Deterministic Algorithms for Maximum Matching on General Graphs in the Semi-Streaming Model

Sumedh Tirodkar[1]
IDSIA, USI-SUPSI, Manno, Switzerland
sumedh.tirodkar@idsia.ch

Abstract

We present an improved deterministic algorithm for Maximum Cardinality Matching on general graphs in the Semi-Streaming Model. In the *Semi-Streaming* Model, a graph is presented as a sequence of edges, and an algorithm must access the edges in the given sequence. It can only use $O(n \operatorname{polylog} n)$ space to perform computations, where n is the number of vertices of the graph. If the algorithm goes over the stream k times, it is called a k-pass algorithm. In this model, McGregor [28] gave the currently best known randomized $(1+\varepsilon)$-approximation algorithm for maximum cardinality matching on general graphs, that uses $(1/\varepsilon)^{O(1/\varepsilon)}$ passes. Ahn and Guha [1] later gave the currently best known deterministic $(1+\varepsilon)$-approximation algorithms for maximum cardinality matching: one on bipartite graphs that uses $O\left(\log\log(1/\varepsilon)/\varepsilon^2\right)$ passes, and the other on general graphs that uses $O(\log n \cdot \operatorname{poly}(1/\varepsilon))$ passes (note that, for general graphs, the number of passes is dependent on the size of the input). We present the first deterministic algorithm that achieves a $(1+\varepsilon)$-approximation on general graphs in only a constant number $\left((1/\varepsilon)^{O(1/\varepsilon)}\right)$ of passes.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Semi Streaming, Maximum Matching

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2018.39

Acknowledgements The author thanks Ashish Chiplunkar, Sagar Kale, Sundar Vishwanathan, and anonymous reviewers for their helpful feedback on the writeup.

1 Introduction

Matching is one of the most well-studied problems in combinatorial optimization. See Schrijver's book [31] and references therein for a comprehensive overview of the classical work. There are polynomial time algorithms known for both weighted and unweighted maximum matching on general graphs [29]. With the advancement of internet and social networks, large amount of data is generated, and often the input graph is so huge that the entire graph may not fit even inside large size RAMs. One way to tackle this problem is to provide the input to the algorithm in pieces. For instance, edges can be provided to the algorithm one by one, or vertices can be provided one by one along with all the edges from that vertex to the previously revealed vertices. The maximum matching problem has been studied in various models in which the input is provided piecewise, for instance, the online preemptive/non-preemptive model (vertex/edge arrival) [15, 11, 23, 7], the dynamic graph model [5, 6, 4, 32], the streaming model [18, 28, 33, 14], etc. In these models, random order [25, 26] arrivals have also been considered.

[1] This work was done when the author was a Post Doctoral Fellow in the School of Technology and Computer Science at TIFR, Mumbai, India.

We study the maximum matching[2] problem in the semi-streaming model. Feigenbaum et al. [18] were the first to consider this problem in the semi-streaming model. 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. A semi-streaming algorithm can use $O(n \operatorname{polylog} n)$ space only, where n is the number of vertices in the input graph. Note that a matching can have size $\Omega(n)$, so $\Omega(n \log n)$ space is necessary. If a semi-streaming algorithm goes over the stream k times, it is called a k-pass algorithm. We say that an algorithm is an α-approximation algorithm if the size of the matching output by the algorithm is at least $\frac{1}{\alpha}$ times the size of an optimum matching, and we say that the matching is α-approximate.

McGregor [28] gave the first $(1+\varepsilon)$-approximation algorithm for maximum matching in this model. This algorithm is randomized, and uses a constant number of passes $\left((1/\varepsilon)^{O(1/\varepsilon)}\right)$. Eggert et al. [13] later improved this result on bipartite graphs by giving a $(1+\varepsilon)$-approximation deterministic algorithm that uses $O(1/\varepsilon^5)$ passes. Ahn and Guha [1] also gave linear programming based $(1+\varepsilon)$-approximation deterministic algorithms. For bipartite graphs, they further improved the results. Their algorithm uses only $O(\log\log(1/\varepsilon)/\varepsilon^2)$ passes. But for general graphs, their algorithm uses $O(\log n \cdot \operatorname{poly}(1/\varepsilon))$ passes. The number of passes is dependent on the size of the input. There are no known $(1+\varepsilon)$-approximation deterministic algorithms on general graphs that use a constant number of passes, and it is not clear how to extend the deterministic algorithms on bipartite graphs [13, 1], to deterministic algorithms on general graphs that use only a constant number of passes. In this paper, we present the first $(1+\varepsilon)$-approximation deterministic algorithm for maximum matching on general graphs that uses only a constant number $\left((1/\varepsilon)^{O(1/\varepsilon)}\right)$ of passes.

We first present a $(1+\varepsilon)$-approximation deterministic algorithm for maximum matching on bipartite graphs, that uses $(1/\varepsilon)^{O(1/\varepsilon)}$-passes, which will help in understanding the main techniques behind the algorithm on general graphs. Note that the algorithm on bipartite graphs in itself does not hold much value, as we have already mentioned earlier that there exist $\operatorname{poly}(1/\varepsilon)$-pass $(1+\varepsilon)$-approximation algorithms on bipartite graphs.

These algorithms build on the techniques used in the two-pass algorithm for maximum matching presented in [20]. In the first pass, the algorithm [20] finds a maximal matching, and in the second pass, it finds a semi-matching between matched and unmatched vertices from the first pass. A (λ_X, λ_Y) *Semi-Matching* is defined as a set of edges such that at most λ_X edges are incident on any vertex in X, and at most λ_Y edges are incident on any vertex in Y, when $X \cap Y = \emptyset$. The algorithm uses this semi-matching to find length 3-augmenting paths in the maximal matching. In the following paragraph, we give a brief overview of how these techniques can be used to find longer augmenting paths in bipartite graphs, and some of the difficulties in extending them to general graphs.

Technical Overview

Let M^* denote some optimal matching in the given graph, and let M denote some maximal matching. For any integer $\ell \geq 1$, a connected component of $M \cup M^*$ that has a path of length $(2\ell+1)$ is called a length $(2\ell+1)$-*augmenting* path (non-augmenting otherwise) with respect to M^*. In general, a length $(2\ell+1)$-augmenting path contains ℓ edges in M, and $(\ell+1)$ edges not in M. We call an edge in M $(2\ell+1)$-*augmentable*, if it belongs to a length $(2\ell+1)$-augmenting path. A length $(2\ell+1)$-*augmentation* is a replacement of edges in M, by edges not in M, from a length $(2\ell+1)$-augmenting path.

[2] Unless explicitly mentioned, Maximum Matching means Maximum Cardinality Matching.

It is widely known, and can also be easily proved that in a maximal matching M, if there are no augmenting paths of length less than $(2/\varepsilon+1)$ with respect to some optimum matching M^*, then the matching M is $(1+\varepsilon)$-approximate. We present deterministic algorithms which output a matching M, that contains a negligible number of augmenting paths of length less than $(1/\varepsilon+1)$ with respect to some optimum matching M^*. We use Lemma 1 to prove that this matching is an $(1+O(\varepsilon))$-approximation to the optimum matching. Informally, the lemma states that if there exist a negligible number of augmenting paths of length less that $(1/\varepsilon+1)$ in a maximal matching M, then M is a good approximation to any optimum matching.

The algorithms for bipartite graphs and general graphs have a common high level idea. Each algorithm begins by finding a maximal matching M in the first pass. Then it first carries out length 3-augmentations of some edges in M, such that after a constant number of passes, the number of remaining 3-augmentable edges in M with respect to any fixed optimal matching M^* is negligible. Then it similarly carries out length 5-augmentations, followed by length 7-augmentations, and so on, up to length $(1/\varepsilon)$-augmentations.

Suppose there are no augmenting paths in M of length less than $(2\ell+1)$. Then, a length $(2\ell+1)$-augmenting path is found as follows. Suppose $au_1v_1u_2v_2u_3\dots v_{\ell-1}u_\ell v_\ell b$ denotes some length $(2\ell+1)$-augmenting path in M with respect to M^*. (Note that edges $u_iv_i \in M$, for all $1 \le i \le \ell$.) In one pass, (a constant fraction of) the outermost edges of any length $(2\ell+1)$-augmenting path (for instance au_1 and $v_\ell b$) are added to a semi-matching S. Now that the algorithm already knows the outermost four edges (for instance $au_1, u_1v_1, u_\ell v_\ell, v_\ell b$) in the augmenting path, it tries to find the length $(2\ell-3)$-augmenting path $v_1u_2v_2u_3\dots v_{\ell-1}u_\ell$. Using this length $(2\ell-3)$-augmenting path $v_1u_2v_2u_3\dots v_{\ell-1}u_\ell$, and edges in M and S, the algorithm finds a length $(2\ell+1)$-augmenting path.

For general graphs, suppose such a procedure is successful in finding a middle length $(2\ell'-3)$ path of some length $(2\ell+1)$-augmenting path. Then how to ensure that the edges stored in the semi-matchings, along with the edges in M, do not form an odd length cycle with this middle length $(2\ell'-3)$ path? We do not face this issue for bipartite graphs as they do not contain any odd length cycles, and so we require new ideas (Directed Semi-Matchings – defined in Section 2) to extend the described techniques to find longer augmenting paths in general graphs.

A Comparison with McGregor's Randomized Algorithm

McGregor's algorithm [28] begins by finding a maximal matching M. To find length $(2\ell+1)$-augmenting paths (for all $\ell \le (1/(2\varepsilon))$), the algorithm randomly partitions the vertices in G into $(\ell+2)$ layers $L_0, L_1, \dots, L_\ell, L_{\ell+1}$. The unmatched vertices in G are added to either L_0 or $L_{\ell+1}$, and the matched vertices are added at random to the layers $L_1, \dots, L_\ell$. The algorithm only stores the edges between two consecutive layers. This makes the graph bipartite, and removes the possibility of any odd length cycle. The algorithm works by finding maximal matchings, first between the first and second levels and then between the vertices in the second that were matched in the first matching, and the third level, and then between the nodes in the third level that were matched in the second matching and the fourth level and so on. This gives node-disjoint $(\ell+1)$-paths, which correspond to length $(2\ell+1)$-augmenting paths in G. The random partitioning operation keeps roughly a $(1/\ell)^{O(\ell)}$ fraction of the length $(2\ell+1)$-augmenting paths, as each such augmenting path "survives" the partitioning by this probability. By repeating this process by a factor of $\ell^{O(\ell)}$, almost all length $(2\ell+1)$-augmenting paths are found. The algorithm finds a $(1+\varepsilon)$-approximate matching with probability $(1-f)$ by running $O(\log(1/f))$ copies of this procedure in parallel.

Like McGregor's algorithm, our algorithm also ensures that a negligible number of $(2\ell+1)$-augmentable edges remain in M (for all $\ell \leq (1/(2\varepsilon))$). As pointed out earlier, the main difficulty in finding augmenting paths in general graphs is due to the existence of odd length cycles (this challenge is present in nearly all variants of the matching problem in non-bipartite graphs, including in the polynomial time algorithms for this problem in the offline setting [29]). McGregor's algorithm handles this issue by considering a random bipartite subgraph of the original graph in each augmentation phase. We rely on the use of directed semi-matchings. The major technical challenge lies in finding which edges need to be stored in the directed semi-matchings. The algorithm needs to recognize and ensure that it does not store edges (in the directed semi-matchings) that form an odd length cycle. This turns out to be non-trivial. Section 4 gives more details.

Note. The random bipartitioning operation (from McGregor's algorithm [28]) in general graphs, keeps roughly a $(1/\ell)^{O(\ell)}$ fraction of the length $(2\ell+1)$-augmenting paths. And hence, this process has to be repeated $\ell^{O(\ell)}$ times to find almost all length $(2\ell+1)$-augmenting paths. So, it is unlikely that this idea can be extended to get a poly$(1/\varepsilon)$-pass $(1+\varepsilon)$-approximation algorithm on general graphs. Our deterministic algorithm explicitly bypasses the barrier of "blossoms" (hereafter, we refer to an odd length alternating (edges not in M and edges in M) cycle that starts and ends at an unmatched vertex as a *blossom*), and hence can be viewed as a step towards achieving a poly$(1/\varepsilon)$-pass $(1+\varepsilon)$-approximation algorithm for general graphs.

Related Work

The algorithm that finds a maximal matching (in which an edge is added to the matching M if there are no edges in M incident on any of its endpoints) is a trivial one-pass 2-approximation algorithm, and no (randomized/deterministic) one-pass algorithm with approximation ratio better than 2 is known for maximum matching although this model was introduced over a decade ago. It remains as one of the major open problems in the streaming community [27]. Goel, Kapralov, and Khanna [19], using connection between streaming complexity and communication complexity, proved that for any $\varepsilon > 0$, a one-pass semi-streaming $(3/2-\varepsilon)$-approximation algorithm does not exist. Later, Kapralov [21], building on those techniques, showed the non-existence of a one-pass semi-streaming $(e/(e-1)-\varepsilon)$-approximation algorithms for any $\varepsilon > 0$. Konrad et al. [25] showed that if the algorithm is allowed one extra pass, then a better than 2-approximate matching can be obtained, by giving a two-pass algorithm. Later, Esfandiari et al. [16] improved these results for bipartite graphs, and Kale and Tirodkar [20] gave improved results on triangle-free as well as general graphs.

Feigenbaum et al. [18] gave a $(3/2+\varepsilon)$-approximation deterministic algorithm on bipartite graphs that uses $O(\log(1/\varepsilon)/\varepsilon)$ passes. Later, Ahn and Guha [1] gave a $(3/2+\varepsilon)$-approximation deterministic algorithm on general graphs that uses $O(\log(1/\varepsilon)/\varepsilon^2)$ passes. Kale and Tirodkar [20] improved both these results by giving a $(3/2+\varepsilon)$-approximation deterministic algorithm on general graphs that use only $O(1/\varepsilon)$ passes.

Feigenbaum et al. [18] gave the first one-pass algorithm for maximum weight matching, with an approximation ratio 6. Subsequent results improved this approximation ratio. Recently in a breakthrough, Paz and Schwartzman [30] gave a $(2+\varepsilon)$-approximation algorithm. The multi-pass version of the problem was considered first by McGregor [28], then by Ahn and Guha [1]. Chakrabarti and Kale [9] and Chekuri et al. [10] 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 been well studied [22, 17, 8, 2]. The Maximum Matching problem has also been studied on dynamic streams (in which edges can be added as well as deleted) [24, 3, 12].

1.1 Organization of the paper

After setting up notation in Section 2, we see a $(1+\varepsilon)$-approximation deterministic algorithm for maximum matching on bipartite graphs in Section 3. Then in Section 4, we see our main result – a $(1+\varepsilon)$-approximation deterministic algorithm on general graphs that uses a constant number of passes.

2 Preliminaries

Let M^* denote some optimal matching in the given graph, and let M denote some maximal matching. We define the following terms which are used in the subsequent sections.

▶ **Definition 2.1.** A (λ_X, λ_Y) *Directed Semi-Matching* is defined as a set of directed edges such that at most λ_X incoming edges are incident on any vertex in X, and at most λ_Y outgoing edges are incident on any vertex in Y.

For a (λ_X, λ_Y) Semi-Matching S, let $\deg_S(x)$ represent the number of edges in S incident on vertex x. For a directed semi-matching S, let $\deg^O_S(x)$ and $\deg^I_S(x)$ represent the number of outgoing and incoming edges, respectively, in S incident on vertex x. For an edge xy in the input stream, both the directed edges xy and yx are considered while finding the directed semi-matchings.

▶ **Definition 2.2.** While finding the middle length $(2\ell'+1)$ path of the length $(2\ell+1)$-augmenting paths, the end points of M which have a length $(\ell-\ell')$ alternating path (with edges in M and edges in the semi-matchings) to unmatched vertices, are called *free vertices*.

(For instance, the green vertices in Figure 1 are free vertices while finding the middle length $(2\ell-3)$-augmenting path.) Suppose $au_1v_1u_2v_2u_3\ldots v_{\ell-1}u_\ell v_\ell b$ denotes some augmenting path of length $(2\ell+1)$ in M with respect to M^*. Then, $v_{(\ell-\ell')/2}u_{(\ell-\ell')/2+1}v_{(\ell-\ell')/2+1}\cdots u_{(\ell+\ell')/2}\ v_{(\ell+\ell')/2}u_{(\ell+\ell')/2+1}$ denotes the middle length $(2\ell'+1)$ path of the augmenting path. Vertex $u_{(\ell-\ell')/2}$ (or $v_{(\ell+\ell')/2+1}$) is a free vertex if there exists a length $(\ell-\ell')$ path to unmatched vertices starting from $u_{(\ell-\ell')/2}$ (or $v_{(\ell+\ell')/2+1}$) via alternate edges in M and the semi-matchings maintained by the algorithm. When $\ell'=\ell$, $V\setminus V(M)$ are the free vertices.

We use the following lemma (similar to Lemma 1 in [28]) in analyzing the performance of the algorithms described in this paper.

▶ **Lemma 1.** *Let M^* be a maximum matching in G. If there are at most $\varepsilon^2|M|$ $(2\ell+1)$-augmentable edges in a maximal matching M with respect to M^*, for any $\ell \le (1/\varepsilon-1)/2$, then M is a $(1+3\varepsilon)$-approximate maximum matching.*

Proof. Let k_ℓ denote the number of $(2\ell+1)$-augmentable edges in M. In any length $(2\ell+1)$-augmenting path, the ratio of number of edges in M^* to the number of edges in M is $\frac{l+1}{l}$.

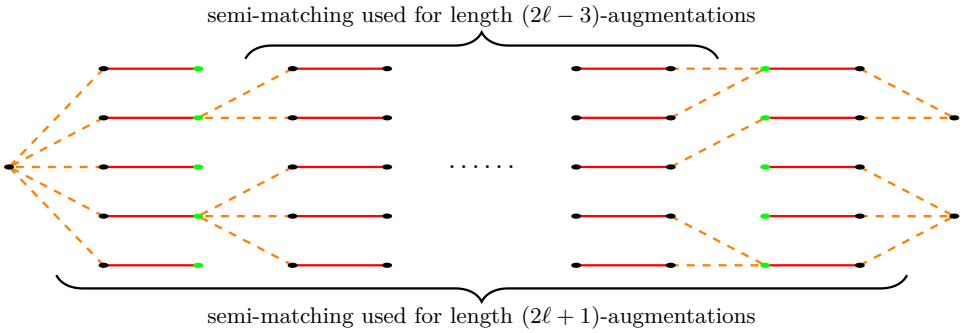

Figure 1 Length $(2\ell+1)$-augmentations. Solid red lines represent edges in M, and dashed orange lines represent edges that belong to S during the recursive steps.

Then,

$$\begin{aligned}
|M^*| &\leq 2k_1 + \frac{3}{2}k_2 + \frac{4}{3}k_3 + \cdots + \frac{(1/\varepsilon-1)/2+1}{(1/\varepsilon-1)/2}k_{(1/\varepsilon-1)/2} \\
&\quad + \frac{(1/\varepsilon+1)/2+1}{(1/\varepsilon+1)/2}(|M| - k_1 - k_2 - \cdots - k_{(1/\varepsilon-1)/2}) \\
&= \left(1 + \frac{2}{1/\varepsilon+1}\right)|M| + \left(2 - \frac{1/\varepsilon+3}{1/\varepsilon+1}\right)k_1 + \left(\frac{4}{3} - \frac{1/\varepsilon+3}{1/\varepsilon+1}\right)k_3 + \ldots \\
&\quad + \left(\frac{1/\varepsilon+1}{1/\varepsilon-1} - \frac{1/\varepsilon+3}{1/\varepsilon+1}\right)k_{(1/\varepsilon-1)/2} \\
&\leq (1+2\varepsilon)|M| + \varepsilon^2 \cdot |M| + \varepsilon^2 \cdot |M| + \cdots + \varepsilon^2 \cdot |M| \\
&\leq (1+3\varepsilon)|M|.
\end{aligned}$$

◀

For the sake of exposition, we ignore floors and ceilings during the analyses of the algorithms presented in Sections 3 and 4.

3 Warming Up: An Algorithm on Bipartite Graphs

In this section, we present a $(1/\varepsilon)^{O(1/\varepsilon)}$-pass $(1+\varepsilon)$-approximation deterministic algorithm for maximum matching on bipartite graphs.

The high level idea for the algorithm is already described in Section 1. What remains to be described is how the length $(2\ell+1)$-augmentations are performed. The function call for length $(2\ell+1)$-augmentations in the matching M assumes that there are no shorter augmenting paths (although there is a small number of them). For each function call, the algorithm finds a $(1/\varepsilon^4, 1)$ semi-matching S in one pass, between the free and matched vertices. For an edge $e \in M$, if there are no edges in S, incident on any of the endpoints of e, add e to M'. For an edge $e \in M$, if there is an edge in S, incident on one of the end points of e, add the other endpoint to the set V_1. Then, the function recursively calls AUGMENT$(\ell-2, V_1, M')$ to find length $(2\ell-3)$-augmenting paths for the matching M' in the graph induced on $V_1 \cup V(M')$. For the recursive call, vertices in V_1 are the free vertices, and vertices in $V(M')$ are the matched vertices. Using these length $(2\ell-3)$-augmenting paths, and edges in S, the algorithm finds length $(2\ell+1)$-augmenting paths in M. Figure 1 gives an illustration of the semi-matchings stored by the algorithm which are used for length $(2\ell+1)$-augmentations.

Algorithm 1 gives a formal description.

Algorithm 1 Deterministic Algorithm on Bipartite Graphs.

Find Maximal Matching M in the first pass.
for $\ell = 1$ `to` $1/(2\varepsilon)$ **do** ▷ To Remove almost all $(2\ell+1)$-augmenting paths
  for `Phase` $p = 1$ `to` $1/\varepsilon^{5\ell}$ **do**
    $M \leftarrow \text{AUGMENT}(\ell, V, M)$.
Output M.
function AUGMENT(ℓ, V, M) ▷ Function of length $(2\ell+1)$-augmentations.
  if $\ell = 0$ **then**
    Find Maximal Matching M' on vertex set V in one pass.
    $M \leftarrow M'$.
  else
    $S \leftarrow \emptyset$, $M' \leftarrow \emptyset$.
    $V_1 \leftarrow V \setminus V(M)$. ▷ Set of free vertices.
    $S \leftarrow \text{SEMI}(1, V(M), 1/\varepsilon^4, V_1)$.
    if $\ell > 1$ **then** ▷ For longer than length 3-augmentations
      $V_1 \leftarrow \emptyset$
      for all edges $uv \in M$ **do**
        if $\exists au \in S$ **then**
          $V_1 \leftarrow V_1 \cup \{v\}$. ▷ Free Vertices in the recursive call.
        if $\nexists$ edge in S on either u or v **then**
          $M' \leftarrow M' \cup \{uv\}$.
      $M' \leftarrow \text{AUGMENT}(\ell - 2, V_1, M')$. ▷ Find length $(2\ell-3)$-augmentations.
    $(2\ell+1)$-augment M greedily using edges in S and M'.
  return M.
function SEMI$(\lambda_X, X, \lambda_Y, Y)$ ▷ Finds a semi-matching in one pass.
  $S \leftarrow \emptyset$.
  for all edges xy such that $x \in X$, $y \in Y$ **do**
    if $\deg_S(x) < \lambda_X$ and $\deg_S(y) < \lambda_Y$ **then**
      $S \leftarrow S \cup \{xy\}$
  return S.

We begin the analysis for any length $(2\ell+1)$-augmentations for the matching M. The function assumes that there are no shorter length augmenting paths in M with respect to any optimal matching. Let E_ℓ denote the set of $(2\ell+1)$-augmentable edges in M with respect to an optimal matching M^*, such that $|E_\ell|$ is maximum.

Bad Edges

Suppose $au_1v_1u_2v_2u_3 \dots v_{\ell-1}u_\ell v_\ell b$ denotes any length $(2\ell+1)$-augmenting path in M with respect to M^*. Then, $v_{(\ell-\ell')/2}u_{(\ell-\ell')/2+1}v_{(\ell-\ell')/2+1} \dots u_{(\ell+\ell')/2}v_{(\ell+\ell')/2}\ u_{(\ell+\ell')/2+1}$ denotes the middle length $(2\ell'+1)$ path of the augmenting path. Edge $u_{(\ell-\ell')/2+1}v_{(\ell-\ell')/2+1} \in M'$ (or $u_{(\ell+\ell')/2}v_{(\ell+\ell')/2} \in M'$) is called a *b*ad edge, if there is no edge in S from $u_{(\ell-\ell')/2+1}$ (or $v_{(\ell+\ell')/2}$) to a free vertex, during the recursive call AUGMENT(ℓ', V_1, M'). Note that V_1 is the set of free vertices during a function call AUGMENT(ℓ', V_1, M').

Let $E^\ell_{B_1}$ denote a set of bad edges u_1v_1 (or $u_\ell v_\ell$) if no edge is added to S during a function call AUGMENT(ℓ, V, M) that is incident from the vertex u_1 (or v_ℓ) to some free vertex (in this case $V_1 = V \setminus V(M)$ are the free vertices).

▶ **Lemma 2.** $|E^{\ell}_{B_1}| \leq 2\varepsilon^4|M|$.

Proof. Suppose there is no edge in S incident from u_1 (or v_ℓ) to any unmatched vertex in V_1. This means that for all the edges incident from u_1 (or v_ℓ) to unmatched vertices, there already were $1/\varepsilon^4$ edges in S incident on the respective unmatched vertices. For $\ell > 1$, each edge in M can have at most one edge in S incident on its endpoints (as there are no length 3-augmentable edges in M). Hence, $|E^{\ell}_{B_1}|/\varepsilon^4 \leq |M|$.

For $\ell = 1$, each edge in M can have at most one edge in S incident on each its endpoints. Hence, $|E^{\ell}_{B_1}|/\varepsilon^4 \leq 2|M|$. ◀

After finding a $(1/\varepsilon^4, 1)$ semi-matching S from unmatched to matched vertices, the function call $\textsc{Augment}(\ell, V, M)$ makes a recursive call $\textsc{Augment}(\ell-2, V_1, M')$. Inside the recursive call, V_1 is the set of free vertices, and $V(M')$ is the set of matched vertices. Let $E^{\ell}_{B_2}$ denote the bad edges formed while finding a $(1/\varepsilon^4, 1)$ semi-matching inside the recursive call. By Lemma 2, $|E^{\ell}_{B_2}| \leq 2\varepsilon^4|M'| \leq 2\varepsilon^4|M|$.

If $au_1v_1u_2v_2u_3 \ldots v_{\ell-1}u_\ell v_\ell b$ denotes any length $(2\ell+1)$-augmenting path in M with respect to M^*, let E^{ℓ}_B denote the set of all the bad edges $u_iv_i \in E_\ell$ formed during one function call $\textsc{Augment}(\ell, V, M)$, and the subsequent recursive calls inside the function. The following lemma bounds the total number of bad edges E^{ℓ}_B in E_ℓ.

▶ **Lemma 3.** $|E^{\ell}_B| \leq \varepsilon^3|M|$.

Proof. We write a recurrence relation to find the total number of bad edges E^{ℓ}_B in E_ℓ.

$$
\begin{aligned}
&\textsc{BadEdges}(\ell) = |E^{\ell}_{B_1}| + \textsc{BadEdges}(\ell-2) \\
&\quad\implies |E^{\ell}_B| \leq 2\varepsilon^4|M| + \textsc{BadEdges}(\ell-2) \qquad \ldots \text{by Lemma 2} \\
&\quad\implies |E^{\ell}_B| \leq 2\varepsilon^4|M| + 2\varepsilon^4|M| + \cdots + 2\varepsilon^4|M| \qquad \ldots \text{at most } (1/(4\varepsilon)) \text{ times.} \\
&\quad\implies |E^{\ell}_B| \leq \varepsilon^3|M|.
\end{aligned}
$$

The function $\textsc{BadEdges}(\ell')$ gives the total number of bad edges in E_ℓ from one function call $\textsc{Augment}(\ell', V_1, M')$. ◀

Now, we give a bound on the total number of augmentations during one function call $\textsc{Augment}(\ell, V, M)$. We say that an augmenting path is "good" if none of the edges in that augmenting path belong to E^{ℓ}_B.

▶ **Lemma 4.** *One function call* $\textsc{Augment}(\ell, V, M)$ *augments at least* $\varepsilon^{4\ell}/2$ *fraction of the total number of good* $(2\ell+1)$*-augmenting paths in* M*.*

Proof. Consider a length $(2\ell+1)$-augmentation of an augmenting path P (for instance, $P := au_1v_1u_2v_2u_3 \ldots v_{\ell-1}u_\ell v_\ell b$). There are at most $(1/\varepsilon^4)^\ell$ length $(2\ell+1)$-augmenting paths (considering edges in M and all edges in S during the recursive calls) with endpoint a (, and at most $(1/\varepsilon^4)^\ell$ length $(2\ell+1)$-augmenting paths with endpoint b). This is because, on any of the free vertices, at most $1/\varepsilon^4$ edges are stored in S during the function call $\textsc{Augment}(\ell, V, M)$, and the subsequent recursive calls. (See Figure 1 for instance.)

After the length $(2\ell+1)$-augmentation of P, none of the other $2/(\varepsilon^4)^\ell - 1$ length $(2\ell+1)$-augmenting paths can be augmented. (Figure 2 gives an illustration.) Thus, the total number of augmentations during one call $\textsc{Augment}(\ell, V, M)$ is at least $\frac{1}{2/(\varepsilon^4)^\ell}$ fraction of the total number of good $(2\ell+1)$-augmenting paths in M. ◀

▶ **Lemma 5.** *After* $\left(1/\varepsilon^{5\ell}\right)$ *function calls* $\textsc{Augment}(\ell, V, M)$*, there are at most* $\varepsilon^2|M|$ *edges remaining in* M *that are* $(2\ell+1)$*-augmentable.*

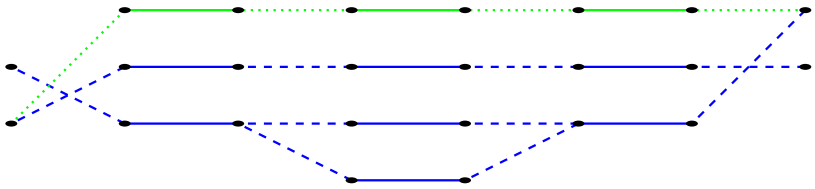

Figure 2 Suppose $\ell = 3$. Green solid lines and dotted lines represent a path P that was augmented, and blue solid lines and dashed lines represent the paths that can no longer be augmented due to that augmentation of path P.

Proof. Lemma 4 shows that, if there are k good $(2\ell+1)$-augmenting paths in M with respect to some M^*, then in each call $\textsc{Augment}(\ell, V, M)$, at least $\varepsilon^{4\ell}$ fraction of these good paths are augmented (ignoring the 1/2 factor). So, after $\log(1/\varepsilon^{4\ell})/\varepsilon^{4\ell}$ passes, the number of good $(2\ell+1)$-augmenting paths remaining in M are at most

$$k(1-\varepsilon^{4\ell})^{(\log(1/\varepsilon^{4\ell})/\varepsilon^{4\ell})} \leq \varepsilon^{4\ell} \cdot k \leq \varepsilon^{4\ell} \cdot |M| \leq \varepsilon^3 |M|.$$

Suppose that all the bad edges E_B^ℓ from the last function call $\textsc{Augment}(\ell, V, M)$ belong to distinct length $(2\ell+1)$-augmenting paths. Then, by Lemma 3, and by the bound on the number of good length $(2\ell+1)$-augmenting paths remaining, at most $2\varepsilon^3|M|$ length $(2\ell+1)$-augmenting paths remain in M.

Since we only consider augmenting paths of length at most $(1/\varepsilon)$, and each augmenting path of length $(1/\varepsilon)$ contains at most $(1/(2\varepsilon))$ edges in M, the total number of $(2\ell+1)$-augmentable edges remaining in M is at most $\varepsilon^2|M|$. ◀

As mentioned earlier, the function call $\textsc{Augment}(\ell, V, M)$ assumes that there are no shorter than length $(2\ell+1)$-augmenting paths. But the analysis does not require this assumption. This assumption is used in the proof of Lemma 2. But Lemma 2 anyways gives an overestimate on the upper bound for $\ell > 1$. Also, during the function call $\textsc{Augment}(\ell, V, M)$, if the algorithm comes across a shorter augmenting path, it is ignored. Thus, using Lemmas 1 and 5, we claim the following result.

▶ **Theorem 6.** *Algorithm 1 is a $(1/\varepsilon)^{O(1/\varepsilon)}$-pass $(1+\varepsilon)$-approximation deterministic algorithm for maximum matching on bipartite graphs.*

Note that during the function call $\textsc{Augment}(\ell, V, M)$, at most one edge is stored in S from a matched vertex to a free vertex. So, there are at most $2|M|$ edges stored in the semi-matchings at any stage. Thus, the algorithm uses $O(n \log n)$ space.

4 Algorithm on General Graphs

In this section, we present a $(1/\varepsilon)^{O(1/\varepsilon)}$-pass $(1+\varepsilon)$-approximation deterministic algorithm for maximum matching on general graphs. The high level idea for the algorithm on general graphs is same as the algorithm on bipartite graphs.

As already mentioned earlier, there do not exist "blossoms" in the bipartite graphs, and the existence of blossoms in the general graphs, makes it difficult to find augmenting paths. We address this issue in the following manner.

While finding the semi-matchings, instead of storing at most one edge on any matched vertex, the algorithm stores at most two edges, and the second edge is chosen carefully. The algorithm needs access to all prior semi-matchings it has stored during the function call

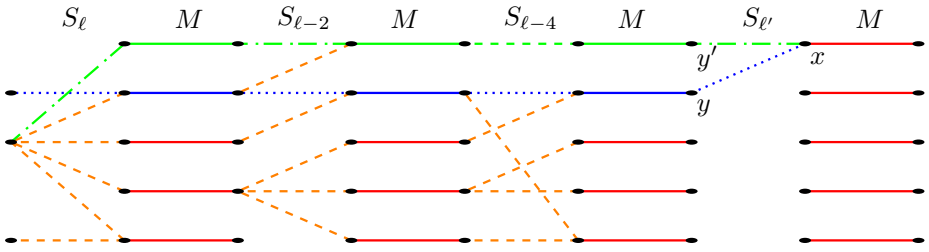

Figure 3 Blue solid and dotted lines represent the path p_{xy}, and green solid and dash-dot lines represent the path $p_{xy'}$. As these paths do not intersect in any vertex other than x, xy is added to $S_{\ell'}$.

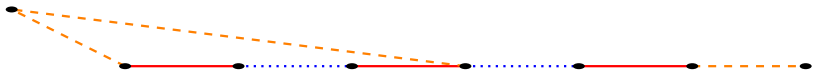

Figure 4 Consider the iteration to find length 7-augmentations in general graphs. All the edges represented by the dashed orange lines will be added to the semi-matching S_3, and in the subsequent iteration if we only consider those edges in M on which there are no edges stored in S_3 (there are none), then we wont be able to find the length 7-augmenting path. So we need to consider all the edges in M in the subsequent iterations. But then $V_1 \cap V(M) \neq \emptyset$.

AUGMENT(ℓ, V, M) to add the second edge. So, rather than using recursive calls inside the function call AUGMENT (ℓ, V, M), we iteratively find semi-matchings. Let $S_{\ell'}$ denote the semi-matching stored during the iteration to find the middle length $(2\ell'+1)$ path of any length $(2\ell+1)$-augmenting path. In this iteration, when the edge xy is read during a pass, such that $x \in V(M)$ and $y \in V_1$, suppose $\exists xy' \in S_{\ell'}$. Let P_{xy} (or $P_{xy'}$) denote the set of paths starting with xy (or xy'), alternating between some edge in M and some edge in $S_{\ell'+2i}$, for i going from 1 to $(\ell-\ell')/2$, ending in some edge in S_ℓ. If $\exists p_{xy} \in P_{xy}$, and $\exists p_{xy'} \in P_{xy'}$, such that p_{xy} and $p_{xy'}$ do not intersect in any vertex except x, then we add xy to $S_{\ell'}$. This ensures that when the middle length $(2\ell'+1)$ path of any length $(2\ell+1)$-augmenting path is found, there exist two non-intersecting paths of length $(\ell-\ell')$, such that each path contains one edge each from $M, S_{\ell'+2}, M, S_{\ell'+4}, \ldots, M, S_\ell$ (in that sequence), and they form length $(2\ell+1)$-augmenting path along with the middle length $(2\ell'+1)$ path. See Figure 3 for an illustration of the process of adding a second edge, incident on vertex x, to $S_{\ell'}$.

For general graphs, unlike bipartite graphs, we require that $V(M)$ and V_1 are not disjoint (see Figure 4 for a reason).

So, the algorithm needs to store directed semi-matchings instead of semi-matchings. Because $V(M)$ and V_1 are not disjoint, while adding an edge xy to any directed semi-matching, it is ensured that there exists a directed path starting with xy containing alternate edges in M and the directed semi-matchings previously stored, such that it does not visit x, thus avoiding formation of a cycle. (See Figures 5 and 6 for an illustration.)

Algorithm 2 gives a formal description.

Bad Edges

In the function call AUGMENT(ℓ, V, M), suppose we are in an iteration to find the middle length $(2\ell'+1)$ path of the length $(2\ell+1)$-augmenting paths in the matching matching M. Let $au_1v_1u_2v_2u_3 \ldots v_{\ell-1}u_\ell v_\ell b$ denote any length $(2\ell+1)$-augmenting path in M with respect to M^*. Then, $v_{(\ell-\ell')/2}u_{(\ell-\ell')/2+1}v_{(\ell-\ell')/2+1} \cdots\ u_{(\ell+\ell')/2}v_{(\ell+\ell')/2}\ u_{(\ell+\ell')/2+1}$ denotes

Figure 5 Red solid lines represent edges in M. Edge xy is not added to a Directed Semi-Matching because there does not exist a directed path starting from xy and not visiting x. The other orange dotted lines represent edges stored in Directed Semi-Matchings in previous iterations.

Figure 6 Red solid lines represent edges in M. Edge xy is added to a Directed Semi-Matching because there exists a directed path starting from xy and not visiting x. The other orange dotted lines represent edges stored in Directed Semi-Matchings in previous iterations.

the middle length $(2\ell' + 1)$ path of the augmenting path. Edge $u_{(\ell-\ell')/2+1}v_{(\ell-\ell')/2+1} \in M$ (or $u_{(\ell+\ell')/2}v_{(\ell+\ell')/2} \in M$) is called a bad edge under one of the following two conditions. (Note that in any iteration, V_1 denotes the set of free vertices.)

1. If there is no directed edge added to $S_{\ell'}$ which is incident from the vertex $u_{(\ell-\ell')/2+1}$ (or $v_{(\ell+\ell')/2}$) to some vertex in V_1, because there already were $(1/\varepsilon^4)$ incoming edges in $S_{\ell'}$ incident on the vertices in V_1 for all the edges incident from $u_{(\ell-\ell')/2+1}$ (or $v_{(\ell+\ell')/2}$).
2. If there is only one directed edge (not in M^*) added to $S_{\ell'}$ that is incident from the vertex $u_{(\ell-\ell')/2+1}$ (or $v_{(\ell+\ell')/2}$) to some vertex in V_1, and a second directed edge is not added to $S_{\ell'}$, because there already were $(1/\varepsilon^4)$ incoming edges in $S_{\ell'}$ incident on the vertices in V_1 for all other edges incident from $u_{(\ell-\ell')/2+1}$ (or $v_{(\ell+\ell')/2}$).

Note. A directed edge which is incident from the vertex $u_{(\ell-\ell')/2+1}$ (or $v_{(\ell+\ell')/2}$) to some vertex in V_1 may not be added to $S_{\ell'}$ also for one of the following two other reasons. First, if the addition of such an edge is going to form a cycle, and second, if the addition of such an edge does not produce two non-intersecting paths from the vertex $u_{(\ell-\ell')/2+1}$ to the unmatched vertices. We argue that the edge $u_{(\ell-\ell')/2+1}v_{(\ell-\ell')/2+1} \in M$ (or $u_{(\ell+\ell')/2}v_{(\ell+\ell')/2} \in M$) is not bad due to either of the above mentioned reasons.

In the first case, such an edge can be ignored, as there already exists a shorter directed path from the vertex $u_{(\ell-\ell')/2+1}$ to some unmatched vertex. The second case needs careful attention. Let $u_{(\ell-\ell')/2+1}v_{(\ell-\ell')/2} \in M^*$ be the edge not added to $S_{\ell'}$, and let $u_{(\ell-\ell')/2+1}v$ be the only edge that is added to $S_{\ell'}$. Either $v_{(\ell-\ell')/2}u_{(\ell-\ell')/2} \in M^*$ is a bad edge or not. If it is a bad edge, we can ignore the edge $u_{(\ell-\ell')/2+1}v_{(\ell-\ell')/2+1} \in M$. Otherwise, there are two sub cases.

1. There are two directed edges from $u_{(\ell-\ell')/2}$ in $S_{\ell'+2}$, which means there are two non-intersecting directed paths p_1 and p_2, from $u_{(\ell-\ell')/2}$ to unmatched vertices. If there exists a directed path from v which intersects p_1 or p_2 or neither of them, then the algorithm should be able to add $u_{(\ell-\ell')/2+1}v_{(\ell-\ell')/2}$ to $S_{\ell'}$. If there exists a directed path from v which intersects both p_1 and p_2, then the edge $u_{(\ell-\ell')/2+1}v_{(\ell-\ell')/2+1}$ is not a bad edge, as it can be used in the length $(2\ell + 1)$-augmentation.
2. There is only one directed edge from $u_{(\ell-\ell')/2}$ in $S_{\ell'+2}$. Suppose there exists a directed path from $u_{(\ell-\ell')/2}$ to unmatched vertices, which does not contain any bad edges (If there does not exist such a path, then we can ignore the edge $u_{(\ell-\ell')/2+1}v_{(\ell-\ell')/2+1} \in M$.)

Algorithm 2 Deterministic Algorithm on General Graphs.

Find Maximal Matching M in the first pass.
for $\ell = 1$ `to` $1/(2\varepsilon)$ **do** ▷ To Remove almost all $(2\ell+1)$-augmenting paths
  for `Phase` $p = 1$ `to` $1/\varepsilon^{5\ell}$ **do**
    $M \leftarrow \text{AUGMENT}(\ell, V, M)$.
Output M.
function AUGMENT(ℓ, V, M) ▷ Function of length $(2\ell+1)$-augmentations.
  $\ell' := \ell$, $V_1 \leftarrow V \setminus V(M)$.
  while $\ell' \geq 0$ **do**
    $S_{\ell'} \leftarrow \emptyset$.
    for all edges ab **do** ▷ One pass to find a directed semi-matching
      for all $xy \in \{ab, ba\}$ **do** ▷ Directed edges ab and ba.
        if $x \in V(M)$, $y \in V_1$, and $\exists$ a directed path starting with xy, with one edge each from $M, S_{\ell'+2}, M, S_{\ell'+4}, \ldots, M, S_\ell$ (in that sequence) that does not visit x **then**
          if $\deg^O_{S_{\ell'}}(x) = 0$ and $\deg^I_{S_{\ell'}}(y) < 1/\varepsilon^4$ **then**
            $S_{\ell'} \leftarrow S_{\ell'} \cup \{xy\}$ ▷ Add directed edge xy.
          else if $\deg^O_{S_{\ell'}}(x) = 1$ and $\deg^I_{S_{\ell'}}(y) < 1/\varepsilon^4$ **then**
            Suppose $\exists xy' \in S_{\ell'}$.
            Let P_{xy} (or $P_{xy'}$) denote the set of directed paths starting with xy (or xy'), with one edge each from $M, S_{\ell'+2}, M, S_{\ell'+4}, \ldots, M, S_\ell$ (in that sequence).
            if $\exists p_{xy} \in P_{xy}$, and $\exists p_{xy'} \in P_{xy'}$, such that p_{xy} and $p_{xy'}$ do not intersect in any vertex except x **then**
              $S_{\ell'} \leftarrow S_{\ell'} \cup \{xy\}$ ▷ Add directed edge xy.
    $V_1 \leftarrow \emptyset$.
    for all edges $uv \in M$ **do**
      if $\exists ua \in S_{\ell'}$ **then** ▷ Directed edge ua.
        $V_1 \leftarrow V_1 \cup \{v\}$. ▷ Set of free vertices for the next iteration.
    $\ell' := \ell' - 2$ ▷ End of While Loop
  $(2\ell+1)$-augment M greedily using edges in $S_\ell, S_{\ell-2}, \ldots$.
return M.

This implies that there is a directed path from v that intersects this directed path, and hence the edge $u_{(\ell-\ell')/2+1}v_{(\ell-\ell')/2+1}$ is not a bad edge, as it can be used in the length $(2\ell+1)$-augmentation (along with the directed edge $u_{(\ell-\ell')/2+1}v \in S_{\ell'}$).

4.1 Analysis

We begin the analysis for any length $(2\ell+1)$-augmentations for the matching M. Let E_ℓ denote the set of $(2\ell+1)$-augmentable edges in M with respect to an optimal matching M^*, such that $|E_\ell|$ is maximum. Also, let $E^\ell_{B_1}$ denote the outermost bad edges of any length $(2\ell+1)$-augmenting path. The following lemma gives an upper bound on $E^\ell_{B_1}$.

▶ **Lemma 7.** $|E^\ell_{B_1}| \leq 4\varepsilon^4|M|$.

Proof. Without loss of generality, lets consider the edge u_1v_1 in a length $(2\ell+1)$-augmenting path $au_1v_1u_2v_2u_3 \ldots v_{\ell-1}u_\ell v_\ell b$. If u_1v_1 is a bad edge, then by the definition of a bad edge, there are $(1/\varepsilon^4)$ edges in S_ℓ responsible. So, $(1/\varepsilon^4)|E^\ell_{B_1}| \leq |S_\ell|$. For $\ell > 1$, each edge in M

can have at most two outgoing edges in S_ℓ incident on its endpoints (as there are no length 3-augmentable edges in M), i.e. $|S_\ell| \leq 2|M|$. Hence, $|E^\ell_{B_1}|/\varepsilon^4 \leq 2|M|$.

For $\ell = 1$, each edge in M can have at most two outgoing edges in S_ℓ incident from each its of endpoints, i.e. $|S_\ell| \leq 4|M|$. Hence, $|E^\ell_{B_1}|/\varepsilon^4 \leq 4|M|$. ◀

Let E^ℓ_B denote the set of the bad edges, formed during the iterations inside the function call $\textsc{Augment}(\ell, V, M)$, in all the length $(2\ell+1)$-augmenting paths. The following lemma bounds the total number of bad edges E^ℓ_B in E_ℓ.

▶ **Lemma 8.** $|E^\ell_B| \leq \varepsilon^3|M|$.

Proof. We write a recurrence relation to find the total number of bad edges E^ℓ_B in E_ℓ.

$$\begin{aligned}
&\textsc{BadEdges}(\ell) = |E^\ell_{B_1}| + \textsc{BadEdges}(\ell-2)\\
&\quad\implies |E^\ell_B| \leq 2\varepsilon^4|M| + \textsc{BadEdges}(\ell-2) \qquad \dots \text{by Lemma 7}\\
&\quad\implies |E^\ell_B| \leq 2\varepsilon^4|M| + 2\varepsilon^4|M| + \dots + 2\varepsilon^4|M| \qquad \dots \text{at most } (1/(4\varepsilon)) \text{ times.}\\
&\quad\implies |E^\ell_B| \leq \varepsilon^3|M|.
\end{aligned}$$

The function $\textsc{BadEdges}(\ell')$ gives the total number of bad edges in E_ℓ formed during the iteration to find middle length $(2\ell'+1)$ path of any length $(2\ell+1)$-augmenting path. ◀

Now, we give a bound on the total number of augmentations during one function call $\textsc{Augment}(\ell, V, M)$. We say that an augmenting path is "good" if none of the edges in that augmenting path belong to E^ℓ_B.

▶ **Lemma 9.** *One function call* $\textsc{Augment}(\ell, V, M)$ *augments at least* $\varepsilon^{4\ell}/2$ *fraction of the total number of good* $(2\ell+1)$*-augmenting paths in* M*.*

Proof. Consider a length $(2\ell+1)$-augmentation of an augmenting path P (for instance, $P := au_1v_1u_2v_2u_3 \dots v_{\ell-1}u_\ell v_\ell b$). There are at most $(1/\varepsilon^4)^\ell$ length $(2\ell+1)$-augmenting paths (considering edges in $M, S_\ell, S_{\ell-2}, \dots$) with the endpoint a (, and at most $(1/\varepsilon^4)^\ell$ length $(2\ell+1)$-augmenting paths with the endpoint b). This is because, on any of the free vertices, at most $1/\varepsilon^4$ edges are stored in any S_i during the function call $\textsc{Augment}(\ell, V, M)$.

After the length $(2\ell+1)$-augmentation of P, none of the other $2/(\varepsilon^4)^\ell - 1$ length $(2\ell+1)$-augmenting paths can be augmented. Thus, the total number of augmentations during one call $\textsc{Augment}(\ell, V, M)$ is at least $\frac{1}{2/(\varepsilon^4)^\ell}$ fraction of the total number of good $(2\ell+1)$-augmenting paths in M. ◀

▶ **Lemma 10.** *After* $(1/\varepsilon^{5\ell})$ *function calls* $\textsc{Augment}(\ell, V, M)$*, the number of* $(2\ell+1)$*-augmentable edges remaining in* M *is at most* $\varepsilon^2|M|$*.*

Proof. Lemma 9 shows that, if there are k good $(2\ell+1)$-augmenting paths in M with respect to some M^*, then in each call $\textsc{Augment}(\ell, V, M)$, at least $\varepsilon^{4\ell}$ fraction of these good paths are augmented (ignoring the 1/2 factor). So, after $\log(1/\varepsilon^{4\ell})/\varepsilon^{4\ell}$ passes, the number of good $(2\ell+1)$-augmenting paths remaining in M are at most

$$k(1-\varepsilon^{4\ell})^{(\log(1/\varepsilon^{4\ell})/\varepsilon^{4\ell})} \leq \varepsilon^{4\ell} \cdot k \leq \varepsilon^{4\ell} \cdot |M| \leq \varepsilon^3|M|.$$

Suppose all the bad edges E^ℓ_B from the last function call $\textsc{Augment}(\ell, V, M)$ belong to distinct length $(2\ell+1)$-augmenting paths. Then, by Lemma 8, and by the bound on the number of good length $(2\ell+1)$-augmenting paths remaining, at most $2\varepsilon^3|M|$ length $(2\ell+1)$-augmenting paths remain in M.

Since we only consider augmenting paths of length at most $(1/\varepsilon)$, and each augmenting path of length $(1/\varepsilon)$ contains at most $(1/(2\varepsilon))$ edges in M, the total number of $(2\ell+1)$-augmentable edges remaining in M is at most $\varepsilon^2|M|$. ◀

As mentioned earlier, the function call $\textsc{Augment}(\ell, V, M)$ assumes that there are no shorter than length $(2\ell+1)$-augmenting paths. But the analysis does not require this assumption. This assumption is used in the proof of Lemma 7. But Lemma 7 anyways gives an overestimate on the upper bound for $\ell > 1$. Also, during the function call $\textsc{Augment}(\ell, V, M)$, if the algorithm comes across a shorter augmenting path, it is ignored. Thus, using Lemmas 1 and 10, we claim the following result.

▶ **Theorem 11.** *Algorithm 2 is a $(1/\varepsilon)^{O(1/\varepsilon)}$-pass $(1+\varepsilon)$-approximation deterministic algorithm for maximum matching on general graphs.*

Note that during the function call $\textsc{Augment}(\ell, V, M)$, at most two edges are stored in any S_ℓ from a matched vertex to free vertices. So, there are at most $2\ell|M|$ edges stored in the directed semi-matchings at any stage. Thus, the algorithm uses $O\left(\frac{1}{\varepsilon} \cdot n \log n\right) = O(n \log n)$ space.

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. 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. 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. Aaron Bernstein and Cliff Stein. Faster Fully Dynamic Matchings with Small Approximation Ratios. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 692–711, 2016. doi:10.1137/1.9781611974331.ch50.
5. Sayan Bhattacharya, Monika Henzinger, and Giuseppe F. Italiano. Deterministic Fully Dynamic Data Structures for Vertex Cover and Matching. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 785–804, 2015. doi:10.1137/1.9781611973730.54.
6. Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai. Fully Dynamic Approximate Maximum Matching and Minimum Vertex Cover in $O(\log^3 n)$ Worst Case Update Time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 470–489, 2017. doi:10.1137/1.9781611974782.30.
7. Niv Buchbinder, Danny Segev, and Yevgeny Tkach. Online Algorithms for Maximum Cardinality Matching with Edge Arrivals. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, pages 22:1–22:14, 2017. doi:10.4230/LIPIcs.ESA.2017.22.
8. 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. doi:10.1007/978-3-662-48350-3_23.

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

10 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. doi:10.1007/978-3-662-47672-7_26.

11 Ashish Chiplunkar, Sumedh Tirodkar, and Sundar Vishwanathan. On Randomized Algorithms for Matching in the Online Preemptive Model. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 325–336, 2015. doi:10.1007/978-3-662-48350-3_28.

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 *Proc. 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1326–1344, 2016. URL: http://dl.acm.org/citation.cfm?id=2884435.2884527.

13 Sebastian Eggert, Lasse Kliemann, Peter Munstermann, and Anand Srivastav. Bipartite Matching in the Semi-streaming Model. *Algorithmica*, 63(1):490–508, 2012. doi:10.1007/s00453-011-9556-8.

14 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. doi:10.1137/100801901.

15 Leah Epstein, Asaf Levin, Danny Segev, and Oren Weimann. Improved Bounds for Online Preemptive Matching. In *30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013, Kiel, Germany*, pages 389–399, 2013. doi:10.4230/LIPIcs.STACS.2013.389.

16 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, December 2016. doi:10.1109/ICDMW.2016.0092.

17 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.

18 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. doi:10.1016/j.tcs.2005.09.013.

19 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.

20 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*, pages 15:1–15:21, 2017. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.15.

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

22 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.

23 R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An Optimal Algorithm for On-line Bipartite Matching. In *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, STOC '90, pages 352–358, New York, NY, USA, 1990. ACM.

24 Christian Konrad. Maximum Matching in Turnstile Streams. In *Proc. 23rd Annual European Symposium on Algorithms*, pages 840–852, 2015. doi:10.1007/978-3-662-48350-3_70.

25 Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum Matching in Semi-Streaming with Few Passes. *CoRR*, abs/1112.0184, 2014. URL: http://arxiv.org/abs/1112.0184.

26 Mohammad Mahdian and Qiqi Yan. Online Bipartite Matching with Random Arrivals: An Approach Based on Strongly Factor-revealing LPs. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 597–606, New York, NY, USA, 2011. ACM. doi:10.1145/1993636.1993716.

27 Andrew McGregor. Problem 60: Single-Pass Unweighted Matchings. http://sublinear.info/index.php?title=Open_Problems:60. Accessed: 2018-02-10.

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

29 Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|V|}|E|)$ Algorithm for Finding Maximum Matching in General Graphs. In *Proceedings of the 21st Annual Symposium on Foundations of Computer Science*, SFCS '80, pages 17–27, Washington, DC, USA, 1980. IEEE Computer Society. doi:10.1109/SFCS.1980.12.

30 Ami Paz and Gregory Schwartzman. A $(2+\epsilon)$-Approximation for Maximum Weight Matching in the Semi-Streaming Model. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2153–2161, 2017. doi:10.1137/1.9781611974782.140.

31 A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.

32 Shay Solomon. Fully Dynamic Maximal Matching in Constant Update Time. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 325–334, 2016. doi:10.1109/FOCS.2016.43.

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