Maximum Matching Sans Maximal Matching: A New Approach for Finding Maximum Matchings in the Data Stream Model

Moran Feldman ✉ 🄳
Department of Computer Science, University of Haifa, Israel

Ariel Szarf ✉
Department of Mathematics and Computer Science, Open University of Israel, Ra'anana, Israel

───── **Abstract** ─────

The problem of finding a maximum size matching in a graph (known as the *maximum matching* problem) is one of the most classical problems in computer science. Despite a significant body of work dedicated to the study of this problem in the data stream model, the state-of-the-art single-pass semi-streaming algorithm for it is still a simple greedy algorithm that computes a maximal matching, and this way obtains ¹/₂-approximation. Some previous works described two/three-pass algorithms that improve over this approximation ratio by using their second and third passes to improve the above mentioned maximal matching. One contribution of this paper continues this line of work by presenting new three-pass semi-streaming algorithms that work along these lines and obtain improved approximation ratios of 0.6111 and 0.5694 for triangle-free and general graphs, respectively.

Unfortunately, a recent work [30] shows that the strategy of constructing a maximal matching in the first pass and then improving it in further passes has limitations. Additionally, this technique is unlikely to get us closer to single-pass semi-streaming algorithms obtaining a better than ¹/₂-approximation. Therefore, it is interesting to come up with algorithms that do something else with their first pass (we term such algorithms non-maximal-matching-first algorithms). No such algorithms are currently known (to the best of our knowledge), and the main contribution of this paper is describing such algorithms that obtain approximation ratios of 0.5384 and 0.5555 in two and three passes, respectively, for general graphs (the result for three passes improves over the previous state-of-the-art, but is worse than the result of this paper mentioned in the previous paragraph for general graphs). The improvements obtained by these results are, unfortunately, numerically not very impressive, but the main importance (in our opinion) of these results is in demonstrating the potential of non-maximal-matching-first algorithms.

1 Introduction

The problem of finding a maximum size matching in a graph (known as the *maximum matching* problem) is one of the most classical problems in computer science, and many polynomial time algorithms have been designed for it over the years (see, e.g., [9, 15, 23]). Due to its central role, the maximum matching problem is often one of the first problems

considered when new computational models are suggested. One such model is the data stream model, which is motivated by Big-Data applications, and has been the subject of an enormous amount of research over the last couple of decades.

In the data stream model, the algorithm receives the input in the form of a stream which it can read sequentially, but due to memory restrictions, the algorithm can store only a small part of this stream. This means that the algorithm has to process (in some sense) the input stream while reading it, and never gets an opportunity to see all the parts of the input at the same time. Traditional algorithms for this model, known as *streaming algorithms*, are allowed only memory that is poly-logarithmic in the natural parameters of the problem. Obtaining a streaming algorithm for a problem is very desirable, but is often not possible. In particular, many graph problems provably do not admit streaming algorithms, and the maximum matching problem is among these problems if one would like an algorithm for the problem to output an (approximately) maximum matching because such a matching might be of linear size in the number of vertices. Nevertheless, non-trivial streaming algorithms have been designed for the maximum matching problem when only the (approximate) size of a maximum matching is desired (see Section 1.1 for details).

The resistance of many graph problems to streaming algorithms has motivated Feigenbaum et al. [19] to suggest semi-streaming algorithms, which are algorithms for the data stream model that are allowed a space complexity of $O(n \log^c n)$ for some constant $c \geq 0$, where n is the number of vertices in the graph. Such algorithms turn out to be a sweet-spot that on the one hand allows many results of interest, and on the other hand, does not lead to triviality because $O(n \log^c n)$ is less than the space necessary for storing the input graph (unless this graph is very sparse). In particular, Feigenbaum et al. [19] observed that one can obtain 1/2-approximation for the maximum matching problem using a simple semi-streaming algorithm that greedily constructs a maximal matching.[1]

The above 1/2-approximation semi-streaming algorithm for the maximum matching problem also has the desirable property that it reads the input stream only once (i.e., it makes a single pass over it). Surprisingly, no single-pass semi-streaming algorithm improving over the approximation ratio of this simple algorithm was suggested in the decade and a half that has already passed since the work of [19] (in contrast, Kapralov [26] showed that no such algorithm can have an approximation ratio better than $1/(1 + \ln 2) \approx 0.59$, improving over previous inapproximability results due to [22, 25]). Given this lack of progress, interest arose in obtaining improved approximation ratios for relaxed versions of the above problem. Perhaps, one of the simplest such relaxations is to allow the algorithm to make a few (usually two or three) sequential passes over the input stream.

The last line of work was introduced by Konrad et al. [29], and was later studied by [18]. The state-of-the-art results for it are summarized in Table 1. We note that beside the state-of-the-art results for general input graphs, Table 1 also gives improved results for bipartite and triangle-free graphs. All the known results in this line of work (to the best of our knowledge) start by greedily constructing a maximal matching during the first pass over the input stream, and then augmenting this matching in the subsequent passes. Recently, Konrad and Naidu [30] showed that this technique has limitations (specifically, even for bipartite graphs, a two-pass semi-streaming algorithm based on this technique cannot obtain a better than 2/3-approximation, which is much more strict than the inapproximability known for general two-pass semi-streaming algorithms [2]). Additionally, and arguably

---

[1] A maximal matching is a matching that is inclusion-wise maximal, and it is well-known that the size of any maximal matching is a 1/2-approximation for the size of a maximum matching.

■ **Table 1** The state-of-the-art approximation ratios for semi-streaming algorithms using two or three passes, and our improvements over these ratios (the number to the right of each improvement is the number of the theorem formally stating it).

Number of Passes	Type of Graphs	State-of-the-Art		This Paper		Approach
Two-Pass	Bipartite	$2 - \sqrt{2} \approx \frac{1}{2} + \frac{1}{11.66}$ ≈ 0.5857	[28]	–		–
	Triangle-Free	$\frac{1}{2} + \frac{1}{16} = 0.5625$	[24]	–		–
	General	$\frac{1}{2} + \frac{1}{32} = 0.53125$	[24]	$\frac{1}{2} + \frac{1}{26} \approx 0.5385$	(1)	non-MMF
Three-Pass	Bipartite	$0.6067 \approx \frac{1}{2} + \frac{1}{9.37}$	[28]	$\frac{1}{2} + \frac{1}{9} \approx 0.6111$	(3)	MMF
	Triangle-Free	$\frac{1}{2} + \frac{1}{10} = 0.6$	[24]	$\frac{1}{2} + \frac{1}{9} \approx 0.6111$	(3)	MMF
	General	$\frac{1}{2} + \frac{81}{1600} \approx \frac{1}{2} + \frac{1}{19.753}$ ≈ 0.5506	[24]	$\frac{1}{2} + \frac{1}{14.4} \approx 0.5694$	(4)	MMF
				$\frac{1}{2} + \frac{1}{18} \approx 0.5555$	(2)	non-MMF

more importantly, multi-pass algorithms that use their first pass for constructing a maximal matching are unlikely to be a step towards a single-pass semi-streaming algorithm with a better than 1/2-approximation guarantee.

Given the above observations, it is natural to believe that the future of the study of semi-streaming algorithms for the maximum matching problem lies in algorithms that use their first pass in a more sophisticated way than simply constructing the traditional maximal matching. We term such algorithms *non-maximal-matching-first algorithms* (or non-MMF algorithms for short). In this paper, we present the first non-MMF algorithms, which leads to improvements over the state-of-the-art both for two and three passes. Admittedly, the improvements we obtain are numerically not very impressive, but their main importance (in our opinion) is in demonstrating the potential of non-MMF algorithms.

To intuitively understand our non-MMF algorithms, one should note that greedily constructing a maximal matching is equivalent to greedily constructing a graph whose connected components are of size at most 2 (where the size of a connected component is defined as the number of vertices in it). Therefore, a natural generalization is to greedily construct in the first pass a graph whose connected components are of size at most 3. There are two intuitive advantages for doing that compared to constructing a maximal matching.

- A connected component of size 3 can contribute two edges to the output matching if it is "augmented" during the next passes with a single additional edge. In contrast, doing the same with a connected component of size 2 requires "augmenting" it with two additional edges. It is important to note that there is a significant conceptual difference between an augmentation of a connected component with one or two edges. Augmenting a connected component with two edges requires finding pairs of edges that augment the *same* connected component, while augmenting with a single edge does not require such a synchronization.
- If we are not able to enjoy the above advantage because many connected components end up being of size 2 rather than 3. Then, the fact that this has happened despite us only restricting the components to be of size at most 3 implies that few edges of a maximum matching intersect only a single connected component of the constructed graph; and therefore, the constructed graph must have many connected components compared to the size of a maximum matching.

Using the above ideas, we prove the following two theorems. The proof of Theorem 1 appears in Section 3, while the adaptation of this proof leading to Theorem 2 is deferred to the full version of this paper [20].

▶ **Theorem 1.** *There exists a non-MMF 2-pass ($7/13 = 1/2 + 1/26$)-approximation semi-streaming algorithm for finding a maximum size matching in a general graph.*

▶ **Theorem 2.** *There exists a non-MMF 3-pass ($5/9 = 1/2 + 1/18$)-approximation semi-streaming algorithm for finding a maximum size matching in a general graph.*

As mentioned above, both Theorems 1 and 2 represent an improvement over the state-of-the-art. However, it turns out that we can further improve over Theorem 2 using new MMF algorithms (i.e., algorithms that construct a maximal matching in their first pass). This leads to the following theorems whose proofs appear in Section 4 and Appendix A, respectively.

▶ **Theorem 3.** *There exists a 3-pass ($11/18 = 1/2 + 1/9$)-approximation semi-streaming algorithm for finding a maximum size matching in a triangle-free graph.[2]*

▶ **Theorem 4.** *There exists a 3-pass ($1/2 + 1/14.4$)-approximation semi-streaming algorithm for finding a maximum size matching in a general graph.*

The algorithms used to prove Theorems 3 and 4 are strongly based on the algorithms suggested by Kale and Tirodkar [24]. For example, the first two passes of the algorithm suggested by Theorem 3 are identical to a two-pass algorithm presented by [24], and the third pass of this algorithm is very similar to the third pass of the three-pass algorithm of [24]. Our novelty, however, is in our ability to analyze the algorithm obtained by putting these two components together.

1.1   Related Work

As mentioned in Section 1, streaming algorithms are not appropriate for the maximum matching problem when the algorithm is required to output an (approximately) maximum matching. However, some non-trivial streaming algorithms are known for this problem when the algorithm is only required to estimate the size of the maximum matching. Kapralov et al. [27] designed a poly-log approximation streaming algorithm for this problem under the assumption that the edges in the input stream are ordered in a uniformly random order. A different line of work [13, 17, 32] considered graphs of bounded arboricity α, comulating with the work of McGregor and Vorotnikova [33], who designed $(\alpha + 2)(1 + \varepsilon)$-approximation streaming algorithm for this problem requiring only $O(\varepsilon^{-2} \log n)$ space. In contrast, Assadi et al. [5] showed that $(1 - \varepsilon)$-approximation of the size of the maximum matching cannot be obtained by a single pass algorithm, even if this algorithm is allowed a semi-streaming space complexity, and [6, 8] lower bounded the number of passes required to obtain such a good approximation using a sub-polynomial space complexity.

Recall that, to date, the best single-pass semi-streaming algorithm for the maximum matching problem is still the natural greedy algorithm, which guarantees $1/2$-approximation. Chitnis et al. [12] presented an exact single-pass algorithm for this problem. However, this algorithm requires $\tilde{O}(k^2)$ memory, where k is an upper bound on the size of the maximum matching (which the algorithm needs to know upfront), and thus, this algorithm is a semi-streaming algorithm only when $k = \tilde{O}(\sqrt{n})$. Given the difficultly to improve over the guarantee of the greedy algorithm using single-pass semi-streaming algorithms, people

---

[2] We recall that every bipartite graph is triangle-free, and therefore, the same result is obtained also for bipartite graphs.

consider also relaxed versions of the maximum matching problem. One standard relaxation is to allow the algorithm to make multiple passes over the input stream. Section 1 surveys algorithms of this kind that use two or three passes. Another line of work considers algorithms that assume a constant (but possibly large) number of passes. The first result of this kind was presented by Feigenbaum et al. [19] (in the same paper that also introduced the notion of semi-streaming algorithms), and guaranteed $(2/3 - \varepsilon)$-approximation using $O(\varepsilon^{-1} \log \varepsilon^{-1})$ passes for bipartite graphs. Later [31] showed how to obtain $(1-\varepsilon)$-approximation for general graphs using $(\varepsilon^{-1})^{O(\varepsilon^{-1})}$ passes, and the number of passes necessary to obtain this guarantee was improved by many further works (see, e.g., [1, 4, 7, 21]). Another standard relaxation for the maximum matching problem is to assume that the edges of the input stream appear in a uniformly random order. The state-of-the-art for this relaxation is a $(2/3+\varepsilon_0)$-approximation single-pass semi-streaming algorithm, where $\varepsilon_0 > 0$ is some absolute constant [3, 10] (see also the references therein for previous works on this relaxation).

The related maximum weight matching problem was also studied heavily in the context of the data stream model. Here, it is not immediately clear that one can obtain a constant approximation ratio using a single-pass semi-streaming algorithm. However, Feigenbaum et al. [19] presented the first such algorithm guaranteeing $1/6$-approximation, and this ratio was improved in series of works [14, 16, 31, 35]. The current state-of-the-art for the problem is $(1/2 - \varepsilon)$-approximation due to Paz and Schwartzman [34]. Since this approximation ratio is essentially identical to the state-of-the-art for the (unweighted) maximum matching problem, any further progress on the maximum weight matching problem (beyond removing the ε) will imply an improvement over the guarantee of the greedy algorithm for the (unweighted) maximum matching problem. It is also worth mentioning that a recent reduction due to Bernstein et al. [11] shows that the reverse is also true in a sense. More specifically, any semi-streaming algorithm for bipartite unweighted graphs can be translated into such an algorithm for weighted graphs with the same number of passes and a loss of only $1 - \varepsilon$ in the approximation guarantee. Naturally, this reduction automatically extends some of our results to the weighted case.

2 Preliminaries

In this section we present the problem that we study more formally, and also introduce the notation used throughout the rest of the paper. We are interested in semi-streaming algorithms for the problem of finding a maximum size matching in a graph $G = (V, E)$ of n vertices. A semi-streaming algorithm for this problem is an algorithm with a space complexity of $O(n \log^c n)$ (for some constant $c \geq 0$) that initially has no knowledge about the edges of E. Instead, the edges of E appear sequentially in an "input stream", and the algorithm may make one or more passes over this input stream. In each pass the algorithm sees the edges one by one, and may do arbitrary calculations after viewing each edge. It is important to note that the space complexity allowed for the algorithm does not suffice for storing all the edges of the graph (unless the graph is very sparse), and this is the reason that the algorithm might benefit from doing multiple passes over the input stream. It is standard to assume that the vertices of V are known upfront, and that each vertex of V can be stored using $O(\log n)$ bits (which implies that every edge of E can also be stored using this asymptotic number of bits).

Throughout the paper, we consider only unweighted graphs and matchings. We also denote by M^* an arbitrary maximum matching of G (i.e., an arbitrary optimal solution for our problem). Notation-wise, we treat M^* (and any other matching considered in the paper)

as a set of the edges included in it. Similarly, when considering a connected component C of a graph, we treat it as a set of the vertices in it, which in particular, implies that $|C|$ is the number of such vertices.

Given a set of edges S or a path P in a graph, we denote by $V(S)$ and $V(P)$ the set of vertices intersecting any edge of S or P, respectively. Similarly, the set of edges included in the path P is denoted by $E(P)$. Often we need to consider collections of paths (or triangles) in a given graph. For clarity, such collections are always denoted using calligraphic letters, and we extend the above notation to such collections. In other words, if $\mathcal{P}$ is a collection of paths, then $V(\mathcal{P})$ and $E(\mathcal{P})$ is the set of vertices and edges, respectively, that are included in these paths. Finally, given a set S of edges and a vertex v, we use $\deg_S(v)$ to denote the degree of the vertex v in the subgraph (V, S).

3   Two-Pass Non-MMF Algorithm

In this section we prove Theorem 1, which we repeat below for convenience.

▶ **Theorem 1.** *There exists a non-MMF 2-pass ($7/13 = 1/2 + 1/26$)-approximation semi-streaming algorithm for finding a maximum size matching in a general graph.*

The algorithm whose existence is guaranteed by Theorem 1 appears as Algorithm 1. In its first pass, this algorithm greedily grows a set P of edges that form either triangles or partial triangles (i.e., isolated edges or paths of length 2). For simplicity, we refer below to the connected components of (V, P) that are not isolated vertices as partial triangles although, technically, they can also be full triangles. In the second pass of Algorithm 1, the algorithm tries to convert the partial triangles of P into more involved structures in one of two ways. To understand these ways, we need to define some terms. First, we designate some of the vertices of every partial triangle as "connection vertices". Specifically, all the vertices of a triangle are considered connection vertices; in a path of length 2 only the two end points are considered to be connection vertices; and finally, in an isolated edge there are no connection vertices. We refer to a partial triangle that was not converted yet into a more involved structure as a "naïve" partial triangle. The first way in which Algorithm 1 tries to convert the partial triangles of P into more involved structures is by greedily adding edges that connect a connection vertex of a naïve partial triangle with an isolated vertex. The set A_1 in the algorithm includes the edges that were added in this way. In parallel, the algorithm also tries a second way to convert the partial triangles of P into more involved structures, which is to greedily add edges that connect a connection vertex of a naïve partial triangle either to a connection vertex of another naïve partial triangle or to an isolated vertex. The set A_2 in the algorithm includes the edges that were added in this way. Upon termination, Algorithm 1 outputs a maximum matching in the set of all the edges that it kept. We recall that given a connected component C of a graph, the notation $|C|$ represents the number of vertices in C.

We begin the analysis of Algorithm 1 by noting that it is indeed a semi-streaming algorithm. The proof of the next observation can be found in the full version of this paper [20].

▶ **Observation 5.** *Algorithm 1 is a semi-streaming algorithm.*

In the rest of this section we analyze the approximation ratio of Algorithm 1. Recall that we use M^* to denote some maximum matching of G. Our first objective in the analysis of the approximation ratio of Algorithm 1 is to lower bound the number of edges of M^* that can potentially be added either to A_1 or to A_2. Towards this goal, we define a charging scheme

▮ **Algorithm 1** MAXIMUM MATCHING VIA GREEDY TRIANGLES – TWO PASSES.

```
// First Pass
```
1 Let $P \leftarrow \varnothing$.
2 **for** *every edge e that arrives* **do**
3   **if** *every connected component of the graph $(V, P \cup \{e\})$ is either an isolated vertex, a path of length at most 2 or a triangle (cycle of size 3)* **then** Add e to P.

```
// Second Pass
```
4 Let $A_1 \leftarrow \varnothing$ and $A_2 \leftarrow \varnothing$.
5 **for** *every edge $(u, v) \notin P$ that arrives* **do**
6   Let C_u and C_v be the connected components of u and v, respectively, in (V, P). We assume without loss of generality that $|C_u| > 1$, otherwise we swap the roles of u and v. `// Note that we cannot have` $|C_u| = |C_v| = 1$ `because the edge` (u, v) `was not added to` P `in the first pass.`
7   **if** *no edge of A_1 intersects C_u and C_v, $|C_v| = 1$ and u is a connection vertex of C_u* **then** Add the edge (u, v) to A_1.
8   **if** *no edge of A_2 intersects C_u and C_v, $|C_v| = 1$ and u is a connection vertex of C_u* **then** Add the edge (u, v) to A_2.
9   **else if** *no edge of A_2 intersects C_u and C_v, and u and v are connection vertices of C_u and C_v, respectively* **then** Add the edge (u, v) to A_2.

10 **return** *a maximum matching in the graph $(V, P \cup A_1 \cup A_2)$*.

π. Under the charging scheme π, every edge $(u, v) \in M^*$ charges the connected components of u and v in (V, P). Each one of these connected components is charged one unit by (u, v), unless it is an isolated edge or an isolated vertex, in which case it is charged only half a unit or nothing by (u, v), respectively. We note that when u and v belong to the same connected component of (V, P), then this connected component is charged twice by (u, v).[3]

The following observation provides an upper bound on the total charged by all the edges of M^* together. Let (#single) be the number of isolated edges in P, (#double) be the number of connected components in (V, P) that are paths of length 2 and (#triangle) be the number of triangles in P.

▶ **Observation 6.** *The total charge according to π is at most (#single) + 3(#double) + 3(#triangle).*

Proof. Every positive amount charged by π is charged to some connected component of (V, P) which is not an isolated vertex. Therefore, to prove the observation we only need to show that every isolated edge of (V, P) is charged at most one unit, and every connected component of (V, P) that is either a path of length 2 or a triangle is charged at most 3 units. Below we are argue that this is indeed the case.

Each connected component C of (V, P) can be charged at most once for every one of its vertices since the fact that M^* is a matching implies that every vertex of C can appear in at most a single edge of M^*. For isolated edges of (V, P), this implies that they can be

---

3   Intuitively, the charge assigned to the connected components of u and v is proportional to the "blame" that can be assigned to them if (u, v) ends up to be outside P. For example, an isolated edge could not alone prevent (u, v) from being added to P, but two such edges (one intersecting u and the other intersecting v) could, together, prevent (u, v) from being added to P. Therefore, we assign a charge of $1/2$ to isolated edges. Observation 7 is based on this intuition.

charged at most twice, and therefore, they are charged at most one unit because they are charged half a unit in each charge. Similarly, connected components of (V, P) that are either paths of length 2 or triangles contain 3 vertices, and therefore, can be charged at most three times. Since every one of these charges is of a single unit, the total charge to each connected component of these kinds is at most 3. ◀

To complement the last observation, let us now describe a simple lower bound on the total charging done by all the edges of M^* according to π. Let (#component-free) be the number of edges of M^* that connect a connection vertex of a connected component of (V, P) to an isolated vertex of (V, P), (#component-component) be the number of edges of M^* that connect connection vertices of two different connected components of (V, P), (#single-single) be the number of edges of M^* whose two end points belong to (not necessarily distinct) isolated edges of (V, P), (#single-component) be the number of edges of M^* that connect a vertex of an isolated edge of (V, P) with a connection vertex of some (other) connected component of (V, P) and (#middle) be the number of edges that either intersect the middle vertex of a length 2 path connected component of (V, P) or are included within a triangle connected component of (V, P). For convenience, the definitions of the notation we use are summarized in Appendix B.

▶ **Observation 7.** *The total charge of all the edges of M^* according to the charging scheme π is at least (#component-free) + 2(#component-component) + (#single-single) + 1.5(#single-component) + (#middle).*

Proof. Since the edges of M^* counted by (#component-free) intersect a connection vertex, they must intersect a connected component of (V, P) which is not an isolated vertex or an isolated edge, and therefore, they charge this connected component one unit. Hence, the total charge by all the edges counted by (#component-free) is at least (#component-free). Similar logic shows that the total charge by all the edges counted by (#component-component), (#single-single), (#single-component) and (#middle) is at least 2(#component-component), (#single-single) , 1.5(#single-component) and (#middle), respectively. The observation now follows since the edges of M^* counted by (#component-free), (#component-component), (#single-single), (#middle) and (#single-component) are distinct. ◀

Combining Observations 6 and 7, we get the following inequality.

$$(\#\text{component-free}) + 2(\#\text{component-component}) + (\#\text{single-single}) \qquad (1)$$
$$+ 1.5(\#\text{single-component}) + (\#\text{middle}) \leq (\#\text{single}) + 3(\#\text{double}) + 3(\#\text{triangle}) \ .$$

In its current form, Inequality (1) is not very useful. We later derive from it a more convenient inequality, but before doing this we need to prove a few other inequalities. Let (#non-M^*-triangles) denote the number of triangle connected components of (V, P) that do not include any edge of M^* within them.

▶ **Lemma 8.** *The following inequalities hold*

$$\begin{aligned} (\#component\text{-}free) + (\#component\text{-}component) + (\#single\text{-}single) \\ + (\#middle) + (\#single\text{-}component) \geq |M^*| \ , \end{aligned} \qquad (2)$$

$$(\#double) + (\#triangle) - (\#non\text{-}M^*\text{-}triangles) \geq (\#middle) \ , \qquad (3)$$

$$2(\#single\text{-}single) + (\#single\text{-}component) \leq 2(\#single) \ , \qquad (4)$$

and they imply together

$$(\#component\text{-}free) + (\#component\text{-}component) + 2(\#single)$$
$$+ (\#double) + (\#triangle) - (\#non\text{-}M^*\text{-}triangles) \geq |M^*| \ .$$

Proof. Since every edge that is included in a connected component of (V, P) which is a path of length 2 must include the middle vertex of this path, every edge $e \in M^*$ that is not counted by either (#component-free), (#component-component), (#single-single), (#single-component) or (#middle) must either connect a vertex of an isolated edge of (V, P) to an isolated vertex or connect two isolated vertices of (V, P). However, such edges cannot exists. Specifically, assume towards a contradiction that (u, v) is an edge of M^* such that u is an isolated vertex of (V, P) and v is either another isolated vertex of (V, P) or belongs to an isolated edge of this graph. Then, the edge (u, v) should have been added by Algorithm 1 to P upon arrival, which contradicts the fact that its end point u ended up as an isolated vertex of (V, P). Hence, every edge $e \in M^*$ is counted by either (#component-free), (#component-component), (#single-single), (#single-component) or (#middle), which implies Inequality (2).

Recall that every edge counted by (#middle) must either be included in a triangle connected component of (V, P) or intersect the middle vertex of a path of length 2 connected component of (V, P). Since M^* is a matching, only one edge of M^* can intersect the middle vertex of a given length 2 path or be included in a given triangle, and therefore, every edge counted by (#middle) can be associated with a distinct path of length 2 or triangle component of (V, P) that is not counted by (#non-M^*-triangles), which implies Inequality (3).

Every edge counted by (#single-single) touches two end-points of isolated edges of (V, P). Similarly, every edge counted by (#single-component) intersects an end-point of an isolated edge of (V, P). Since every end-point of an isolated edge of (V, P) can be touched by at most a single edge of M^* because M^* is a matching, this implies that the number of end points of the isolated edges of (V, P) is at least $2(\#single\text{-}single) + (\#single\text{-}component)$. However, this number is also equal to $2(\#single)$, which implies Inequality (4). ◄

The last inequality in the previous lemma provides a lower bound on (#component-free)+ (#component-component), and one can view (#component-free)+(#component-component) as a count of edges of M^* that have potential to be added to A_2 in Algorithm 1. The next lemma is the promised derivative of Inequality (1), and it provides a lower bound on (#component-free). Observe that (#component-free) is a count of edges of M^* that have the potential to be added to A_1.

▶ **Lemma 9.** $2|M^*| \leq (\#component\text{-}free) - (\#non\text{-}M^*\text{-}triangles) + 2(\#single) + 4(\#double) + 4(\#triangle)$.

Proof. Adding twice Inequality (2) to Inequality (1), we get

$$2|M^*| - (\#component\text{-}free) - (\#single\text{-}single) - 0.5(\#single\text{-}component) - (\#middle)$$
$$\leq (\#single) + 3(\#double) + 3(\#triangle) \ .$$

The lemma now follows by adding Inequality (3) and half of Inequality (4) to the last inequality. ◄

So far we have shown lower bounds on the size of the sets of edges that have a potential to be added to A_1 or A_2 by Algorithm 1. Our next step is to lower bound the size of the sets A_1 and A_2 that Algorithm 1 ends up constructing using this potential.

(a) Path of length 2.   **(b)** M^*-triangle.
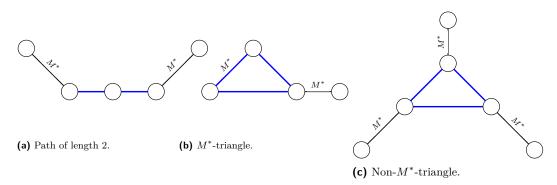
(c) Non-M^*-triangle.

Figure 1 A graphical study of the maximum number of M^* edges that can intersect connection vertices of various types of partial triangles. Sub-figures (a) and (b) show that at most two such edges can intersect the connection vertices of a path of length 2 and an M^*-triangle (i.e., a triangle that includes an edge of M^*). Sub-figure (c) shows that the connection vertices of a non-M^*-triangle can intersect up to 3 edges of M^*.

▶ **Lemma 10.** $3|A_1| \geq$ *(#component-free)* $-$ *(#non-M^*-triangles).*

Proof. We say that an edge e of M^* counted by (#component-free) is excluded by an edge $f \in A_1$ if e and f intersect the same connected component of (V, P). One can observe that every edge e counted by (#component-free) is excluded by some edge of A_1 (possibly itself) when Algorithm 1 terminates because otherwise Algorithm 1 would have added e to A_1, which would have resulted in e excluding itself. Therefore, we can upper bound (#component-free) by counting the number of edges excluded by the edges of A_1.

Let (u, v) be an edge of A_1, and assume without loss of generality that v is the end point of this edge which is an isolated vertex of (V, P). This implies that u is a connection vertex of a connected component C_u of (V, P) which is either a path of length 2 or a triangle. If C_u is a path of length 2, then the edge (u, v) can exclude only edges counted by (#component-free) that intersect either v or a connection vertex of C_u, and there can be only 3 such edges because M^* is a matching (see Figure 1a). Next, consider the case in which C_u is a triangle which is not counted by (#non-M^*-triangles). In this case there can be at most 2 edges of M^* intersecting C_u (see Figure 1b), and since (u, v) can exclude only edges that intersect either C_u or v, we get that it can exclude at most 3 edges.[4] It remains to consider the case in which C_u is a triangle counted by (#non-M^*-triangles). In this case, (u, v) can again exclude every edge of M^* that intersects C_u or v, and this time there can be at most 4 such edges (see Figure 1c). Combining all the above, we get that the number of edges excluded by all the edges of A_1 is at most

$$3|A_1| + |\{e \in A_1 \mid e \text{ intersects a triangle counted by (\#non-}M^*\text{-triangles)}\}| \ .$$

As explained above, this expression is an upper bound on (#component-free). Furthermore, since A_1 includes at most a single edge intersecting every connected component of (V, P), the second term in this expression is upper bounded by (#non-M^*-triangles). Therefore, we get

$$(\#\text{component-free}) \leq 3|A_1| + (\#\text{non-}M^*\text{-triangles}) \ .$$

The lemma now follows by rearranging this inequality.   ◀

---

[4] One of the two M^* edges intersecting C_u is guaranteed to be an edge of the triangle itself since the triangle is not counted by (#non-M^*-triangles). Since such edges cannot be counted by (#component-free), we get that the edge (u, v) can exclude at most 2 edges of (#component-free) rather than 3. However, we ignore this observation as it does not lead to a better approximation guarantee.

The next corollary now follows by combining Lemmata 9 and 10.

▶ **Corollary 11.** $2|M^*| \leq 3|A_1| + 2(\#single) + 4(\#double) + 4(\#triangle)$.

▶ **Lemma 12.** *It holds that* $4|A_2| \geq (\#component\text{-}component) + (\#component\text{-}free) - (\#non\text{-}M^*\text{-}triangles)$.

The proof of Lemma 12 is quite similar to the proof of Lemma 10. Therefore, and due to space constrained, we defer it to Appendix C. The next corollary now follows by combining Lemma 12 and the final inequality in Lemma 8.

▶ **Corollary 13.** $|M^*| \leq 4|A_2| + 2(\#single) + (\#double) + (\#triangle)$.

Let us now denote $L = (\#single) + (\#double) + (\#triangle) + \max\{|A_1|, |A_2|\}$. We argue below that L is a lower bound on the size of the solution produced by Algorithm 1. However, before proving this, let us show first that L is large.

▶ **Lemma 14.** $L \geq 7/13|M^*|$.

Proof. Plugging the definition of L into Corollaries 11 and 13 yields the inequalities $2|M^*| \leq 3L - (\#single) + (\#double) + (\#triangle)$ and $|M^*| \leq 4L - 2(\#single) - 3(\#double) - 3(\#triangle)$. Adding the first of these inequalities three times to the second one gives $7|M^*| \leq 13L - 5(\#single) \leq 13L$, where the second inequality holds since $(\#single)$ is non-negative by definition. The lemma now follows by rearranging the above inequality. ◀

As promised, we now argue that Algorithm 1 produces a matching of size at least L.

▶ **Lemma 15.** *Algorithm 1 outputs a matching of size at least L.*

Proof. Since Algorithm 1 outputs a maximum matching in $(V, P \cup A_1 \cup A_2)$, to prove the lemma it suffices to show that the graph $(V, P \cup A_1)$ includes a matching of size $(\#single) + (\#double) + (\#triangle) + |A_1|$ and the graph $(V, P \cup A_2)$ includes a matching of size $(\#single) + (\#double) + (\#triangle) + |A_2|$. We prove below only the claim regarding $(V, P \cup A_2)$. The claim regarding $(V, P \cup A_1)$ can be proved analogously.

Let H be the number of edges in A_2 that connect two non-isolated vertices of (V, P). Then, we classify the connected components of $(V, P \cup A_2)$ as follows, and show how to build a large matching M based on this classification.

- $(V, P \cup A_2)$ includes $(\#single) + (\#double) + (\#triangle) - |A_2| - H$ connected components that are (i) not an isolated node, and (ii) appear also in (V, P). Each one of these connected components contains at least one edge, and therefore, can contribute some edge to M.
- $(V, P \cup A_2)$ includes $|A_2| - H$ connected components that consist of a connected component C of (V, P) that has connection vertices and an edge e connecting a connection vertex of C to an isolated vertex of (V, P). One can observe that the combination of C and e must be either a path of length 3 or a triangle and an edge attached to one of its vertices, and in both cases this combined connected component contains two vertex disjoint edges which it can contribute to the matching M.
- $(V, P \cup A_2)$ includes H connected components that consist of two connected components C_1, C_2 of (V, P) that have connection vertices and an edge e connecting a connecting vertex of C_1 with a connecting vertex of C_2. There are three shapes that the connected component obtained in this way can take: a path of length 5, a triangle with a path of length 3 attached to one of its vertices or two triangles and an edge connecting them. However, one can observe that all these shapes include three vertex disjoint edges that can be contributed to the matching M.

By collecting from every connected component of $(V, P \cup A_2)$ the edges that it can contribute to M according to the above analysis, we get a matching in $(V, P \cup A_2)$ of size at least

$$[(\#\text{single}) + (\#\text{double}) + (\#\text{triangle}) - |A_2| - H] + 2[|A_2| - H] + 3H$$
$$= (\#\text{single}) + (\#\text{double}) + (\#\text{triangle}) + |A_2| \ . \qquad \blacktriangleleft$$

Lemmata 14 and 15 imply together the following corollary. Together with Observation 5, this corollary implies Theorem 1.

▶ **Corollary 16.** *Algorithm 1 is a* $^7/_{13}$*-approximation algorithm.*

Before concluding this section, we note that Theorem 2 is proved in the full version of this paper [20] by splitting the second pass of Algorithm 1 into two passes. One pass that constructs A_1, and a second pass that constructs A_2, while making sure not to use again connected components of (V, P) already used by A_1.

4   Three-Pass Algorithm for Triangle-Free Graphs

In this section we prove Theorem 3, which we repeat here for convenience.

▶ **Theorem 3.** *There exists a 3-pass* $(^{11}/_{18} = {}^1/_2 + {}^1/_9)$*-approximation semi-streaming algorithm for finding a maximum size matching in a triangle-free graph.*

We refer to the algorithm whose existence is guaranteed by Theorem 3 as TRIANGLE-FREEALG. In its first pass, TRIANGLEFREEALG constructs a maximal matching M_0 of G. Formally, the pseudocode for this pass appears as Algorithm 2.

◾ **Algorithm 2** TRIANGLEFREEALG – FIRST PASS.

---
1 Let $M_0 \leftarrow \varnothing$.
2 **for** *every edge e that arrives* **do**
3   │   Add e to M_0 if it does not intersect any edge that already belongs to M_0.

We say that an edge $e \in E$ is a *wing* if e includes exactly one vertex of $V(M_0)$. Intuitively, the reason we are interested in wings is that one can obtain an augmenting path[5] for M_0 by combining an edge $(u, v) \in M_0$ with two wings: one wing that intersects u and one wing that intersects v. The second pass of TRIANGLEFREEALG grows a set W of wings. Since we hope to construct multiple augmenting paths using these wings, the algorithm makes sure to limit the number of wings in W that intersect any given vertex u (specifically, the algorithm allows only a single wing in W to intersect u if $u \in V(M_0)$, and otherwise it allows up to two wings of W to intersect u). The pseudocode of this second pass appears as Algorithm 3.

Algorithm 3 also includes a post-processing step in which a set $\mathcal{P}_1$ of augmenting paths (with respect to M_0) is constructed using W. This is done by constructing an auxiliary multi-graph G_A over the vertices of $V \setminus V(M_0)$ in which there is an edge between two nodes $u, v \in V \setminus V(M_0)$ for every path $P_{u,v}$ of length 3 in $W \cup M_0$ between them. One can note that every such path $P_{u,v}$ must be an augmenting path consisting of an edge $e \in M_0$ and two wings from W: one intersecting u and an end-point of e, and the other intersecting v and the other end-point of e. Algorithm 3 finds a maximum size matching M_A in G_A, and then sets $\mathcal{P}_1$ to be the collection of (augmenting) paths corresponding to the edges of M_A.

---

[5] A path P is an augmenting path for a matching M if $M \oplus E(P)$ is a valid matching of size $|M| + 1$.

Algorithm 3 TRIANGLEFREEALG – SECOND PASS.

---

1 Let $W \leftarrow \varnothing$.
2 for *every edge e that arrives* **do**
3   **if** *e intersects exactly one vertex $u \in V(M_0)$* **then**
4   Let v denote the other end-point of e (i.e., the end-point that is not u).
5   **if** $\deg_W(u) < 1$ *and* $\deg_W(v) < 2$ **then**   Add e to W.

    `// Post-processing`
6 Let G_A be a multi-graph over the vertices $V \setminus V(M_0)$. For every path $P_{u,v}$ of length 3 in $W \cup M_0$ between two vertices $u, v \in V \setminus V(M_0)$, we add an edge (u, v) to the graph G_A. `// This is a multi-graph because there might be multiple such paths between a pair of vertices of` $V \setminus V(M_0)$.
7 Find a maximum size matching M_A in G_A.
8 Let $\mathcal{P}_1 \leftarrow \{P_{u,v} \mid (u,v) \in M_A\}$.

Consider now an edge $e \in M_0$ that does not appear in any path of $\mathcal{P}_1$ and is connected by some wing $w \in W$ to some vertex $u \notin V(M_0) \cup V(\mathcal{P}_1)$. The pair e, w can be extended into an augmenting path if one can find another wing w' connecting the other end of e (the end that does not intersect w) to a vertex $v \notin V(M_0) \cup V(\mathcal{P}_1)$ that is not u. The third pass of TRIANGLEFREEALG greedily constructs a collection $\mathcal{P}_2$ of augmenting paths in this way. A pseudocode of this pass appears as Algorithm 4. After completing the pass, Algorithm 4 returns the matching obtained by augmenting M_0 with the augmenting paths of $\mathcal{P}_1$ and $\mathcal{P}_2$.

Algorithm 4 TRIANGLEFREEALG – THIRD PASS.

---

1 Let $\mathcal{P}_2 \leftarrow \varnothing$.
2 for *every edge w' that arrives* **do**
3   **if** *there exist 4 vertices $u, a, b, v \in V \setminus (V(\mathcal{P}_1) \cup V(\mathcal{P}_2))$ such that: (i) $u \notin V(M_0)$, (ii) $w' = (u, a)$, (iii) $(a, b) \in M_0$ and (iv) $(b, v) \in W$* **then**
4   Add the path u, a, b, v to $\mathcal{P}_2$. `// Note that` $u \neq v$ `because otherwise` u, a, b, v `would have been a triangle.`

5 return $M_0 \oplus \left(\bigcup_{P \in \mathcal{P}_1 \cup \mathcal{P}_2} E(P) \right)$.

We begin the analysis of TRIANGLEFREEALG with the following lemma, which shows that this algorithm returns a matching, and also gives a basic lower bound on the size of this matching. Due to space constraints, the proof of this lemma is deferred to Appendix C.

▶ **Lemma 17.** *The paths in $\mathcal{P}_1$ and $\mathcal{P}_2$ are vertex disjoint, and therefore, the output of* TRIANGLEFREEALG *is a matching of size $|M_0| + |\mathcal{P}_1| + |\mathcal{P}_2|$.*

Using the last lemma we can also bound the space complexity of Algorithm 4. The technical proof of the next corollary can be found in the full version of this paper [20].

▶ **Corollary 18.** TRIANGLEFREEALG *is a semi-streaming algorithm.*

It remains to analyze the approximation ratio of TRIANGLEFREEALG. Our analysis roughly follows the flow of the algorithm, and thus, we begin by observing that the matching M_0 constructed in the first pass of this algorithm is of size at least $|M^*|/2$ (recall that M^* is a maximum size matching of G) because M_0 is a maximal matching of G by construction.

In its second pass, TRIANGLEFREEALG constructs the set W of wings. Our next objective is to lower bound the size of W. Towards this goal, we need to define W_M to be the set of all edges of M^* that are wings (we recall that an edge e is a wing if exactly one of its end points appear in $V(M_0)$).

▶ **Observation 19.** $|W_M| \geq 2(|M^*| - |M_0|)$.

Proof. Since M_0 is a maximal matching, every edge of M^* intersects at least one edge of M_0. Hence, every edge of W_M includes a single end-point of an edge of M_0, and every edge of $M^* \setminus W_M$ includes two end-points of edges of M_0 (the two end-points might belong to different edges or to the same edge), which implies $|M_0| \geq (|W_M| + 2|M^* \setminus W_M|)/2 = |M^*| - |W_M|/2$. Rearranging this inequality completes the proof of the observation. ◄

▶ **Lemma 20.** $|W| \geq \frac{2}{3}|W_M| \geq \frac{4}{3}(|M^*| - |M_0|)$.

Proof. Let $I = V(M_0) \cap V(W_M)$, and let I_F be the set of vertices in I that do not appear in any edge of W. Every vertex $a \in I_F \subseteq I$ must belong to some wing $w(a) \in W_M$ by the definition I. However, this wing was not added to W (because $a \in I_F$), which implies that the condition in Line 5 of Algorithm 3 evaluated to FALSE when $w(a)$ arrived. Since a is not covered by any edge of W (i.e., $\deg_W(v) = 0$), the fact that this condition evaluated to FALSE implies that the end point of $w(a)$ that does not belong to $V(M_0)$ must have a degree of 2 under W. Formally, if we denote by $u(a)$ the end point of $w(a)$ that does not belong to $V(M_0)$, then we must have $\deg_W(u(a)) = 2$.

We now observe that (i) every wing in W_M contains a disjoint vertex of $V \setminus V(M_0)$ because W_M is a subset of the optimal matching M^*, and (ii) every wing in W contains only one vertex of $V \setminus V(M)$ because it is a wing. These two observations imply together

$$|W| \geq \sum_{a \in I_F} \deg_W(u(a)) = 2|I_F| \ . \tag{5}$$

In contrast, since (i) every wing in W contains a single vertex of $V(M_0)$, and (ii) all the vertices of $I \setminus I_F \subseteq V(M_0)$ appear in some wing of W,

$$|W| \geq |I| - |I_F| = |W_M| - |I_F| \ , \tag{6}$$

where the equality holds since every edge of W_M is a wing, and therefore, intersects a single vertex of $V(M_0)$. The lemma now follows by adding two copies of Inequality (6) to Inequality (5). ◄

We now get to the analysis of the third pass of TRIANGLEFREEALG, and our first goal in this analysis is to identify a set of paths that have a potential (in some sense) to end up in $\mathcal{P}_2$. Let $\mathcal{P}'$ be the set of paths of length 3 in G that consist of a wing of W_M followed by an edge of M_0 and then a wing of W. We think of the paths in $\mathcal{P}'$ as directed from their W_M to their W edge, and consider two paths that differ only in their direction to be different paths. This is important because if there is an edge $e \in M_0$ incident to two edges $w_1, w_2 \in W \cap W_M$, then the path w_1, e, w_2 fulfills the requirements to belong to $\mathcal{P}'$ both when w_1 is considered the first edge in it and when w_2 is considered the first edge of the path. Thus, the fact that we treat the direction of the path as part of the path's definition allows both the paths w_1, e, w_2 and w_2, e, w_1 to appear in $\mathcal{P}'$.

▶ **Observation 21.** $|\mathcal{P}'| \geq \frac{10}{3}|M^*| - \frac{16}{3}|M_0|$.

Proof. Since $\deg_W(a) \le 1$ for every vertex $a \in V(M_0)$, there are $|W|$ end-points of M_0 that intersect an edge of W. Let us denote these end-points by V_W, and for every end-point $a \in V_W$, we denote by $b(a)$ the other end-point of the same edge of M_0. Formally, $V_W = V(M_0) \cap V(W)$, and $b(a)$ is the single element of the set $\{b \mid (a, b) \in M_0\}$. One can now observe that $\mathcal{P}'$ includes a (distinct) path for every wing of W_M that intersect $b(a)$ for some vertex $a \in V_W$. Therefore,

$$
\begin{aligned}
|\mathcal{P}'| &= |\{b(a) \mid a \in V_W\} \cap V(W_M)\}| \\
&\ge |\{b(a) \mid a \in V_W\}| + |V(W_M) \cap V(M_0)\}| - |V(M_0)| = |W| + |W_M| - |V(M_0)| \\
&\ge \tfrac{4}{3}(|M^*| - |M_0|) + 2(|M^*| - |M_0|) - |V(M_0)| = \tfrac{10}{3}|M^*| - \tfrac{16}{3}|M_0| \ ,
\end{aligned}
$$

where the first equality holds since $\{b(a) \mid a \in V_W\}$ is a subset of $V(M_0)$, and the last inequality follows from Observation 19 and Lemma 20. ◄

A path in $\mathcal{P}'$ has a potential to be added to $\mathcal{P}_2$ only if none of its vertices appears in $\mathcal{P}_1$. Let $\mathcal{P}''$ be the set of such paths (formally, $\mathcal{P}'' = \{P \in \mathcal{P}' \mid V(P) \cap V(\mathcal{P}_1) = \varnothing\}$). The following lemma lower bounds the size of $\mathcal{P}''$.

▶ **Lemma 22.** $|\mathcal{P}''| \ge |\mathcal{P}'| - 6|\mathcal{P}_1| \ge \tfrac{10}{3}|M^*| - \tfrac{16}{3}|M_0| - 6|\mathcal{P}_1|.$

Proof. The second inequality of the lemma follows from Observation 21, and therefore, we concentrate on proving the first inequality. Towards this goal, assume that $P' \in \mathcal{P}'$ is a path that intersects with a path $P_1 \in \mathcal{P}_1$ on an internal vertex. Since the middle edge of both paths is an edge of M_0, this implies that the two paths intersect on both their internal vertices. Furthermore, since both end-edges of P_1 and one end-edge of P' belong to W, there must be an internal vertex $a \in V(M_0)$ of both paths that intersects an edge of W in both paths. However, since $\deg_W(a) \le 1$, the edges of W intersecting a in both paths must be identical, which implies that the paths P' and P_1 intersect also on some end-point. Since P' and P_1 where chosen as general paths of $\mathcal{P}'$ and $\mathcal{P}_1$, respectively, that intersect on an internal node, this implies that the difference $|\mathcal{P}'| - |\mathcal{P}''|$ is equal to the number of paths in $\mathcal{P}'$ that intersect a path of $\mathcal{P}_1$ in an *end-point*. The rest of the proof is devoted to proving that the last number is at most $6|\mathcal{P}_1|$.

Since each path of $\mathcal{P}_1$ has only two end points, to prove that the paths of $\mathcal{P}_1$ intersect at most $6|\mathcal{P}_1|$ paths of $\mathcal{P}'$ at an end-point, it suffices to show that every vertex of $V \setminus V(M_0)$ can appear in at most 3 paths of $\mathcal{P}'$. To see why that is the case, consider an arbitrary vertex $u \in V \setminus V(M_0)$. If u belongs to some path $P' \in \mathcal{P}'$, then it must be in one of two roles as follows.

- If u is the last vertex of the path, then the last edge of the path is an edge $e \in W$ that includes u, and the other edges of the path P' are the single edge of M_0 intersecting e and the single edge of W_M intersecting e. Note that this means that the identity of the entire path is determined by the edge e, and therefore, the number of paths of $\mathcal{P}'$ in which u is the last vertex can be upper bounded by $\deg_W(u) \le 2$.
- If u is the first vertex of the path, then the first edge of the path is the single edge $e \in W_M$ that includes u, and the other edges of the path are the single edge $e' \in M_0$ that intersect e and the single edge $e'' \in W$ that intersects e'. Hence, the entire path is determined by the fact that u is its first vertex, and therefore, there can be only a single path in $\mathcal{P}'$ in which u is the first vertex. ◄

Originally, all the paths of $\mathcal{P}''$ can be picked in the third pass of TRIANGLEFREEALG (Algorithm 4) since they are vertex disjoint from the paths of $\mathcal{P}_1$. However, as Algorithm 4 starts to add paths to $\mathcal{P}_2$, it stops being possible to add some paths of $\mathcal{P}''$ to $\mathcal{P}_2$. Still, we

can lower bound the size of $\mathcal{P}_2$ in terms of the size of $\mathcal{P}''$. The proof of the next lemma is based on a logic similar to the one used in the previous proof. Thus, we defer this proof to Appendix C.

▶ **Lemma 23.** $|\mathcal{P}_2| \geq \frac{1}{6}|\mathcal{P}''| \geq \frac{5}{9}|M^*| - \frac{8}{9}|M_0| - |\mathcal{P}_1|$.

▶ **Corollary 24.** *The size of the output of* TRIANGLEFREEALG *is* $|M_0| + |\mathcal{P}_1| + |\mathcal{P}_2| \geq \frac{11}{18}|M^*| = (\frac{1}{2} + \frac{1}{9})|M^*|$.

Proof. The size of the output of TRIANGLEFREEALG is $|M_0| + |\mathcal{P}_1| + |\mathcal{P}_2|$ by Lemma 17, thus, we only need to lower bound this sum. To do this, note that

$$|M_0| + |\mathcal{P}_1| + |\mathcal{P}_2| \geq |M_0| + |\mathcal{P}_1| + \{\frac{5}{9}|M^*| - \frac{8}{9}|M_0| - |\mathcal{P}_1|\}$$
$$= \frac{5}{9}|M^*| + \frac{1}{9}|M_0| \geq \frac{5}{9}|M^*| + \frac{1}{18}|M^*| = \frac{11}{18}|M^*| \ ,$$

where the first inequality follows from Lemma 23, and the second inequality follows from the observation made at the beginning of this section (namely, that $|M_0|$ is a 1/2-approximation for $|M^*|$ because M_0 is a maximal matching).   ◀

Theorem 3 now follows from Corollaries 18 and 24.

5   Conclusion and Future Work

We have presented in this paper a new approach for semi-streaming algorithms for the maximum matching problem, and showed that this approach can be used to improve the state-of-the-art in two and three passes. Our approach calls for a more sophisticated logic in the first pass rather than simply building a maximal matching in a greedy fashion, as is done by previous algorithms. In our implementation of this approach, we greedily built in this pass connected components of size 3 (recall that greedily building a maximal matching is equivalent to greedily building connected components of size 2). Similarly, one can try to greedily construct in the first pass larger connected component, which we believe is likely to yield even better approximation guarantees. However, the analysis of algorithms based on such a first pass is likely to be inelegant, and to require a lot of case analysis since larger components allow many more configurations compared to smaller components. It might also be interesting to try to come up with an interesting algorithm that uses a completely different kind of logic in its first pass.

In addition to the above, we have used in this paper the traditional technique to improve over the state-of-the-art for three passes. Further improving the approximation ratio of two-pass and three-pass algorithms (or proving that this is not possible), is a nice question that is still open. We conclude by recalling that the most basic open question in this field of research is still breaking the (almost trivial) 1/2-approximation for single-pass algorithms. We hope that our new approach will lead to progress on both the above open questions.

───── **References** ─────

1   Kook Jin Ahn and Sudipto Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. *ACM Trans. Parallel Comput.*, 4(4):17:1–17:40, 2018. `doi:10.1145/3154855`.

2   Sepehr Assadi. A two-pass (conditional) lower bound for semi-streaming maximum matching. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 708–742. SIAM, 2022. `doi:10.1137/1.9781611977073.32`.

3　　Sepehr Assadi and Soheil Behnezhad. Beating two-thirds for random-order streaming matching. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 19:1–19:13, 2021. `doi:10.4230/LIPIcs.ICALP.2021.19`.

4　　Sepehr Assadi, Arun Jambulapati, Yujia Jin, Aaron Sidford, and Kevin Tian. Semi-streaming bipartite matching in fewer passes and optimal space. *arXiv e-prints*, pages arXiv–2011, 2020.

5　　Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1723–1742. SIAM, 2017. `doi:10.1137/1.9781611974782.113`.

6　　Sepehr Assadi, Gillat Kol, Raghuvansh R. Saxena, and Huacheng Yu. Multi-pass graph streaming lower bounds for cycle counting, max-cut, matching size, and other problems. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 354–364. IEEE, 2020. `doi:10.1109/FOCS46700.2020.00041`.

7　　Sepehr Assadi, S. Cliff Liu, and Robert E. Tarjan. An auction algorithm for bipartite matching in streaming and massively parallel computation models. In *4th Symposium on Simplicity in Algorithms (SOSA)*, pages 165–171, 2021. `doi:10.1137/1.9781611976496.18`.

8　　Sepehr Assadi and Vishvajeet N. Graph streaming lower bounds for parameter estimation and property testing via a streaming XOR lemma. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 612–625. ACM, 2021. `doi:10.1145/3406325.3451110`.

9　　Michel L. Balinski and Jaime Gonzalez. Maximum matchings in bipartite graphs via strong spanning trees. *Networks*, 21(2):165–179, 1991. `doi:10.1002/net.3230210203`.

10　Aaron Bernstein. Improved bounds for matching in random-order streams. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 12:1–12:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.ICALP.2020.12`.

11　Aaron Bernstein, Aditi Dudeja, and Zachary Langley. A framework for dynamic matching in weighted graphs. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 668–681. ACM, 2021. `doi:10.1145/3406325.3451113`.

12　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 *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1326–1344, 2016. `doi:10.1137/1.9781611974331.ch92`.

13　Graham Cormode, Hossein Jowhari, Morteza Monemizadeh, and S. Muthukrishnan. The sparse awakens: Streaming algorithms for matching size estimation in sparse graphs. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, pages 29:1–29:15, 2017. `doi:10.4230/LIPIcs.ESA.2017.29`.

14　Michael S. Crouch and Daniel M. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 28 of *LIPIcs*, pages 96–104, 2014. `doi:10.4230/LIPIcs.APPROX-RANDOM.2014.96`.

15　Jack Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of research of the National Bureau of Standards B*, 69(125-130):55–56, 1965.

16　Leah Epstein, Asaf Levin, Julián Mestre, and Danny Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM J. Discret. Math.*, 25(3):1251–1265, 2011. `doi:10.1137/100801901`.

17   Hossein Esfandiari, MohammadTaghi Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. *ACM Trans. Algorithms*, 14(4):48:1–48:23, 2018. `doi:10.1145/3230819`.

18   Hossein Esfandiari, MohammadTaghi Hajiaghayi, and Morteza Monemizadeh. Finding large matchings in semi-streaming. In Carlotta Domeniconi, Francesco Gullo, Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, editors, *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain*, pages 608–614. IEEE Computer Society, 2016. `doi:10.1109/ICDMW.2016.0092`.

19   Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005. `doi:10.1016/j.tcs.2005.09.013`.

20   Moran Feldman and Ariel Szarf. Maximum matching sans maximal matching: A new approach for finding maximum matchings in the data stream model. *CoRR*, abs/2109.05946, 2021. `arXiv:2109.05946`.

21   Manuela Fischer, Slobodan Mitrovic, and Jara Uitto. Deterministic $(1+\epsilon)$-approximate maximum matching with poly$(1/\epsilon)$ passes in the semi-streaming model and beyond. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 248–260. ACM, 2022. `doi:10.1145/3519935.3520039`.

22   Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 468–485, 2012. `doi:10.1137/1.9781611973099.41`.

23   John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973. `doi:10.1137/0202019`.

24   Sagar Kale and Sumedh Tirodkar. Maximum matching in two, three, and a few more passes over graph streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, volume 81 of *LIPIcs*, pages 15:1–15:21, 2017. `doi:10.4230/LIPIcs.APPROX-RANDOM.2017.15`.

25   Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1679–1697, 2013. `doi:10.1137/1.9781611973105.121`.

26   Michael Kapralov. Space lower bounds for approximating maximum matching in the edge arrival model. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1874–1893, 2021. `doi:10.1137/1.9781611976465.112`.

27   Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 734–751, 2014. `doi:10.1137/1.9781611973402.55`.

28   Christian Konrad. A simple augmentation method for matchings with applications to streaming algorithms. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 117 of *LIPIcs*, pages 74:1–74:16, 2018. `doi:10.4230/LIPIcs.MFCS.2018.74`.

29   Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, volume 7408 of *Lecture Notes in Computer Science*, pages 231–242, 2012. `doi:10.1007/978-3-642-32512-0_20`.

30 Christian Konrad and Kheeran K. Naidu. On two-pass streaming algorithms for maximum bipartite matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 19:1–19:18, 2021. `doi:10.4230/LIPIcs.APPROX/RANDOM.2021.19`.

31 Andrew McGregor. Finding graph matchings in data streams. In *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX) and 9th International Workshop on Randomization and Computation (RANDOM)*, pages 170–181, 2005. `doi:10.1007/11538462_15`.

32 Andrew McGregor and Sofya Vorotnikova. Planar matching in streams revisited. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 17:1–17:12, 2016. `doi:10.4230/LIPIcs.APPROX-RANDOM.2016.17`.

33 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)*, volume 61 of *OASICS*, pages 14:1–14:4, 2018. `doi:10.4230/OASIcs.SOSA.2018.14`.

34 Ami Paz and Gregory Schwartzman. A $(2 + \epsilon)$-approximation for maximum weight matching in the semi-streaming model. *ACM Trans. Algorithms*, 15(2):18:1–18:15, 2019. `doi:10.1145/3274668`.

35 Mariano Zelke. Weighted matching in the semi-streaming model. *Algorithmica*, 62(1-2):1–20, 2012. `doi:10.1007/s00453-010-9438-5`.

A Three-Pass Algorithm for General Graphs

In this section we prove Theorem 4, which we repeat here for convenience.

▶ **Theorem 4.** *There exists a 3-pass $(1/2 + 1/14.4)$-approximation semi-streaming algorithm for finding a maximum size matching in a general graph.*

The algorithm that we use to prove Theorem 4 is given as Algorithm 5. Since this algorithm is very similar to the algorithm TriangleFreeAlg presented in Section 4, we use below the terminology and notation defined in the last section.

Intuitively, the reason why TriangleFreeAlg does not apply to general graphs is that given an edge $(a, b) \in M_0$, a wing $(u, a) \in W_M$ and a wing $(b, v) \in W$, we are not guaranteed that these three edges form an augmenting path for the matching M_0 because they might represent a triangle. To overcome this hurdle, Algorithm 5 constructs two sets of edges in its second pass: a set W_1 constructed exactly like the set W in TriangleFreeAlg, and a set W_2 constructed in the same way, but while excluding the edges of W_1. Since W_1 and W_2 are disjoint, given an edge $(a, b) \in M_0$ and a wing $(u, a) \in W_M$, at most one of the sets W_1 or W_2 can contain a wing that forms a triangle together with these two edges, which intuitively allows us to bound the deterioration in the approximation guarantee resulting from the existence of such triangles.

We note that the analysis of TriangleFreeAlg up to Lemma 20 applies to Algorithm 5 with a single modification. Namely, Lemma 20 provides a lower bound on the size of the set W, which translates into an identical lower bound on the size of the corresponding set W_1 in Algorithm 5.

In the rest of this section, it will be convenient to work with the set W_2' constructed by Algorithm 6 (note that Algorithm 6 is used for analysis purposes only). Intuitively, W_2' is constructed in the same general way in which W_1 and W_2 are constructed; however, while all the edges of the input stream are considered in the construction of W_1, and only the edges of $E \setminus W_1$ are considered in the construction of W_2, the construction of W_2' takes into account the edges of $(E \setminus W_1) \cup W_M$.

■ **Algorithm 5** MAXIMUM MATCHING VIA AUGMENTING PATHS – GENERAL GRAPHS.

    // First Pass

1 Let $M_0 \leftarrow \varnothing$.

2 **for** *every edge e that arrives* **do**

3   Add e to M_0 if it does not intersect any edge that already belongs to M_0.

    // Second Pass

4 Let $W_1 \leftarrow \varnothing$, $W_2 \leftarrow \varnothing$.

5 **for** *every edge e that arrives* **do**

6   **if** *e intersects exactly one vertex $u \in V(M_0)$* **then**

7   Let v denote the other end-point of e (i.e., the end-point that is not u).

8   **if** $\deg_{W_1}(u) < 1$ *and* $\deg_{W_1}(v) < 2$ **then** Add e to W_1.

9   **else if** $\deg_{W_2}(u) < 1$ *and* $\deg_{W_2}(v) < 2$ **then** Add e to W_2.

    // Post-processing

10 Let G_A be a multi-graph over the vertices $V \setminus V(M_0)$. For every path $P_{u,v}$ of length 3 in $W_1 \cup W_2 \cup M_0$ between two distinct vertices $u, v \in V \setminus V(M_0)$, we add an edge (u, v) to the graph G_A. // This is a multi-graph because there might be multiple such paths between a pair of vertices of $V \setminus V(M_0)$.

11 Find a maximum size matching M_A in G_A.

12 Let $\mathcal{P}_1 \leftarrow \{P_{u,v} \mid (u, v) \in M_A\}$.

    // Third Pass

13 Let $\mathcal{P}_2 \leftarrow \varnothing$.

14 **for** *every edge w' that arrives* **do**

15   **if** *there exist 4 vertices $u, a, b, v \in V \setminus (V(\mathcal{P}_1) \cup V(\mathcal{P}_2))$ such that: (i) $u \notin V(M_0)$, (ii) $w' = (u, a)$, (iii) $(a, b) \in M_0$, (iv) $(b, v) \in W_1 \cup W_2$ and (v) $u \neq v$* **then**

16   Add the path u, a, b, v to $\mathcal{P}_2$.

17 **return** $M_0 \oplus \left(\bigcup_{P \in \mathcal{P}_1 \cup \mathcal{P}_2} E(P) \right)$.

Since W_2' is a subset of $W_1 \cup W_2$ by construction, the set $W_1 \cup W_2$ that is often referred to by Algorithm 5 is identical to the set $W_1 \cup W_2'$. Furthermore, one can observe that the lower bound proved by Lemma 20 for W_1 applies also to W_2' because all the edges of W_M are considered for addition to W_2' at some point (either during the construction of W_2 or in Algorithm 6). This implies the following observation.

▶ **Observation 25.** $|W_1| + |W_2'| \geq \frac{4}{3}|W_M|$.

We now define a multi-set $\mathcal{P}'$ similar to the set of the same name used in the analysis of TRIANGLEFREEALG. Specifically, $\mathcal{P}'$ includes every triangle or path obtained by combining an edge $(u, a) \in W_M$, an edge $(a, b) \in M_0$ and an edge (b, v) of either W_1 or W_2'. Moreover, if there are multiple options to obtain a path or triangle in this way, then the multiplicity of the path or triangle in $\mathcal{P}'$ will be equal to the number of these options. To make this point clearer, we provide a pseudocode for constructing $\mathcal{P}'$ as Algorithm 7 (again, Algorithm 7 is used for analysis purposes only).

▶ **Observation 26.** $|\mathcal{P}'| \geq \frac{20}{3}|M^*| - \frac{32}{3}|M_0|$.

■ **Algorithm 6** Construction of W_2'.

---

1 Let $W_2' \leftarrow W_2$.
2 **for** *every edge $(u, v) \in W_1 \cap W_M$* **do**
3 $\quad$ Assume without loss of generality that u is the end point of (u, v) that belongs to $V(M_0)$.
4 $\quad$ **if** $\deg_{W_2'}(u) < 1$ *and* $\deg_{W_2'}(v) < 2$ **then** Add (u, v) to W_2'.

■ **Algorithm 7** Construction of $\mathcal{P}'$.

---

1 Let $\mathcal{P}' \leftarrow \varnothing$.
2 **for** *every edge $(u, a) \in W_M$* **do**
3 $\quad$ **for** *every edge $(a, b) \in M_0$* **do**
4 $\quad\quad$ **for** *every edge $(b, v) \in W_1$* **do** Add the path/triangle $(u, a), (a, b), (b, v)$ to $\mathcal{P}'$.
5 $\quad\quad$ **for** *every edge $(b, v) \in W_2'$* **do** Add the path/triangle $(u, a), (a, b), (b, v)$ to $\mathcal{P}'$.

Proof. Repeating the proof of Observation 21, we get that at least $|W_1| + |W_M| - |V(M_0)|$ paths are added to $\mathcal{P}'$ in Line 4 of Algorithm 7, and at least $|W_2'| + |W_M| - |V(M_0)|$ paths are added to $\mathcal{P}'$ in Line 5 of Algorithm 7. Therefore,

$$|\mathcal{P}'| \geq |W_1| + |W_2'| + 2|W_M| - 2|V(M_0)| \geq (\tfrac{4}{3} + 2)|W_M| - 2|V(M_0)|$$
$$\geq 2(\tfrac{4}{3} + 2)(|M^*| - |M_0|) - 2|V(M_0)| = \tfrac{20}{3}|M^*| - \tfrac{32}{3}|M_0| \ ,$$

where the second inequality follows from Observation 25, and the last inequality follows from Observation 19. ◄

An element (path or triangle) of $\mathcal{P}'$ has a potential to be added to $\mathcal{P}_2$ by Algorithm 5 only if it is a path (i.e., not a triangle) and none of its vertices appears in $\mathcal{P}_1$. Let $\mathcal{P}''$ be the multi-set of such paths. The following lemma lower bounds the size of $\mathcal{P}''$.

▶ **Lemma 27.** $|\mathcal{P}''| \geq |\mathcal{P}'| - 12|\mathcal{P}_1| - |M_0| \geq \tfrac{20}{3}|M^*| - \tfrac{35}{3}|M_0| - 12|\mathcal{P}_1|$.

Proof. The second inequality of the lemma follows from Observation 26, and therefore, we concentrate on proving the first inequality. Let $\tilde{\mathcal{P}}'$ be the multi-set of paths/triangles from $\mathcal{P}'$ that do not intersect any vertex of $\mathcal{P}_1$. Repeating the proof of Lemma 22, we get that $\tilde{\mathcal{P}}'$ contains all the paths/triangles added to $\mathcal{P}'$ by Line 4 of Algorithm 7 except for up to $6|\mathcal{P}_1|$ paths/triangles, and the same is true for the paths/triangles added to $\mathcal{P}'$ by Line 5 of Algorithm 7. Since every path/triangle in $\mathcal{P}'$ was added to this mutli-set by either Line 4 or Line 5 of Algorithm 7, we get $|\tilde{\mathcal{P}}'| \geq |\mathcal{P}'| - 12|\mathcal{P}_1|$.

Since $\mathcal{P}''$ includes every path of $\tilde{\mathcal{P}}'$, to complete the proof of the lemma it remains to show that $\tilde{\mathcal{P}}'$ contains at most $|M_0|$ triangles. To see this, we recall that every triangle (or path) in $\tilde{\mathcal{P}}'$ must include a single edge of M_0, and we claim that no two triangles in $\tilde{\mathcal{P}}'$ can share this edge (and therefore, the number of triangles is upper bounded by the number of edges in M_0). Assume towards a contradiction that this claim does not hold, i.e., that there exist two triangles $T_1, T_2 \in \tilde{\mathcal{P}}'$ sharing an edge $e \in M_0$. Each one of these triangles must include one edge of W_M. Let e_1 and e_2 denote the edges of W_M in T_1 and T_2, respectively, and let e_1' the single edge of T_1 which is not e or e_1 and e_2' be the single edge of T_2 which is not either e or e_2. We now need to consider two cases. The first case is when $e_1 = e_2$. In this case e_1' and e_2' must be also identical, and cannot belong to W_M because $e_1 = e_2$ belongs to

W_M and W_M is a subset of the matching M^*. However, this leads to a contradiction because one of the edges e_1' or e_2' must belong to W_1, and the other of these edges must belong to W_2', and the sets W_1 and W_2' can intersect only on edges of W_M.

It remains to consider the case in which $e_1 \neq e_2$. Let u_1, u_2 be the end-points of these edges, respectively, that do not belong to the edge e of M_0. Since $e_1 \neq e_2$ are edges of the W_M, which is a subset of the matching M^*, u_1 and u_2 must be distinct. Consider now the path e_1', e, e_2'. One can observe that this is indeed a path because (i) $u_1 \neq u_2$ and (ii) the fact that e_1 and e_2 are vertex disjoint implies that e_1' and e_2' intersect different end-points of e. Furthermore, since $T_1, T_2 \in \tilde{\mathcal{P}}'$, this path does not intersect any vertex of $\mathcal{P}_1$, and thus, its existence contradicts the maximality of the matching M_A constructed by Algorithm 5 because both e_1' and e_2' belong to $W_1 \cup W_2' = W_1 \cup W_2$. ◀

We are now ready to lower bound the number of augmenting paths found by Algorithm 5 during its third pass.

▶ **Lemma 28.** $|P_2| \geq |\mathcal{P}''|/12 \geq \frac{5}{9}|M^*| - \frac{35}{36}|M_0| - |\mathcal{P}_1|$.

Proof. The proof of the lemma is very similar to the proof of Lemma 23, except that now every path of $\mathcal{P}_2$ might get a charge of up to 12 because the paths of $\mathcal{P}''$ originally added to $\mathcal{P}'$ by Line 4 of Algorithm 7 can contribute up to 6 to this charge, and the same goes for the paths of $\mathcal{P}''$ originally added to $\mathcal{P}'$ by Line 5 of this algorithm. ◀

Theorem 4 now follows from Corollary 18 and the next corollary.

▶ **Corollary 29.** *The size of the matching produced by Algorithm 5 is at least* $(\frac{1}{2} + \frac{1}{14.4})|M^*|$.

Proof. By Lemma 17, the size of the matching produced by Algorithm 5 is at least

$$|M_0| + |\mathcal{P}_1| + |\mathcal{P}_2| \geq \tfrac{5}{9}|M^*| + \tfrac{1}{36}|M_0| \geq \tfrac{5}{9}|M^*| + \tfrac{1}{72}|M^*| = (\tfrac{1}{2} + \tfrac{1}{14.4})|M^*| \ ,$$

where the first inequality holds by Lemma 28, and the second inequality holds since M_0 (as a maximal matching) is of size at least $\frac{1}{2}|M^*|$. ◀

B   Notation Summary

The next table summarizes the notation used in the analyses of our non-`MMF` algorithms.

Notation	Explanation
(#single)	The number of isolated edges in (V, P).
(#double)	The number of connected components in (V, P) that are paths of length 2.
(#triangle)	The number of triangles in (V, P).
(#component-free)	The number of edges of M^* that connect a connection vertex of a connected component of (V, P) to an isolated vertex of (V, P).
(#component-component)	The number of edges of M^* that connect connection vertices of two different connected components of (V, P).
(#single-single)	The number of edges of M^* whose two end points belong to (not necessarily distinct) isolated edges of (V, P).
(#single-component)	The number of edges of M^* that either connect a vertex of an isolated edge of (V, P) with a connection vertex of some (other) connected component of (V, P).
(#middle)	The number of edges that either (i) intersect the middle vertex of a length 2 path connected component of (V, P), or (ii) are included within a triangle connected component of (V, P).

C Omitted Proofs

▶ **Lemma 12.** *It holds that $4|A_2| \geq$ (#component-component) + (#component-free) −
(#non-M^*-triangles).*

Proof. The proof of Lemma 12 is very similar to the proof of Lemma 10, and therefore,
we only sketch it. We first define that an edge $e \in A_2$ excludes an edge f of M^* counted
by either (#component-component) or (#component-free) if they both intersect the same
connected component of (V, P). Like in the proof of Lemma 10, it can be argued that
(#component-component) + (#component-free) is upper bounded by the total number of
edges of M^* excluded by the edges of A_2, and on the other hand, every edge e of A_2 excludes
up to $4 + T(e)$ edges, where $T(e)$ is the number of triangles counted by (#non-M^*-triangles)
that intersect e. Therefore,

$$(\text{\#component-component}) + (\text{\#component-free}) \leq \sum_{e \in A_2} [4 + T(e)]$$

$$\leq 4|A_2| + (\text{\#non-}M^*\text{-triangles}) \ ,$$

where the second inequality holds since every connected component of (V, P) intersects only
a single edge of A_2. The lemma now follows by rearranging the last inequality. ◀

▶ **Lemma 17.** *The paths in $\mathcal{P}_1$ and $\mathcal{P}_2$ are vertex disjoint, and therefore, the output of
TriangleFreeAlg is a matching of size $|M_0| + |\mathcal{P}_1| + |\mathcal{P}_2|$.*

Proof. Given the above discussion, it is clear that all the paths in $\mathcal{P}_1 \cup \mathcal{P}_2$ are augmentation
paths with respect to M_0, which implies that the first part of the lemma indeed implies
the second part. Furthermore, one can observe that the condition in Line 3 of Algorithm 4
guarantees that the paths in $\mathcal{P}_2$ are vertex disjoint from each other and from the paths of
$\mathcal{P}_1$. Thus, to complete the proof of the lemma, it remains to argue that the paths in $\mathcal{P}_1$ are
also vertex disjoint.

Recall that the end-points of every path in $\mathcal{P}_1$ belong to $V \setminus V(M_0)$ and the internal points
of these paths belong to $V(M_0)$. Hence, to show that the paths in $\mathcal{P}_1$ are vertex disjoint, it
suffices to argue this separately for their end-points and their internal nodes. Every path
$P_{u,v} \in \mathcal{P}_1$ corresponds to an edge (u, v) in the matching M_A. Since the end-points of the
path $P_{u,v}$ are also the end-points of this edge, we get that the paths in $\mathcal{P}_1$ must have disjoint
end-points because M_A is a matching. Consider now some path $P_{u,v} \in \mathcal{P}_1$, and let us denote
the internal nodes of this path by a and b. Since a and b appear only in the edge (a, b) of M_0
(because M_0 is a matching), we get that if one of them belongs to a path of $\mathcal{P}_1$, then the other
belongs to this path as well. Furthermore, by Line 5 of Algorithm 3, $\deg_W(a) = \deg_W(b) = 1$,
which implies that any path of $\mathcal{P}_1$ that includes the nodes a and b as internal nodes must in
fact be identical to $P_{u,v}$ itself. Hence, no two paths in $\mathcal{P}_1$ share internal nodes. ◀

▶ **Lemma 23.** $|\mathcal{P}_2| \geq \frac{1}{6}|\mathcal{P}''| \geq \frac{5}{9}|M^*| - \frac{8}{9}|M_0| - |\mathcal{P}_1|$.

Proof. We begin the proof by observing that no edge $e \in M_0$ is connect by two distinct
wings $w_1, w_2 \in W$ to vertices of $V \setminus (V(M_0) \cup V(\mathcal{P}_1))$. Assume towards a contradiction that
this is not true, then there is an edge e in G_A corresponds to the path P defined as w_1, e, w_2.
Since M_A is a maximum matching in G_A, it must include at least one edge that contains
some end-point of P (otherwise, the edge corresponding to P could be added to M_A, which
violates its maximality); which contradicts the definition of either w_1 or w_2.

For every path $P'' \in \mathcal{P}''$, let us charge a cost of 1 to some path of $\mathcal{P}_2$ that intersects
it. To see why such a path must exist, let us denote by e_M the edge of P'' that belongs to
W_M (the first edge of P''). When e_M arrives, the path P'' was one candidate to be added

to $\mathcal{P}_2$ by Algorithm 4. If this candidate was still feasible at this time (in the sense that it was vertex disjoint from $\mathcal{P}_2$), then Algorithm 4 must have added either P'' to $\mathcal{P}_2$ or another path that includes e_M. In either case, following the arrival of e_M, some path intersecting P'' (which is possibly P'' itself) appears in $\mathcal{P}_2$ – and can be charged.

Our next goal is to show that the total cost charged to any single path of $\mathcal{P}_2$ is at most 6, which implies the lemma because the total cost charged to all the paths of $\mathcal{P}_2$ is exactly $|\mathcal{P}''|$. We do that by making two observations.

- Since $\mathcal{P}'' \subseteq \mathcal{P}'$, we get by the proof of Lemma 22 that at most 3 paths of $\mathcal{P}''$ can include any given vertex $u \in V \setminus V(M_0)$.
- Our second observation is that, if a path $P'' \in \mathcal{P}''$ intersects a path $P_2 \in \mathcal{P}_2$, then they must intersect on an end-point of P_2. Assume towards a contradictions that they only intersect on an internal node a. Since the middle edges of both paths are edges of M_0 that include a, both internal edges must be the same. Let us denote this internal edge by e. Furthermore, as explained above, there can be only a single edge $w \in W$ that intersects e and does not include a vertex of $V(\mathcal{P}_1)$. This edge must belong also to both paths, and therefore, the end-point of w that does not belong to $V(M_0)$ is an end-point of both P'' and P_2.

Combining the above two observations, we get that, for every path $P_2 \in \mathcal{P}_2$, only paths of $\mathcal{P}''$ intersecting an end-point of P_2 can charge a cost to P_2, and there can be at most 3 paths of $\mathcal{P}''$ intersecting each such end-point. Since P_2 has only two end-points, this implies that at most 6 paths of $\mathcal{P}''$ can charge P_2. ◀