Brief Announcement: Graph Matching in Massive Datasets

Soheil Behnezhad[*]
University of Maryland

Mahsa Derakhshan[*]
University of Maryland

Hossein Esfandiari[†]
University of Maryland

Elif Tan[‡]
Ankara University

Hadi Yami[*]
University of Maryland

ABSTRACT

In this paper we consider the maximum matching problem in large bipartite graphs. We present a new algorithm that finds the maximum matching in a few iterations of a novel edge sampling technique. This algorithm can be implemented in big data settings such as *streaming setting* and *MapReduce setting*, where each iteration of the algorithm maps to one pass over the stream, or one MapReduce round of computation, respectively. We prove that our algorithm provides a $1 - \varepsilon$ approximate solution to the maximum matching in $1/\varepsilon$ rounds which improves the prior work in terms of the number of passes/rounds. Our algorithm works even better when we run it on real datasets and finds the exact maximum matching in 4 to 8 rounds while sampling only about %1 of the edges.

1 INTRODUCTION

The best offline algorithm known for the maximum matching problem runs in $O(m\sqrt{n})$ time and $O(m)$ space [13] for a graph G with n vertices and m edges. For massive graphs, this is quite inefficient, especially when the input graph is dense (i.e., $m \gg n$). To overcome this problem, there are several recent attempts to design efficient algorithms in *streaming* and/or *distributed* settings (e.g. MapReduce)[2–4, 7, 9, 11, 12]. All of these algorithms give approximate solutions to the maximum matching, i.e., for some $0 < \alpha \leq 1$ they report a solution which is at least α fraction of the maximum matching.

In streaming setting, a graphs is given as a sequence of edges. The algorithm is allowed to take a few passes over the input (usually constant or logarithmic), and has access to a local memory significantly smaller than the input graph (i.e. $o(n^2)$).

McGregor [12] provided a $1 - \varepsilon$ approximation algorithm for unweighted general graphs and a $0.5 - \varepsilon$ approximation algorithm for weighted graphs in constant (exponential in $1/\varepsilon$) passes over the input, using $\tilde{O}(n)$ space. For bipartite unweighted graphs, Ahn and Guha [1] improved the number of passes to $O(1/\varepsilon^2 \log\log 1/\varepsilon)$. On a related note, Ahn and Guha [2] later presented a MapReduce algorithm that achieves a $1 - \varepsilon$ approximation in $O(p/\varepsilon)$ rounds, using $O(n^{1+1/p})$ local space.

There are also some recent attempts towards estimating the size of the maximum matching in these settings [3, 4, 7, 9] and some attempts to find the maximum matching when its size is small [4–6].

In this paper, we present an algorithm that finds the maximum matching in a few iterations of a novel edge sampling technique. Our algorithm iteratively samples the edges of the input graph based on the minimum vertex cover of the currently sampled edges. In fact, each iteration of the algorithm maps to one pass over the stream in the streaming setting, or one Map-Reduce round of computation. We prove that after $1/\varepsilon$ iterations our algorithm gives a $1-\varepsilon$ approximation, improving the result of Ahn and Guha [1] in terms of the number of passes.

The simplicity of our algorithm makes it implementable in common big data settings such as streaming and Map-Reduce models. Our experiments indicate that on real world datasets, it works much better than our theoretical guarantees. More precisely, it finds the exact maximum matching in 4 to 8 of rounds, while sampling only about %1 of edges in total.

2 NOTATIONS

The input, denoted by $\mathcal{G}$, is an undirected bipartite graph with n vertices. Throughout this paper, $\mathcal{V}(\mathcal{G})$ denotes the vertex set of $\mathcal{G}$ and $\mathcal{E}(\mathcal{G})$ denotes the edge set of $\mathcal{G}$. For any arbitrary subgraph H of $\mathcal{G}$, $\mathcal{G} \setminus H$ is a subgraph of $\mathcal{G}$ where $\mathcal{V}(\mathcal{G} \setminus H) = \mathcal{V}(\mathcal{G}) - \mathcal{V}(H)$ and an edge $\{u, v\} \in \mathcal{E}(\mathcal{G})$ is in $\mathcal{E}(\mathcal{G} \setminus H)$ if and only if u and v are not in $\mathcal{V}(H)$. We may also abuse this notation and use $\mathcal{G} \setminus V$ when V is a subset of $\mathcal{V}(\mathcal{G})$ and not a sub-graph of $\mathcal{G}$. Moreover, we denote the set of edges in the maximum matching of a graph G by $\mathcal{M}_G$.

3 ALGORITHM

In this section we explain our algorithm to find a maximum matching in bipartite graphs (Figure 1). The input graph $\mathcal{G}$ and a real number α are given. Our algorithm consists of

[*]Supported in part by NSF CAREER award CCF-1053605, NSF BIGDATA grant IIS-1546108, NSF AF:Medium grant CCF-1161365, DARPA GRAPHS/AFOSR grant FA9550-12-1-0423, and another DARPA SIMPLEX grant. Emails: {soheil,mahsaa,hadiyami}@cs.umd.edu.
[†]Supported in part by Google PhD Fellowship in Market Algorithms. Email: hossein@cs.umd.edu.
[‡]Supported by TUBITAK-2219 International Postdoctoral Research Scholarship Programme. Email: etan@ankara.edu.tr.

SPAA '17, July 24-26, 2017, Washington DC, USA
© 2017 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4593-4/17/07.
https://doi.org/10.1145/3087556.3087601

```
 1: procedure MaximumMatchingBySampling($\mathcal{G}$, $\alpha$)
 2:     $\mathcal{S}_0, \mathcal{C}_0 \leftarrow \emptyset$
 3:     $i \leftarrow 1$
 4:     while $|\mathcal{E}(\mathcal{G} \setminus \mathcal{C}_{i-1})| > 0$ do
 5:         $U \leftarrow n\alpha$ edges sampled u.a.r from $\mathcal{G} \setminus \mathcal{C}_{i-1}$
 6:         $\mathcal{S}_i \leftarrow \mathcal{S}_{i-1} \cup U$
 7:         $\mathcal{C}_i \leftarrow$ vertices in minimum vertex cover of $\mathcal{S}_i$
 8:         $i \leftarrow i + 1$
 9:     end while
10:     return maximum matching of $\mathcal{S}_{i-1}$
11: end procedure
```

Figure 1: Maximum Matching via Sampling

several rounds, and in each round, we add some edges to our sample until there are no more valid edges to add. More precisely, $\mathcal{S}_i$ denotes our sample at round i and $\mathcal{C}_i$ denotes a minimum vertex cover of $\mathcal{S}_i$. Initially $\mathcal{S}_0$ and $\mathcal{C}_0$ are empty sets. For any $i > 0$, $\mathcal{S}_i$ is the union of $\mathcal{S}_{i-1}$ and a sample of at most $n\alpha$ edges chosen uniformly at random from the candidate set $\mathcal{E}(\mathcal{G} \setminus \mathcal{C}_{i-1})$. This means in the first round the candidate edges that we sample from is the edge set of $\mathcal{G}$, and $\mathcal{S}_1$ is a set of $n\alpha$ edges that are chosen uniformly at random from $\mathcal{E}(\mathcal{G})$. We continue this until round r where there are no edges in the candidate set $\mathcal{E}(\mathcal{G} \setminus \mathcal{C}_r)$. This implies that all the edges in $\mathcal{G}$ are covered by $\mathcal{C}_r$. Note that in a bipartite graph the size of the minimum vertex cover is equal to the size of the maximum matching by Kőnig's theorem [10]. Hence we have $|\mathcal{M}_{\mathcal{S}_r}| = |\mathcal{C}_r|$. On the other hand $\mathcal{C}_r$ is also a vertex cover of $\mathcal{G}$ since it covers every edge in $\mathcal{G}$, therefore the size of the maximum matching of $\mathcal{G}$ is at most $|\mathcal{C}_r|$. This means $|\mathcal{M}_{\mathcal{G}}| = |\mathcal{M}_{\mathcal{S}_r}|$ and it suffices to return the maximum matching of our last sample when there are no edges in the candidate set.

4 ANALYSIS

In this section we analyze Algorithm 1 and show that by setting $\alpha = \sqrt{n}$, our algorithm gives a $(1-\varepsilon)$-approximation for the maximum matching problem in streaming setting, using $O(\frac{n^{1.5}}{\varepsilon})$ space and $O(\frac{1}{\varepsilon})$ passes.

We first prove that every edge that is not covered by the vertex cover in the i-th round is added to the sample in the next round, with high probability. We define the notion of *critical induced subgraphs* as follows:

Definition 4.1 (Critical Induced Subgraph). An induced subgraph $\mathcal{K}$ of $\mathcal{G}$ is *critical*, if $|\mathcal{E}(\mathcal{K})| \geq n^{1.5}$.

We show that the probability that Algorithm 1 does not sample any edge from an arbitrary critical induced subgraph in the first round is too small that even the union bound over all critical induced subgraphs is still very low. This means that with high probability any critical induced subgraph has at least one edge in $\mathcal{S}_1$.

Lemma 4.2. *With probability at least $1 - (\frac{2}{e})^n$, any critical induced subgraph of $\mathcal{G}$ has at least one edge in $\mathcal{S}_1$, if $\alpha = \sqrt{n}$.*

Proof. Let $\mathcal{K}$ be an arbitrary critical induced subgraph of $\mathcal{G}$. For any arbitrary edge w of $\mathcal{G}$, let $\bar{\mathrm{P}}(w, \mathcal{S}_1)$ denote the probability that w is not in $\mathcal{S}_1$. We also use $\bar{\mathrm{P}}(\mathcal{K}, \mathcal{S}_1)$ to denote the probability that none of the edges of $\mathcal{K}$ are in $\mathcal{S}_1$. By fixing a random edge x of $\mathcal{K}$ we first give an upper bound for $\bar{\mathrm{P}}(x, \mathcal{S}_1)$ to further show $\bar{\mathrm{P}}(\mathcal{K}, \mathcal{S}_1)$ is very small.

Note that we sample $n\sqrt{n}$ edges of $\mathcal{E}(\mathcal{G})$ uniformly at random in the first step, thus $\bar{\mathrm{P}}(x, \mathcal{S}_1) = 1 - \frac{n\sqrt{n}}{|\mathcal{E}(\mathcal{G})|}$. This means $\bar{\mathrm{P}}(x, \mathcal{S}_1)$ has the highest value when $|\mathcal{E}(\mathcal{G})|$ is large, but the number of the edges in a graph with n vertices could not be more than n^2. Therefore

$$\bar{\mathrm{P}}(x, \mathcal{S}_1) < 1 - \frac{n\sqrt{n}}{n^2} = 1 - \frac{1}{\sqrt{n}}. \quad (1)$$

We take advantage of the fact that all the edges in $\mathcal{S}_1$ are chosen uniformly at random to give an upper bound for $\bar{\mathrm{P}}(\mathcal{K}, \mathcal{S}_1)$ using $\bar{\mathrm{P}}(x, \mathcal{S}_1)$. Boolean random variables in $\{X_1, \ldots, X_n\}$ are *negatively correlated* if:

$$\Pr[\bigwedge_{i \in S} X_i = 1] \leq \prod_{i \in S} \Pr[X_i] \qquad \forall S \subseteq \{1, \ldots, n\} \quad (2)$$

For any edge w of $\mathcal{G}$, we define the boolean random variable $\bar{\mathrm{X}}(w, \mathcal{S}_1)$ to be 1 if w is not in $\mathcal{S}_1$ and 0 otherwise (note that $\bar{\mathrm{P}}(w, \mathcal{S}_1) = \Pr[\bar{\mathrm{X}}(w, \mathcal{S}_1)]$). Since the edges in $\mathcal{S}_1$ are chosen uniformly at random, the boolean random variables in $\mathcal{X}_{\mathcal{G}} = \{\bar{\mathrm{X}}(y, \mathcal{S}_1) \mid y \in \mathcal{E}(\mathcal{G})\}$ are negatively correlated [8], therefore the probability that none of the edges of $\mathcal{K}$ are in $\mathcal{S}_1$, is not more than $(1 - n^{-0.5})^{|\mathcal{E}(\mathcal{K})|}$, more formally let $\mathcal{X}_{\mathcal{K}} = \{\bar{\mathrm{X}}(y, \mathcal{S}_1) \mid y \in \mathcal{E}(\mathcal{K})\}$, since $\mathcal{X}_{\mathcal{K}} \subseteq \mathcal{X}_{\mathcal{G}}$ by (2):

$$\Pr[\bigwedge_{r \in \mathcal{X}_{\mathcal{K}}} r = 1] \leq \prod_{r \in \mathcal{X}_{\mathcal{K}}} \Pr[r] = \bar{\mathrm{P}}(x, \mathcal{S}_1)^{|\mathcal{E}(\mathcal{K})|}$$

and by (1):

$$\bar{\mathrm{P}}(x, \mathcal{S}_1)^{|\mathcal{E}(\mathcal{K})|} \leq (1 - \frac{1}{\sqrt{n}})^{|\mathcal{E}(\mathcal{K})|}$$

therefore:

$$\Pr[\bigwedge_{r \in \mathcal{X}_{\mathcal{K}}} r = 1] = \bar{\mathrm{P}}(\mathcal{K}, \mathcal{S}_1) \leq (1 - \frac{1}{\sqrt{n}})^{|\mathcal{E}(\mathcal{K})|}. \quad (3)$$

Note that $\mathcal{K}$ is a critical induced subgarph of $\mathcal{G}$ and by definition $|\mathcal{E}(\mathcal{K})| \geq n\sqrt{n}$, thus $\bar{\mathrm{P}}(\mathcal{K}, \mathcal{S}_1) \leq (1 - n^{-0.5})^{n\sqrt{n}}$ and by using the fact that $(1 - \frac{1}{x})^x < \frac{1}{e}$ for any real number $x \geq 1$, we get:

$$\bar{\mathrm{P}}(\mathcal{K}, \mathcal{S}_1) \leq \left(1 - \frac{1}{\sqrt{n}}\right)^{n\sqrt{n}} = \left(\left(1 - \frac{1}{\sqrt{n}}\right)^{\sqrt{n}}\right)^n < (\frac{1}{e})^n.$$

On the other hand every subset of $\mathcal{V}(\mathcal{G})$ is equivalent to an induced subgraph of $\mathcal{G}$. Therefore the total number of critical induced subgraphs of $\mathcal{G}$ does not exceed 2^n. This means the probability that there is at least one critical induced subgraph of $\mathcal{G}$ that has no edge in $\mathcal{S}_1$, which is equal to the union bound of $\bar{\mathrm{P}}(\mathcal{K}, \mathcal{S}_1)$ for all possible critical induced subgraphs, is not more than $2^n \times (\frac{1}{e})^n = (\frac{2}{e})^n$. $\square$

Using Lemma 4.2, we give an $n^{1.5}$ upper bound for the size of $\mathcal{E}(\mathcal{G}/\mathcal{C}_i)$ with high probability. This means every edge in $\mathcal{G}/\mathcal{C}_i$ could be added to the sample in the next round.

Lemma 4.3. *With probability at least $1 - (\frac{2}{e})^n$ every edge in $\mathcal{G}/\mathcal{C}_{i-1}$ is in $\mathcal{S}_i$, for any $i > 1$, If $\alpha = \sqrt{n}$.*

Proof. Note that since $\alpha = \sqrt{n}$, in Line 6 of Algorithm 1, $n\sqrt{n}$ edges of $\mathcal{G}/\mathcal{C}_{i-1}$ are added to the sample. Therefore if the number of edges in $\mathcal{G}/\mathcal{C}_{i-1}$ is less than $n\sqrt{n}$ (which we prove with high probability is the case), $\mathcal{S}_i$ will contain them all as desired.

To prove $|\mathcal{E}(\mathcal{G}/\mathcal{C}_{i-1})| \le n\sqrt{n}$ with high probability, recall that by Lemma 4.2, all critical induced subgraphs of $\mathcal{G}$ have at least one edge in $\mathcal{S}_1$ with probability $1 - (\frac{2}{e})^n$. We show if $|\mathcal{E}(\mathcal{G}/\mathcal{C}_{i-1})| > n\sqrt{n}$ there exists a critical induced subgraph of $\mathcal{G}$ that does not have any edge in $\mathcal{S}_1$, and therefore it happens only with probability $(\frac{2}{e})^n$. To do so, we first prove if an edge e is in $\mathcal{S}_1$, then it cannot be in $\mathcal{G}/\mathcal{C}_{i-1}$. To see this, note that e has to be in $\mathcal{S}_{i-1}$ since the sample set at each round is a super set of the sample set at the previous round (Line 6 of Algorithm 1). This means the vertex cover $\mathcal{C}_{i-1}$ of $\mathcal{S}_{i-1}$, contains at least one of the vertices of e to cover it, which implies at least one of the vertices of e is not in $\mathcal{G}/\mathcal{C}_{i-1}$, and therefore $e \notin \mathcal{E}(\mathcal{G}/\mathcal{C}_{i-1})$. Note that if the number of edges in $\mathcal{G}/\mathcal{C}_{i-1}$ is more than $n\sqrt{n}$, then $\mathcal{G}/\mathcal{C}_{i-1}$ is a critical induced subgraph of $\mathcal{G}$ by Definition 4.1 which does not have any edge in $\mathcal{S}_1$ (or otherwise that edge could not have appeared in $\mathcal{G}/\mathcal{C}_{i-1}$). By Lemma 4.2 this happens only with probability $(\frac{2}{e})^n$.

We proved with probability at least $1 - (\frac{2}{e})^n$, the number of edges in candidate set $\mathcal{G}/\mathcal{C}_{i-1}$ is too small that we can sample them all. $\square$

Theorem 4.5 states that after $\frac{1}{\varepsilon}$ rounds, the maximum matching of the sample is a $(1-\varepsilon)$-approximation for the maximum matching of $\mathcal{G}$ with high probability. To prove this, we first show in Lemma 4.4, that at least $|\mathcal{M}_\mathcal{G}| - |\mathcal{M}_{\mathcal{S}_i}|$ edges of $\mathcal{M}_\mathcal{G}$ are in $\mathcal{S}_{i+1} - \mathcal{S}_i$ with high probability. Then using this lemma we show that before reaching the (1-ε)-approximation, at each round, the size of the maximum matching of the sample increases by at least $\varepsilon|\mathcal{M}_\mathcal{G}|$. Using this observation we give an upper bound for the number of rounds before reaching the approximation.

Lemma 4.4. *If $|\mathcal{M}_{\mathcal{S}_i}| < |\mathcal{M}_\mathcal{G}|$, and we set $\alpha = \sqrt{n}$, with probability at least $1 - (\frac{2}{e})^n$, at least $|\mathcal{M}_\mathcal{G}| - |\mathcal{M}_{\mathcal{S}_i}|$ edges of $\mathcal{M}_\mathcal{G}$ are in $\mathcal{S}_{i+1} - \mathcal{S}_i$.*

Proof. Let k_i denote $|\mathcal{M}_\mathcal{G}| - |\mathcal{M}_{\mathcal{S}_i}|$. Note that $|\mathcal{C}_i| = |\mathcal{M}_{\mathcal{S}_i}|$ since any subgraph of a bipartite graph is also a bipartite graph and the size of the minimum vertex cover of any bipartite graph is equal to the size of its maximum matching. Therefore $|\mathcal{M}_\mathcal{G}| - |\mathcal{C}_i| = k_i$. This means $\mathcal{C}_i$ does not cover at least k_i edges of $\mathcal{M}_\mathcal{G}$, since no two edges of $\mathcal{M}_\mathcal{G}$ could be covered with the same vertex. By Lemma 4.3 every edge in $\mathcal{G}/\mathcal{C}_i$ is in $\mathcal{S}_{i+1}$ with probability at least $1 - (\frac{2}{e})^n$, therefore at least k_i edges of $\mathcal{M}_\mathcal{G}$ are added to $\mathcal{S}_{i+1}$ with probability at least $1 - (\frac{2}{e})^n$. $\square$

Theorem 4.5. *$\mathcal{M}_{\mathcal{S}_r}$ is a (1-ε)-approximation for $\mathcal{M}_\mathcal{G}$ with probability at least $1 - (\frac{2}{e})^{\frac{n}{\varepsilon}}$, if $r \ge \frac{1}{\varepsilon}$ and $\alpha = \sqrt{n}$.*

Dataset	Nodes	Edges	Maximum Matching
`Wiki-Vote`	14230	13595753	7115
`Facebook`	8078	5608736	4039
`Enron-Email`	73384	60599542	36692
`Twitter`	58316	195413186	60810
`Slashdot`	154720	189796532	77360

Table 1: Properties of the datasets

Proof. Let k_i denote $|\mathcal{M}_\mathcal{G}| - |\mathcal{M}_{\mathcal{S}_i}|$. Note that in Line 6 of Algorithm 1 every edge in $\mathcal{S}_i$ is in $\mathcal{S}_{i+1}$, thus $|\mathcal{M}_{\mathcal{S}_i}| \le |\mathcal{M}_{\mathcal{S}_{i+1}}|$. This implies that the size of the matching of the sample is increasing at each round. Therefore for any $m' \le |\mathcal{M}_\mathcal{G}|$ there exists a round i such that $m' \le \mathcal{M}_{\mathcal{S}_t}$.

Using Lemma 4.4 we find a lower bound for the size of the maximum matching in round r based on k_r. Lemma 4.4 states that in round i of this algorithm with probability at least $1 - (\frac{2}{e})^n$, $\mathcal{S}_{i+1}$ includes at least k_i new edges from $\mathcal{M}_\mathcal{G}$. Since for each round j where $j \le i$ we have $k_j \ge k_i$, in each round before r with probability at least $1 - (\frac{2}{e})^n$, at least k_r new edges from $\mathcal{M}_\mathcal{G}$ are added to the sampled graph. Therefore the total number of edges form $\mathcal{M}_\mathcal{G}$ that are guaranteed to be in the $\mathcal{S}_r$ with probability at least $1 - (\frac{2}{e})^{\frac{n}{\varepsilon}}$ is greater than $(r-1) \times \varepsilon|\mathcal{M}_\mathcal{G}|$. Considering these, we have the equation below which shows after round r the minimum number of $\mathcal{M}_\mathcal{G}$ edges the sampled graph has, is smaller than the number of its maximum matching size.

$$(r-1) \times \varepsilon|\mathcal{M}_\mathcal{G}| < (1-\varepsilon)|\mathcal{M}_\mathcal{G}|$$

This equation states that the number of maximum rounds for having a $(1-\varepsilon)$-approximation with probability at least $1 - (\frac{2}{e})^{\frac{n}{\varepsilon}}$ is $\frac{1}{\varepsilon}$. Therefore we have a (1-ε)-approximation for $\mathcal{M}_\mathcal{G}$ with probability at least $1 - (\frac{2}{e})^{\frac{n}{\varepsilon}}$. $\square$

5 EMPIRICAL RESULTS

In this section we describe our experiments on several datasets.

5.1 Datasets

We present our empirical results on the 2-hop neighborhood graph of `Twitter`, `Facebook`, `Slashdot`, `Wiki-Vote`, and `Enron-Email` datasets from *SNAP*[1], a publicly available dataset collection. Note that these networks are not necessarily bipartite. To make them bipartite, we create two partitions by duplicating the original graph's vertex set and adding an edge from a vertex v in a partition to vertex u in the other partition if there is an edge from v to u in the original graph.

Table 1 shows some of the relevant properties of our datasets.

5.2 Experiments

In order to practically evaluate Algorithm 1, we run it on our datasets for different values of α in the range 1-10 to find a maximum matching. We also run it on different induced

[1] http://snap.stanford.edu/data/

α	1	2	5	10	20
`Wiki-Vote`	8	6	5	4	3
`Facebook`	4	3	2	3	2
`Enron-Email`	7	6	4	4	3
`Twitter`	8	7	5	4	4
`Slashdot`	8	6	4	4	4

Table 2: The number of rounds until we find the maximum matching for different values of α.

α	1	2	5	10	20
`Wiki-Vote`	0.7%	0.8%	1.6%	2.5%	2.7%
`Facebook`	0.6%	0.7%	1.4%	1.6%	2.9%
`Enron-Email`	0.6%	0.8%	1.3%	2.2%	2.7%
`Twitter`	0.3%	0.5%	0.7%	1.3%	1.7%
`Slashdot`	0.6%	0.7%	1.1%	1.7%	2.2%

Table 3: The ratio of sampled edges until we find the maximum matching for different values of α.

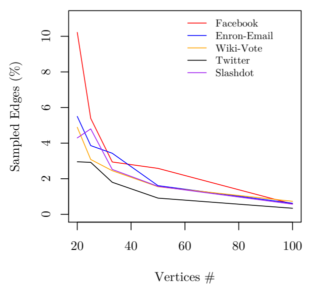
subgraphs of the datasets to compare the results for different number of vertices. To find an induced subgraph of our datasets, we first randomly choose a subset of vertices in the original network and construct the bipartite graph for those users (this means for any selected user we have two vertices).

Some of the important implications of our empirical results are as follows:

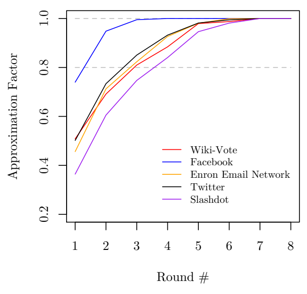
Different number of vertices. as the number of vertices of the induced subgraph increases, the ratio of edges sampled by the algorithm to find a maximum matching decreases accordingly. This property of our algorithm is specially very important for solving large input graphs, which is indeed the main focus of our approach. This property could be observed in Figure 2 (a).

Different values of α. For very large values of α, Algorithm 1 clearly samples all the edges in the first round. However, by decreasing α, although the expected number of rounds needed to find the maximum matching increases, the ratio of edges that we sample decreases. This could be observed in Table 2 and Table 3. Note that even in the worst case (where $\alpha = 1$) the number of rounds does not exceed 8 on any dataset. The above-mentioned property means one can pick a desirable α to balance the number of rounds versus the ratio of sampled edges.

Approximation factor in different rounds. Note that after each round, the approximation factor gets better and better until when we find the maximum matching. According to Figure 2 (b), even when $\alpha = 1$, we can get a 0.8-approximation for all datasets by at most 4 rounds and it takes at most 7 rounds to find the optimal maximum matching.



(a)



(b)

Figure 2

REFERENCES

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

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

[3] M. Bury and C. Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. *arXiv preprint arXiv:1505.02019*, 2015.

[4] R. Chitnis, G. Cormode, H. Esfandiari, M. Hajiaghayi, A. McGregor, M. Monemizadeh, and S. Vorotnikova. Kernelization via sampling with applications to dynamic graph streams. *arXiv preprint arXiv:1505.01731*, 2015.

[5] R. Chitnis, G. Cormode, H. Esfandiari, M. Hajiaghayi, and M. Monemizadeh. Brief announcement: New streaming algorithms for parameterized maximal matching & beyond. In *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures*, pages 56–58. ACM, 2015.

[6] R. Chitnis, G. Cormode, M. Hajiaghayi, and M. Monemizadeh. Parameterized streaming: maximal matching and vertex cover. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1234–1251. SIAM, 2015.

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

[8] K. Joag-Dev and F. Proschan. Negative association of random variables with applications. *The Annals of Statistics*, pages 286–295, 1983.

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

[10] D. Konig. Gráfok és mátrixok. matematikai és fizikai lapok, 38: 116–119, 1931.

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

[12] A. McGregor. Finding graph matchings in data streams. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 170–181. Springer, 2005.

[13] S. Micali and V. V. Vazirani. An $o(\sqrt{|v|}|e|)$ algoithm for finding maximum matching in general graphs. In *Foundations of Computer Science, 1980., 21st Annual Symposium on*, pages 17–27. IEEE, 1980.