z}_{i,t} = \{z_i, \ldots, z_t\}$ of thresholds $z_s = \min_{j \in \widehat{K}_s'} r_{j,s}$ since its arrival.

Theorem 1. $\widehat{S}_{i,t}$ *is an unbiased estimator of $S_{i,t}$.*

Proof. Trivially $\widehat{S}_{i,t} = 0 = S_{i,t}$ for $t < i$. For $t \geq i$ let $z_{i,t} = \min_{j \in \widehat{K}_t \setminus \{i\}} r_{j,t}$. Observe $i \in \widehat{K}_t$ iff $r_{i,s}$ is not the smallest priority in any $\widehat{K}_s'$ for all $s \in [i, t]$. In other words

$$\{i \in \widehat{K}_t\} = \cap_{s \in [i,t]}\{\frac{w_{i,s}}{\beta_i} > z_{i,s}\} = \{\beta_i < \min_{s \in [i,t]} \frac{w_{i,s}}{z_{i,s}}\}$$

Thus $\mathbb{P}[i \in \widehat{K}_t | \mathcal{Z}_{i,t}] = \tilde{q}_{i,t} := \min\{1, \min_{s \in [i,t]} w_{i,s}/z_{i,s}\}$. Note that $\tilde{q}_{i,t} = q_{i,t}$ when $i \in \widehat{K}_t$ since then $z_{i,s} = z_s$ for all

$s \in [i, t]$. Hence when $t \geq i$

$$\mathbb{E}[\widehat{S}_{i,t}|\mathcal{Z}_{i,t}] = \mathbb{P}[I(i \in \widehat{K}_t)|\mathcal{Z}_{i,t}]/\tilde{q}_{i,t} = 1 = S_{i,t} \quad (1)$$

independent of $\mathcal{Z}_{i,t}$ and hence $\mathbb{E}[\widehat{S}_{i,t}] = S_{i,t}$. $\qquad \square$

Let $z_t^* = \max \mathcal{Z}_{m,t}$. Theorem 2 shows $p_{i,t} = \min\{1, \min_{i \leq s \leq t} \frac{w_{i,s}}{z_s^*}\}$ can be used in place of $q_{i,t}$. This simplifies computation since: (a) each $p_{i,t}$ uses the same z_t^*; (b) updates of $p_{i,t}$ can be deferred until t at which $w_{i,t}$ increases.

Theorem 2. *If $t \rightarrow w_{i,t}$ is non-decreasing for each i then $q_{i,t} = p_{i,t}$ and hence $\widehat{S}_{i,t} = I(i \in \widehat{K}_t)/p_{i,t}$ for all $t \geq i$.*

Proof. Let d_t denote the edge discarded during processing arrival t. By assumption, i is admitted to $\widehat{K}_i$ and since $w_{j,t}$ is non-decreasing in t, $z_i = z_{i,i} = w_{d_i,i}/\beta_{d_i} \geq w_{d_i,s}/\beta_{d_i} > z_s$ for all $s \in [d_i, i]$ in order that $d_i \in K_s$ for all $s \in [d_i, i-1]$. Iterating the argument we obtain that $z_i \geq \mathcal{Z}_{m,i}$ and hence $z_i = z_i^*$ and $p_{i,i} = q_{i,i}$. The argument is completed by induction. Assume $p_{i,s} = q_{i,s}$ for $s > i$. If in addition $z_{t+1} > z_t^*$, then $z_{t+1}^* = z_{t+1}$ and hence $p_{i,s+1} = q_{i,s+1}$. If $z_{s+1} \leq z_s^*$ then $z_s = z_{s+1}^*$ and hence $w_{i,s+1}/z_{s+1} \geq w_{i,s+1}/z_{s+1}^* \geq w_{i,s}/z_{s+1}^* = w_{i,s}/z_s^*$. Thus we replace z_s by z_{s+1}^* in the definition of $q_{i,s+1}$ but use of either leaves its value unchanged, since by hypothesis both exceed $q_{i,s} \leq w_{i,s}/z_{i,s}^*$. $\qquad \square$

2.2 Edge Sampling Weights

We now specify the weights used for edge selection. The total similarity of node $u \in U$ is $\mathcal{C}(u) = \sum_{u' \in U} C(u, u') = \sum_{v \in \Gamma(u)}(|\Gamma(v)|-1)$. Thus, the effective contributions of an edge (u, v) to the total similarities $\mathcal{C}(u)$ and $\mathcal{C}(v)$ are $|\Gamma(v)|-1$ and $|\Gamma(u)|-1$ respectively. This relation indicates that if we wish to sample nodes $u \in U, v \in V$ with high total similarities $\mathcal{C}(u)$ and $\mathcal{C}(v)$ as vertices in the edge sample, we should sample nodes with high degrees $|\Gamma(u)|$ and $|\Gamma(v)|$. For adaptive sampling, an edge $e = (u, v) \in \widehat{K}_t'$ has weight

$$w(u, v) = |\widehat{\Gamma}_t(u)| + |\widehat{\Gamma}_t(v)| \quad (2)$$

where $\widehat{\Gamma}_t(u)$, $\widehat{\Gamma}_t(v)$ are the neighbor sets of u, v in the graph $\widehat{G}_t'$ induced by $\widehat{K}_t'$. We also consider a non-adaptive variant in which edges weights are computed on arrival as above, but remain fixed thereafter.

2.3 Unbiased Estimation of Similarity Weights

Consider first generating the exact similarity C_t from the truncated stream K_t. Each arriving edge $e_i = (u, v)$, $i \leq t$ contributes to $C_t(u, u')$ through wedges (u, v, u') for $v \in \Gamma_t(u) \cap \Gamma_t(u')$. Thus to compute $C_t(u, u')$ we count the number of such wedges occurring up to time t, *i.e.*,

$$C_t(u, u') = \sum_{i=1}^{t} \sum_{v \in \Gamma_i(u) \cap \Gamma_i(u')} \big(I(u(e_i) = u))S_{(u',v),i-1} + I(u(e_i) = u'))S_{(u,v),i-1}\big) \quad (3)$$

where $u(e)$ denotes the initial node of edge e. By linearity, we obtain an unbiased estimate $\widehat{C}_t$ of C_t by replacing each $S_{(u,v),i-1}$ by its unbiased estimate $\widehat{S}_{(u,v),i-1}$. Each arriving edge $e_i = (u, v)$ generates an increment to $\widehat{C}_t(u, u')$ for all edges (u', v) in $\widehat{K}_i$, the increment size being the corresponding value of $\widehat{S}_{(u',v),i-1}$, namely, $1/p_{(u',v),i-1}$.

Algorithm 1: Adaptive Sampling for Bipartite Projection

Input: Stream of Bipartite Graph Edges in $U \times V$;
Edge Sample Size m; Similarity Sample Size n;
Output: Sample Similarity Edges $\widehat{K}_U, \widehat{K}_V$; Estimate $\widehat{C}$

```
1  Procedure SIMADAPT(m, n)
2      K = ∅; z* = 0 ;
3      AGGREGATE.INITIALIZE (n) ;
4      while (new edge (u, v)) do
5          foreach (u' ∈ Γ(v)) do
6              UPDATEEDGE((u', v), z*) ;
7              AGGREGATE.ADD ((u', u), 1/p(u', v))
8          foreach (v' ∈ Γ(u)) do
9              UPDATEEDGE((u, v'), z*) ;
10             AGGREGATE.ADD ((v', v), 1/p(u, v'))
11         if |K| < m then
12             INSERTEDGE (u, v) ;
13         else if
               w(u, v)/β(u, v) < min_{(u',v')∈K} w(u', v')/β(u', v')
           then
14             z* = max{z*, w(u, v)/β(u, v)}
15         else
16             INSERTEDGE((u, v)) ;
17             (u*, v*) = arg min_{(u',v')∈K} w(u', v')/β(u', v') ;
18             z* = max{z*, w(u*, v*)/β(u*, v*)} ;
19             DELETEEDGE((u*, v*))

20 Procedure UPDATEEDGE((ũ, ṽ), z̃)
21     if (z̃ > 0) then
22         p(ũ, ṽ) = min{p(ũ, ṽ), w(ũ, ṽ)/z̃}

23 Procedure INSERTEDGE(ũ, ṽ)
24     foreach (u'' ∈ Γ(ṽ)) do
25         UPDATEEDGE((u'', ṽ), z*) ; w(u'', ṽ) ++
26     foreach (v'' ∈ Γ(ũ)) do
27         UPDATEEDGE((ũ, v''), z*) ; w(ũ, v'') ++
28     K = K ∪ {(ũ, ṽ)}; w(ũ, ṽ) = |Γ(ũ)| + |Γ(ṽ)|; p(ũ, ṽ) = 1

29 Procedure DELETEEDGE(ũ, ṽ)
30     K = K \ {(ũ, ṽ)}; Delete p(ũ, ṽ)
```

```
31 Procedure SIMQUERY()
32     (K̂_U ∪ K̂_V, Ĉ) = AGGREGATE.QUERY ()
```

2.4 Aggregation of Similarity Updates

The above construction recasts the problem of reconstituting the sums $\{\widehat{C}_t(u, u') : (u, u') \in U \times U\}$ as the problem of aggregating the stream of key-value pairs

$$\left\{\left((u, u'), p_{(u',v),i-1}^{-1}\right) : i \in [t], e_i = (u, v), (u', v) \in \widehat{K}_i\right\}$$

Exact aggregation would entail allocating storage for every key (u, u') in the stream. Instead, we use weighted sample-based aggregation to provide unbiased estimates of the $\widehat{C}_t(u, u')$ in fixed storage. Specific aggregation algorithms with this property include Adaptive Sample & Hold [Estan and Varghese, 2002], Stream VarOpt [Cohen *et al.*, 2011] and Priority-Based Aggregation (PBA) [Duffield *et al.*, 2017]. Each of these schemes is weighted, inclusion of new items having probability proportional to the size $1/p_{(u,v),i-1}$ of an update. Weighted sketch-based methods such as L_p sampling [Andoni *et al.*, 2011; Monemizadeh and Woodruff, 2010] could also be used, but with space factors that grow polylogarithmically in the inverse of the bias, they are less able to take advantage of smoothing from aggregation.

Estimation Variance. Inverse probability estimators [Horvitz and Thompson, 1952] like those in Theorem 1 furnish unbiased variance estimators computed directly from the estimators themselves; see [Tillé, 2006]. For $\widehat{S}_{i,t}$ this takes the form $\widehat{V}_{i,t} = \widehat{S}_{i,t}(p_{i,t}^{-1} - 1)$, with the unbiasedness property $\mathbb{E}[\widehat{V}_{i,t}] = \text{Var}(\widehat{S}_{i,t})$. These have performed well in graph stream applications [Ahmed *et al.*, 2017]. The approach extends to the composite estimators with sample-based aggregates, using variance bounds and estimators established for the methods listed above, combined via the Law of Total Variance. We omit the details.

2.5 Algorithms

Alg. 1 defines SIMADAPT which implements Adaptive Priority Sampling for bipartite edges, and generates and aggregates a stream of similarity updates. It accepts two reservoir size parameters: m for streaming bipartite edges, and n for similarity matrix estimates. Aggregation of similarity increments uses the class AGGREGATE, which has three methods. INITIALIZE initializes sampling in a reservoir of a given size; ADD aggregates an update to the similarity estimate; QUERY returns the estimate of the similarity graph at any point in the stream. Each arriving edge generates similarity updates for each adjacent edge as inverse probabilities (lines 7, 10).

The bipartite edge sample is maintained in a priority queue K based on increasing order of edge priority, which for each (u, v) is computed as the quotient of the edge weight $w(u, v)$ (the sum of the degrees of u and v) and a permanent random number $\beta(u, v) \in (0, 1]$, generated on demand as a hash of the unique edge identifier (u, v). An arriving edge is inserted (line 12)) if the current occupancy is less than m. Otherwise, if its priority is less than the current minimum, it is discarded and the threshold z^* updated (line 14). If not, the arriving edge replaces the edge of minimum priority (lines 16–19). Edge insertion increments the weights of each adjacent edge (lines 25 and 27). Since $w_{i,t}$ and z_t^* are non-decreasing in t, the update of $p_{i,t}$ (*i.e.*, $p_{i,t} = \min\{p_{i,t-1}, w_{i,t}/z_t^*\}$) (line 22) is deferred until $w_{i,t}$ increases (lines 25, 27) or $p_{i,t}$ is used in a similarity update (lines 6, 9).

A variant SIMFIXED uses (non-adaptive) sampling for bipartite edge sampling with fixed weights. It is obtained by modifying Algorithm 1 as follows. Since weights are not updated, the update and increment steps are omitted (lines 6 and 25–27). Edge probabilities are computed on demand as $p(u, v) = \min\{1, w(u, v)/z^*\}$. We compare with SIMUNIF, a variant of SIMFIXED with unit weights.

Data Structure and Time Cost. We implement the priority queue as a *min-heap* [Cormen *et al.*, 2001] where the root position points to the edge with the lowest priority. Access to the lowest priority edge is $O(1)$. Edge insertions are $O(\log m)$ worst case. In SIMADAPT, each insertion of an edge (u, v) increments the weights of its neighboring edges. Each weight increment may change its edge's priority, requiring its position in the priority queue to be updated. The worst case cost for heap update is $O(\log m)$. But since the priority is incremented, the edge is bubbled down by exchanging with its lowest priority child if that has lower priority.

dataset	Bipartite				Similarity	
	$|U|+|V|$	$|K|$	$d_{\max}$	d_{avg}	$|K_U|$	$|R_U|$
RATING	2M	6M	12K	5	204M	203
MOVIE	62K	3M	33K	90	1.2M	6,797
GITHUB	122K	440K	4K	7	22.3M	156

Table 1: Datasets and characteristics. Bipartite graph: $|U|+|V|$: #nodes, $|K|$: #edges, $d_{\max}$: max. degree, d_{avg}: average degree. Similarity graph: $|K_U|$: #edges in source similarity, $|R_U|$ # dense ranks = #distinct weights in source similarity graph.

Space Cost. The space requirement is $O(|\widehat{V}|+|\widehat{U}|+m+n)$, where $|\widehat{U}|+|\widehat{V}|$ is the number of nodes in the reservoir, with m and n the capacities of the edge and similarity reservoirs.

3 Evaluation

Datasets. Our evaluations use three datasets comprising bipartite real-world graphs publicly available at Network Repository [Rossi and Ahmed, 2015]. Basic properties are listed in Table 1. In the bipartite graph $G = (U, V, K)$, $|U|+|V|$ is the number of nodes in both partitions, $|K|$ is the number of edges, $d_{\max}$ and d_{avg} are maximum and average degrees. $|K_U|$ is the number of edges in the source partition and $|R_U|$ the number of dense ranks, i.e. the number of distinct similarity values. In RATING (*rec-amazon-ratings*) an edge indicates a user rated a product; in MOVIE (*rec-each-movie*) that a user reviewed a movie, and in GITHUB (*rec-github*) that a user is a member of a project. The experiments used a 64-bit desktop equipped with an Intel® Core™ i7 Processor with 4 cores running at 3.6 GHz.

Accuracy Metrics. Since applications such as recommendation systems rank based on similarity, our metrics focus on accuracy in determining higher similarities that dominate recommendations with metrics that have been used in the literature; see e.g., [Gunawardana and Shani, 2009].

Dense Rankings and their Correlation. We compare estimated and actual rankings of the similarities. We use *dense ranking* in which edges with the same similarity have the same rank, and rank values are consecutive. Dense ranking is insensitive to permutations of equal similarity edges and reduces estimation noise. We use the integer part of the estimated similarity to reduce noise. To assess the linear relationship between the actual and estimated ranks we use Spearman's rank correlation on top-k actual ranks. For each edge $e = (u, u')$ in the actual similarity graph, let r_e and $\widehat{r}_e$ denote the dense ranks of $C(u, u')$ and $\lfloor \widehat{C}(u, u') \rfloor$. Cor($k$) is the top-$k$ rank correlation, i.e., over pairs $\{(r_e, \widehat{r}_e) : e \in K_{U,k}\}$ where $K_{U,k} = \{e \in K_U : r_e \leq k\}$.

Weighted Relative Error. We summarize relative errors by weighting by the actual edge similarity, and for each k, we compute the top-k *weighted relative error* WRE(k) as,

$$\sum_{(u,u')\in K_{U,k}} \left| \widehat{C}(u, u') - C(u, u') \right| \Big/ \sum_{(u,u')\in K_{U,k}} C(u, u')$$

Baseline Methods. We compare against two baseline methods. First, SIMPLE takes a uniform sample of the bipartite

dataset	metric	SIMFIXED		SIMADAPT	
		top-100	top-Max	top-100	top-Max
RATING	WRE	0.027	0.089	0.012	0.072
	1-Cor	*0.021*	*0.022*	*0.009*	*0.012*
MOVIE	WRE	0.006	0.122	0.002	0.135
	1-Cor	*0.004*	*0.018*	*0.001*	*0.025*
GITHUB	WRE	0.094	0.128	0.064	0.120
	1-Cor	*0.100*	*0.069*	*0.046*	*0.053*

Table 2: Performance of SIMADAPT, SIMFIXED with $f_m = 10\%$ edge sampling, $f_n = 10\%$ PBA. WRE and $1 - \text{Cor}$. Max rank is $\{200, 6400, 150\}$ for $\{$RATING, MOVIE, GITHUB$\}$

edge stream, and forms an unbiased estimate of $C(u, u')$ by $|\widehat{\Gamma}(u) \cap \widehat{\Gamma}(u')|/p^2$ where p is the bipartite edge sampling rate. Second, we compare with sampling-based approach to link prediction in graph streams recently proposed in [Zhao *et al.*, 2016], which investigated several similarity metrics. We use CNHASH to denote its common neighbor (CN) estimate adapted to the bipartite graph setting. CNHASH uses a separate edge sample per node of the full graph, sampling a fixed maximum reservoir size L per node using min-hashing to coordinate sampling across different nodes in to order promote selection of common neighbors. Similarity estimates are computed across node pairs. Unlike our methods, CN-HASH does not offer a fixed bound on the total edge sample size in the streaming case because neither the number of nodes nor the distribution of edges is known in advance. We attribute space costs for CNHASH using constant space per vertex property of the sketch described in [Zhao *et al.*, 2016], and map this to an equivalent edge sampling rate f_m, normalizing with the space-per-edge costs of each method. For a sample aggregate size n, we apply our metrics to the CN-HASH similarity estimates of the top-n true similarity edges.

Experimental Setup. We applied SIMADAPT, SIMFIXED and SIMUNIF to each dataset, using edge sample reservoir size m a fraction f_m of the total edges, and sample aggregation reservoir size n a fraction f_n of the edges of the actual similarity graph. MOVIE and GITHUB used $f_m \in \{5\%, 10\%, 15\%, 20\%, 25\%, 30\%\}$. The RATING achieved the same accuracy with smaller sampling rates $\{1\%, 5\%, 10\%\}$. Second stage sampling fractions were $f_n \in \{5\%, 10\%, 15\%, 100\%\}$, where 100% is exact aggregation.

4 Results

Comparison of Proposed Methods. For SIMADAPT and SIMFIXED, Table 2 summarizes the metrics WRE and $1-\text{Cor}$ applied to $\{$RATING, MOVIE, GITHUB$\}$ for both top-100 and maximal dense ranks of $\{200, 6400, 150\}$ respectively. The sampling rates are $f_m = 10\%$ for bipartite edges $f_n = 10\%$ PBA for similarity edges. SIMADAPT performs noticeably better for the top-100 dense ranks, with errors ranging from 0.1% to 6.4% representing an error reduction of between 32% and 73% relative to SIMFIXED. The methods have similar accuracy up to maximal ranks.

Accuracy and Bipartite Edge Sample Rate f_m. Figure 1 shows metric dependence on edge sample rate f_m for top-100 dense ranks, using WRE on MOVIE (right) and $1 - \text{Cor}$
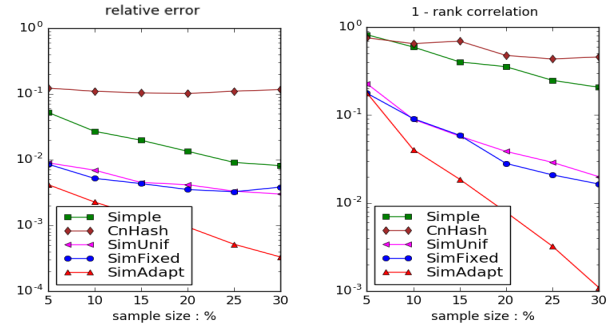

Figure 1: Dependence on bipartite edge sample rate f_m. Left: WRE on MOVIE. Right: $1 - \text{Cor}(k)$ on GITHUB. Top-100 dense ranks.
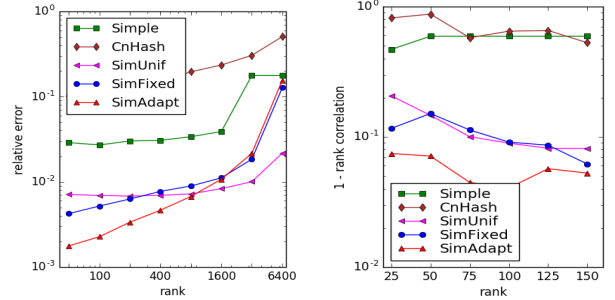

Figure 2: Dependence on rank. Left: WRE on MOVIE. Right: $1 - \text{Cor}(k)$ on GITHUB. Top-k dense ranks for k up to top-Max.

on GITHUB (left). Each figure has curves for SIMADAPT, SIMFIXED and SIMUNIF (and baseline methods CNHASH and SIMPLE discussed below). We observe that SIMADAPT obtains up to an order of magnitude reduction in both metrics for $f_m \geq 20\%$.

Accuracy and Similarity Rank. Figure 2 displays the same metric/data combinations as Figure 1 with $f_m = 10\%$ for top-k ranks as a function of k. As expected, SIMADAPT is most accurate for lower ranks that it is designed to sample well, with WRE 0.002 for MOVIE at $k = 50$ growing to about 0.2 at maximum rank considered. SIMUNIF performed slightly better at high ranks, we believe because it was directing relatively more resources to high rank edges.

Accuracy and Aggregation Sampling Rate f_n. In all datasets the PBA second stage had little effect on accuracy for sampling rates f_n down to about 10% or less under a wide variety of parameter settings. Figure 3 shows results for SIMFIXED applied to MOVIE at fraction $f_m = 10\%$ and PBA sampling rates of 5% and 15%, specifically WRE and $1 - \text{Cor}$ for the top-k dense ranks, as a function of k. For k up to several hundred, even 5% PBA sampling has little or no effect, while errors roughly double when nearly all ranks are included. SIMUNIF, and to a lesser extent SIMFIXED, exhibited more noise, even at *higher* bipartite sampling rates f_m, which we attribute to a greater key diversity of updates (being less concentrated on high similarities) competing for space. Indeed, this noise was absent with exact aggregation.

Baseline Comparisons. Figures 1 and 2 include metric curves for the baseline methods SIMPLE and CNHASH.
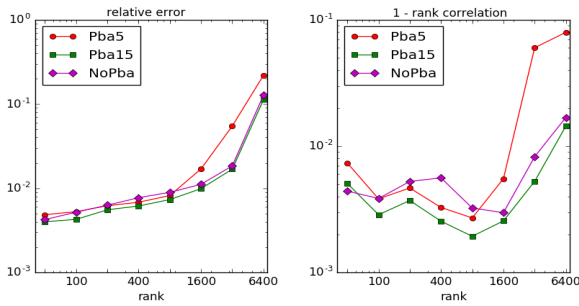
Figure 3: Sample based aggregation w/ PBA $f_n = 5\%, 15\%$, and none. Left: WRE. Right: $1-\mathrm{Cor}$, as function of top-k ranks. Dataset MOVIE with SIMFIXED $f_m = 10\%$ bipartite edge sampling.

SIMADAPT and SIMFIXED typically performed better than SIMPLE by at least an order of magnitude. In some experiment with higher edge sample rate f_m, SIMUNIF was less accurate than SIMPLE, we believe due to the noise described above; Our methods performed noticeably better than CN-HASH in all cases, while CNHASH was often no better than SIMPLE. The reasons for this are two-fold. First, in the streaming context, CNHASH does not make maximal use of its constant space per vertex for nodes whose degree is less than maximum L. However, *even counting only the stored edges*, CNHASH performs worse than our methods for storage use equivalent to our edge sampling rate $f_m < 15\%$. This second reason is the interaction of reservoir design with graph properties. Using shared edge buffer, SIMADAPT and SIM-FIXED devote resources to high adjacency edges associated with high similarity in a sparse graph. Edges incident at high degree nodes are more likely to acquire future adjacencies.

Noise reduction was employed for similarity estimates comprising a small number of updates. These exhibit noise from inverse probability estimators without the benefit of smoothing. We maintained an update count per edge and filtered estimates with count below a threshold. Most benefit was obtained by filtering estimates of update count below 10; this was used in all experiments reported above. Figure 4 compares the effects of no filtering (top) with filtering at threshold 10 (right) applied to SIMADAPT with $f_m = 10\%$ edge sample, for the approximately 1,000 similarity edges in the top 100 estimated dense ranks. The left column shows actual and forecast weights. Without filtering, noise in the estimated similarity curve is due to a few edges whose estimated similarity greatly exceeds the actual similarity due to estimation noise, These are largely absent after filtering. The right column shows a scatter of (actual, forecast) ranks. Observe the cluster of edges with high actual rank (i.e. lower actual weight) and overestimated weight present with no filtering, that are removed by filtering.

5 Related Work

A number of problems specific to bipartite graphs have recently attracted attention in the streaming or semi-streaming context. The classic problem of bipartite matching has been considered for semi-streaming [Eggert *et al.*, 2012; Kliemann, 2011] and streaming [Goel *et al.*, 2012] data. Identifying top-k queries in graphs streams has been studied in [Pan and Zhu, 2012]. The Adaptive Graph Priority Sampling
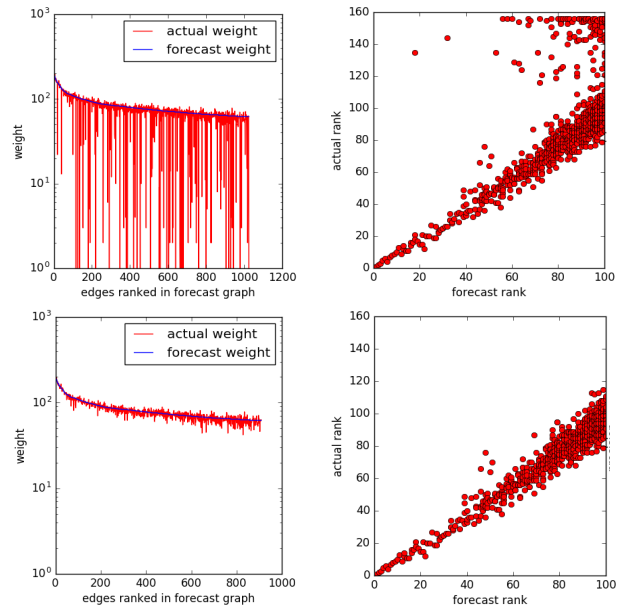

Figure 4: Noise and Filtering. GITHUB $f_m = 10\%$. Top-100 dense ranked edges. Top: No filtering. Bottom: filter threshold 10. Left: forecast and actual weights. Right: scatter of (forecast, actual) ranks.

of this paper builds on the graph priority sampling framework GPS in [Ahmed *et al.*, 2017] while the second sample aggregation method appears in [Duffield *et al.*, 2017]. Graph stream sampling for subgraph counting is addressed in [Ahmed *et al.*, 2017; Jha *et al.*, 2015; Stefani *et al.*, 2017; Zakrzewska and Bader, 2017; Ahmed *et al.*, 2014a] amongst others; see [Ahmed *et al.*, 2014b] for a review. [Zhao *et al.*, 2016] is closer to our present work in that it provides a sample-based estimate of the CN count, albeit not specialized to the bipartite context. We make a detailed comparison of design and performance of [Zhao *et al.*, 2016] with our proposed approach in in Section 4.

6 Summary and Conclusion

This paper has proposed a sample-based estimator of the similarity (or projection graph) induced by a bipartite edge stream, i.e., the weighted graph whose edge weights or similarities are the numbers of common neighbors of its endpoint nodes. The statistical properties of real-world bipartite graphs provide an opportunity for weighted sampling that devotes resources to nodes with high similarity edges in the projected graph. Our proposed algorithm provides unbiased estimates of similarity graph edges in fixed storage without prior knowledge of the graph edge stream. With a relatively small sample of bipartite and the similarity graph edges (10% in each case), and with the enhancement of count based filtering of similarity edges at threshold 10, the sampled similarity edge set reproduces the actual similarities of sampled edges with errors of about 10^{-2} for top-100 dense estimate ranked edges, rising to an error of about 10^{-1} when most estimated edges are considered. Indeed, for the parameters used, the rank distribution of the sampled similarity graph is very similar to that of the actual graph for all but the highest ranks.

References

[Ahmed *et al.*, 2014a] Nesreen K. Ahmed, Nick Duffield, Jennifer Neville, and Ramana Kompella. Graph sample and hold: A framework for big-graph analytics. In *SIGKDD*, 2014.

[Ahmed *et al.*, 2014b] Nesreen K. Ahmed, Jennifer Neville, and Ramana Kompella. Network sampling: From static to streaming graphs. *In TKDD*, 8(2):1–56, 2014.

[Ahmed *et al.*, 2017] Nesreen K. Ahmed, Nick Duffield, Theodore L. Willke, and Ryan A. Rossi. On sampling from massive graph streams. *Proc. VLDB*, 10(11):1430–1441, August 2017.

[Andoni *et al.*, 2011] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Streaming algorithms via precision sampling. In *IEEE FOCS*, pages 363–372, Oct 2011.

[Cohen *et al.*, 2011] Edith Cohen, Nick Duffield, Haim Kaplan, Carsten Lund, and Mikkel Thorup. Efficient stream sampling for variance-optimal estimation of subset sums. *SIAM J. Comput.*, 40(5):1402–1431, September 2011.

[Cormen *et al.*, 2001] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. 2nd edition, 2001.

[Duffield *et al.*, 2017] Nick Duffield, Yunhong Xu, Liangzhen Xia, Nesreen K. Ahmed, and Minlan Yu. Stream aggregation through order sampling. In *CIKM*, 2017.

[Eggert *et al.*, 2012] Sebastian Eggert, Lasse Kliemann, Peter Munstermann, and Anand Srivastav. Bipartite matching in the semi-streaming model. *Algorithmica*, 63:490–508, 2012.

[Estan and Varghese, 2002] Cristian Estan and George Varghese. New directions in traffic measurement and accounting. In *SIGCOMM*, 2002.

[Fouss *et al.*, 2007] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE TKDE*, 19(3):355–369, 2007.

[Goel *et al.*, 2012] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proc. SODA '12*, pages 468–485, Philadelphia, PA, USA, 2012.

[Gunawardana and Shani, 2009] Asela Gunawardana and Guy Shani. A survey of accuracy evaluation metrics of recommendation tasks. *J. Mach. Learn. Res.*, 10, 2009.

[Herlocker *et al.*, 2004] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[Horvitz and Thompson, 1952] Daniel Horvitz and Donovan Thompson. A generalization of sampling without replacement from a finite universe. *J. of the American Stat. Assoc.*, 47(260):663–685, 1952.

[Jha *et al.*, 2015] Madhav Jha, C. Seshadhri, and Ali Pinar. A space-efficient streaming algorithm for estimating transitivity and triangle counts using the birthday paradox. *ACM Trans. Knowl. Discov. Data*, 9(3):15:1–15:21, 2015.

[Kliemann, 2011] Lasse Kliemann. *Matching in Bipartite Graph Streams in a Small Number of Passes*, pages 254–266. Springer, Berlin, Heidelberg, 2011.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD*, pages 426–434, 2008.

[Liben-Nowell and Kleinberg, 2007] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.*, 58(7):1019–1031, 2007.

[Monemizadeh and Woodruff, 2010] Morteza Monemizadeh and David Woodruff. 1-pass relative-error lp-sampling with applications. In *SODA*, 2010.

[Muthukrishnan, 2005] S. Muthukrishnan. *Data streams: Algorithms and applications*. Now Publishers Inc, 2005.

[Ning *et al.*, 2015] Xia Ning, Christian Desrosiers, and George Karypis. *A Comprehensive Survey of Neighborhood-Based Recommendation Methods*, pages 37–76. Springer US, Boston, MA, 2015.

[Pan and Zhu, 2012] Shirui Pan and Xingquan Zhu. Continuous top-k query for graph streams. In *Proc. CIKM '12*, pages 2659–2662, New York, NY, USA, 2012.

[Rossi and Ahmed, 2015] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015. `http://networkrepository.com`.

[Salton *et al.*, 1993] Gerard Salton, James Allan, and Chris Buckley. Approaches to passage retrieval in full text information systems. In *ACM SIGIR*, pages 49–58, 1993.

[Stefani *et al.*, 2017] Lorenzo De Stefani, Alessandro Epasto, Matteo Riondato, and Eli Upfal. TriÈst: Counting local and global triangles in fully dynamic streams with fixed memory size. *ACM TKDD*, 11(4):43:1–43:50, 2017.

[Tillé, 2006] Yyes Tillé. *Sampling Algorithms*. Springer-Verlag, 2006.

[Zakrzewska and Bader, 2017] Anita Zakrzewska and David A Bader. Streaming graph sampling with size restrictions. In *IEEE/ACM ASONAM*, 2017.

[Zhao *et al.*, 2016] Peixiang Zhao, Charu Aggarwal, and Gewen He. Link prediction in graph streams. In *ICDE*, pages 553–564, 2016.

[Zhou *et al.*, 2007] Tao Zhou, Jie Ren, Matúš Medo, and Yi-Cheng Zhang. Bipartite network projection and personal recommendation. *Physical Review E*, 76(4):046115, 2007.