Algorithmica (2019) 81:4275–4292
https://doi.org/10.1007/s00453-019-00593-6

Maximum Matching on Trees in the Online Preemptive and the Incremental Graph Models

Sumedh Tirodkar[1] · Sundar Vishwanathan[2]

Received: 28 September 2017 / Accepted: 28 May 2019 / Published online: 11 June 2019

Abstract

We study the Maximum Cardinality Matching (MCM) and the Maximum Weight Matching (MWM) problems, on trees and on some special classes of graphs, in the online preemptive and the incremental graph models. In the *Online Preemptive* model, the edges of a graph are revealed one by one and the algorithm is required to always maintain a valid matching. On seeing an edge, the algorithm has to either accept or reject the edge. If accepted, then the adjacent edges are discarded, and all rejections are permanent. In this model, the complexity of the problems is settled for deterministic algorithms (McGregor, in: Proceedings of the 8th international workshop on approximation, randomization and combinatorial optimization problems, and proceedings of the 9th international conference on randomization and computation: algorithms and techniques, APPROX'05/RANDOM'05, Springer, Berlin, pp. 170–181, 2005; Varadaraja, in: Automata, languages and programming: 38th international colloquium, ICALP 2011, Zurich, Switzerland, proceedings, part I, pp. 379–390, 2011. https://doi.org/10.1007/978-3-642-22006-7_32). Epstein et al. (in: 30th international symposium on theoretical aspects of computer science, STACS 2013, Kiel, Germany, pp. 389–399, 2013. https://doi.org/10.4230/LIPIcs.STACS.2013.389) gave a 5.356-competitive randomized algorithm for MWM, and also proved a lower bound on the competitive ratio of $(1+\ln 2)\approx 1.693$ for MCM. The same lower bound applies for MWM. In the *Incremental Graph* model, at each step an edge is added to the graph, and the algorithm is supposed to quickly update its current matching. Gupta (in: 34th international conference on foundation of software technology and theoretical computer science, FSTTCS 2014, 15–17 Dec 2014, New Delhi, India, pp. 227–239, 2014. https://doi.org/10.4230/LIPIcs.FSTTCS.2014.227) proved that for any $\epsilon \le 1/2$, there exists an algorithm that maintains a $(1+\epsilon)$-approximate MCM for an incremental bipartite graph in an amortized update time of $O\left(\frac{\log^2 n}{\epsilon^4}\right)$. No $(2-\epsilon)$-approximation algorithm with a worst case update time of $O(1)$ is known in this model, even for special classes of graphs. In this paper we show that some of the results can be improved for trees, and

✉ Sumedh Tirodkar
sumedhtirodkar@gmail.com

Extended author information available on the last page of the article

for some special classes of graphs. In the online preemptive model, we present a $64/33$-competitive randomized algorithm (which uses only two bits of randomness) for MCM on trees. Inspired by the above mentioned algorithm for MCM, we present the main result of the paper, a randomized algorithm for MCM with a worst case update time of $O(1)$, in the incremental graph model, which is $3/2$-approximate (in expectation) on trees, and 1.8-approximate (in expectation) on general graphs with maximum degree 3. Note that this algorithm works only against an oblivious adversary. We derandomize this algorithm, and give a $(3/2 + \epsilon)$-approximate deterministic algorithm for MCM on trees, with an amortized update time of $O(1/\epsilon)$. We also present a minor result for MWM in the online preemptive model, a 3-competitive randomized algorithm (that uses only $O(1)$ bits of randomness) on growing trees (where the input revealed upto any stage is always a tree, i.e. a new edge never connects two disconnected trees).

Keywords Online preemptive model · Incremental dynamic graph model · Primal-dual analysis

1 Introduction

The *Maximum (Cardinality/Weight) Matching* problem is one of the most extensively studied problems in Combinatorial Optimization. See Schrijver's book [13] and references therein for a comprehensive overview of classic work. A *matching* $M \subseteq E$ is a set of disjoint edges. Traditionally the problem was studied in the offline setting where the entire input is available to the algorithm beforehand. But over the last few decades it has been extensively studied in various other models where the input is revealed in pieces, like the vertex arrival model (adversarial and random), the edge arrival model (adversarial and random), streaming and semi-streaming models, the online preemptive model, etc. [4–6,9–11]. In this paper, we study the Maximum Cardinality Matching (MCM) and the Maximum Weight Matching (MWM) problems, on trees and on some special classes of graphs, in the *Online Preemptive* model, and in the *Incremental Graph* model. (Refer to Sect. 1.2 for a comparison between the two models.)

In the online preemptive model, the edges arrive online in an arbitrary order, and the algorithm is supposed to accept or reject an edge on arrival. If accepted, the algorithm can reject it later, and all rejections are permanent. The algorithm is supposed to always maintain a valid matching. There is a 5.828-competitive deterministic algorithm due to McGregor [11] for MWM, and a tight lower bound for deterministic algorithms due to Varadaraja [15]. Epstein et al. [5] gave a 5.356-competitive randomized algorithm for MWM, and also proved a 1.693 lower bound on the competitive ratio achievable by any randomized algorithm for MCM. No better lower bound is known for MWM.

In [3], the authors gave the first randomized algorithm with competitive ratio $(28/15)$ less than 2 for MCM in the online preemptive model on growing trees (defined in Sect. 1.1). In Sect. 2, we extend their algorithm to give a $64/33$-competitive randomized (which uses only two bits of randomness) algorithm for MCM on trees. Although the algorithm is an extension of the one for growing trees in [3], it motivates the algorithm (described in Sect. 3) for MCM in the incremental graph model.

Note that the adversary presenting the edges in the online preemptive model is oblivious, and does not have access to the random choices made by the algorithm.

In recent years, algorithms for approximate MCM in dynamic graphs have been the focus of many studies due to their wide range of applications. Here [1,2,8,14] is a non-exhaustive list some of the studies. The objective of these dynamic graph algorithms is to efficiently process an online sequence of update operations, such as edge insertions and deletions. The algorithm has to quickly maintain an approximate maximum matching despite an adversarial order of edge insertions and deletions. Dynamic graph problems are usually classified according to the types of updates allowed: incremental models allow only insertions, decremental models allow only deletions, and fully dynamic models allow both. We study MCM in the incremental model. Gupta [7] proved that for any $\epsilon \le 1/2$, there exists an algorithm that maintains a $(1+\epsilon)$-approximate MCM on bipartite graphs in the incremental model in an amortized update time of $O\left(\frac{\log^2 n}{\epsilon^4}\right)$. We present a randomized algorithm for MCM in the incremental model with a worst case update time of $O(1)$, which is 3/2-approximate (in expectation) on trees, and 1.8-approximate (in expectation) on general graphs with maximum degree 3. This algorithm works only against an oblivious adversary. We derandomize this algorithm, and give a $(3/2+\epsilon)$-approximate deterministic algorithm for MCM on trees, with an amortized update time of $O(1/\epsilon)$. Note that the algorithm of Gupta [7] is based on multiplicative weights update method (the method assigns initial weights to the dual variables, and updates these weights multiplicatively and iteratively), and it seems unlikely that a better running time analysis for special classes of graphs is possible.

We present a minor result in Sect. 4, a 3-competitive randomized algorithm (which uses only $O(1)$ bits of randomness) for MWM on growing trees in the online preemptive model. Although growing trees are a very restricted class of graphs, there are a couple of reasons to study the performance of the algorithm on this class of input. Firstly, almost all lower bounds, including the one due to Varadaraja [15] for MWM are on growing trees. Secondly, even for this restricted class, the analysis is involved. We use the primal-dual technique for analyzing the performance of this algorithm, and show that this analysis is indeed tight by giving an example, for which the algorithm achieves the competitive ratio 3. We describe the algorithm for general graphs, but are only able to analyze it for growing trees, and new ideas are needed to prove a better bound for general graphs.

1.1 Preliminaries

We use primal-dual techniques to analyze the performance of all the randomized algorithms described in this paper. Here are the well known Primal and Dual formulations of the matching problem.

Primal LP	Dual LP
$\max \sum_e w_e x_e$	$\min \sum_v y_v$
$\forall v : \sum_{v \in e} x_e \le 1$	$\forall e \equiv (u,v) : y_u + y_v \ge w_e$
$x_e \ge 0$	$y_v \ge 0$

For MCM, $w_e = 1$ for any edge. Any matching M implicitly defines a feasible primal solution. If an edge $e \in M$, then $x_e = 1$, otherwise $x_e = 0$.

Suppose an algorithm outputs a matching M, then let P be the corresponding primal feasible solution. Let D denote some feasible dual solution. The following claim can be easily proven using weak duality.

Claim 1 *If $D \leq \alpha \cdot P$, then the algorithm is α-competitive.*

If M is any matching, then for an edge e, $X(M, e)$ denote the set of edges in M that conflict with e. We will say that a vertex (resp. an edge) is *covered* by a matching M if there is an edge in M which is incident with (resp. adjacent to) the vertex (resp. edge). We also say that an edge is *covered* by a matching M if it belongs to M.

In the online preemptive model, growing trees are trees where a new edge has exactly one vertex common with already revealed edges.

1.2 Online Preemptive Model Versus Incremental Graph Model

There are two main differences between these models. Firstly, in the online preemptive model, once an edge is rejected/removed from the matching maintained by the algorithm, it cannot be added into its matching, whereas in the incremental graph model, rejected/removed edges can be added to the matching later on. Secondly, there is no restriction on how much time an algorithm in the online preemptive model can use to process a revealed edge, whereas in the incremental graph model, the algorithm is supposed to process the revealed edge fast. The term "fast" is used loosely, and is specific to any problem. For example, an MCM on general graphs can be found in time $O(m\sqrt{n})$ when the entire input is available [12]. But for dynamic graphs, every time an edge is inserted, the algorithm is expected to maintain a matching, approximate if not exact, in time smaller than the time required by the optimal offline algorithm for MCM (say, for instance, in $O(\text{polylog}\, n)$ amortized time).

2 MCM in the Online Preemptive Model

In this section, we present a randomized algorithm (that uses only 2 bits of randomness) for MCM on trees in the online preemptive model.

The algorithm maintains four matchings M_1, M_2, M_3, M_4, and it tries to ensure that a large number of input edges are covered by some or other matchings. (Here, the term "large number" is used vaguely. Suppose more than four edges are incident with a vertex, then at most four of them will belong to matchings, one to each.) One of the four matchings is output uniformly at random. A more formal description of the algorithm follows.

Algorithm 1 Randomized Algorithm for MCM on Trees

```
1. Pick $l \in \{1, 2, 3, 4\}$ uniformly at random.
2. The algorithm maintains four matchings: $M_1$, $M_2$, $M_3$, and $M_4$.
3. On arrival of an edge $e$, the processing happens in two phases.
    (a) The Augment phase. The new edge $e$ is added to each $M_i$ in which there are no edges adjacent
        to $e$.
    (b) The Switching phase. For $i = 2, 3, 4$, in order, $M_i \leftarrow M_i \backslash X(M_i, e) \cup \{e\}$, provided this decreases
        the quantity $\sum_{j \in [4], i \neq j, X(M_i, e) \subseteq X(M_j, e)} |M_i \cap M_j|$.
4. Output $M_l$.
```

Although l is picked randomly at the beginning of the algorithm, this concrete value is not known to the adversary (as we assume that the adversary is oblivious, and cannot look at the random choices made by the algorithm).

Note that in the Switching phase, the expected size of the matching stored by the algorithm might decrease. For example, consider two disjoint edges e_1 and e_2 that have been revealed. Each of them will belong to all four matchings. So the expected size of the matching stored by the algorithm is 2. Now, if an edge e is revealed that intersects both e_1 and e_2, then e will be added to M_2 and M_3. The expected size of the matching is now 1.5. The important thing to notice here is that the decrease is not too much, and we are able to prove that the competitive ratio of the algorithm still remains below 2.

We begin with the following observations.

1. After an edge is revealed, its endpoints are covered by all four matchings.
2. If an edge e does not belong to any matching, then there exists four edges e_1, e_2, e_3, e_4 that intersect with e, such that $\forall i \in [4]$, e_i is contained in M_i. And also, there does not exist a pair of edges among these edges, which belong to the same matching. Otherwise, in the Switching phase, the edge e would be added to some matching.
3. Every edge is covered by at least three matchings.

At any stage, after a revealed edge has been processed, all the edges that have been revealed until then, can be classified in to one of the following three types. (Note that this classification can change over the course of input.) An edge is called *internal* if there are edges incident with both its endpoints which belong to some matching (not necessarily the same). An edge is called a *leaf edge* either if one of its endpoints is a leaf or if all the edges incident with one of its endpoints do not belong to any matching. An edge is called *bad* if its endpoints are covered by only three matchings (counting multiplicities).

We begin by proving some properties about the algorithm. The key structural lemma that keeps "influences" of bad edges local is given below.

Lemma 2 *At most five consecutive vertices on a path can have bad edges incident with them.*

According to Lemma 2, there can be at most four consecutive internal bad edges or at most five bad leaf edges incident with five consecutive vertices of a path. Lemma 2 is proved in "Appendix".

Once all edges have been seen, we distribute the primal charge among the dual variables, and use the primal-dual framework to infer the competitive ratio. If the endpoints of every edge are covered with four matchings, then the distribution of dual charge is easy. However we do have bad edges, and would like the edges in matchings to contribute more to the endpoints of these edges. Then, the charge on the other endpoint would be less and we need to balance this through other edges. We present the details as follows.

Lemma 3 *There exists an assignment of the primal charge to the dual variables such that the dual constraint for each edge* $e \equiv (u, v)$ *is satisfied at least* $\frac{33}{64}$ *in expectation, i.e.* $\mathbb{E}[y_u + y_v] \geq \frac{33}{64}$.

Proof Root the tree at an arbitrary vertex. For any edge $e \equiv (u, v)$, let v be the parent vertex, and u be the child vertex. The dual variable assignment is done after the entire input is seen, as follows.

Dual variable management An edge e will distribute its primal charge between its endpoints. The exact values are discussed below. In general, we look to transfer all of the primal charge to the parent vertex. But this does not work and we need a finer strategy. This is detailed below.

- If e does not belong to any matching, then it does not contribute to the values of dual variables.
- If e belongs to a single matching then, depending on the situation, one of 0, ϵ, 2ϵ, 3ϵ, 4ϵ, or 5ϵ of its primal charge will be assigned to u and the rest will be assigned to v.
- If e belongs to two matchings, then at most 6ϵ of its primal charge will be assigned to u as required. The rest is assigned to v.
- If e belongs to three or four matchings, then its entire primal charge is assigned to v.

We will show that $y_u + y_v \geq 2 + \epsilon$ for such an edge, when summed over all four matchings. The value of ϵ is chosen later.

For the sake of analysis, if there are bad leaf edges incident with both the endpoints of an internal edge, then we treat it like a bad internal edge. We need to do this because a bad leaf edge might need to transfer its entire primal charge to the vertex on which there are edges which do not belong to any matching (refer to Case 3 below). Note that the endpoints of the internal edge would still be covered by three matchings, even if we assume that the bad leaf edges do not exist on its endpoints. The analysis breaks up into eight cases.

Case 1 Suppose e does not belong to any matching. There must be a total of at least 4 edges incident with u and v besides e, each belonging to a distinct matching. Of these 4, at least a total of 3, say e_1, e_2, and e_3, must be from children of u and v, to u and v respectively. The edges e_1, e_2, and e_3, each assign a charge of at least $1 - 5\epsilon$ to y_u and y_v, respectively. Therefore, $y_u + y_v \geq 3 - 15\epsilon \geq 2 + \epsilon$.

Case 2 Suppose e is a bad leaf edge that belongs to a single matching, and internal edges are incident with v. By Observation 3, there must exist an edge e_1 from a child vertex of v to v, which belongs to a single matching, and another edge e_2, also belonging to a single matching from v to its parent vertex. The edge e assigns a charge of 1 to y_v. If e_1 assigns a charge of 1 or $1-\epsilon$ or $1-2\epsilon$ or $1-3\epsilon$ or $1-4\epsilon$ to y_v, then e_2 assigns ϵ or 2ϵ or 3ϵ or 4ϵ or 5ϵ respectively to y_v. In either case, $y_u + y_v = 2+\epsilon$. The key fact is that e_1 could not have assigned 5ϵ to its child vertex, as that would imply that there are more than five consecutive vertices on a path having bad edges incident with them, which is a contradiction to Lemma 2.

Case 3 Suppose e is a bad leaf edge that belongs to a single matching, and internal edges are incident with u. This implies that there are two edges e_1 and e_2 from children of u to u, each belonging to a single distinct matching. The edge e assigns a charge of 1 to y_v. Both e_1 and e_2 assign a charge of at least $1-4\epsilon$ to y_u. In either case, $y_u + y_v \geq 3 - 8\epsilon \geq 2+\epsilon$. The key fact is that neither e_1 nor e_2 could have assigned more than 4ϵ to their corresponding child vertices. Since, then by Lemma 2 (as in *Case 2*), e cannot be a bad edge.

Case 4 Suppose e is an internal bad edge. This implies (by Lemma 2) that there is an edge e_1 from a child vertex of u to u, which belongs to a single matching. Also, there is an edge e_2, from v to its parent vertex (or from a child vertex v to v), which also belongs to a single matching. The edge e assigns its remaining charge (1 or $1-\epsilon$ or $1-2\epsilon$ or $1-3\epsilon$ or $1-4\epsilon$) to y_v. If e_1 assigns a charge of 1 or $1-\epsilon$ or $1-2\epsilon$ or $1-3\epsilon$ or $1-4\epsilon$ to y_u, then e_2 assigns ϵ or 2ϵ or 3ϵ or 4ϵ or 5ϵ respectively to y_v. In either case, $y_u + y_v = 2+\epsilon$. The key fact is that e_1 could not have assigned 5ϵ to its child vertex. Since, then by Lemma 2 (as in *Case 2*), e cannot be a bad edge.

Case 5 Suppose e is not a bad edge, and it belongs to a single matching. Then either one of the follwing subcases is possible.

- There are at least two edges e_1 and e_2 from child vertices of u or v to u or v respectively.
- There is e_1 on u and e_2 on v, each belonging to a single matching.
- There is one edge e_3 from a child vertex of u or v to u or v, respectively, which belongs to two matchings.
- There is one edge e_4 from a child vertex of u or v to u or v, respectively, which belongs to single matching, and one edge e_5 from v to its parent vertex which belongs to two matchings.

In either case, $y_u + y_v \geq 3 - 10\epsilon \geq 2+\epsilon$.

Case 6 Suppose e is a bad edge that belongs to two matchings, and an internal edge is incident with u or v. This implies that there is an edge e_1, from a child vertex of u to u or from v to its parent vertex, which belongs to a single matching. The edge e assigns a charge of 2 to y_v, and the edge e_1 assigns a charge of ϵ to y_u or y_v respectively. Thus, $y_u + y_v = 2+\epsilon$.

Case 7 Suppose e is not a bad edge and it belongs to two matchings. This means that either there is an edge e_1 from a child vertex of u to u, which belongs to at least one matching, or there is an edge from child vertex of v to v that belongs to at least one matching, or there is an edge from v to its parent vertex which belongs to two matchings. The edge e assigns a charge of 2 among y_u and y_v. The neighboring edges assign a charge of ϵ to y_u or y_v (depending on which vertex it is incident with), yielding $y_u + y_v \geq 2 + \epsilon$.

Case 8 Suppose, e belongs to 3 or 4 matchings, then trivially $y_u + y_v \geq 2 + \epsilon$.

From the above cases, $y_v + y_v \geq 3 - 15\epsilon$ and $y_u + y_v \geq 2 + \epsilon$. The best value for the competitive ratio is obtained when $\epsilon = \frac{1}{16}$, yielding $\mathbb{E}[y_u + y_v] \geq \frac{33}{64}$. □

Lemma 3 immediately implies Theorem 4 using Claim 1.

Theorem 4 *Algorithm 1 is a $\frac{64}{33}$-competitive randomized algorithm for finding an MCM on trees.*

3 MCM in the Incremental Graph Model

In this section, we present our main result, a randomized algorithm (that uses only $O(1)$ bits of randomness) for MCM in the incremental graph model, which is 3/2-approximate (in expectation) on trees, and is 1.8-approximate (in expectation) on general graphs with maximum degree 3, with $O(1)$ worst case update time per edge. It is inspired by the randomized algorithm for MCM on trees described in Sect. 2. In the online preemptive model, we cannot add edges to the matching which were discarded earlier, which results in the existence of bad edges. But in the incremental graph model, there is no such restriction. For some $i \in [3]$, let $e \equiv (u, v) \in M_i$ be switched with some edge $e' \equiv (u, u')$, i.e. $M_i \leftarrow M_i \backslash \{e\} \cup \{e'\}$. If there is an edge $e'' \equiv (v, v') \in M_j$ for $i \neq j$, then we can add e'' to M_i if possible. Using this simple trick, we get a better approximation ratio in this model, and also the analysis becomes significantly simpler. Details follow.

Algorithm 2 Randomized Algorithm for MCM

```
1. Pick $l \in \{1, 2, 3\}$ uniformly at random..
2. The algorithm maintains three matchings: $M_1$, $M_2$, and $M_3$.
3. When an edge $e$ is inserted, the processing happens in two phases.
   (a) The Augment phase. The new edge $e$ is added to each $M_i$ in which there are no edges adjacent to $e$.
   (b) The Switching phase. For $i = 2, 3$, in order, $M_i \leftarrow M_i \backslash X(M_i, e) \cup \{e\}$, provided that it decreases the quantity $\sum_{j \in [3], i \neq j, X(M_i, e) \subseteq X(M_j, e)} |M_i \cap M_j|$.
       For every edge $e'$ discarded from $M_i$, add edges on the other endpoint of $e'$ in $M_j$ ($\forall j \neq i$) to $M_i$ if possible.
4. Output the matching $M_l$ on query.
```

Note that the endpoints of every edge will be covered by all three matchings, and hence all three matchings are maximal.

We again use the primal-dual technique to analyze the performance of this algorithm on trees.

Lemma 5 *There exists an assignment of the primal charge amongst the dual variables such that the dual constraint for each edge* $e \equiv (u, v)$ *is satisfied at least* $\frac{2}{3}$ *in expectation.*

Proof Root the tree at an arbitrary vertex. For any edge $e \equiv (u, v)$, let v be the parent vertex, and u be the child vertex. The dual variable assignment is done at the end of the input/on query, as follows.

- If e does not belong to any matching, then it does not contribute to the values of dual variables.
- If e belongs to a single matching, then its entire primal charge is assigned to v as $y_v = 1$.
- If e belongs to two matchings, then its entire primal charge is assigned equally amongst u and v, as $y_u = 1$ and $y_v = 1$.
- If e belongs to three matchings, then its entire primal charge is assigned to v as $y_v = 3$.

The analysis breaks up into three cases.

Case 1 Suppose e does not belong to any matching. There must be a total of at least 2 edges incident amongst u and v besides e, each belonging to a distinct matching, from their respective children. Therefore, $y_u + y_v \geq 2$.

Case 2 Suppose e belongs to a single matching. Then either there is an edge e' incident with u or v which belongs to a single matching, from their respective children, or there is an edge e'' incident with u or v which belongs to two matchings. In either case, $y_u + y_v \geq 2$.

Case 3 Suppose e belongs to two or three matchings, then $y_u + y_v \geq 2$ trivially. □

Lemma 5 immediately implies Theorem 6 using Claim 1.

Theorem 6 *Algorithm 2 is a* $\frac{3}{2}$*-approximate (in expectation) randomized algorithm for MCM on trees, with a worst case update time of* $O(1)$.

We also analyze Algorithm 2 for general graphs with maximum degree 3, and prove the following Theorem.

Theorem 7 *Algorithm 2 is a* 1.8*-approximate (in expectation) randomized algorithm for MCM on general graphs with maximum degree* 3*, with a worst case update time of* $O(1)$.

We use the following lemma to prove Theorem 7.

Lemma 8 *There exists an assignment of the primal charge amongst the dual variables such that the dual constraint for each edge* $e \equiv (u, v)$ *is satisfied at least* $\frac{5}{9}$ *in expectation.*

Proof The dual variable assignment is done at the end of the input/or query, as follows.

- If e does not belong to any matching, then it does not contribute to the value of dual variables.

- If e belongs to a single matching, then there are two subcases.
  1. W.l.o.g., if u is covered by a single matching, then the primal charge $x_e = 1$ is divided as follows. y_u is assigned $1/2 + \epsilon$, and y_v is assigned $1/2 - \epsilon$.
  2. If both u and v are covered by at least two matchings, then the primal charge $x_e = 1$ is divided equally among the dual variables y_u and y_v.
- If e belongs to two or three matchings, then its entire primal charge is divided equally amongst u and v.

The analysis breaks up into three cases.

Case 1 Suppose e belongs to a single matching. Then, $y_u + y_v \geq 1 + 1/2 - \epsilon + 1/2 - \epsilon = 2 - 2\epsilon \geq 3/2 + \epsilon$.

Case 2 Suppose e does not belong to any matching. Then u and v must be covered by a total of at least 3 matchings (counting multiplicities). W.l.o.g., if u is covered by a single matching, then v has to be covered by at least two matchings. Hence, $y_u = 1/2 + \epsilon$, and $y_v \geq 1$. Otherwise, both u and v are covered by at least two matchings, then $y_u \geq 1$ and $y_v \geq 1$. Therefore, $y_u + y_v \geq 3/2 + \epsilon$.

Case 3 Suppose e belongs to two or three matchings, then $y_u + y_v \geq 3/2 + \epsilon$ trivially.

The proof of the lemma is complete with $\epsilon = 1/6$. □

Lemma 8 immediately implies Theorem 7 using Claim 1.

3.1 A Deterministic Algorithm

Note that Algorithm 2 only works against an oblivious adversary. In this section, we derandomize Algorithm 2 to give a $(3/2 + \epsilon)$-approximation deterministic algorithm, for MCM on trees, with an amortized update time of $O(1/\epsilon)$, for any $\epsilon \leq 1/2$.

Algorithm 3 Deterministic Algorithm for MCM

```
1. Let $\epsilon \in (0, 1/2]$ be some input parameter, and $c = 1$.
2. The algorithm maintains four matchings: $M_1, M_2, M_3$ and a support matching $M_4$.
3. When an edge $e$ is inserted, the processing happens in four phases.
    (a) The Augment phase. The new edge $e$ is added to each $M_i$ in which there are no edges adjacent to $e$.
    (b) The Switching phase. For $i = 2, 3$, in order, $M_i \leftarrow M_i \setminus X(M_i, e) \cup \{e\}$, provided $|X(M_i, e)| = 1$ and it decreases the quantity $\sum_{j \in [3], i \neq j} |M_i \cap M_j|$.
        For the edge $e'$ that is discarded from $M_i$, add edges on the other endpoint of $e'$ in $M_j$ ($\forall j \in [4], j \neq i$) to $M_i$ if possible.
    (c) The Support phase. If the edge $e$ was not added to any $M_i, \forall i \in [3]$, in the Augment or Switching phase, then add it to the support matching $M_4$ if there are no edges adjacent to it in $M_4$.
    (d) The ChangeCurr phase. If $|M_c| < \left(|M_i| + |M_j|\right) / (2(1 + \epsilon))$ such that $i, j, c \in [3]$ and $i \neq j \neq c$, then set $c = k$ if $M_k$ is the matching of maximum size among $M_1, M_2, M_3$.
4. On query, output matching $M_c$.
```

Note that there are two more phases in this algorithm than in Algorithm 2, and there is also a minor modification in the description of the Switching phase. These changes are done to ensure that the size of any matching maintained by the algorithm never decreases (which helps with the analysis as pointed out later). An edge can be added to M_i in the Switching phase only if it has one conflicting edge in M_i. But this can result in M_2 and M_3 not being maximal matchings (which is again required in the analysis as pointed out later). The only way this can happen is if some edge e is not added to any matching in the Augment phase, and later on, after the Switching phase, its endpoints are not covered by M_2 and M_3. We add such an edge to the support matching M_4, and this edge is later added to M_2 and M_3 in the Switching phase, thereby ensuring their maximality. With these modifications, the approximation ratio claimed by Theorem 6 still holds on average size of the matchings M_1, M_2, M_3 stored by this algorithm.

We prove the following theorem for Algorithm 3.

Theorem 9 *Algorithm 3 is $\left(\frac{3}{2}+\epsilon\right)$-approximate for MCM on trees, with an amortized update time of $O(1/\epsilon)$.*

Proof We first prove the approximation ratio, and then argue about the update time per edge.

Step (3c) in Algorithm 3 ensures that at each stage $|M_c| \geq \left(|M_i|+|M_j|\right)/(2(1+\epsilon))$, such that $i, j, c \in [3]$ (where $i \neq j \neq c$), and M_c is the current matching which will be output by the algorithm on query. Let M be the optimum matching at any stage. Theorem 6 implies that

$$\begin{aligned} \frac{|M_c|+|M_i|+|M_j|}{3} &\geq \frac{2}{3}|M| \\ \implies |M_c| + 2(1+\epsilon)|M_c| &\geq 2|M| \\ \implies \left(\frac{3}{2}+\epsilon\right)|M_c| &\geq |M|. \end{aligned}$$

Note that the approximation ratio trivially holds after the first edge is inserted as we set $c = 1$ (and will hold even if we set $c = 2$ or 3, because the first edge is added to all three matchings M_1, M_2, M_3).

In the Augment or the Switching phase, $O(1)$ time is spent per edge. Let M'', M_c'', M_i'', M_j'' represent the matchings M, M_c, M_i, M_j immediately after c was updated, and let M', M_c', M_i', M_j' represent the respective matchings immediately after the previous time c was updated. In the ChangeCurr phase, at most $2|M''|$ time is potentially spent (because the size of any matching stored by the algorithm is at most $|M''|$) while changing the current matching output by the algorithm. But we show that this happens very rarely. If $|M''| \geq 2|M'|$, then at least $|M''|/2$ edges have been inserted between the two most recent updates of c. This implies an amortized update time of $O(1)$ per edge.

Now suppose $|M''| < 2|M'|$. Immediately after the previous update of c, $|M_c'| \geq (|M_i'|+|M_j'|)/2$ (because M_c is the maximum size matching among M_1, M_2, and M_3). Just before c is updated in the ChangeCurr phase, $|M_c''| < (|M_i''|+|M_j''|)/(2(1+\epsilon))$. So, the change in the value of $|M_c|$ is at most

$$\frac{|M_i''|+|M_j''|}{2(1+\epsilon)} - \frac{|M_i'|+|M_j'|}{2}.$$

But this value is at least zero, as the size of any matching can never decrease by the description of the algorithm. Hence,

$$\begin{aligned}
&\frac{|M_i''| + |M_j''|}{2(1+\epsilon)} - \frac{|M_i'| + |M_j'|}{2} \geq 0 \\
\implies &|M_i''| + |M_j''| - (1+\epsilon)(|M_i'| + |M_j'|) \geq 0 \\
&\implies (|M_i''| - |M_i'|) + (|M_j''| - |M_j'|) \geq \epsilon(|M_i'| + |M_j'|) \\
&\implies (|M_i''| - |M_i'|) + (|M_j''| - |M_j'|) \geq \epsilon|M'| \qquad \ldots M_i', M_j' \text{ are maximal.}
\end{aligned}$$

Thus, along with the fact that $|M''| < 2|M'|$, $\Omega(\epsilon|M''|)$ edges have been inserted between the two most recent updates of the value of c. This implies an amortized update time of $O(1/\epsilon)$ per edge, and finishes the proof. □

4 MWM in the Online Preemptive Model

In this section, we present a randomized algorithm (that uses only $O(1)$ bits of randomness) for MWM in the online preemptive model, and analyze its performance for growing trees. The algorithm is motivated by the deterministic algorithm for MWM due to McGregor [11]. McGregor's algorithm is easy to describe—if the weight of the new edge is more than $(1+\gamma)$ times the weight of the conflicting edges in the current matching, then evict them and add the new edge. The algorithm is $(1+\gamma)(2+1/\gamma)$-competitive, and attains the best competitive ratio of $3+2\sqrt{2} \approx 5.828$ for $\gamma = \frac{1}{\sqrt{2}}$. It achieves this competitive ratio for the following example. Start by presenting an edge of weight $x_0 = 1$ to the algorithm. This edge will be added to the matching. Assume inductively that after iteration i, the algorithm's matching has only the edge of weight x_i. In iteration $i+1$, present an edge of weight $y_{i+1} = (1+\gamma)x_i$ on one endpoint of x_i (we slightly abuse the notation here, and say that x_i is also the name of the edge of weight x_i). This edge will not be accepted in the algorithm's matching. Give an edge of weight $x_{i+1} = (1+\gamma)x_i + \epsilon$ on the other endpoint of x_i. This edge will be accepted in the algorithm's matching, and x_i will be evicted. This process terminates for some large n, letting $x_{n+1} = (1+\gamma)x_n$. The edge of weight x_{n+1} will not be accepted in the algorithm's matching. The algorithm will hold only the edge of weight x_n, whereas the optimum matching would include the edges of weights $y_1, \ldots, y_{n+1}, x_{n+1}$. It can be easily inferred that this gives the required lower bound on the competitive ratio.

Notice that the edges presented in the example depended on γ. To beat this, we maintain two matchings, with γ values γ_1 and γ_2 respectively, and choose one at random. We describe the algorithm next.

Algorithm 4 Randomized Algorithm for MWM

```
1. Maintain two matchings $M_1$ and $M_2$. Let $j = 1$ with probability $p$, and $j = 2$ otherwise.
2. On receipt of an edge $e$:
   For $i = 1, 2$, if $w(e) > (1+\gamma_i)w(X(M_i, e))$, then $M_i = M_i \backslash X(M_i, e) \cup \{e\}$.
3. Output $M_j$.
```

Note that we cannot just output the best of two matchings because that could violate the constraints of the online preemptive model.

4.1 Analysis

We use the primal-dual technique to analyze the performance of this algorithm. The primal-dual technique used to analyze McGregor's deterministic algorithm for MWM described in [3] is fairly straightforward. However the management becomes complicated with the introduction of randomness, and we are only able to analyze the algorithm in a very restricted class of graphs, which are growing trees.

Theorem 10 *The expected competitive ratio of Algorithm 4 on growing trees is*

$$\max\left\{\frac{1+\gamma_1}{p}, \frac{1+\gamma_2}{1-p}, \frac{(1+\gamma_1)(1+\gamma_2)(1+2\gamma_1)}{p\cdot\gamma_1+(1-p)\gamma_2+\gamma_1\gamma_2}\right\},$$

where p is the probability to output M_1.

We maintain both primal and dual variables along with the run of the algorithm. Consider a round in which an edge $e \equiv (u, v)$ is revealed, where v is the new vertex. Before e is revealed, let e_1 and e_2 be the edges incident with u which belong to M_1 and M_2 respectively. If such an e_i does not exist, then we may assume $w(e_i) = 0$. The primal and dual variables are updated as follows.

- If e is rejected by both matchings, we set the primal variable $x_e = 0$, and the dual variable $y_v = 0$.
- e is added to M_1 only, then we set the primal variable $x_e = p$, and the dual variable $y_u = \max(y_u, \min((1+\gamma_1)w(e), (1+\gamma_2)w(e_2)))$, and $y_v = 0$.
- If e is added to M_2 only, then we set the primal variable $x_e = 1-p$, and the dual variable $y_u = \max(y_u, \min((1+\gamma_1)w(e_1), (1+\gamma_2)w(e)))$, and $y_v = 0$.
- If e is added to both matchings, then we set the primal variable $x_e = 1$, and the dual variables $y_u = \max(y_u, (1+\gamma_1)w(e))$ and $y_v = (1+\gamma_1)w(e)$.
- If an edge e' is evicted from M_1 (or M_2), we decrease its primal variable $x_{e'}$ by p (or $(1-p)$ respectively), and the corresponding dual variables are unchanged.

We begin with three simple observations.

1. The cost of the primal solution is equal to the expected weight of the matching maintained by the algorithm.
2. The dual variables never decrease. Hence, if a dual constraint is feasible once, it remains so.
3. $y_u \geq \min((1+\gamma_1)w(e_1), (1+\gamma_2)w(e_2))$.

The idea behind the analysis is to prove a bound on the ratio of the dual cost and the primal cost while maintaining dual feasibility. By Observation 2, to ensure dual feasibility, it is sufficient to ensure feasibility of the dual constraint of the new edge. If the new edge e is not accepted in any M_i, then $w(e) \leq \min((1+\gamma_1)w(e_1), (1+\gamma_2)w(e_2))$. Hence, the dual constraint is satisfied by Observation 3. Otherwise, it can be seen that the dual constraint is satisfied by the updates performed on the dual variables.

The following lemma implies Theorem 10 using Claim 1.

Lemma 11 $\frac{\Delta Dual}{\Delta Primal} \leq \max\left\{\frac{1+\gamma_1}{p}, \frac{1+\gamma_2}{1-p}, \frac{(1+\gamma_1)(1+\gamma_2)(1+2\gamma_1)}{p\cdot\gamma_1+(1-p)\gamma_2+\gamma_1\gamma_2}\right\}$ *after every round.*

We will use the following simple technical lemma to prove Lemma 11.

Lemma 12 $\frac{ax+b}{cx+d}$ *increases with* x *iff* $ad - bc \geq 0$.

Proof of Lemma 11 There are four cases to be considered.

1. If edge e is accepted in M_1, but not in M_2, then $(1+\gamma_1)w(e_1) < w(e) \leq (1+\gamma_2)w(e_2)$. By Observation 3, before e was revealed, $y_u \geq (1+\gamma_1)w(e_1)$. After e is accepted in M_1, $\Delta\text{Primal} = p(w(e) - w(e_1))$, and $\Delta\text{Dual} \leq (1+\gamma_1)(w(e) - w(e_1))$. Hence,
$$\frac{\Delta\text{Dual}}{\Delta\text{Primal}} \leq \frac{(1+\gamma_1)}{p}.$$
2. If edge e is accepted in M_2, but not in M_1, then $(1+\gamma_2)w(e_2) < w(e) \leq (1+\gamma_1)w(e_1)$. By Observation 3, before e was revealed, $y_u \geq (1+\gamma_2)w(e_2)$. After e is accepted in M_2, $\Delta\text{Primal} = (1-p)(w(e) - w(e_2))$, and $\Delta\text{Dual} \leq (1+\gamma_2)(w(e) - w(e_2))$. Hence,
$$\frac{\Delta\text{Dual}}{\Delta\text{Primal}} \leq \frac{(1+\gamma_2)}{1-p}.$$
3. Suppose edge e is accepted in both the matchings, and $(1+\gamma_1)w(e_1) \leq (1+\gamma_2)w(e_2) < w(e)$. By Observation 3, before e was revealed, $y_u \geq (1+\gamma_1)w(e_1)$. After e is accepted in both the matchings, $\Delta\text{Dual} \leq (1+\gamma_1)(2w(e) - w(e_1))$. The change in the primal cost is
$$\begin{aligned}\Delta\text{Primal} &\geq w(e) - p\cdot w(e_1) - (1-p)\cdot w(e_2)\\ &\geq w(e) - p\cdot w(e_1) - (1-p)\cdot\frac{w(e)}{1+\gamma_2}\\ &= \frac{p+\gamma_2}{1+\gamma_2}w(e) - p\cdot w(e_1).\\ \frac{\Delta\text{Dual}}{\Delta\text{Primal}} &\leq (1+\gamma_1)\frac{2w(e)-w(e_1)}{\frac{p+\gamma_2}{1+\gamma_2}w(e) - p\cdot w(e_1)}.\end{aligned}$$
By Lemma 12, this value increases, for a fixed $w(e)$, with $w(e_1)$ if $\gamma_2 \leq \frac{p}{1-2p}$, and its worst case value is obtained when $(1+\gamma_1)w(e_1) = w(e)$. Thus,
$$\begin{aligned}\frac{\Delta\text{Dual}}{\Delta\text{Primal}} &\leq (1+\gamma_1)\frac{2(1+\gamma_1)(1+\gamma_2) - (1+\gamma_2)}{(p+\gamma_2)(1+\gamma_1) - p(1+\gamma_2)}\\ &= (1+\gamma_1)(1+\gamma_2)\frac{1+2\gamma_1}{p\cdot\gamma_1 + (1-p)\gamma_2 + \gamma_1\gamma_2}.\end{aligned}$$

4. Suppose e is accepted in both the matchings, and $(1+\gamma_2)w(e_2) \leq (1+\gamma_1)w(e_1) < w(e)$. By Observation 3, before e was revealed, $y_u \geq (1+\gamma_2)w(e_2)$. The following bound can be proven similarly.

$$\frac{\Delta\text{Dual}}{\Delta\text{Primal}} \leq (1+\gamma_1)(1+\gamma_2)\frac{1+2\gamma_1}{p\cdot\gamma_1+(1-p)\gamma_2+\gamma_1\gamma_2}.$$

□

The following theorem is an immediate consequence of Theorem 10.

Theorem 13 *Algorithm 4 is a 3-competitive randomized algorithm for MWM on growing trees, when $p = 1/3$, $\gamma_1 = 0$, and $\gamma_2 = 1$; and this bound is tight.*

The input for which Algorithm 4 is 3-competitive is as follows. Start by presenting an edge of weight $x_0 = 1$. It will be added to both M_1 and M_2. Assume inductively that currently both matchings only contain an edge of weight x_i. Present an edge of weight $y_{i+1} = x_i$ on one endpoint of x_i. This edge will not be accepted in either of the matchings. Present an edge of weight $x_{i+1} = 2\cdot x_i + \epsilon$ on the other endpoint of x_i. This edge will be accepted in both the matchings, and x_i will be evicted. For a sufficiently large value n, let $x_{n+1} = x_n$. So the edge of weight x_{n+1} will not be accepted in either of the matchings. Both the matchings will hold only the edge of weight x_n, whereas the optimum matching would include the edges of weights $y_1, \ldots, y_{n+1}, x_{n+1}$. The weight of the matching stored by the algorithm is 2^n, whereas the weight of the optimum matching is $\approx 3\cdot 2^n$ (we have ignored the ϵ terms here). This gives the competitive ratio 3.

Note In the analysis of Algorithm 4 for growing trees, we crucially use the following fact in the dual variable assignment. If an edge $e \notin M_i$ for some i, then a new edge incident with its leaf vertex will definitely be added to M_i, and it suffices to assign a zero charge to the corresponding dual variable. This is not necessarily true for more general classes of graphs, and new ideas are needed to analyze the performance for those classes.

Acknowledgements The first author would like to thank Ashish Chiplunkar for helpful suggestions to improve the competitive ratio of Algorithm 4, and also to improve the presentation of Sect. 4.

Appendix: Proof of Lemma 2

We crucially use the following lemma to prove Lemma 2.

Lemma 14 (a) *If an edge e belongs only to M_4 at the end of input, then bad edges cannot be incident with both its endpoints.*
(b) *Also, if an edge e was added to M_4 only in the Switching phase, then e cannot be a bad edge.*

Proof There are two cases to consider.

1. Suppose e was added to M_4 only when it was revealed. Then on one of its endpoints either there should be two edges incident (other than e), such that each of them belongs to a single matching, or there should be one edge which belongs to two matchings. In either case, the edges incident with that endpoint of e should have neighboring edges which belong to some matching (by the description of algorithm). And hence, these edges cannot be bad.
2. Suppose e was added to M_4 as well as to some other matching when it was revealed. If e belonged to three matchings when it was revealed, then its neighboring edge will have its endpoints covered by at least four matching edges, and this number can never go below four. If e belonged to two matchings when it was revealed, then it

   – (a) either has one neighboring edge which belongs to two matchings,

   – (b) or one neighboring edge on each of its endpoints, each belonging to distinct matchings,

   – (c) or two neighboring edges on one of its endpoints, such that both of them belong to distinct matchings.

   In Case (a), this neighboring edge should have a neighboring edge on its other endpoint which belongs to some matching, and hence it cannot be a bad edge. In Case (b), each of these edges should have at least two neighboring edges of their own on their respective other endpoint, which belong to certain matching. Hence, not both of these edges can be bad. In Case (c), both these edges should have neighboring edges of their own on their respective other endpoint, which belong to certain matchings. Hence, both these edges cannot be bad.

For the second part of the lemma, if edge e was added to M_4 in the Switching phase, then this means that e will have three neighboring edges e_1,e_2, and e_3, belonging to M_1, M_2, and M_3, respectively. This is because e will be added to M_4 in the Switching phase only if it is not added to M_2 or M_3 in the Switching phase, which means there are edges which belong only to M_2 and M_3 respectively. □

Proof of Lemma 2 There are two cases to consider.

1. Suppose there is a bad leaf edge e which belongs to M_4. If e is added to M_4 in the Switching phase, then e cannot be a bad edge (by part (b) of Lemma 14). So, e has to be added to M_4 in the Augment phase for it to be a bad leaf edge in the future.
   - If e was added to M_4 alone when revealed, then it must have neighbors e_1 and e_2 such that both of them do not belong to M_4. Then, they must have had neighboring edges e'_1 and e'_2 respectively which belonged to M_4 (at some stage). If e''_1 (and/or e''_2) switches e'_1 (and/or e'_2 respectively) out of M_4, then e''_1 (and/or e''_2 respectively) cannot be a bad edge (by part (b) of Lemma 14). Otherwise, the lemma holds due to part (a) of Lemma 14.
   - If e was added to two matchings (M_4 being one of them) when it was revealed, and finally has only one internal neighboring edge e_1, then e_1 will have a

neighboring edge e_2 on its other endpoint. Either e_2 belongs to M_4 or its neighboring edge e'_2 on the other endpoint belongs to M_4. The lemma holds if finally e'_2 belongs to M_4 (by part (a) of Lemma 14) or if finally the neighboring edge e''_2 of e'_2 belongs to M_4 (by part (b) of Lemma 14). (The proof for this case will also work for the case when e was revealed first as a single disconnected edge, and then e_1 was revealed on one of its endpoints.)

- If e was added to two or three matchings (M_4 being one of them) when it was revealed, and finally has two internal neighboring edges e_1 and e_2, then e_1 and e_2 must have neighboring edges e'_1 and e'_2 respectively which belonged to M_4 (at some stage). If e''_1 (and/or e''_2) switches e'_1 (and/or e'_2 respectively) out of M_4, then e''_1 (and/or e''_2 respectively) cannot be a bad edge (by part (b) of Lemma 14). Otherwise, the lemma holds due to part (a) of Lemma 14.

2. Let e_1 and e_2 be two bad internal edges which do not belong to M_4. Then, they must have had neighboring edges e'_1 and e'_2 respectively which belonged to M_4 (at some stage). If e''_1 (and/or e''_2) switches e'_1 (and/or e'_2 respectively) out of M_4, then e''_1 (and/or e''_2 respectively) cannot be a bad edge (by part (b) of Lemma 14). Otherwise, the Lemma holds due to part (a) of Lemma 14. □

References

1. Bernstein, A., Stein, C.: 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, 10–12 Jan 2016, pp. 692–711. https://doi.org/10.1137/1.9781611974331.ch50 (2016)
2. Bhattacharya, S., Henzinger, M., Nanongkai, D.: 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, 16–19 Jan 2017, pp. 470–489. https://doi.org/10.1137/1.9781611974782.30 (2017)
3. Chiplunkar, A., Tirodkar, S., Vishwanathan, S.: On randomized algorithms for matching in the online preemptive model. In: Algorithms-ESA 2015—23rd Annual European Symposium, Patras, Greece, 14–16 Sept 2015, Proceedings, pp. 325–336. https://doi.org/10.1007/978-3-662-48350-3_28 (2015)
4. Epstein, L., Levin, A., Mestre, J., Segev, D.: Improved approximation guarantees for weighted matching in the semi-streaming model. SIAM J. Discrete Math. **25**(3), 1251–1265 (2011)
5. Epstein, L., Levin, A., Segev, D., Weimann, O.: Improved bounds for online preemptive matching. In: 30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013, Kiel, Germany, pp. 389–399. https://doi.org/10.4230/LIPIcs.STACS.2013.389 (2013)
6. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: On graph problems in a semi-streaming model. Theor. Comput. Sci. **348**(2), 207–216 (2005)
7. Gupta, M.: Maintaining approximate maximum matching in an incremental bipartite graph in polylogarithmic update time. In: 34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, 15–17 Dec 2014, New Delhi, India, pp. 227–239. https://doi.org/10.4230/LIPIcs.FSTTCS.2014.227 (2014)
8. Gupta, M., Peng, R.: Fully dynamic $(1+\epsilon)$-approximate matchings. In: Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, FOCS '13, pp. 548–557, Washington, DC, USA, 2013. IEEE Computer Society (2013). https://doi.org/10.1109/FOCS.2013.65
9. Kalyanasundaram, B., Pruhs, K.: Online weighted matching. J. Algorithms **14**(3), 478–488 (1993). https://doi.org/10.1006/jagm.1993.1026
10. Karp, R.M., Vazirani, U.V., Vazirani, V.V.: An optimal algorithm for on-line bipartite matching. In: Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing, STOC '90, pp. 352–358, New York, NY, USA. ACM (1990)

11. McGregor, A.: Finding graph matchings in data streams. In: Proceedings of the 8th International Workshop on Approximation, Randomization and Combinatorial Optimization Problems, and Proceedings of the 9th International Conference on Randomization and Computation: Algorithms and Techniques, APPROX'05/RANDOM'05, pp. 170–181. Springer, Berlin (2005)
12. Micali, S., Vazirani, V.V.: 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, pp. 17–27, Washington, DC, USA. IEEE Computer Society (1980). https://doi.org/10.1109/SFCS.1980.12
13. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin (2003)
14. Solomon, S.: Fully dynamic maximal matching in constant update time. In: IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, Oct 9–11 2016, Hyatt Regency, New Brunswick, New Jersey, USA, pp. 325–334. https://doi.org/10.1109/FOCS.2016.43 (2016)
15. Varadaraja, A.B.: Buyback problem: approximate matroid intersection with cancellation costs. In: Automata, Languages and Programming: 38th International Colloquium, ICALP 2011, Zurich, Switzerland, Proceedings, Part I, pp. 379–390. https://doi.org/10.1007/978-3-642-22006-7_32 (2011)

Affiliations

Sumedh Tirodkar[1] · **Sundar Vishwanathan**[2]

Sundar Vishwanathan
sundar@cse.iitb.ac.in

[1] Tata Institute of Fundamental Research, Mumbai, India

[2] CSE, IIT Bombay, Mumbai, India