Maximum Matching in Two, Three, and a Few More Passes Over Graph Streams

Sagar Kale[1] and Sumedh Tirodkar[2]

1 **Dept. of Computer Science, Dartmouth College, Hanover, NH, USA**
`sag@cs.dartmouth.edu.`

2 **School of Technology and Computer Science, TIFR, Mumbai, India**
`sumedh.tirodkar@tifr.res.in`

Abstract

We consider the maximum matching problem in the semi-streaming model formalized by Feigenbaum, Kannan, McGregor, Suri, and Zhang [13] that is inspired by giant graphs of today. As our main result, we give a two-pass $(1/2 + 1/16)$-approximation algorithm for triangle-free graphs and a two-pass $(1/2 + 1/32)$-approximation algorithm for general graphs; these improve the approximation ratios of $1/2 + 1/52$ for bipartite graphs and $1/2 + 1/140$ for general graphs by Konrad, Magniez, and Mathieu [20]. In three passes, we are able to achieve approximation ratios of $1/2 + 1/10$ for triangle-free graphs and $1/2 + 1/19.753$ for general graphs. We also give a multi-pass algorithm where we bound the number of passes *precisely*—we give a $(2/3 - \varepsilon)$-approximation algorithm that uses $2/(3\varepsilon)$ passes for triangle-free graphs and $4/(3\varepsilon)$ passes for general graphs. Our algorithms are simple and combinatorial, use $O(n \log n)$ space, and (can be implemented to) have $O(1)$ update time per edge.

For general graphs, our multi-pass algorithm improves the best known *deterministic* algorithms in terms of the number of passes:

- Ahn and Guha [1] give a $(2/3-\varepsilon)$-approximation algorithm that uses $O(\log(1/\varepsilon)/\varepsilon^2)$ passes, whereas our $(2/3 - \varepsilon)$-approximation algorithm uses $4/(3\varepsilon)$ passes;
- they also give a $(1-\varepsilon)$-approximation algorithm that uses $O(\log n \cdot \mathrm{poly}(1/\varepsilon))$ passes, where n is the number of vertices of the input graph; although our algorithm is $(2/3-\varepsilon)$-approximation, our number of passes do not depend on n.

Earlier multi-pass algorithms either have a large constant inside big-O notation for the number of passes [9] or the constant cannot be determined due to the involved analysis [22, 1], so our multi-pass algorithm should use much fewer passes for approximation ratios bounded slightly below $2/3$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Semi Streaming, Maximum Matching

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

Maximum matching is a well-studied problem in a variety of computational models. We consider it in the semi-streaming model formalized by Feigenbaum, Kannan, McGregor, Suri, and Zhang [13] that is inspired by generation of ginormous graphs in recent times. A graph stream is an (adversarial) sequence of the edges of a graph, and a semi-streaming algorithm must access the edges in the given order and use $O(n \operatorname{polylog} n)$ space only, where n is the number of vertices; note that a matching can have size $\Omega(n)$, so $\Omega(n \log n)$ space is necessary. The number of times an algorithm goes over a stream of edges is called the number of

© Sagar Kale and Sumedh Tirodkar;
licensed under Creative Commons License CC-BY

passes. A trivial (1/2)-approximation algorithm that can be easily implemented as a one-pass semi-streaming algorithm is to output a maximal matching. Since the formalization of the semi-streaming model more than a decade ago, the problem of finding a better than (1/2)-approximation algorithm or proving that one cannot do better has baffled researchers [21]. In a step towards resolving this, Goel, Kapralov, and Khanna [14] proved that for any $\varepsilon > 0$, a one-pass semi-streaming $(2/3+\varepsilon)$-approximation algorithm does not exist; Kapralov [16], building on those techniques, showed non-existence of one-pass semi-streaming $(1-1/e+\varepsilon)$-approximation algorithms for any $\varepsilon > 0$. A natural next question is: Can we do better in, say, two passes or three passes? In answering that, Konrad, Magniez, and Mathieu [20] gave three-pass and two-pass algorithms that output matchings that are better than (1/2)-approximate. In this work, we give algorithms that improve their approximation ratios for two-pass and three-pass algorithms. We also give a multi-pass algorithm that does better than the best known multi-pass algorithms for at least initial few passes. We are able to bound the number of passes precisely: we give a $(2/3-\varepsilon)$-approximation algorithm that uses $2/(3\varepsilon)$ passes for triangle-free graphs and $4/(3\varepsilon)$ passes for general graphs. Earlier works either have a large constant inside the big-O notation for the number of passes [9] or the constant cannot be determined due to the involved analysis [22, 1]. For example, the $(1-\varepsilon)$-approximation algorithm by Eggert et al. [9] potentially uses $288/\varepsilon^5$ passes, and for the $(1-\varepsilon)$-approximation algorithms by McGregor [22] and Ahn and Guha [1], the constants inside the big-O bound cannot be determined due to the involved analysis. The $(2/3-\varepsilon)$-approximation algorithm by Feigenbaum et al. [13] uses $O(\log(1/\varepsilon)/\varepsilon)$ passes, which is $O(\log(1/\varepsilon))$ factor larger than the number of passes we use to get the same approximation ratio. Our algorithms are simple and combinatorial, use $O(n \log n)$ space, and (can be implemented to) have $O(1)$ update time per edge. We also give an explicit and tight analysis of the three-pass algorithm by Konrad et al. [20] that is reminiscent of Feigenbaum et al.'s [13] multi-pass algorithm.

Technical overview:

If we can find a matching M such that there are no augmenting paths of length 3 in $M \cup M^*$, where M^* is a maximum matching, then M is (2/3)-approximate, i.e., $(1/2 + 1/6)$-approximate. This is because, in each connected component of $M \cup M^*$, the ratio of M-edges to M^*-edges is at least 2/3. This is the basis for the $(2/3-\varepsilon)$-approximation algorithm by Feigenbaum et al. [13] that uses $O(\log(1/\varepsilon)/\varepsilon)$ passes. The same idea is used by Konrad et al. [20] in the analysis of their two-pass algorithms. In the first pass, they find a maximal matching M_0 and some subset of support edges, say S. If M_0 is so bad that $M_0 \cup M^*$ is almost entirely made up of augmenting paths of length 3 (i.e., $|M_0| \approx |M^*|/2$), then by the end of the second pass, they manage to augment (using length-3 augmentations) a constant fraction of M_0 using S and a fresh access to the edges, resulting in a better than (1/2)-approximation. On the other hand, if M_0 is not so bad, then they already have a good matching. One limitation this idea faces is that a fraction of the edges in S may become useless for an augmentation if both its endpoints get matched in M_0 by the end of the first pass. Our main result is a two-pass algorithm (described in Section 5) that differs in two ways from the former approach. Firstly, in the first pass, we only find a maximal matching M_0 so that in the second pass, where we maintain a set S of support edges, S would not contain "useless" edges. Secondly, any augmentation in our algorithm happens immediately when an edge arrives if it forms an augmenting path of length 3 with edges in M_0 and S.

Our results:

In light of the discussion so far, one way to evaluate an algorithm is how much advantage it gains over the (1/2)-approximate maximal matching found in the first pass. We summarize our two-pass and three-pass results in Table 1 and multi-pass results in Table 2. We stress that we are able to bound the number of passes *precisely*, without big-O notation. For general graphs, our multi-pass algorithm improves the best known *deterministic* algorithms in terms of number of passes—see the third multi-row of Table 2. We note that our multi-pass algorithm is not just a repetition of the second pass of our two-pass algorithm. Such a repetition will give an asymptotically worse number of passes (see, for example, the multi-pass algorithm due to Feigenbaum et al. [13]; the first row of Table 2). We carefully choose the parameters for each pass to get the required number of passes. Also note that Table 1 shows *advantages* over a maximal matching—an algorithm is said to have advantage α if it is a $(1/2+\alpha)$-approximation algorithm (because a maximal matching is (1/2)-approximate).

Table 1 Advantages over a maximal matching—advantage α means $(1/2+\alpha)$-approximation.

<table>
<tr><th>Problem</th><th>Previous work</th><th>Advantage</th><th>Advantage in this work</th></tr>
<tr><td>Bipartite two-pass</td><td>Esfandiari et al. [11]</td><td>1/12</td><td rowspan="2">Not considered separately</td></tr>
<tr><td>Bipartite three-pass</td><td>Esfandiari et al. [11]</td><td>1/9.52</td></tr>
<tr><td>Triangle-free two-pass</td><td colspan="2" rowspan="2">Not considered separately</td><td>1/16 (in Section 5)</td></tr>
<tr><td>Triangle-free three-pass</td><td>1/10 (in Appendix A)</td></tr>
<tr><td>General two-pass</td><td>Konrad et al. [20]</td><td>1/140</td><td>1/32 (in Section 5)</td></tr>
<tr><td>General three-pass</td><td colspan="2">Not considered separately</td><td>1/19.753 (in Appendix B)</td></tr>
</table>

Table 2 Multi-pass algorithms—see Section 6.

Graph	Results	Approx	# Passes
Bipartite	Feigenbaum et al. [13]	$2/3-\varepsilon$	$O(\log(1/\varepsilon)/\varepsilon)$
	Eggert et al.[9]	$1-\varepsilon$	$288/\varepsilon^5$
	Ahn and Guha [1]	$1-\varepsilon$	$O(\log\log(1/\varepsilon)/\varepsilon^2)$
Triangle free	This work (in Section 6)	$2/3-\varepsilon$	$2/(3\varepsilon)$
General	McGregor [22] *randomized*	$1-\varepsilon$	$O((1/\varepsilon)^{1/\varepsilon})$
	Ahn and Guha [1]	$2/3-\varepsilon$	$O(\log(1/\varepsilon)/\varepsilon^2)$
	Ahn and Guha [1]	$1-\varepsilon$	$O(\log n \cdot \mathrm{poly}(1/\varepsilon))$
	This work (in Section 6)	$2/3-\varepsilon$	$4/(3\varepsilon)$

Note of independent work

The work of Esfandiari et al. [11] who claim better approximation ratios for *bipartite graphs* in two passes and three passes is independent and almost concurrent. Our work differs in several aspects. We consider triangle-free graphs (superset of bipartite graphs) and general graphs, and we additionally consider multi-pass algorithms. Also, their algorithm has a post-processing step that uses time $O(\sqrt{n}\cdot|E|)$, whereas our algorithms can be implemented to have $O(1)$ update time per edge. One further detail about this appears in Appendix D.

1.1 Related Work

Karp, Vazirani, and Vazirani [18] gave the celebrated $(1-1/e)$-competitive randomized online algorithm for bipartite graphs in the vertex arrival setting. Goel et al. [14] gave the first one-pass deterministic algorithm with the same approximation ratio, i.e., $1-1/e$, in the semi-streaming model in the vertex arrival setting. For the rest of this section, results involving ε hold for any $\varepsilon > 0$. As mentioned earlier, Goel, Kapralov, and Khanna [14] proved nonexistence of one-pass $(2/3+\varepsilon)$-approximation semi-streaming algorithms, which was extended to $(1-1/e+\varepsilon)$-approximation algorithms by Kapralov [16]. On the algorithms side, nothing better than outputting a maximal matching, which is $(1/2)$-approximate, is known. Closing this gap is considered an outstanding open problem in the streaming community [21].

On the multi-pass front, in the semi-streaming model, Feigenbaum et al. [13] gave a $(2/3-\varepsilon)$-approximation algorithm for bipartite graphs that uses $O(\log(1/\varepsilon)/\varepsilon)$ passes; McGregor [22] improved it to give a $(1-\varepsilon)$-approximation algorithm for general graphs that uses $O((1/\varepsilon)^{1/\varepsilon})$ passes. For bipartite graphs, this was again improved by Eggert et al. [9] who gave a $(1-\varepsilon)$-approximation $O((1/\varepsilon)^5)$-pass algorithm. Ahn and Guha [1] gave a linear-programming based $(1-\varepsilon)$-approximation $O(\log\log(1/\varepsilon)/\varepsilon^2)$-pass algorithm for bipartite graphs. For general graphs, their $(1-\varepsilon)$-approximation algorithm uses number of passes proportional to $\log n$, so it is worse than that of McGregor [22].

For the problem of one-pass weighted matching, there is a line of work starting with Feigenbaum et al. [13] giving a 6-approximation semi-streaming algorithm. Subsequent results improved this approximation ratio: see McGregor [22], Zelke [24], Epstein et al. [10], Crouch and Stubbs [8], Grigorescu et al.[15], and most recently in a breakthrough, giving a $(2+\varepsilon)$-approximation semi-streaming algorithm, Paz and Schwartzman [23]. The multi-pass version of the problem was considered first by McGregor [22], then by Ahn and Guha [1]. Chakrabarti and Kale [5] and Chekuri et al. [6] consider a more general version of the matching problem where a submodular function is defined on the edges of the input graph.

The problem of estimating the *size* of a maximum matching (instead of outputting the actual matching) has also been considered. We mention Kapralov et al. [17], Esfandiari et al. [12], Bury and Schwiegelshohn [4], and Assadi et al. [2].

In the dynamic streams, edges of the input graph can be removed as well. The works of Konrad [19], Assadi et al. [3], and Chitnis et al. [7] consider the maximum matching problem in dynamic streams.

1.2 Organization of the Paper

After setting up notation in Section 2, we give a tight analysis of the three-pass algorithm for bipartite graphs by Konrad et al. [20] in Section 3. In Section 4, we see our simple two-pass algorithm for triangle-free graphs. Then in Section 5, we see our main result—the improved two-pass algorithm, and then we see the multi-pass algorithm in Section 6. The results that are not covered in the main sections are covered in the appendix.

2 Preliminaries

We work on graph *streams*. The input is a sequence of edges (stream) of a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges; a bipartite graph is denoted as $G = (A, B, E)$. A streaming algorithm may go over the stream a few times (multi-pass) and use space $O(n \operatorname{polylog} n)$, where $n = |V|$. In this paper, we give algorithms that make two, three, or a few more passes over the input graph stream. A matching M is a subset of edges such that each vertex has at most one edge in M incident to it. The maximum cardinality matching problem, or maximum matching, for short, is to find a largest matching in the given graph. Our goal is to design streaming algorithms for maximum matching.

For a subset F of edges and a subset U of vertices, we denote by $U(F) \subseteq U$ the set of vertices in U that have an edge in F incident on them. Conversely, we denote by $F(U) \subseteq F$ the set of edges in F that have an endpoint in U. For a subset F of edges and a vertex $v \in V(F)$, we denote by $N_F(v)$ the set of v's neighbors in the graph $(V(F), F)$, and we define $\deg_F(v) := |N_F(v)|$.

In the first pass, our algorithms compute a *maximal* matching which we denote by M_0. We use M^* to indicate a matching of maximum cardinality. Assume that M_0 and M^* are given. For $i \in \{3, 5, 7, \dots\}$, a connected component of $M_0 \cup M^*$ that is a path of length i is called an i-augmenting path (nonaugmenting otherwise). We say that an edge in M_0 is 3-augmentable if it belongs to a 3-augmenting path, otherwise we say that it is non-3-augmentable.

▶ **Lemma 1** (Lemma 1 in [20]). *Let $\alpha \geqslant 0$, M_0 be a maximal matching in G, and M^* be a maximum matching in G such that $|M_0| \leqslant (1/2 + \alpha)|M^*|$. Then the number of 3-augmentable edges in M_0 is at least $(1/2 - 3\alpha)|M^*|$, and the number of non-3-augmentable edges in M_0 is at most $4\alpha|M^*|$.*

Proof. Let the number of 3-augmentable edges in M_0 be k. For each 3-augmentable edge in M_0, there are two edges in M^* incident on it. Also, each non-3-augmentable edge in M_0 lies in a connected component of $M_0 \cup M^*$ in which the ratio of the number of M^*-edges to the number of M_0-edges is at most $3/2$. Hence,

$$\begin{aligned}
|M^*| &\leqslant 2k + \frac{3}{2}(|M_0| - k) && \text{since \# non-3-augmentable edges} = |M_0| - k\,, \\
&\leqslant 2k + \frac{3}{2}\left(\left(\frac{1}{2} + \alpha\right)|M^*| - k\right) && \text{because } |M_0| \leqslant (1/2+\alpha)|M^*|\,, \\
&= \frac{1}{2}k + \left(\frac{3}{4} + \frac{3}{2}\alpha\right)|M^*|\,,
\end{aligned}$$

which, after simplification, gives $k \geqslant (1/2 - 3\alpha)|M^*|$. And the number of non-3-augmentable edges in M_0 is $|M_0| - k \leqslant |M_0| - (1/2 - 3\alpha)|M^*| \leqslant (1/2 + \alpha - 1/2 + 3\alpha)|M^*| = 4\alpha|M^*|$. ◀

We make the following simple, yet crucial, observation.

▶ Observation 1. Let M_0 be a maximal matching. Then $V(M_0)$ is a vertex cover, and there is no edge between any two vertices in $V \setminus V(M_0)$. Therefore, even if the input graph is not a bipartite graph, the set of edges incident on $V \setminus V(M_0)$, i.e., $E(V \setminus V(M_0))$ give rise to a bipartite graph with bipartition $(V \setminus V(M_0), V(M_0))$.

For all the algorithms in this paper, it can be verified that their space complexity is $O(n \log n)$ and update time per edge is $O(1)$. We also ignore floors and ceilings for the sake of exposition.

3 Analyzing the Three Pass Algorithm for Bipartite Graphs

We analyze the three-pass algorithm for bipartite graphs given by Konrad et al. [20], i.e., Algorithm 1 by considering the distribution of lengths of augmenting paths. We also give a tight example.

Algorithm 1 Three-pass algorithm for bipartite graphs due to Konrad et al. [20]

1: In the first pass, find a maximal matching M_0.
2: In the second pass, find a maximal matching
  - M_A in $F_2 := \{ab : a \in A(M_0), b \in B \setminus B(M_0)\}$ (see Figure 1).
3: In the third pass, find a maximal matching
  - M_B in $F_3 := \{ab : a \in A \setminus A(M_0)$ and $\exists a' \in A(M_A)$ such that $a'b \in M_0\}$.
4: Augment M_0 using edges in M_A and M_B and return the resulting matching M.

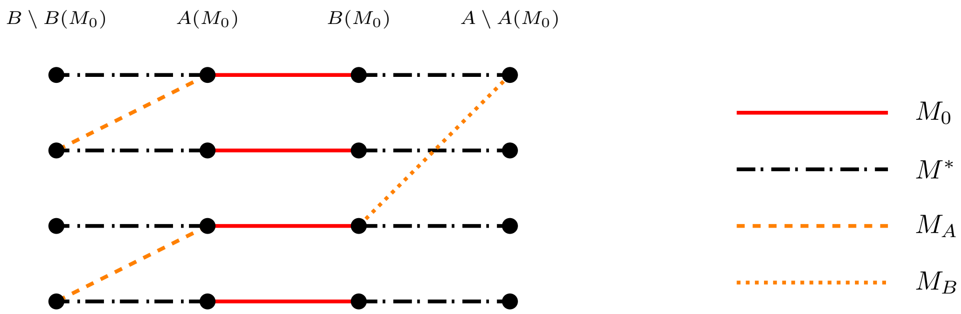

Figure 1 Example: state of variables in an execution of Algorithm 1.

▶ **Theorem 2.** *Algorithm 1 is a three-pass, semi-streaming,* $(1/2 + 1/10)$*-approximation algorithm for maximum matching in bipartite graphs.*

Proof. Without loss of generality, let M^* be a maximum matching such that all nonaugmenting connected components of $M_0 \cup M^*$ are single edges. For $i = \{3, 5, 7, \ldots\}$, let k_i denote the number of i-augmenting paths in $M_0 \cup M^*$, and let $k = |M_0 \cap M^*|$. Then

$$|M_0| = k + \sum_i \frac{i-1}{2} k_i \quad \text{and} \quad |M^*| = k + \sum_i \frac{i+1}{2} k_i \,. \tag{1}$$

Consider an i-augmenting path $b_1 a_1 b_2 a_2 b_3 \cdots b_{(i+1)/2} a_{(i+1)/2}$ in $M_0 \cup M^*$, where for each j, we have $a_j \in A$ and $b_j \in B$. We call the vertex $a_{(i-1)/2}$ a good vertex, because an edge in M_A incident to $a_{(i-1)/2}$ can potentially be augmented using the edge $b_{(i+1)/2} a_{(i+1)/2}$. To elaborate, consider the set of all edges in M_A incident on good vertices; call it M'_A. Consider the set of edges of the type $b_{(i+1)/2} a_{(i+1)/2}$ from each i-augmenting path; call it M_F. Note that M_F is a matching. Then we can augment M_0 using M'_A and M_F by as much as $|M'_A|$.

There is a matching of size $\sum_i k_i$ in F_2 formed by edges of the type $b_1 a_1$ from each i-augmenting path. Since M_A is maximal in F_2, we have $|M_A| \geqslant (\sum_i k_i)/2$. Now, the number of good vertices is $\sum_i k_i$; therefore, the number of bad (i.e., not good) vertices is $|M_0| - \sum_i k_i$. So the number of edges in M_A incident on good vertices (see Figure 2)

$$|M'_A| \geqslant \frac{\sum_i k_i}{2} - \left(|M_0| - \sum_i k_i\right) = \frac{3}{2} \sum_i k_i - |M_0| \,.$$

Let $B_G := \{b \in B : \exists a \in A(M'_A)$ such that $ab \in M_0\}$. Let $M'_F \subseteq M_F$ be defined as $M'_F := \{ba \in M_F : b \in B_G\}$. Then we know that $|M'_F| = |M'_A|$ and $M'_F \subseteq M_F \subseteq F_3$. Since we select a maximal matching in F_3 in the third pass,

$$|M_B| \geqslant \frac{|M'_F|}{2} = \frac{|M'_A|}{2} = \frac{3}{4} \sum_i k_i - \frac{|M_0|}{2} \,. \tag{2}$$

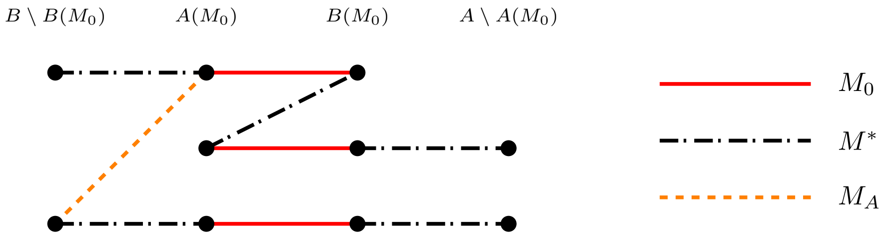

Figure 2 Tight example for Algorithm 1: M_A has only one edge that lands on a bad vertex and cannot be augmented in the third pass. So $|M| = |M_0| = 3$ and $|M^*| = 5$.

So the output size

$$\begin{aligned} |M| &= |M_0| + |M_B| \\ &\geqslant |M_0| + \frac{3}{4}\sum_i k_i - \frac{|M_0|}{2} && \text{by (1) and (2),} \\ &= \frac{|M_0|}{2} + \frac{3}{4}(|M^*| - |M_0|) && \text{by (1), } \sum_i k_i = |M^*| - |M_0|\,, \end{aligned}$$

i.e., $|M| \geqslant 3|M^*|/4 - |M_0|/4$, but we also have $|M| \geqslant |M_0|$, hence

$$|M| \geqslant \max\left\{|M_0|, \frac{3}{4}|M^*| - \frac{1}{4}|M_0|\right\}.$$

So the bound is minimized when $|M_0| = 3|M^*|/4 - |M_0|/4 = 3|M^*|/5 = (1/2 + 1/10)|M^*|$. ◀

As we can see in the proof above, the worst case happens when $|M| = |M_0| = 3|M^*|/5$. Setting $k_3 = k_5 \geqslant 1$, $k = 0$, and $k_i = 0$ for $i > 5$ gives us the tight example shown in Figure 2.

4 A Simple Two Pass Algorithm for Triangle Free Graphs

Before seeing our main result, we see a simple two pass algorithm for triangle-free graphs. The function Semi() in Algorithm 2 greedily computes a subset of edges such that each vertex in X has degree at most one and each vertex in Y has degree at most λ; we call such a subset a (λ, X, Y)-semi-matching (Konrad et al. [20] call this a λ-bounded semi-matching). In Algorithm 2, we find a maximal matching M_0 in the first pass, and, in the second pass, we find a $(\lambda, V(M_0), V \setminus V(M_0))$-semi-matching S. After the second pass, we greedily augment edges in M_0 one by one using edges in S.

▶ **Theorem 3.** *Algorithm 2 is a two-pass, semi-streaming, $(1/2 + 1/20)$-approximation algorithm for maximum matching in triangle-free graphs.*

Proof. As in the proof of Theorem 2, let M^* be a maximum matching such that all nonaugmenting connected components of $M_0 \cup M^*$ are single edges. For $i = \{3, 5, 7, \ldots\}$, let k_i denote the number of i-augmenting paths in $M_0 \cup M^*$, and let k denote the number of edges in $M^* \cap M_0$.

Consider an i-augmenting path $x_1y_1x_2y_2x_3 \cdots x_{(i+1)/2}y_{(i+1)/2}$ in $M_0 \cup M^*$. We call the vertices $y_1 \in V(M_0)$ and $x_{(i+1)/2} \in V(M_0)$ good vertices, because the edges $x_1y_1 \in M^*$ and $x_{(i+1)/2}y_{(i+1)/2} \in M^*$ can potentially be added to S by our algorithm. Denote by V_G the

Algorithm 2 Two-pass algorithm for triangle-free graphs

In the first pass: $M_0 \leftarrow$ maximal matching
In the second pass: $S \leftarrow \textsc{Semi}(\lambda, V(M_0), V \setminus V(M_0))$ (see Figure 3).
After the second pass, augment M_0 greedily using edges in S to get M; output M.
function $\textsc{Semi}(\lambda, X, Y)$ ▷ based on Algorithm 7 in Konrad et al. [20]
    $S \leftarrow \emptyset$
    foreach edge xy such that $x \in X$, $y \in Y$ **do**
        if $\deg_S(x) = 0$ and $\deg_S(y) \leqslant \lambda - 1$ **then**
            $S \leftarrow S \cup \{xy\}$

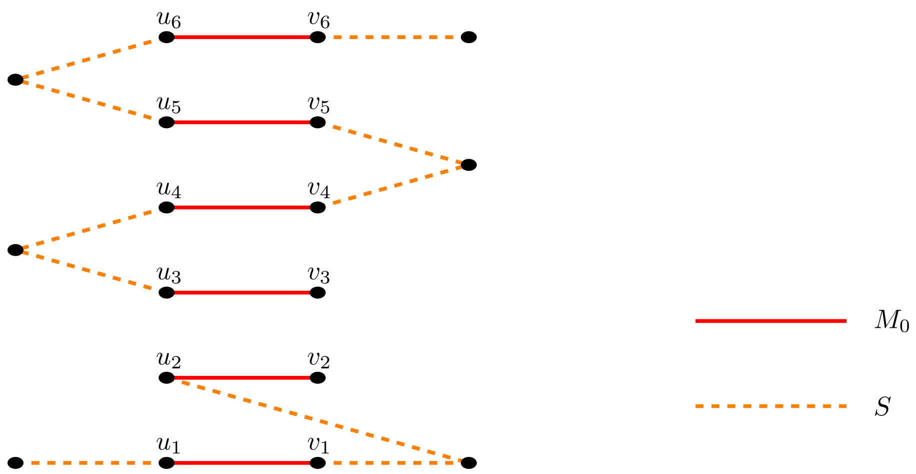

Figure 3 Example showing M_0 and S at the end of the second pass of Algorithm 2 with $\lambda = 2$. When we greedily augment M_0 after the second pass, we may choose to augment u_5v_5 and lose two possible augmentations of edges u_4v_4 and u_6v_6.

set of good vertices and by $V_B := V(M_0) \setminus V_G$ the set of bad vertices. Then $|V_G| = 2\sum_i k_i$. Note that $V_G \cap V_B = \emptyset$ and $V_G \cup V_B = V(M_0)$ by definition.

Let $V_{\mathrm{NC}} := V_G \setminus V(S)$ be the set of good vertices *not covered* by S. An edge $uv \in M^*$ with $u \in V \setminus V(M_0)$ and $v \in V_{\mathrm{NC}}$ was not added to S, because $\deg_S(u) = \lambda$. Hence

$$\lambda |V_{\mathrm{NC}}| \leqslant |V(M_0)| - |V_{\mathrm{NC}}| \quad \text{i.e.,} \quad |V_{\mathrm{NC}}| \leqslant \frac{2}{\lambda+1}|M_0|\,, \tag{3}$$

because at most $|V(M_0)| - |V_{\mathrm{NC}}|$ vertices in $V(M_0)$ are covered by S. Now,

$$\begin{aligned}
|V(M_0) \setminus V(S)| &= |V_G \setminus V(S)| + |V_B \setminus V(S)| && \because V_G \cap V_B = \emptyset \text{ and } V_G \cup V_B = V(M_0),\\
&\leqslant |V_{\mathrm{NC}}| + |V_B| && \because V_{\mathrm{NC}} = V_G \setminus V(S), |V_B \setminus V(S)| \leqslant |V_B|,\\
&\leqslant \frac{2}{\lambda+1}|M_0| + |V(M_0)| - |V_G| && \text{by (3) and } \because |V_B| = |V(M_0)| - |V_G|,\\
&= \frac{2}{\lambda+1}|M_0| + |V(M_0)| - 2\sum_i k_i && \text{because } |V_G| = 2\sum_i k_i\,.
\end{aligned}$$

Using $|V(M_0)| = |V(M_0) \setminus V(S)| + |V(M_0) \cap V(S)|$ and the above, we get

$$|V(M_0) \cap V(S)| \geqslant |V(M_0)| - \left(\frac{2}{\lambda+1}|M_0| + |V(M_0)| - 2\sum_i k_i\right)$$
$$= 2\left(\sum_i k_i - \frac{1}{\lambda+1}|M_0|\right). \tag{4}$$

We observe that at most $|M_0|$ vertices in $V(M_0)$ (one endpoint of each edge) can be covered by S without having both endpoints of an edge in M_0 covered. Hence, at least $|V(M_0) \cap V(S)| - |M_0|$ edges in M_0 have both their endpoints covered by S, which, by (4), is at least

$$2\left(\sum_i k_i - \frac{1}{\lambda+1}|M_0|\right) - |M_0| = 2\sum_i k_i - \frac{\lambda+3}{\lambda+1}|M_0|. \tag{5}$$

After the second pass, when we greedily augment an edge from the above edges, i.e., edges whose both endpoints are covered by S, we may potentially lose $2(\lambda-1)$ other augmentations (see Figure 3). To see this, consider $uv \in M_0$ such that $u, v \in V(S)$ and $au \in S$ and $vb \in S$. The graph is triangle free, so we know that $a \neq b$, and we *can* augment M_0 using the 3-augmenting path $auvb$; but we may lose at most $\lambda - 1$ edges incident to a in S and at most $\lambda - 1$ edges incident to b in S. Therefore the number of augmentations c we get after the second pass is at least $1/(2\lambda - 1)$ times the right hand side of (5), i.e.,

$$c \geqslant \frac{2}{2\lambda-1}\sum_i k_i - \frac{\lambda+3}{(2\lambda-1)(\lambda+1)}|M_0|.$$

So the output size $|M| = |M_0| + c$, and using the above bound on c and simplifying we get:

$$|M| \geqslant \frac{2}{2\lambda-1}\sum_i k_i + \frac{2(\lambda^2-2)}{(2\lambda-1)(\lambda+1)}|M_0|;$$

substituting $\sum_i k_i = |M^*| - |M_0|$, by (1), in the above,

$$|M| \geqslant \frac{2}{2\lambda-1}|M^*| + \frac{2(\lambda^2-\lambda-3)}{(2\lambda-1)(\lambda+1)}|M_0|.$$

Using $\lambda = 3$ and the fact that M_0 is 2-approximate, we get

$$|M| \geqslant \frac{2}{5}|M^*| + \frac{3}{10}|M_0| \geqslant \frac{2}{5}|M^*| + \frac{3}{20}|M^*| = \frac{11}{20}|M^*| = \left(\frac{1}{2} + \frac{1}{20}\right)|M^*|.$$ ◀

5 Improved Two Pass Algorithm

We present our main result that is a two pass algorithm in this section. In the first pass, we find a maximal matching M_0. In the second pass, we maintain a set S of support edges xy, such that $x \in V \setminus V(M_0)$, $y \in V(M_0)$, and $\deg_S(y) \leqslant \lambda_{\mathrm{M}}$ and $\deg_S(x) \leqslant \lambda_{\mathrm{U}}$, where $\lambda_{\mathrm{M}} \geqslant 1$ and $\lambda_{\mathrm{U}} \geqslant 1$ are parameters denoting maximum degree allowed in S for matched and unmatched vertices (with respect to M_0), respectively. Whenever a new edge forms a 3-augmenting path with an edge in M_0 and an edge in S, we augment. We store the vertices involved in a 3-augmentation in the variable I. We ignore a new edge if it is incident to a vertex in I. Unused support edges that are incident to a vertex in I become "useless"; hence to address this, we store the endpoints of M_0 edges that share an endpoint with such useless edges in the variable I_B, and we ignore a new edge if it is incident to a vertex in I_B. Algorithm 3 gives a formal description.

Algorithm 3 Improved two-pass algorithm: input graph G

In the first pass, find a maximal matching M_0.
if G is triangle-free **then**
    Return Improve-Matching($M_0, 2, 1$)
else
    Return Improve-Matching($M_0, 4, 2$)
function Improve-Matching($M_0, \lambda_{\mathrm{U}}, \lambda_{\mathrm{M}}$)
    $M \leftarrow M_0$, $S \leftarrow \emptyset$, $I \leftarrow \emptyset$ and $I_B \leftarrow \emptyset$
    foreach edge xy in the stream **do**
        if x or $y \in I \cup I_B$ **then**
            Continue, i.e., ignore xy.
        else if $x \in V(M_0)$ and $y \in V(M_0)$ **then**
            Continue, i.e., ignore xy.
        else if there exist v and b such that $yv \in M_0$ and $vb \in S$ **then**
            $M \leftarrow M \setminus \{yv\} \cup \{xy, vb\}$ ▷ a 3-augmentation
            Let $I_x \leftarrow \{u_x, v_x : xu_x \in S \text{ and } u_xv_x \in M_0\}$.
            Let $I_b \leftarrow \{u_b, v_b : u_bv_b \in M_0 \text{ and } v_bb \in S\}$.
            Then $I \leftarrow I \cup \{x, y, v, b\}$ and $I_B \leftarrow I_B \cup I_x \cup I_b$.
        else
            Without loss of generality, assume that $x \in V \setminus V(M_0)$ and $y \in V(M_0)$.
            if $\deg_S(x) < \lambda_{\mathrm{U}}$ and $\deg_S(y) < \lambda_{\mathrm{M}}$ **then** ▷ See Figure 4.
                $S \leftarrow S \cup \{xy\}$ ▷ **Note:** Once an edge is added to S, it is never removed from it.
    Return M.

Setting up a charging scheme to lower bound the number of augmentations

We first lay the groundwork and give a charging scheme.

▶ Observation 2. For general graphs (that are possibly not triangle-free), we need to set $\lambda_{\mathrm{M}} \geqslant 2$.

To see why, suppose $\lambda_{\mathrm{M}} = 1$. Let uv be a 3-augmentable edge in M_0. Then, for the edge uv, we might end up storing the edges ub and vb in S, and the edge uv would not get augmented. If $\lambda_{\mathrm{M}} \geqslant 2$, and we store at least λ_{M} edges incident to u, then an edge incident to v will not form a triangle with at least one of those and uv would get augmented. So, for general graphs, we need to set $\lambda_{\mathrm{M}} \geqslant 2$.

Let $|M_0| = (1/2 + \alpha)|M^*|$. For a 3-augmentable edge $uv \in M_0$, let $auvb$ be the 3-augmenting path such that $au, vb \in M^*$. Without loss of generality, assume that au arrived before vb. Then we make the following observation.

▶ Observation 3. When au arrived, it may not be added to S for one of the following reasons:

- The vertex a was already matched.
- There were λ_{M} edges incident to u in S.
- There were λ_{U} edges incident to a in S.

We call some edges in M_0 *good*, some *partially* good, and some *bad*. An edge is good if it got augmented. An edge $uv \in M_0$ is bad if it is 3-augmentable, not good, and vertex a or b had λ_{U} edges incident to them in S when edge au or vb arrived. An edge $uv \in M_0$ is partially good if it is 3-augmentable, but neither good nor bad ("partially" good because, as we will

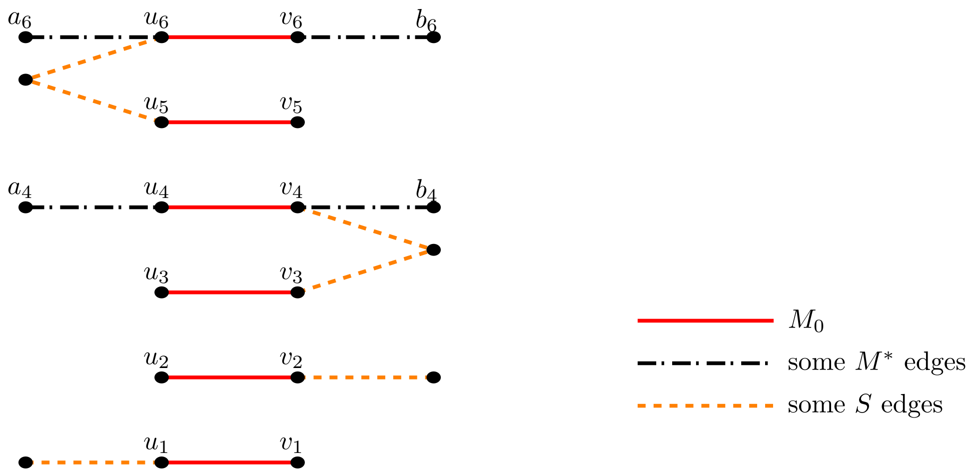

Figure 4 Example showing M_0 and some of the edges in M^* and S during the second pass of Algorithm 3 for triangle-free graphs with $\lambda_{\mathrm{U}} = 2$ and $\lambda_{\mathrm{M}} = 1$. At most one of u_i and v_i can have positive degree in S, because we would rather augment $u_i v_i$ instead of adding the latter edge to S. By our convention, $a_4 u_4$ arrived before $v_4 b_4$, and $a_6 u_6$ arrived before $v_6 b_6$. Since $a_4 u_4$ was not added to S, we have $\deg_S(a_4) = \lambda_{\mathrm{U}}$ (S edges incident to a_4 are not shown).

see later, we can hold some good edge $u'v' \in M_0$ responsible for uv not getting augmented). Note that all 3-augmentable edges get some label according to our labeling. We require the following lemma to describe the charging scheme.

▶ **Lemma 4.** *Suppose au was not added to S because there were already λ_{M} edges incident to u in S. If, later, uv did not get augmented when vb arrived, then*
- *b was already matched via augmenting path $a''u''v''b$, or*
- *there exists $a'u \in S$ and $u'v' \in M_0$ such that a' was matched via augmenting path $a'u'v'b'$.*

Proof. When au arrived, $|N_S(u)| \geqslant \lambda_{\mathrm{M}}$. If b was unmatched when vb arrived, then some $a' \in N_S(u) \setminus \{b\}$ must have been matched, otherwise we would have augmented uv. Now for triangle-free graphs $b \notin N_S(u)$, so $|N_S(u) \setminus \{b\}| = |N_S(u)| \geqslant 1$, and for general graphs, by Observation 2, $\lambda_{\mathrm{M}} \geqslant 2$, so $|N_S(u) \setminus \{b\}| \geqslant \lambda_{\mathrm{M}} - 1 \geqslant 1$. ◀

Charging Scheme.

As alluded to earlier, we charge a partially good edge to some good edge. Recall that for a 3-augmentable edge $uv \in M_0$, we denote by $au, vb \in M^*$ the edges that form the 3-augmenting path with uv such that au arrived before vb. We use Observation 3 and consider the following cases. See Figure 5.

- Suppose au was not added to S because a was already matched. Then, let $u'v' \in M_0$ was augmented using $au'v'b'$. If $\deg_S(a) \leqslant \lambda_{\mathrm{U}} - 1$, then we charge uv to $u'v'$. Otherwise, uv is bad.
- Suppose au was not added to S because $\deg_S(u) = \lambda_{\mathrm{M}}$. Then we use Lemma 4. We either charge uv to $u'v'$, or if $\deg_S(b) \leqslant \lambda_{\mathrm{U}} - 1$, then we charge uv to $u''v''$. Otherwise, uv is bad.
- Suppose au was not added to S because $\deg_S(a) = \lambda_{\mathrm{U}}$, then uv is bad.
- Otherwise, au was added to S, but uv did not get augmented when vb arrived. Then:

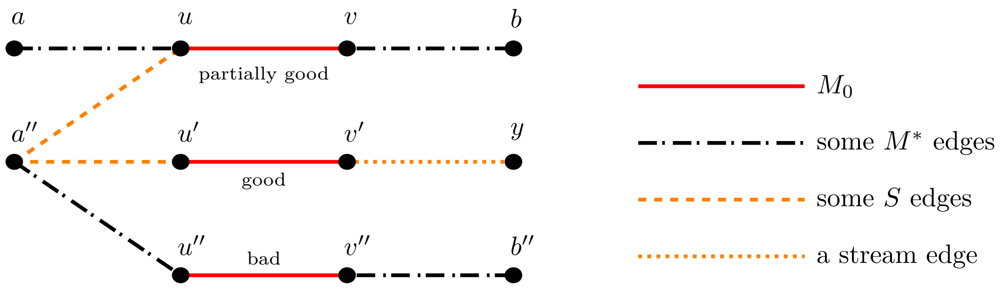

Figure 5 Example showing a good edge, a bad edge, and a partially good edge. We use parameters $\lambda_U = 2$ and $\lambda_M = 1$, so we are in the triangle-free case. The edge $u'v'$ is not 3-augmentable but was augmented using $a''u'v'y$, so $u'v'$ is a good edge. The edge $u''v''$ is a 3-augmentable edge that was not augmented and when $a''u''$ arrived, $\deg_S(a'') = 2$, so $u''v''$ is a bad edge. For uv, we did not take au in S, because $\deg_S(u) = 1$, so uv is a partially good edge, and we can charge uv to $u'v'$ using Lemma 4.

- Either there exists $a' \in N_S(u)$ that was matched via augmenting path $a'u'v'b'$ (note that a' may be same as a), then we charge uv to $u'v'$;
- or b was already matched via augmenting path $a''u''v''b$, and vb was ignored; in this case, if $\deg_S(b) \leqslant \lambda_U - 1$, then we charge uv to $u''v''$, otherwise, uv is bad.

We now bound the number of bad edges in M_0 from above.

▶ **Lemma 5.** *The number of bad edges is at most $\lambda_M|M_0|/\lambda_U$.*

Proof. We claim that for any $uv \in M_0$, $\deg_S(u) + \deg_S(v) \leqslant \lambda_M$, hence $|S| \leqslant \lambda_M|M_0|$. A short argument is that the $(\lambda_M + 1)$th edge would cause an augmentation and will not be added to S. Let us assume the claim. By the definition of a bad edge, λ_U edges in S are "responsible" for one bad edge in M_0. Also, an edge au' (or $v''b$, resp.) in S can be responsible for at most one bad edge that can only be uv if $au \notin S$ (or if $vb \notin S$, resp.; considering the 3-augmenting path $auvb$). Hence, the total number of bad edges is at most $|S|/\lambda_U \leqslant \lambda_M|M_0|/\lambda_U$. Now we prove the claim.

We first prove for triangle-free graphs by contradiction. Let $\deg_S(u) + \deg_S(v) > \lambda_M$, and let $vy \in S$ be the (λ_M+1)th edge incident to one of u and v that was added to S. Since $\lambda_M \geqslant 1$ and $\deg_S(v) \leqslant \lambda_M$, we have $\deg_S(u) \geqslant 1$, i.e. $N_S(u) \neq \emptyset$. Now when vy arrived:

- the vertex y was unmatched, otherwise vy would not be added to S;
- no vertex $x \in N_S(u)$ was matched, otherwise $u, v \in I_B$, and vy would not be added to S.

The above implies that when vy arrived, due to some $x \in N_S(u)$ the if condition on Line 14 became true, and we augmented uv via $xuvy$ instead of adding vy to S. This is a contradiction.

For general graphs, we argue by contradiction slightly informally for the sake of brevity. By Observation 2, for general graphs, $\lambda_M \geqslant 2$. Let $\deg_S(u) + \deg_S(v) > \lambda_M \geqslant 2$. Let vy be the second edge incident to one of u and v that was added to S; the first edge can be xu or vy'.

Suppose xu was the first edge. If $x \neq y$, then we would have augmented uv via $xuvy$ instead of adding vy to S—a contradiction. If $x = y$, then after vy was processed, $N_S(u) = N_S(v) = \{y\}$, and a third edge incident to one of u and v would not be added to S, because it would have formed a 3-augmenting path with either yu or vy, resulting in a contradiction that $\deg_S(u) + \deg_S(v) = 2$.

Otherwise, suppose vy' was the first edge; then $N_S(v) = \{y, y'\}$ after vy was processed. Since eventually $\deg_S(u) + \deg_S(v) \geqslant \lambda_M + 1 \geqslant 3$ and $\deg_S(u), \deg_S(v) \leqslant \lambda_M$, we would eventually have $\deg_S(u) \geqslant 1$, so let $xu \in S$. When xu arrived, it would have formed an 3-augmenting path with either vy or vy' (here, taking care of the fact that one of y and y' can be same as x), resulting in a contradiction that xu was not added to S.

Thus, we get the claim and complete the proof. ◀

As a consequence, we get the following.

▶ Observation 4. In any call to Improve-Matching(), we need to set $\lambda_U > \lambda_M$, i.e., $\lambda_U \geqslant 2$.

To see why, suppose $\lambda_U \leqslant \lambda_M$. Then by Lemma 5, potentially all 3-augmentable edges in M_0 could become bad edges.

Recall that a 3-augmentable edge is good, partially good, or bad; so by Lemmas 1 and 5,

$$\begin{aligned}
\#\text{ good or partially good edges} &\geqslant \left(\frac{1}{2} - 3\alpha\right)|M^*| - \frac{\lambda_M |M_0|}{\lambda_U} \\
&= \left(\frac{1}{2} - 3\alpha\right)|M^*| - \frac{\lambda_M}{\lambda_U}\left(\frac{1}{2} + \alpha\right)|M^*| \\
&= \left(\frac{\lambda_U - \lambda_M}{2\lambda_U} - \left(\frac{3\lambda_U + \lambda_M}{\lambda_U}\right)\alpha\right)|M^*|\,. \qquad (6)
\end{aligned}$$

In the following lemma, we bound the number of partially good edges in M_0 that are charged to one good edge.

▶ **Lemma 6.** *At most $2\lambda_U - 1$ partially good edges in M_0 are charged to one good edge in M_0.*

Proof. Suppose $uv \in M_0$ was augmented by edges xu and vy such that xu arrived before vy, then $xu \in S$. Now $|N_S(x)|, |N_S(y)| \leqslant \lambda_U$. Since $xu \in S$, we have $|N_S(x) \setminus \{u\}| \leqslant \lambda_U - 1$. Let $B := (N_S(x) \setminus \{u\}) \cup N_S(y)$, then $|B| \leqslant 2\lambda_U - 1$. Now, the set of partially good edges that are charged to uv is a subset of $M_0(B)$. Observing that $|M_0(B)| \leqslant |B| \leqslant 2\lambda_U - 1$ finishes the proof. ◀

The following lemma characterizes the improvement given by Improve-Matching().

▶ **Lemma 7.** *Let $|M_0| = (1/2 + \alpha)|M^*|$ and $M = $ Improve-Matching$(M_0, \lambda_U, \lambda_M)$, then*

$$|M| \geqslant \left(\frac{1}{2} + \frac{\lambda_U - \lambda_M}{4\lambda_U^2} + \left(1 - \frac{3\lambda_U + \lambda_M}{2\lambda_U^2}\right)\alpha\right)|M^*| \geqslant \left(\frac{1}{2} + \frac{\lambda_U - \lambda_M}{4\lambda_U^2}\right)|M^*|\,.$$

Proof. By (6) and Lemma 6, the total number of augmentations during one call to Improve-Matching() is at least

$$\frac{1}{2\lambda_U}\left(\frac{\lambda_U - \lambda_M}{2\lambda_U} - \left(\frac{3\lambda_U + \lambda_M}{\lambda_U}\right)\alpha\right)|M^*| = \left(\frac{\lambda_U - \lambda_M}{4\lambda_U^2} - \left(\frac{3\lambda_U + \lambda_M}{2\lambda_U^2}\right)\alpha\right)|M^*|\,.$$

Hence, we get the following bound on the size of the output matching M:

$$\begin{aligned}
|M| &\geqslant |M_0| + \left(\frac{\lambda_U - \lambda_M}{4\lambda_U^2} - \frac{3\lambda_U + \lambda_M}{2\lambda_U^2}\alpha\right)|M^*| \\
&= \left(\frac{1}{2} + \frac{\lambda_U - \lambda_M}{4\lambda_U^2} + \left(1 - \frac{3\lambda_U + \lambda_M}{2\lambda_U^2}\right)\alpha\right)|M^*| \quad \text{because } |M_0| = (1/2 + \alpha)|M^*|\,, \\
&\geqslant \left(\frac{1}{2} + \frac{\lambda_U - \lambda_M}{4\lambda_U^2}\right)|M^*| \quad \text{since } \lambda_U \geqslant 2 \text{ by Observation 4}\,.
\end{aligned}$$

◀

Now we state and prove our main result.

▶ **Theorem 8.** *Algorithm 3 uses two passes and has an approximation ratio of* $1/2 + 1/16$ *for triangle-free graphs and an approximation ratio of* $1/2 + 1/32$ *for general graphs for maximum matching.*

Proof. After the second pass, the output size $|M| \geqslant (1/2 + (\lambda_{\mathrm{U}} - \lambda_{\mathrm{M}})/(4\lambda_{\mathrm{U}}^2))|M^*|$ due to Lemma 7; we use $\lambda_{\mathrm{U}} = 2$ and $\lambda_{\mathrm{M}} = 1$ for triangle-free graphs and $\lambda_{\mathrm{U}} = 4$ and $\lambda_{\mathrm{M}} = 2$ (see Observation 2) for general graphs to get the claimed approximation ratios. ◀

6 Multi Pass Algorithm

We run the function Improve-Matching() in Algorithm 3 with increasing values of λ_{U}, and the approximation ratio converges to $1/2+1/6$. We note that this multi-pass algorithm is not just a repetition of the function Improve-Matching(). Such a repetition will give an asymptotically worse number of passes (see, for example, the multi-pass algorithm due to Feigenbaum et al. [13]). We carefully choose the parameter λ_{U} for each pass to get the required number of passes.

Algorithm 4 Multi-pass algorithm: input graph G

In the first pass, find a maximal matching M_1.
$M \leftarrow M_1$
if G is triangle-free **then**
  for $i = 2$ to $\lceil 2/(3\varepsilon) \rceil$ **do**
    $M \leftarrow$ Improve-Matching$(M, i, 1)$
else
  for $i = 2$ to $\lceil 4/(3\varepsilon) \rceil$ **do**
    $M \leftarrow$ Improve-Matching$(M, i+1, 2)$
Return M.

▶ **Theorem 9.** *For any* $\varepsilon > 0$*, Algorithm 4 is a semi-streaming* $(1/2+1/6-\varepsilon)$*-approximation algorithm for maximum matching that uses* $2/(3\varepsilon)$ *passes for triangle-free graphs and* $4/(3\varepsilon)$ *passes for general graphs.*

Proof. We prove the theorem for triangle-free case; the general case is similar. Let M_i be the matching computed by Algorithm 4 after ith pass, and let $p := \lceil 2/(3\varepsilon) \rceil$, so $\varepsilon \leqslant 2/(3p)$. Since M_1 is maximal, it is $(1/2)$-approximate. Let $\alpha_1 := 0$, and for $i \in \{2, 3, \ldots, p\}$, let

$$\alpha_i := \frac{i-1}{4i^2} + \left(1 - \frac{3i+1}{2i^2}\right)\alpha_{i-1}$$

(see Lemma 7 with $\lambda_{\mathrm{U}} = i$ and $\lambda_{\mathrm{M}} = 1$). Then, by Lemma 7 and the logic of Algorithm 4, for $i \in [p]$, the matching M_i is $(1/2 + \alpha_i)$-approximate (by a trivial induction). Now we bound α_p by induction. We claim that for $i \in [p]$,

$$\alpha_i \geqslant \frac{1}{6} - \frac{2}{3i},$$

which we prove by induction on i.

Base case: For $i = 1$, we have $1/6 - \alpha_1 = 1/6 - 0 = 1/6 \leqslant 2/(3 \cdot 1)$.

For inductive step, we want to show that

$$\frac{1}{6} - \alpha_i = \frac{1}{6} - \frac{i-1}{4i^2} - \left(1 - \frac{3i+1}{2i^2}\right)\alpha_{i-1} \leqslant \frac{2}{3i}\,,$$

which is implied by the following (using inductive hypothesis)

$$\frac{1}{6} - \frac{i-1}{4i^2} + \left(1 - \frac{3i+1}{2i^2}\right)\left(\frac{2}{3(i-1)} - \frac{1}{6}\right) \leqslant \frac{2}{3i}\,,$$

implied by
$$\frac{1}{6} - \frac{i-1}{4i^2} + \left(\frac{2i^2-3i-1}{2i^2}\right)\left(\frac{4-i+1}{6(i-1)}\right) \leqslant \frac{2}{3i}\,,$$

multiplying both sides by $12i^2(i-1)$, we then need to show that,

$$2i^2(i-1) - 3(i-1)^2 + (2i^2-3i-1)(-i+5) \leqslant 8i(i-1)\,,$$

implied by $2i^3 - 2i^2 - 3(i^2-2i+1) + (-2i^3+10i^2+3i^2-15i+i-5) \leqslant 8i^2-8i\,,$

implied by $2i^3 - 5i^2 + 6i - 3 + (-2i^3+13i^2-14i-5) \leqslant 8i^2-8i\,,$

implied by $8i^2-8i-8 \leqslant 8i^2-8i\,,$

which is true, so we get the claim. Therefore $\alpha_p \geqslant 1/6 - 2/(3p) \geqslant 1/6 - \varepsilon$, and by our earlier observation, M_p is $(1/2+\alpha_p)$-approximate, and this finishes the proof for triangle-free case. The proof for general case is very similar. We define $p := \lceil 4/(3\varepsilon)\rceil$ and $\alpha_1 := 0$, and for $i \in \{2,3,\dots,p\}$, we define

$$\alpha_i := \frac{i-1}{4(i+1)^2} + \left(1 - \frac{3(i+1)+2}{2(i+1)^2}\right)\alpha_{i-1}\,,$$

i.e., we use $\lambda_{\mathrm{U}} = i+1$ and $\lambda_{\mathrm{M}} = 2$. The corresponding claim then is that for $i \in [p]$,

$$\alpha_i \geqslant \frac{1}{6} - \frac{4}{3i}\,,$$

which can be verified by induction on i. ◀

Acknowledgements. We thank Sundar Vishwanathan and Ashish Chiplunkar for helpful discussions. The first author would like to thank his advisor Amit Chakrabarti and Andrew McGregor for helpful discussions.

References

1 Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. *Inf. Comput.*, 222:59–79, January 2013. URL: `http://dx.doi.org/10.1016/j.ic.2012.10.006`, `doi:10.1016/j.ic.2012.10.006`.

2 Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1723–1742, 2017. URL: `http://epubs.siam.org/doi/abs/10.1137/1.9781611974782.113`, `arXiv:http://epubs.siam.org/doi/pdf/10.1137/1.9781611974782.113`, `doi:10.1137/1.9781611974782.113`.

3 Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *Proc. 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1345–1364, 2016. URL: `http://dl.acm.org/citation.cfm?id=2884435.2884528`.

4 Marc Bury and Chris Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. In *Proc. 23rd Annual European Symposium on Algorithms*, pages 263–274, 2015. URL: http://dx.doi.org/10.1007/978-3-662-48350-3_23, doi:10.1007/978-3-662-48350-3_23.

5 Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: matchings, matroids, and more. *Mathematical Programming*, 154(1):225–247, 2015. URL: http://dx.doi.org/10.1007/s10107-015-0900-7, doi:10.1007/s10107-015-0900-7.

6 Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *Proc. 42nd International Colloquium on Automata, Languages and Programming*, pages 318–330, 2015. URL: http://dx.doi.org/10.1007/978-3-662-47672-7_26, doi:10.1007/978-3-662-47672-7_26.

7 Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In *Proc. 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1326–1344, 2016. URL: http://dl.acm.org/citation.cfm?id=2884435.2884527.

8 Michael Crouch and Daniel M. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In *Proc. 17th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, volume 28, pages 96–104, 2014. doi:http://dx.doi.org/10.4230/LIPIcs.APPROX-RANDOM.2014.96.

9 Sebastian Eggert, Lasse Kliemann, Peter Munstermann, and Anand Srivastav. Bipartite matching in the semi-streaming model. *Algorithmica*, 63(1):490–508, 2012. URL: http://dx.doi.org/10.1007/s00453-011-9556-8, doi:10.1007/s00453-011-9556-8.

10 Leah Epstein, Asaf Levin, Julian Mestre, and Danny Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM Journal on Discrete Mathematics*, 25(3):1251–1265, 2011. URL: http://epubs.siam.org/doi/abs/10.1137/100801901, arXiv:http://epubs.siam.org/doi/pdf/10.1137/100801901, doi:10.1137/100801901.

11 H. Esfandiari, M. Hajiaghayi, and M. Monemizadeh. Finding large matchings in semi-streaming. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 608–614, Dec 2016. doi:10.1109/ICDMW.2016.0092.

12 Hossein Esfandiari, Mohammad T. Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. In *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1217–1233, 2015. URL: http://dl.acm.org/citation.cfm?id=2722129.2722210.

13 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2):207–216, December 2005. URL: http://dx.doi.org/10.1016/j.tcs.2005.09.013, doi:10.1016/j.tcs.2005.09.013.

14 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proc. 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 468–485, 2012. URL: http://dl.acm.org/citation.cfm?id=2095116.2095157.

15 Elena Grigorescu, Morteza Monemizadeh, and Samson Zhou. Streaming weighted matchings: Optimal meets greedy. *CoRR*, abs/1608.01487, 2016. URL: http://arxiv.org/abs/1608.01487.

16 Michael Kapralov. Better bounds for matchings in the streaming model. In *Proc. 24th Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2013.

17 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proc. 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 734–751, 2014. URL: http://dl.acm.org/citation.cfm?id=2634074.2634129.

18 Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proc. 22nd Annual ACM Symposium on the Theory of Computing*, pages 352–358, 1990.

19 Christian Konrad. Maximum matching in turnstile streams. In *Proc. 23rd Annual European Symposium on Algorithms*, pages 840–852, 2015. URL: http://dx.doi.org/10.1007/978-3-662-48350-3_70, doi:10.1007/978-3-662-48350-3_70.

20 Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. *In Proc. 15th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, pages 231–242, 2012 and CoRR*, abs/1112.0184, 2014. URL: http://arxiv.org/abs/1112.0184.

21 Andrew McGregor. Problem 60: Single-pass unweighted matchings. http://sublinear.info/index.php?title=Open_Problems:60. Accessed: 2017-02-16.

22 Andrew McGregor. Finding graph matchings in data streams. In *Proc. 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 170–181, 2005. URL: http://dx.doi.org/10.1007/11538462_15, doi:10.1007/11538462_15.

23 Ami Paz and Gregory Schwartzman. A $(2 + \varepsilon)$-approximation for maximum weight matching in the semi-streaming model. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2153–2161, 2017. URL: http://epubs.siam.org/doi/abs/10.1137/1.9781611974782.140, arXiv:http://epubs.siam.org/doi/pdf/10.1137/1.9781611974782.140, doi:10.1137/1.9781611974782.140.

24 Mariano Zelke. Weighted matching in the semi-streaming model. In *Proc. 25th International Symposium on Theoretical Aspects of Computer Science*, pages 669–680, 2008.

A Three Pass Algorithm for Triangle Free Graphs

For completeness, we present our three-pass algorithm for triangle-free graphs.

Algorithm 5 Three-pass algorithm for triangle-free graphs

1: In the first pass, find a maximal matching M_0.
2: In the second pass, find a maximal matching M_1 in $F_1 := \{uv : u \in V \setminus V(M_0), v \in V(M_0)\}$.
3: After the second pass:
- $M_1' \leftarrow$ arbitrary largest subset of M_1 such that there is no 3-augmenting path in $M_1' \cup M_0$ with respect to M_0
- $V_2 \leftarrow \{x \in V(M_0) : \exists v, w$ such that $vw \in M_1'$ and $wx \in M_0\}$
- For $x \in V_2$, denote by $P(x)$ the vertex v such that there exists w with $vw \in M_1'$ and $wx \in M_0$. See x and $P(x)$ in Figure 6.

4: In the third pass: $F_2 := \{xy : x \in V_2, y \in V \setminus V(M_0)\}$
5: $M_2 \leftarrow \emptyset$
6: **for** edge $xy \in F_2$ **do**
7:     **if** x, and y are unmarked **then**
8:         $M_2 \leftarrow M_2 \cup \{xy\}$; since the graph is triangle free, $y \neq P(x)$, and we can augment M_0 using xy.
9:         Mark $P(x)$, x, y, and $P^{-1}(y)$ (if exists).
10: Let M be largest of M_3 and M_3' which are computed below.
- Augment M_0 using edges in M_1 to get M_3.
- Augment M_0 using edges in M_1' and M_2 to get M_3'.

11: Output M.

▶ **Theorem 10.** *Algorithm 5 is a three-pass, semi-streaming,* $(1/2 + 1/10)$*-approximation algorithm for maximum matching in triangle-free graphs, and the analysis is tight.*

Proof. Let $|M_0| = (1/2 + \alpha)|M^*|$. The number of edges in M^* incident on $V(M^*) \setminus V(M_0)$ is

$$|V(M^*) \setminus V(M_0)| \geqslant |V(M^*)| - |V(M_0)| = 2|M^*| - 2|M_0| = (1 - 2\alpha)|M^*| ; \tag{7}$$

and these edges also belong to F_1. Since M_1 is a maximal matching in F_1,

$$|M_1| \geqslant (1 - 2\alpha)|M^*|/2 = (1/2 - \alpha)|M^*| . \tag{8}$$

Let c be the number of 3-augmenting paths in $M_1 \cup M_0$, so $|M_1'| = |M_1| - c$ by the definition of M_1'. By Lemma 1, there are at most $4\alpha|M^*|$ non-3-augmentable edges in M_0. So at least $|M_1| - c - 4\alpha|M^*|$ edges of M_1' are incident on 3-augmentable edges of M_0. Therefore there is a matching of size at least $|M_1| - c - 4\alpha|M^*|$ in F_2; consider one such matching M_F. We claim that $|M_2| \geqslant |M_F|/4$. See Figure 6. Let $xy \in M_2$; we note that xy disallows at most four edges in M_F from being added to M_2 due to the (at most) four marks that it adds, because a marked vertex can disallow at most one edge in M_F (due to it being a matching), which shows the claim. Hence:

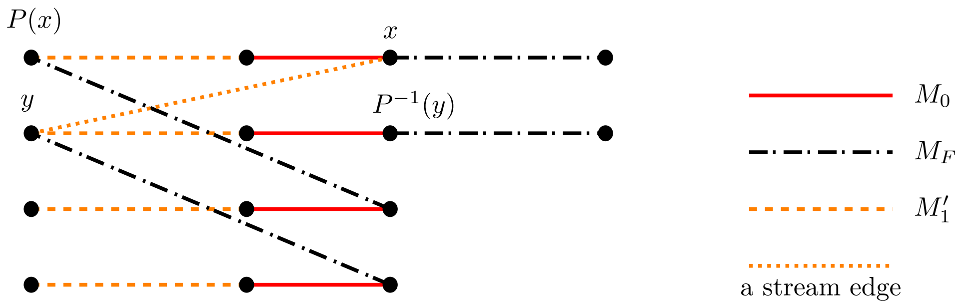

Figure 6 An edge $xy \in M_2$ disallows at most four edges in M_F from being added to M_2.

$$\begin{aligned} |M_2| &\geqslant \frac{|M_F|}{4} \\ &\geqslant \frac{|M_1| - c - 4\alpha|M^*|}{4} \\ &\geqslant \frac{1}{4}\left(\left(\frac{1}{2} - \alpha\right)|M^*| - c - 4\alpha|M^*|\right) && \text{by (8)}, \\ &= \frac{1}{4}\left(\left(\frac{1}{2} - 5\alpha\right)|M^*| - c\right). \end{aligned}$$

Now, each edge in M_2 gives one augmentation after the second pass. To see this, we observe that for any $x \in V_2$, at any point in the algorithm, x and $P(x)$ are either both marked or both unmarked. So when an edge $xy \in M_2$ arrives, x and y are unmarked, and $P(x)$ and $P^{-1}(y)$ (if it exists) are also unmarked, otherwise one of x and y would have been marked and xy would not have been added to M_2. Since both $P(x)$ and $P^{-1}(y)$ were unmarked, we can use the augmenting path $\{M_1'(\{P(x)\}), M_0(\{x\}), xy\}$. Hence we get at least

$$\max\left\{c, \frac{1}{4}\left(\left(\frac{1}{2} - 5\alpha\right)|M^*| - c\right)\right\}$$

augmentations after the third pass. This is minimized by setting

$$\begin{aligned} c &= \frac{1}{4}\left(\left(\frac{1}{2} - 5\alpha\right)|M^*| - c\right) \\ &= \frac{1}{5}\left(\left(\frac{1}{2} - 5\alpha\right)|M^*|\right) \\ &= \left(\frac{1}{10} - \alpha\right)|M^*|. \end{aligned}$$

So we get the following bound:

$$|M| \geqslant |M_0| + \left(\frac{1}{10} - \alpha\right)|M^*| \geqslant \left(\frac{1}{2} + \alpha\right)|M^*| + \left(\frac{1}{10} - \alpha\right)|M^*| = \left(\frac{1}{2} + \frac{1}{10}\right)|M^*|. \blacktriangleleft$$

The tight example is shown in Figure 7.

B Three Pass Algorithm for General Graphs

We find a maximal matching M_1 in the first pass. Then we use Improve-Matching() function from Algorithm 3, i.e.,

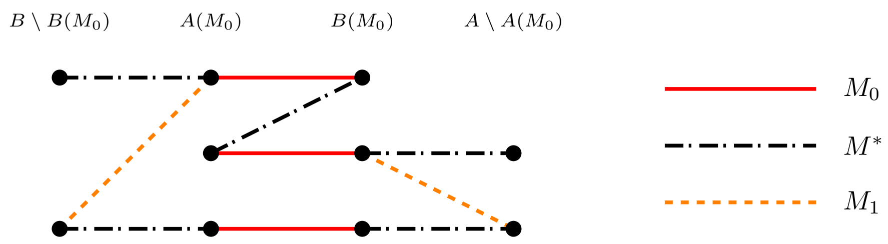

Figure 7 Tight example for Algorithm 5: M_1 has only two edges that land on bad vertices and cannot be augmented in the third pass. So $|M| = |M_0| = 3$ and $|M^*| = 5$.

- in the second pass, $M_2 \leftarrow$ IMPROVE-MATCHING$(M_1, 4, 2)$, and
- in the third pass, $M_3 \leftarrow$ IMPROVE-MATCHING$(M_2, 5, 2)$.

We observe that M_1 is (1/2)-approximate. Then by double application of Lemma 7, we get that M_3 is $(1/2 + 81/1600) \approx (1/2 + 1/19.753)$-approximate.

C Three Pass Algorithm for Bipartite Graphs: Suboptimal Analysis

We now give an analysis of Algorithm 1 that shows approximation ratio of only $1/2 + 1/18$ that is based on Konrad et al.'s [20] analysis for their two-pass algorithm for bipartite graphs. Afterward, we demonstrate that by not considering the distribution of lengths of augmenting paths, we may prove an approximation ratio of at most $1/2 + 1/14$. The better and tight analysis appears in Section 3.

▶ **Theorem 11.** *Algorithm 1 is a three-pass, semi-streaming,* $(1/2 + 1/18)$*-approximation algorithm for maximum matching in bipartite graphs.*

Proof. As usual, let $|M_0| = (1/2 + \alpha)|M^*|$. Since M_0 is a maximal matching, there are $|B(M^*) \setminus B(M_0)|$ edges of M^* that are also in F_2. We have

$$|B(M^*) \setminus B(M_0)| \geqslant |B(M^*)| - |B(M_0)| = |M^*| - |M_0|\,,$$

and since M_A is maximal, we then get the following:

$$|M_A| \geqslant \frac{1}{2}|B(M^*) \setminus B(M_0)| \geqslant \frac{1}{2}(|M^*| - |M_0|) = \frac{1}{2}\left(1 - \left(\frac{1}{2} + \alpha\right)\right)|M^*| = \frac{1}{2}\left(\frac{1}{2} - \alpha\right)|M^*|. \tag{9}$$

By Lemma 1, there are at most $4\alpha|M^*|$ non-3-augmentable edges in M_0. Which means that at least $|M_A| - 4\alpha|M^*|$ edges of M_A are incident on 3-augmentable edges of M_0; therefore there is a matching of size at least $|M_A| - 4\alpha|M^*|$ in F_3. Since we output a maximal matching in F_3, we get at least $(1/2)(|M_A| - 4\alpha|M^*|)$ augmentations after the third pass. So we get the following bound:

$$\begin{aligned}
|M| &\geqslant |M_0| + \frac{1}{2}(|M_A| - 4\alpha|M^*|) \\
&\geqslant |M_0| + \frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2} - \alpha\right) - 4\alpha\right)|M^*| && \text{by (9)}, \\
&= |M_0| + \left(\frac{1}{8} - \frac{9}{4}\alpha\right)|M^*| \\
&= \left(\frac{1}{2} + \alpha\right)|M^*| + \left(\frac{1}{8} - \frac{9}{4}\alpha\right)|M^*| && \text{because } |M_0| = (1/2+\alpha)|M^*|, \\
&= \left(\frac{1}{2} + \frac{1}{8} - \frac{5}{4}\alpha\right)|M^*|.
\end{aligned}$$

We also have $|M| \geqslant |M_0| = (1/2 + \alpha)|M^*|$. As α increases, the former bound deteriorates and the latter improves, so the worst case α is when these two bounds are equal, which happens at $\alpha = 1/18$, and the approximation ratio we get is $1/2 + 1/18$. ◀

C.1 Improved Analysis Without Considering Longer Augmenting Paths

We can analyze Algorithm 1 better if we bound $|M_A|$ more carefully. The claim is that at least $(1/2 - 7\alpha)|M^*|/2$ edges of M_A are incident on 3-augmentable edges of M_0. Let $A_G \subseteq A(M_0)$ be the set of vertices in A that are endpoints of 3-augmentable edges of M_0; also, let $A_N = A(M_0) \setminus A_G$. So there is a matching of size at least $|A_G|$ in F_2 that covers A_G. Any maximal matching in F_2 has at least $(|A_G| - |A_N|)/2$ edges that are incident on A_G. To see the claim, we use the facts $|A_G| \geqslant (1/2 - 3\alpha)|M^*|$ and $|A_N| \leqslant 4\alpha|M^*|$. So there is a matching of size at least $(1/2 - 7\alpha)|M^*|/2$ in F_3. We output a maximal matching in F_3; hence we get at least $(1/2 - 7\alpha)|M^*|/4$ augmentations after the third pass. So we get the following bound:

$$\begin{aligned}
|M| &\geqslant |M_0| + \frac{1}{4}\left(\frac{1}{2} - 7\alpha\right)|M^*| \\
&= \left(\frac{1}{2} + \alpha\right)|M^*| + \frac{1}{4}\left(\frac{1}{2} - 7\alpha\right)|M^*| \\
&= \left(\frac{1}{2} + \frac{1}{8} - \frac{3}{4}\alpha\right)|M^*|.
\end{aligned}$$

where the second inequality is by (9). We also have $|M| \geqslant |M_0| = (1/2 + \alpha)|M^*|$, so the worst case α is when these two bounds are equal, which happens at $\alpha = 1/14$ and the approximation ratio we get is $1/2 + 1/14$, and we get the following theorem.

▶ **Theorem 12.** *Algorithm 1 is a three-pass, semi-streaming, $(1/2 + 1/14)$-approximation algorithm for maximum matching in bipartite graphs.*

D A Note on the Analysis by Esfandiari et al.

We demonstrate with an example that the analysis of the algorithm by Esfandiari et al. [11] given for bipartite graphs cannot be extended for triangle-free graphs to get the same approximation ratio. See Figure 8. Lemma 6 in their paper, as they correctly claim, holds only for bipartite graphs and not for triangle-free graphs. Our algorithm in Section 4 is essentially the same algorithm except for the post-processing step; we augment the maximal matching computed in the first pass greedily, whereas they use an offline maximum matching algorithm. We have highlighted some other comparison points in Section 1.

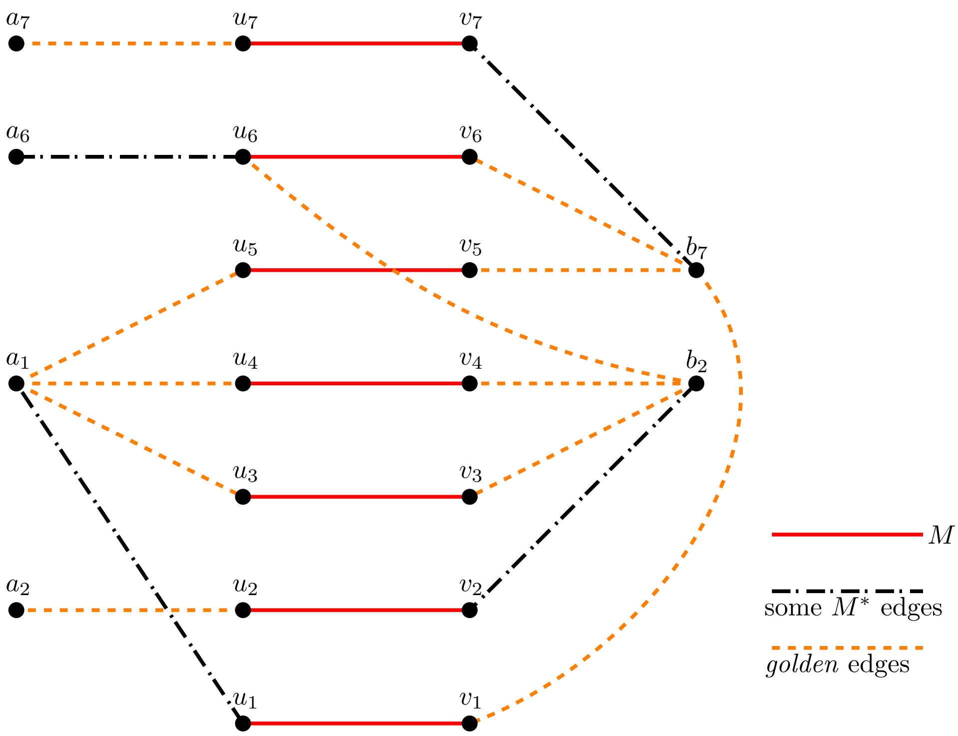

Figure 8 Example demonstrating that Lemma 6 in Esfandiari et al. [11] does not hold when the input graph is not bipartite but is triangle-free. We use $k = 3$. For an M edge u_iv_i, there are two M^* edges incident on it, which are a_iu_i and v_ib_i, and some of the M^* edges are not shown, but all golden edges are shown, which we call support edges or denote by S in our terminology. It can be seen from this example that their algorithm is not a $(1/2 + 1/12)$-approximation algorithm for triangle free graphs, because out of the seven 3-augmentable edges in M, only one will get augmented, thereby giving a worse approximation ratio.