Survey on Matching in the Graph-Stream Model

Mrinal Anand
IIT Gandhinagar
mrinal.anand@iitgn.ac.in

Kishen N. Gowda
IIT Gandhinagar
kishen.gowda@iitgn.ac.in

Vraj Patel
IIT Gandhinagar
vraj.patel@iitgn.ac.in

Abstract

In recent years, tremendous amount of network data has been generated and graph is a natural way to represent many of these datasets. Due to the massive size of these graphs, it is not possible to store the entire graph into main memory, creating the need for use of methods like data streaming and distributed processing to generate insights for such graphs. In this survey we mainly focus on work done on the matching problem for graph streams under both single pass and multi pass streaming settings.

Keywords and phrases Semi-Streaming Model, Graph Streams, Approximation Algorithms, Maximum Matching, Multi-pass

1 Introduction

In the past two decades, the world has witnessed a huge improvement in computational hardware in terms of processing power and storage capacity of machines. As a result there has been a rise in mining of massive real world datasets for insights. Most of these massive dataset can be naturally represented in the form of graphs, e.g. social network can be represented as tabular data as well as a graph but graphs are a more natural choice for representation of social network. The YouTube social network dataset [22] had a total of 1,134,890 nodes and 2,987,624 edges in the year 2012. Some other examples of graph datasets are road networks, bitcoin network, citation network etc.

However, analysing these massive graphs via classical algorithms can be challenging task given the sheer size of graph. There are mainly two approaches to handle these massive graphs - (i) Data Streaming and (ii) Distributed Processing. In data streaming model the input is stream of data which is read sequentially using only sub-linear space. In case of graph streaming the input can be either sequence of edges or nodes (possibly both) of a graph. The main advantages of using streaming algorithm is that it can handle massive data with limited amount of *random access memory*, other benefit of streaming model is that it can many-a-time be adapted to be used in an online setting for real-time computation scenarios.

Streaming algorithms use polylogarithmic [$\mathcal{O}(\text{poly} \log n)$] memory space in the size of the input (n). It has been shown in [6] that polylogarithmic memory is insufficient for many graph problems. Even the basic bipartiteness or connectivity problems in graph require $\Omega(n)$ space in memory. Therefore most of the recent work [7, 19] involves semi-streaming model that uses $\mathcal{O}(n \operatorname{poly} \log n)$ space where n is number of nodes in the graph. Under semi-streaming setting most of the graph problems becomes reasonable to solve.

Maximum Matching is a very well-studied fundamental problem in combinatorial optimization. In an unweighted bipartite graph, the optimization problem is to find a maximum cardinality matching. Maximum Matching can be defined as a set of edges of maximum size such that no two adjacent edges are selected. Maximum Bipartite Matching is a special case of maximum matching, in which graph is bipartite. It has been shown that greedy approach for maximal matching yields $\frac{1}{2}$ approximation of maximum matching. It has been proved in [9] that approximation ratio better than $(1 - \frac{1}{e})$ is not possible for semi-streaming models. While we can't do better than $(1 - \frac{1}{e})$ but it still have some scope for improvement.

In this survey, we will be mainly focusing on recent advancement in the field of Maximum Matching and Maximum Bipartite Matching using semi-streaming models. We will be discussing results for single pass and multiple pass settings under both random and adversarial order assumptions.

2 Preliminaries

Let $G = (V, E)$ be a graph with vertex set V and edge set E. We also denote $V(G)$ to be the vertex set and $E(G)$ to be the edge set of graph G. If G is bipartite with bipartition A and B then we write $G = (A, B, E)$ and we denote $V = A \cup B$. Let $n = |V|$ and $m = |E|$. For an edge $e \in E$ with endpoints $u, v \in V$, we denote e by (u, v). Given a vertex $v \in V$, $\deg_G(v)$ denotes the degree of the vertex v (no. of edges incident on v) in the graph $G = (V, E)$. We sometimes use $\deg(v)$ when the graph in context is clear. Given a subset of edges $F \subseteq E$, $V(F)$ denotes set of endpoints of edges in F. In case of a bipartite graph $G = (A, B, E)$, $A(F)$ (or $B(F)$) denotes the set of endpoints of edges in F which belong to A(or B). Also, given a vertex v, $\deg_F(v)$ denotes the no. of edges in F which are incident on v.

▶ **Definition 1. (Matching)**: *A matching in a graph $G = (V, E)$ is a subset of edges $M \subseteq E$ such that $\forall v \in V : \deg_M(v) \leqslant 1$. A maximum matching $M^\star$ is a matching such that for any other matching M', $|M^\star| \geqslant |M'|$. A maximal matching M is a matching that is inclusive-wise maximal, i.e. $\forall e \in E \setminus M$, $M \cup \{e\}$ is not a matching.*

For a subset of edges $F \subseteq E$, $\operatorname{opt}(F)$ denotes the maximum matching in the graph G restricted to edges F. We may substitute $\operatorname{opt}(G)$ for $\operatorname{opt}(E)$ and $M^\star$ for $\operatorname{opt}(G)$. For a set of vertices S and a set of edges F, let $S(F)$ denote the subset of vertices of S covered (or matched) by F. Further, $\overline{S(F)} := S \setminus S(F)$. For $S \subseteq V$, $G[S]$ denotes the graph induced on the vertex set S. We write $\operatorname{opt}(S)$ for $\operatorname{opt}(G[S])$, i.e. a maximum matching in $G[S]$. In case of bipartite graphs, for $S_A \subseteq A$ and $S_B \subseteq B$ we write $\operatorname{opt}(S_A, S_B)$ for $\operatorname{opt}(G[A_S \cup B_S])$. Moreover, for two sets S_1 and S_2, $S_1 \oplus S_2$ denotes the symmetric difference $((S_1 \setminus S_2) \cup (S_2 \setminus S_1))$ of the two sets. For a matching M and a vertex v, $M(v) = u$ if either $(u, v) \in M$ or $(v, u) \in M$. Also for given a graph G and a matching M of G, a vertex v is called a *matched vertex (or node)* if there is an edge incident on v in M and *free vertex (or node)* otherwise.

▶ **Definition 2. (Augmenting Path)**: *Let $p \geqslant 3$ be an odd integer. Then a length p augmenting path with respect to a matching M in a graph $G = (V, E)$ is a path $P = (v_1, \ldots, v_{p+1})$ such that $v_1, v_{p+1} \notin V(M)$ and for $i \leqslant \frac{p-1}{2}$, $(v_{2i}, v_{2i+1}) \in M$ and $(v_{2i-1}, v_{2i}) \notin M$.*

In other words, an augmenting path with respect to a matching is a path of odd length which alternates between traversing an edge not in the matching and an edge that is not in the matching and the starting and ending points are not in the matching in consideration. Since an augmenting path is of odd length and it starts and ends by traversing an edge not in the matching, the number of edges not in the matching (say E_{odd}) is exactly one more than the number of edges in the matching (say E_{even}). What this means is that if there is an augmenting path with respect to a matching in M, then $M' = M \setminus E_{even} \cup E_{odd}$ is a matching of size $|M| + 1$.

Now we introduce some notation and terminology which is commonly used in the graph streaming literature. The graph stream for an input graph $G(V, E)$ to be a sequence of edges arriving one at a time in some order. We define n to be the number of vertices in the graph. Let $\Pi(G)$ be the set of all possible permutations of edges in E. The input for the algorithms is some $\pi \in \Pi(G)$. Further, $\pi[i]$ denotes the i-th edge of π, $\pi[i, j]$ represents the sequence of edges $\pi[i], \pi[i + 1], \ldots, \pi[j]$, $\pi(i, j]$ represents the sequence of edges $\pi[i + 1, j]$ and $\pi[i, j)$ represents the sequence of edges $\pi[i, j - 1]$. When i, j are real numbers, we define $\pi[i]$ to be the same as $\pi[\lfloor i \rfloor]$ and thus $\pi[i, j] = \pi[\lfloor i \rfloor, \lfloor j \rfloor]$.

A k-pass semi-streaming algorithm with update time $t(n)$ is one where for any given $\pi \in \Pi(G)$, the algorithm goes through (or reads) the entire stream π at most k times (i.e. makes at most k "passes" over π) while maintaining a random access memory of size $\mathcal{O}(n \text{ polylog } n)$ and taking at most $\mathcal{O}(t(n))$ time between consecutive reads. One read operation on any stream is assumed to take constant time.

In the analysis of semi-streaming algorithms, assumptions on the ordering of the data in the stream plays a big role in the analysis. Much of the earliest literature on streaming considered the adversarial order of data i.e. the order of the stream is chosen by an adversary to yield the worst possible running time for the algorithm. Therefore, the guarantees given were always with respect to the worst case scenarios. Such adversarial order assumption is not a very practical assumption to work with as in reality, there is no such adversaries who are in control of the data and in some sense, the ordering of the data can be assumed to be random. This is the core reasoning behind the definition of random order streams where the stream is assumed to be randomly samples from all possible orderings and the expected guarantee is analysed and reported under this assumption.

In the current context of matching, even in the semi-streaming model, the problem cannot be solved exactly in constant number of passes. Therefore, any semi-streaming model for the maximum matching problem which takes constant number of passed returns an approximate matching. Therefore, the guarantee in consideration here will be the approximation ratio. A deterministic algorithm $\mathbf{A}$ that solves the matching problem is said to be c-approximate if for any given input graph G, $\mathbf{A}$ outputs a matching M such that $|M| \leqslant c \cdot |\text{opt}(G)|$. In the context of random-order streams, the deterministic algorithm is said to be c-approximate if the expected approximation ratio is c, that is $\mathbb{E}[|M|] \geqslant c \cdot |\text{opt}(G)|$ (where the expectation is taken over all possible arrival orders). When considering randomized semi-streaming algorithms for matching, the algorithm is said to be c-approximate if $\mathbb{E}[|M|] \geqslant c \cdot |\text{opt}(G)|$ where the expectation is taken over the internal random coins rather than the arrival order.

Do note that in all these contexts, during the analysis of the guarantees, the worst case is considered against the graph G is always in consideration (i.e. the graph is considered adversarial/worst case).

3 Semi-Streaming Algorithms for Unweighted Maximum Matching

The simplest and the earliest algorithm for MAXIMUM MATCHING in the semi-streaming model is the Greedy matching algorithm (Algorithm 1) which does the following: Start with an empty matching, when an edge arrives, if both the endpoints of the edge are not already part of any edge in the matching, add the edge to the matching. This Greedy algorithm is a 1 pass 0.5-approximate algorithm which takes linear space and has $\mathcal{O}(\infty)$ update time. The fact that it is a 0.5 approximation follows from the fact that for every edge (u, v) belonging to a maximum matching, the greedy matching being constructed must contain an edge with at least one of u or v as an endpoint.

Algorithm 1 `Greedy`(π) [15]

```
  Require: The input stream $\pi$ is an edge stream of a graph $G = (V, E)$
1 begin
2     $M \leftarrow \varnothing$
3     while edge stream not empty do
4         $e = v_1 v_2 \leftarrow$ next edge in stream
5         if $\{v_1, v_2\} \cap V(M) = \varnothing$ then
6             $M \leftarrow M \cup \{e\}$
7     return $M$
```

The surprising fact is that to this date, this simple algorithm remains the best one-pass deterministic algorithm for MAXIMUM MATCHING as well as MAXIMUM BIPARTITE MATCHING in the adversarial stream order setting.

3.1 Algorithms with Multiple Passes

The standard approach adapted to construct bigger matchings is to find augmenting paths. McGregor [17] presented a $\frac{1}{1+\epsilon}$ approximation for MAXIMUM MATCHING on random stream order. The algorithm first finds a maximal matching using greedy algorithm in one pass followed by multiple passes to construct augmenting paths. To minimize the no. of passes required to construct the augmenting paths, the author constructs an auxiliary layered graph G'. This graph is constructed based on a simple randomized procedure. After constructing the graph G', augmenting paths are constructed over multiple passes. The number of passes required in this algorithm is a complicated function depending heavily on ϵ (see [17]).

Based on the idea explained (in detail) in Algorithm 2 (described in due course), an $\mathcal{O}\left(\frac{\log \frac{1}{\epsilon}}{\epsilon}\right)$-pass semi-streaming algorithm that computes a $\frac{2}{3} - \epsilon$ approximation to the MAXIMUM BIPARTITE MATCHING problem was presented in [7].

3.2 Algorithms with fewer passes

In this section, we will survey semi-streaming algorithms for MAXIMUM MATCHING and MAXIMUM BIPARTITE MATCHING with few ($\leqslant 3$) passes.

To improve the Greedy Algorithm, a simple strategy is to compute a maximal matching M_G in one pass, and then utilize the second and third passes to find 3-augmenting paths. The existence of such 3-augmenting paths is based on the following lemma:

▶ **Theorem 3.** *([15]) Let $\epsilon \geqslant 0$. Let M be a maximal matching of G s.t. $|M| \leqslant \left(\frac{1}{2}+\epsilon\right)|M^\star|$. Then M contains at least $\left(\frac{1}{2}-3\epsilon\right)|M^\star|$ 3-augmentable paths.*

The 3-augmenting paths are found as follows. Firstly, compute a maximal matching M_G using the `Greedy` algorithm. According to Theorem 3, either M_G is better than $\frac{1}{2}$ approximation, or if M_G is close to $\frac{1}{2}$ approximation, then most of the edges in M_G are 3-augmentable. So, in a second pass, compute a maximal matching M_L between the matched vertices $B(M_G)$ and the free vertices $\overline{A(M_G)}$ using the `Greedy` algorithm. M_L will be at least of size $\frac{1}{2}$ times the number of 3-augmenting paths. Edges from M_L will serve as the start of length 3 augmenting paths. Finally, in a third pass, complete these 3-augmenting paths with edges of M_G and M_L by computing another maximal matching M_R.

Algorithm 2 Three Pass Bipartite Matching Algorithm [15]

```
Require: The input stream $\pi$ is an edge stream of a bipartite graph $G = (A, B, E)$
1 begin
2     $M_G, M_L, M_R \leftarrow \varnothing$
3     1st pass: $M_G \leftarrow$ Greedy($\pi$)
4     $G_L \leftarrow$ Induced Graph between $\overline{A(M_G)}$ and $B(M_G)$
5     2nd pass: $M_L \leftarrow$ Greedy($\pi \cap G_L$)
6     $G_R \leftarrow$ Induced Graph between $\overline{B(M_G)}$ and $\{a \in A(M_G) : M_G(a) \in B(M_L)\}$
7     3rd pass: $M_R \leftarrow$ Greedy($\pi \cap G_R$)
8     $M \leftarrow$ matching obtained from $M_G$ augmented by $M_L \cup M_R$
9     return $M$
```

Konrad et. al. [15] presented a series of algorithms with one or two passes for the Maximum Bipartite Matching problem. They essentially simulate the three-pass algorithm (Algorithm 2) in one or two passes and bound the performance based on some crucial properties of the Greedy algorithm.

Firstly, they claim and prove that if in expectation over all input edges sequences the matching computed by the Greedy algorithm is close to a $\frac{1}{2}$ approximation, then Greedy builds this matching early on. In other words, Greedy converges quickly (see Lemma 2 in [15]).

Based on this property, they presented the following deterministic one-pass 0.505 approximation semi-streaming algorithm for the Maximum Bipartite Matching problem on *random stream orders* (Algorithm 3). Essentially, the input stream π is split into three phases: $\pi[1, \alpha m], \pi(\alpha m, \beta m]$ and $\pi(\beta m, m]$ (for $0 < \alpha < \beta < 1$), where these three phases correspond to the three passes in Algorithm 2. The matchings obtained in these three phases are denoted by M_0, M_1 and M_2 respectively. Either the Greedy algorithm already performs well, or the matching obtained by Greedy is close to a $\frac{1}{2}$ approximation. In the latter case, the Greedy algorithm converges early on, hence M_0 is a good representative of the greedy solution. Also, by Theorem 3 almost all the edges of M_0 are 3-augmentable. Thus, the 3-augmenting paths are found with the help of the matchings M_1 and M_2 in a similar manner as in Algorithm 2.

Furthermore, they showed that this algorithm can be adapted to general graphs yielding a deterministic one-pass 0.503 approximation semi-streaming algorithm for the Maximum Matching problem on *random stream orders*.

Secondly, Konrad et. al. present the following property of the Greedy algorithm (See Theorem 4). According to this property, in expectation, any maximal matching of the graph induced on

Algorithm 3 Single Pass Bipartite Matching Algorithm [15]

```
Require: The input stream $\pi$ is an edge stream of a bipartite graph $G = (A, B, E)$
1  begin
2      $\alpha \leftarrow 0.4312, \beta \leftarrow 0.7595$
3      $M_G \leftarrow \texttt{Greedy}(\pi)$
4      $M_0 \leftarrow \texttt{Greedy}(\pi[1, \alpha m])$, matching obtained by Greedy on the first $\lfloor \alpha m \rfloor$ edges
5      $F_1 \leftarrow$ Induced Graph between $\overline{A(M_0)}$ and $B(M_0)$
6      $M_1 \leftarrow \texttt{Greedy}(F_1 \cap \pi(\alpha m, \beta m])$, matching obtained by Greedy on edges
         $\lfloor \alpha m \rfloor + 1$ through $\lfloor \beta m \rfloor$ that intersect $F_1$
7      $F_2 \leftarrow$ Induced Graph between $\overline{B(M_0)}$ and $\{a \in A(M_0) : M_0(a) \in B(M_1)\}$
8      $M_2 \leftarrow \texttt{Greedy}(F_1 \cap \pi(\beta m, m])$, matching obtained by Greedy on edges $\lfloor \beta m \rfloor + 1$
         through $m$ that intersects $F_2$
9      $M \leftarrow$ matching obtained from $M_0$ augmented by $M_1 \cup M_2$
10     return larger of the two matchings $M_G$ and $M$
```

On Line 3, M_G is computed in parallel

a subset $A' \subseteq A$ and B contains a big fraction of the edges which are part of 3-augmenting paths of edges in a maximal matching of G.

▶ **Theorem 4.** *([15]) Let $0 < p \leqslant 1$, let $G = (A, B, E)$ be a bipartite graph. Let A' be an independent random sample such that $\Pr[a \in A'] = p, \forall a \in A$. Let F be the induced bipartite graph between A' and B. Then for any input stream π*

$$\mathbb{E}_{A'}[|\texttt{Greedy}(F \cap \pi)|] \geqslant \frac{p}{p+1}|\mathrm{opt}(G)|$$

Based on the above property, Konrad et. al. present the following randomized two-pass 0.519 approximation semi-streaming algorithm to solve the Maximum Bipartite Matching problem on *arbitrary stream orders* (Algorithm 4). Either the greedy algorithm already performs well, or the matching obtained by Greedy is close to a $\frac{1}{2}$ approximation. In the latter case, applying Theorem 4, an independent random sample $A' \subseteq A$ is picked such that $\Pr[a \in A'] = p$ for all a. In the first pass, the algorithm computes a greedy matching M_0 of G and a matching M' of the graph induced on the vertices $A' \cup B$. Some edges of M' will serve as a start of 3-augmenting paths, call them $M_1 (\subseteq M')$. In a second pass these 3-augmenting paths with edges of M_0 and M_1 are completed with another maximal matching M_2.

Furthermore, they present a deterministic two-pass algorithm on the same line as the above randomized algorithm. Instead of considering a maximal matching of a random subset $A' \subseteq A$, here an edge set S is chosen such that it contains edges which are part of 3-augmenting paths with edges of M_0 (obtained from Greedy matching). The set S is an incomplete λ-bounded semi-matching, i.e. $S \subseteq E$ such that $\deg_S(a) \leqslant 1$ and $\deg_S(b) \leqslant \lambda$, for all $a \in A$ and $b \in B$. A similar theorem like Theorem 4 is claimed and proved. This gives a deterministic two-pass 0.519 approximation semi-streaming algorithm to solve the Maximum Bipartite Matching problem on *arbitrary stream orders*. Further, they adapted this algorithm to general graphs giving a deterministic two-pass 0.5071 approximation semi-streaming algorithm for the Maximum Matching problem on *arbitrary stream orders*.

Gamlath et. al. and Konrad [8, 12] presented improved one-pass algorithms for Maximum Bipartite Matching (0.512 approximation and 0.539 approximation, respectively) on random stream orders. The algorithm by Gamlath et. al. works for the entire class of Triangle-free

Algorithm 4 Two Pass Bipartite Matching Algorithm [15]

```
   Require : The input stream $\pi$ is an edge stream of a bipartite graph $G = (A, B, E)$
 1 begin
 2     $p \leftarrow \sqrt{2} - 1$
 3     Pick an independent random sample $A' \subseteq A$ such that $\Pr[a \in A'] = p, \forall a \in A$
 4     $F_1 \leftarrow$ Induced Graph between $A'$ and $B$
 5     $1^{st}$pass: $M_0 \leftarrow \texttt{Greedy}(\pi)$ and $M' \leftarrow \texttt{Greedy}(F_1 \cap \pi)$
 6     $M_1 \leftarrow \{e \in M' : e$ is between $B(M_0)$ and $\overline{A(M_0)}\}$
 7     $F_2 \leftarrow$ Induced Graph between $\overline{B(M_0)}$ and $\{a \in A(M_0) : M_0(a) \in B(M_1)\}$
 8     $2^{nd}$pass: $M_2 \leftarrow \texttt{Greedy}(F_2 \cap \pi)$
 9     $M \leftarrow$ matching obtained from $M_0$ augmented by $M_1 \cup M_2$
10     return $M$
```

graphs (a super-set of Bipartite Graphs). Their algorithm also works on finding 3-augmenting paths. Gamlath et. al. also adapted their technique to general graphs and presented a one-pass 0.506 approximation algorithm on random stream orders. The work by Konrad [12] is based on finding augmenting paths along with a residual sparsity property of the random order Greedy Matching Algorithm [14]. Konrad also presented a two pass 0.5857 approximation algorithm for Maximum Bipartite Matching on arbitrary stream orders, as a side result of his technique used to finding augmenting paths.

Recently in SODA '20, Farhadi et. al. [5] presented a deterministic one-pass 0.6 approximation algorithm for Maximum Bipartite Matching on random stream orders. Unlike previous algorithms, this algorithm finds augmenting paths with length 5. They also provide a black-box reduction from the general Maximum Matching problem to the Maximum Bipartite Matching problem when edges arrive in random order, i.e. assuming that the approximation ratio of the Maximum Bipartite Matching problem is p on random stream orders, we get an algorithm for Maximum Matching with approximation $\frac{2p}{2p+1}$ with high probabilty on the number of vertices n. Hence, this results in a one-pass 0.545 approximation algorithm for Maximum Matching on random stream orders.

4 Empirical Comparison

For the empirical results, we focus on Maximum Bipartite Matching. The datasets which are used are the IMDB dataset and the NotreDameActors dataset from the SuiteSparse Matrix Collection which are large bipartite graphs. [1] The experiment involved running each of the algorithms in consideration on the same shuffled stream created. We report the average over 10 shuffles of the stream for each algorithm. All of the algorithms implemented are for bipartite matching from Konrad et. al. [15].

[1]Implementation can be found in this Github Repository: https://github.com/alphatron1999/GraphStreamingMatching

Table 1 Size of the Datasets Used

Dataset	Number of Nodes	Number of Edges	Max. Size Matching
IMDB	1,324,748	3,782,463	250,516
NotreDameActors (NDA)	520,223	1,470,404	114,762

Table 2 Results Obtained (here R: Randomized, D: Deterministic)

Dataset	Optimum	Greedy	One Pass	Two Pass (R)	Two Pass (D)	Three Pass
IMDB	250,516	215,997.6	215,997.6	227,317	238,831.2	240,097.6
NDA	114,762	97,293.2	97,293.2	100,744.5	106,448.6	108,300.5

Notice that the result for Greedy Algorithm and One Pass Deterministic are exactly the same and this matches up with the theoretical expectation because according to the lemma the One Pass algorithm only works better than Greedy when the Greedy algorithm gives close to worst case performance but for these dataset, Greedy algorithm itself performs really well.

5 Conclusion

In this work, we surveyed some of the pioneer algorithms for Maximum Matching in the semi-streaming model. We discussed in detail some general and latest techniques used as well as some important properties of the Greedy Algorithm (which played a key role in improving the Greedy Algorithm). We also conducted an empirical comparison between some of these algorithms on massive real world datasets.

5.1 Related Work:

Significant work has been done for the Weighted version of Maximum Matching. One of the earliest work was by Feigenbaum et. al. [7] who presented a one-pass $\frac{1}{6}$ approximation for Weighted Maximum Matching for arbitrary stream orders. Later, McGregor [17] gave a $\frac{1}{2+\epsilon}$ approximation algorithm with multiple passes (heavily dependent on ϵ) on random stream orders. Recently, Paz et. al. [20] gave a $\frac{1}{2+\epsilon}$ approximation in single pass on random stream orders. Techniques like Linear Programming were used by Ahn et. al. [1] to give a $1-\epsilon$ approximation with multiple passes on arbitrary stream orders.

A slightly different line of research focuses on dynamic streaming, in which removal of edges from graph is also allowed. Some works in this area are [13, 2]. Also another interesting problem is the online version of Maximum Matching. Earliest work was by Vazirani et. al who gave a $1-\frac{1}{e}$ approximation. Some latest work on online matching is in [21].

5.2 Future Directions:

There are many future directions of research in the specific case of Matching itself. First directions is to break the $\frac{1}{2}$ approximation barrier which has been an open problem for a long time. Secondly, finding improved upper bounds is also an interesting line of research (maybe $\frac{1}{2}$ approximation is the actual upper bound, instead of $1 - \frac{1}{e}$). Also, improving the algorithms in special settings like random stream orders, multiple passes and special classes of graphs is also an interesting direction. It would also be convenient if this survey could be improved to a more general survey on matching and include techniques from the various variants of the problem.

References

[1] Kook Jin Ahn and Sudipto Guha. 'Linear Programming in the Semi-streaming Model with Application to the Maximum Matching Problem'. In: *CoRR* abs/1104.2315 (2011). arXiv: `1104.2315`. URL: `http://arxiv.org/abs/1104.2315`.

[2] Sepehr Assadi et al. 'Maximum matchings in dynamic graph streams and the simultaneous communication model'. English (US). In: *27th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*. Ed. by Robert Krauthgamer. Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms. 27th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016 ; Conference date: 10-01-2016 Through 12-01-2016. Association for Computing Machinery, Jan. 2016, pp. 1345–1364.

[3] Vladimir Boginski, Sergiy Butenko and Panos Pardalos. 'Network models of massive datasets'. In: *Comput. Sci. Inf. Syst.* 1 (Jan. 2004), pp. 75–89. DOI: `10.2298/CSIS0401075B`.

[4] Sebastian Eggert et al. 'Bipartite Matching in the Semi-streaming Model'. In: *Algorithmica* 63 (2011), pp. 490–508.

[5] Alireza Farhadi et al. 'Approximate Maximum Matching in Random Streams'. In: *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '20. Salt Lake City, Utah: Society for Industrial and Applied Mathematics, 2020, pp. 1773–1785.

[6] Joan Feigenbaum et al. 'Graph Distances in the Data-Stream Model'. In: *SIAM Journal on Computing* 38.5 (2009), pp. 1709–1727. DOI: `10.1137/070683155`. eprint: `https://doi.org/10.1137/070683155`. URL: `https://doi.org/10.1137/070683155`.

[7] Joan Feigenbaum et al. 'On Graph Problems in a Semi-Streaming Model'. In: *Theoretical Computer Science* 348 (Dec. 2005), pp. 207–216. DOI: `10.1016/j.tcs.2005.09.013`.

[8] Buddhima Gamlath et al. 'Weighted Matchings via Unweighted Augmentations'. In: *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*. PODC '19. Toronto ON, Canada: Association for Computing Machinery, 2019, pp. 491–500. ISBN: 9781450362177. DOI: `10.1145/3293611.3331603`. URL: `https://doi.org/10.1145/3293611.3331603`.

[9] Ashish Goel, Michael Kapralov and Sanjeev Khanna. 'On the communication and streaming complexity of maximum bipartite matching'. In: *Proceedings of the 2012 Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 468–485. DOI: `10.1137/1.9781611973099.41`. eprint: `https://epubs.siam.org/doi/pdf/10.1137/1.9781611973099.41`. URL: `https://epubs.siam.org/doi/abs/10.1137/1.9781611973099.41`.

[10] Sagar Kale, Sumedh Tirodkar and Sundar Vishwanathan. 'Maximum Matching in Two Passes, Three Passes, and a Few More Passes Over Graph Streams'. In: (Feb. 2017).

[11] R. M. Karp, U. V. Vazirani and V. V. Vazirani. 'An Optimal Algorithm for On-Line Bipartite Matching'. In: *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*. STOC '90. Baltimore, Maryland, USA: Association for Computing Machinery, 1990, pp. 352–358. ISBN: 0897913612. DOI: 10.1145/100216.100262. URL: https://doi.org/10.1145/100216.100262.

[12] Christian Konrad. 'A Simple Augmentation Method for Matchings with Applications to Streaming Algorithms'. In: *MFCS*. 2018.

[13] Christian Konrad. 'Maximum Matching in Turnstile Streams'. In: *CoRR* abs/1505.01460 (2015). arXiv: 1505.01460. URL: http://arxiv.org/abs/1505.01460.

[14] Christian Konrad. 'MIS in the Congested Clique Model in $O(\log\log\Delta)$ Rounds'. In: 2018.

[15] Christian Konrad, Frédéric Magniez and Claire Mathieu. 'Maximum Matching in Semi-streaming with Few Passes'. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Ed. by Anupam Gupta et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 231–242. ISBN: 978-3-642-32512-0.

[16] Andrew McGregor. 'Graph Stream Algorithms: A Survey'. In: *SIGMOD Rec.* 43.1 (May 2014), pp. 9–20. ISSN: 0163-5808. DOI: 10.1145/2627692.2627694. URL: https://doi.org/10.1145/2627692.2627694.

[17] Andrew Mcgregor. 'Finding Graph Matchings in Data Streams'. In: *APPROX-RANDOM*. 2005.

[18] Andrew McGregor and Sofya Vorotnikova. 'A Simple, Space-Efficient, Streaming Algorithm for Matchings in Low Arboricity Graphs'. In: *1st Symposium on Simplicity in Algorithms (SOSA 2018)*. Ed. by Raimund Seidel. Vol. 61. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 14:1–14:4. ISBN: 978-3-95977-064-4. DOI: 10.4230/OASIcs.SOSA.2018.14. URL: http://drops.dagstuhl.de/opus/volltexte/2018/8295.

[19] S. Muthukrishnan. 'Data streams: algorithms and applications'. In: *Foundations and Trends in Theoretical Computer Science* 1 (2003).

[20] Ami Paz and Gregory Schwartzman. 'A $(2+\epsilon)$-Approximation for Maximum Weight Matching in the Semi-Streaming Model'. In: *CoRR* abs/1702.04536 (2017). arXiv: 1702.04536. URL: http://arxiv.org/abs/1702.04536.

[21] Erik Vee, Sergei Vassilvitskii and Jayavel Shanmugasundaram. 'Optimal Online Assignment with Forecasts'. In: *Proceedings of the 11th ACM Conference on Electronic Commerce*. EC '10. Cambridge, Massachusetts, USA: Association for Computing Machinery, 2010, pp. 109–118. ISBN: 9781605588223. DOI: 10.1145/1807342.1807360. URL: https://doi.org/10.1145/1807342.1807360.

[22] Jaewon Yang and Jure Leskovec. 'Defining and Evaluating Network Communities based on Ground-truth'. In: *CoRR* abs/1205.6233 (2012). arXiv: 1205.6233. URL: http://arxiv.org/abs/1205.6233.