On Randomized Algorithms for Matching in the Online Preemptive Model

Ashish Chiplunkar AshishC@cse.iitb.ac.in
Sumedh Tirodkar SumedhT@cse.iitb.ac.in
Sundar Vishwanathan Sundar@cse.iitb.ac.in

Abstract

We investigate the power of randomized algorithms for the maximum cardinality matching (MCM) and the maximum weight matching (MWM) problems in the online preemptive model. In this 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. The complexity of the problem is settled for deterministic algorithms [5, 7]. However, it is wide open for randomized algorithms. We initiate a systematic study of the same.

We present a primal-dual analysis for the deterministic algorithm due to [5], and extend this technique to barely random algorithms for MCM on paths and growing trees. Next, we identify certain natural classes of randomized online algorithms, and of input models, and prove lower bounds on the competitive ratio achievable for these classes. We also present the best possible $\frac{3}{2}$-competitive randomized algorithm for MCM on paths.

1 Introduction

Matching has been a central problem in combinatorial optimization. Indeed, algorithm design in various models of computations, sequential, parallel, streaming, etc., have been influenced by techniques used for matching. We study the maximum cardinality matching (MCM) and the maximum weight matching (MWM) problems in the online preemptive model. In this model, edges $e_1, \ldots, e_m$ of a graph, possibly weighted, are presented one by one. An algorithm is required to output a matching $M_i$ after the arrival of each edge $e_i$. This model constrains an algorithm to accept/reject an edge as soon as it is revealed. If accepted, the adjacent edges, if any, have to be discarded.

An algorithm is said to have a competitive ratio $\alpha$ if for all graphs and all sequences of edges, the cost of the matching maintained by the algorithm is at least $\frac{1}{\alpha}$ times the cost of the offline optimum.

The deterministic complexity of this problem is settled. For MCM, it is easily seen that no deterministic algorithm can have a competitive ratio less than $2$ and that, an algorithm which maintains a maximal matching is $2$-competitive. For MWM, McGregor [5] gave a deterministic algorithm with a competitive ratio of $3 + 2\sqrt{2} \approx 5.828$, which was later proved to be the best possible by Varadaraja [7]. Very little is known on the power of randomness for this problem. Recently, Epstein et. al. [2] proved a lower bound of $1 + \ln 2 \approx 1.693$ on the competitive ratio of randomized algorithms for MCM. This is the best lower bound known even for MWM. Epstein et. al. [2] also give a $5.356$-competitive randomized algorithm for MWM. In this paper, we initiate a systematic study of the power of randomness for this problem. As a first step towards improving the upper bound, we present barely random algorithms, (which use only a constant number of bits of randomness), for MCM on paths, on growing trees with maximum degree $3$, and on growing trees with no restriction on maximum degree, which achieve competitive ratios $\frac{3}{2}$, $\frac{12}{7}$, and $\frac{28}{15}$ respectively. In the growing tree model, at each step, a new edge connects a new vertex to the already existing tree. We remark that all lower bounds on deterministic algorithms are on growing trees, and so this is a natural candidate to begin investigation. The problem is already non trivial in this model as our analysis indicates. Note that all the above mentioned algorithms can be considered as randomized greedy algorithms in the offline setting. Proving an approximation ratio, even slightly better than $2$ is known to be very difficult for such algorithms. For example, see [6].

The optimal maximal matching algorithm for MCM, and McGregor’s [5] optimal deterministic algorithm for MWM, are both local algorithms. The choice of whether a new edge is accepted or rejected is
only based on the new edge and conflicting edges stored by the algorithm. It is natural to add randomness to such local algorithms, and to ask whether they do better than the known deterministic lower bounds. A obvious way to add randomness is to accept/reject the new edge with certain probability, which is only dependent on the new edge and the conflicting edges stored by the algorithm. The choice of adding a new edge is independent of the previous coin tosses used by the algorithm. We call such algorithms as randomized local algorithms. We prove negative results, showing that randomized local algorithms cannot do better than optimal deterministic algorithms.

The idea behind the randomized algorithm given by Epstein et. al. [2] is the following. For a parameter \( \theta \), they round the weights of the edges to powers of \( \theta \) randomly, and then they update the matching using a deterministic algorithm. The weights get distorted by a factor \( \frac{\ln \theta}{\theta - 1} \) in the rounding step, and the deterministic algorithm has a competitive ratio of \( 2 + \frac{2}{\ln \theta} \) on \( \theta \)-structured graphs, i.e., graphs with edge weights being powers of \( \theta \). The overall competitive ratio of the randomized algorithm is \( \frac{\ln \theta}{\theta - 1} \cdot \left( 2 + \frac{2}{\ln \theta} \right) \) which is minimized at \( \theta \approx 5.356 \). A natural way to reduce this competitive ratio is by getting an improved algorithm for \( \theta \)-structured graphs. However, we prove that the competitive ratio \( 2 + \frac{2}{\ln \theta} \) is indeed tight for \( \theta \)-structured graphs, as long as \( \theta \geq 4 \), for deterministic algorithms.

We also present the best possible randomized algorithms for MCM on paths, that achieves a competitive ratio of \( \frac{4}{3} \), which matches the lower bound.

The objective of this study is to understand the complexity of randomized algorithms in the online preemptive model. The technical contributions of this paper have two broad themes. One is to highlight model specific bounds. The other important contribution is the use of the primal-dual method for this model. We believe that the key difficulty in analyzing randomized algorithms lies in the management of dual variables. Even in the growing tree model, the primal-dual based analysis is considerably involved. We begin by analyzing McGregor’s deterministic algorithm [5] for MWM in this framework.

2 A Primal-Dual Analysis of a deterministic algorithm for MWM

In this section, we give a primal-dual analysis for the deterministic algorithm due to [5] for maximum weight matching problem in the online preemptive model. The algorithm is as follows.

**Algorithm 1** Deterministic Algorithm for MWM

1. Fix a parameter \( \gamma \).
2. If the new edge \( e \) has weight greater than \((1 + \gamma)\) times the weight of the edges currently adjacent to \( e \), then include \( e \) and discard the adjacent edges.

2.1 Analysis

**Lemma 1.** [5] The competitive ratio of this algorithm is \((1 + \gamma)(2 + \frac{1}{\gamma})\).

We use the primal-dual technique to prove the same competitive ratio. This analysis technique is different from the one in [1]. In [1], the primal variables once set to a certain value are never changed whereas in our analysis the primal variables may change during the run of algorithm. The primal and dual LPs for the maximum weight matching problem are as follows.

| Primal LP | Dual LP |
|-----------|---------|
| \( \max \sum_e w_e x_e \)  | \( \min \sum_e y_e \) |
| \( \forall v \ldots \sum_{e \in \delta(v)} x_e \leq 1 \)  | \( \forall e \ldots y_u + y_v \geq w_e \) |
| \( x_e \geq 0 \)  | \( y_e \geq 0 \) |

We maintain primal and dual variables along with the run of algorithm. On arrival of an edge and processing it, we maintain the following invariants.
• The dual LP is always feasible.
• For each edge $e \equiv (u, v)$ in the current matching, $y_u \geq (1 + \gamma)w(e)$ and $y_v \geq (1 + \gamma)w(e)$.
• The change in cost of the dual solution is at most $(1 + \gamma)(2 + \frac{1}{\gamma})$ times the change in cost of primal solution.

These invariants imply that the competitive ratio of the algorithm is $(1 + \gamma)(2 + \frac{1}{\gamma})$.

We start with 0 as the initial primal and dual solutions. Consider a round in which an edge $e$ of weight $w$ is given. Assume that all the above invariants hold before this edge is given. Whenever an edge $e \equiv (u, v)$ is accepted by the algorithm, we assign values $x_e = 1$ to the primal variable and $y_u = \max(y_u, (1 + \gamma)w(e))$, $y_v = \max(y_v, (1 + \gamma)w(e))$ to the dual variables of its end points. Whenever an edge $e$ is evicted, we change its primal variable $x_e = 0$. The dual variables never decrease. Hence, if a dual constraint is feasible once, it remains so. We will now show that the invariants are always satisfied. These are three cases.

1. If the edge $e \equiv (u, v)$ has no conflicting edges in the current matching, then it is accepted by the algorithm in current matching $M$. We assign $x_e = 1, y_u = \max(y_u, (1 + \gamma)w(e))$ and $y_v = \max(y_v, (1 + \gamma)w(e))$. Hence, $y_u \geq (1 + \gamma)w(e)$ and $y_v \geq (1 + \gamma)w(e)$. And hence, the dual constraint $y_u + y_v \geq w(e)$ is feasible. The change in the dual cost is at most $2(1 + \gamma)w(e)$. The change in the primal cost is $w(e)$. So, the change in the dual cost is at most $(1 + \gamma)(2 + \frac{1}{\gamma})$ times the change in the cost of the primal solution.

2. If the edge $e \equiv (u, v)$ has conflicting edges $X(M, e)$ and $w(e) \leq (1 + \gamma)w(X(M, e))$, then it is rejected by the algorithm. That happens when $y_u + y_v \geq (1 + \gamma)w(X(M, e))$, and the dual constraint for edge $e$ is satisfied.

3. If the edge $e \equiv (u, v)$ had conflicting edges $X(M, e)$ and $w(e) > (1 + \gamma)w(X(M, e))$, then it is accepted by the algorithm in the current matching $M$ and $X(M, e)$ is/are evicted from $M$. We only need to show that the change in dual cost is at most $(1 + \gamma)(2 + \frac{1}{\gamma})$ times the change in the primal cost. The change in primal cost is $w(e) - w(X(M, e))$. The change in dual cost is at most $2(1 + \gamma)w(e) - (1 + \gamma)w(X(M, e))$. Hence the ratio is at most

$$\frac{2(1 + \gamma)w(e) - (1 + \gamma)w(X(M, e))}{w(e) - w(X(M, e))}$$

$$= 2(1 + \gamma) + \frac{(1 + \gamma)w(X(M, e))}{w(e) - w(X(M, e))}$$

$$< 2(1 + \gamma) + \frac{w(e)}{w(e) - w(X(M, e))}$$

$$\leq 2(1 + \gamma) + (1 + \frac{1}{\gamma})$$

$$= (1 + \gamma)(2 + \frac{1}{\gamma})$$

Here, the management of the dual variables was straightforward. The introduction of randomization complicates matters considerably, and we are only able to analyze the algorithm in the very restricted setting of paths and “growing trees”.

3 Barely Random Algorithms for MCM

In this section, we present barely random algorithms for the MCM on paths and growing trees. These algorithms use only a constant number of bits of randomness. We begin by stating the following theorem.

**Theorem 1.** There exists a barely random algorithm for the MCM on paths which is $\frac{3}{2}$-competitive, and no barely random algorithm can do better.
The barely random algorithm for paths and the proof of above theorem is deferred to the appendix (section A). Next, we present barely random algorithms for growing trees with maximum degree 3 and growing trees with no restriction on the maximum degree. We use the primal-dual technique to analyze the performance of all these algorithms. The primal and dual LPs for MCM are as follows.

| Primal LP | Dual LP |
|-----------|---------|
| \[ \max \sum_e x_e \] | \[ \min \sum_v y_v \] |
| \[ \forall v \ldots \sum_{v \in e} x_e \leq 1 \] | \[ \forall e \ldots y_u + y_v \geq 1 \] |
| \[ x_e \geq 0 \] | \[ y_v \geq 0 \] |

3.1 Randomized Algorithm for Growing Trees with maximum degree 3

In this section, we give a barely random algorithm for growing trees, with maximum degree 3. We beat the lower bound of 2 for MCM on the performance of any deterministic algorithm, for this class of inputs. The edges are revealed in online fashion such the one new vertex is revealed per edge, (except for the first edge). Any vertex in the input graph has maximum degree 3.

Algorithm 2 Randomized Algorithm for Growing Trees with \( \Delta = 3 \)

1. The algorithm maintains 3 matchings \( M_1, M_2, M_3 \).
2. On receipt of an edge \( e \), the processing happens in two phases.
   (a) The augment phase. Here, the new edge \( e \) is added to each \( M_i \) such that there is no edge in \( M_i \) sharing an end point with \( e \).
   (b) The switching phase. For \( i = 2, 3 \), in order, \( e \) is added to \( M_i \) and the conflicting edge is discarded, provided it decreases the quantity \( \sum_{i,j \in [3], i \neq j} |M_i \cap M_j| \).
3. Output a matching \( M_i \) with probability \( \frac{1}{3} \).

We make following simple observations.

- There cannot be an edge which is not in any matching.
- Call an edge “bad” if its end points are covered by only two matchings. Indeed, an edge whose none of the end points are leaves, cannot be “bad”.
- An edge incident on a vertex of degree 3 cannot be “bad”, because there will be a distinct edge belonging to every matching.

We begin by proving a few simple lemmas regarding the algorithm.

**Lemma 2.** There cannot be “bad” edges incident on both vertices of an edge.

**Proof.** Note that a “bad” edge is created when an edge is revealed on a leaf node of another edge which belongs to two or three matchings currently and finally belongs to only one matching.

Let \( e \) be an edge which currently belongs to three matchings, which means it is the first edge revealed. Now if an edge \( e_1 \) is revealed on a vertex of \( e \), then \( e_1 \) would be added to one matching, and \( e \) would be removed from that matching, (in the switching phase of the algorithm). For \( e_1 \) to be a bad edge, \( e \) should be switched out of one more matching. This can happen only if there are two more edges revealed on the other vertex of \( e \). This means there cannot be “bad” edges on both sides of \( e \).

Let \( e \) belongs to two matchings. Then \( e \) already has a neighboring edge \( e_2 \) which belongs to some matching. When \( e_1 \) is revealed on the leaf vertex of \( e \), it will be added to one matching, in the augment phase. Now for \( e_1 \) to be “bad”, \( e \) should switch out of some matching. This can only happen if there is one more edge \( e_3 \) revealed on the common vertex of \( e \) and \( e_2 \). Again, the lemma holds.

**Lemma 3.** If a vertex has three edges incident on it, then at most one of these edges can have a “bad” neighboring edge.
Proof. Out of the three edges incident on a vertex, only one could have belonged to two matchings at any step during the run of algorithm. Hence, only that edge which belonged to two matchings at some stage during the run of algorithm can have a “bad” neighboring edge.

Theorem 2. The randomized algorithm for finding MCM on growing trees with maximum degree 3 is $\frac{12}{7}$-competitive.

Proof. $M_1, M_2, M_3$ are valid matchings and hence correspond to valid primal solutions. For each edge $e \equiv (u, v)$ in some matching $M_i$, we distribute a charge of $x_e = 1$ amongst dual variables $y_u$ and $y_v$ of its end points. We prove that for each edge $e$, $y_u + y_v \geq \frac{7}{4}$. Thus, $\mathbb{E}[y_u + y_v] \geq \frac{7}{12}$. Hence, this algorithm has a competitive ratio $\frac{12}{7}$. All the dual variables are initialized to 0. Suppose $e \equiv (u, v) \in M_i$ for some $i \in [3]$. Then distribution of primal charge $x_e$ amongst dual variables $y_u$ and $y_v$ of its end points is done as follows. If there is a “bad” edge incident on $u$, then edge $e$ transfer $\frac{3}{4}$ of its primal charge to $y_u$ and rest of it to $y_v$. Else, edge $e$ transfer its primal charge equally between $y_u$ and $y_v$.

We look at three cases and then prove that $y_u + y_v \geq \frac{7}{4}$ for each edge $e \equiv (u, v)$.

1. Edge $e \equiv (u, v)$ is “bad”. $e \in M_i$ for some $i \in [3]$. $e$ will have some neighboring edge $e_1$ such that $e_1 \in M_j$ for $j \in [3]$ and $i \neq j$. Let the common vertex between $e$ and $e_1$ be $v$. Then $e_1$ will transfer $\frac{3}{4}$ of its primal charge to $y_v$. Thus, $y_u + y_v = \frac{7}{4}$.

2. Edge $e \equiv (u, v)$ is present in a single matching and not “bad”. This case has four sub cases.

   (a) $e$ has one neighboring edge $e_1$. Then $e_1$ should belong to two matchings.

   (b) $e$ has two neighboring edges $e_1$ and $e_2$ both belonging to only one matching. If these are both on the same side of $e$, then at most one of them could have a “bad” neighboring edge (by lemma 3). If these are on opposite sides of $e$, then none of them can have a “bad” neighboring edge.

   (c) $e$ has three neighboring edges $e_1, e_2$, and $e_3$, such that $e_1$ and $e_2$ are on one side of $e$, and $e_3$ is on another side of $e$. At most one of $e_1$ and $e_2$ can have a “bad” neighboring edge (by lemma 3).

   (d) $e$ has four neighboring edges $e_1, e_2, e_3$, and $e_4$, such that $e_1$ and $e_2$ are on one side of $e$, and $e_3$ and $e_4$ are on another side of $e$.

   We can see that in all the above sub cases, $y_u + y_v \geq \frac{7}{4}$.

3. Edge $e \equiv (u, v)$ belongs to two or three matchings. Then, $y_u + y_v \geq \frac{7}{4}$ trivially.

This proves that we have a $\frac{12}{7}$-competitive randomized algorithm for finding MCM on growing trees with maximum degree 3.

3.2 Randomized Algorithm for MCM on Growing Trees

In this section, we give a barely random algorithm for growing trees. By using only two bits of randomness, we beat the lower bound of 2 for MCM on the performance of any deterministic algorithm, for this class of inputs. The edges are revealed in online fashion such the one new vertex is revealed per edge, (except for the first edge).
Algorithm 3 Randomized Algorithm for Growing Trees

1. The algorithm maintains four matchings: $M_1, M_2, M_3$, and $M_4$.

2. On receipt of an edge $e$, the processing happens in two phases.
   
   (a) The augment phase. Here, the new edge $e$ is added to each $M_i$ such that there is no edge in $M_i$ sharing an end point with $e$.

   (b) The switching phase. For $i = 2, 3, 4$, in order, $e$ is added to $M_i$ and the conflicting edge is discarded, provided it decreases the quantity $\sum_{i,j \in [4], i \neq j} |M_i \cap M_j|$.

3. Output a matching $M_i$ with probability $\frac{1}{4}$.

We may assume that all edges that do not belong to any matching are leaf edges. This helps in simplifying the analysis. Suppose there is an edge $e$ which does not belong to any matching, but is not a leaf edge. We break the tree into two subtrees at edge $e$. The edge $e$ belongs to the tree in which it has 4 neighboring edges.

We make the following simple observations.

- When an edge is revealed, its end points are covered by all 4 matchings.
- An edge $e$ that does not belong to any matching, has 4 distinct neighboring edges on one of its end points such that each of these edges belong to a distinct matching. The condition must hold when the edge was revealed, and cannot change subsequently.

An edge is called “internal” if there are edges incident on both its end points. An edge is called “bad” if its end points are covered by only 3 matchings. We begin by proving few lemmas about the algorithm.

**Lemma 4.**
1. The end points of an “internal” edge are covered by at least four edges which are in some $M_i$.

2. If $p, q, r$ are 3 consecutive vertices on a path, then “bad” edges cannot be incident on all 3 of these vertices, (as in figure 7).

The proof of this lemma is presented in the appendix (section [B]).

**Theorem 3.** The randomized algorithm for finding MCM on growing trees is $\frac{28}{15}$-competitive.

A simple analysis like the one in section 3.1 will not work here. (For a reason, see appendix section [C]).

The analysis of this algorithm proceeds in two steps. First we allocate ranks to the vertices of tree and then with the help of these ranks, we assign values to the dual variables and use the primal-dual technique to prove the competitive ratio of this algorithm.

1. **Ranking:** Consider a vertex $v$. Let $v_1, \ldots, v_k$ be the neighbors of $v$. For each $i$, let $d_i$ denote the maximum distance from $v$ to any leaf if there was no edge between $v$ and $v_i$. The rank of $v$ is defined as the minimum of all the $d_i$. Observe that the rank of $v$ is one plus the second highest rank among the neighbors of $v$. Thus there can be at most one neighbor of vertex $v$ which has rank at least the rank of $v$. All leaves have rank 0. Rank 1 vertices have at most 1 non-leaf neighbor.
2. Dual Variable Management: Consider an edge $e$ from a vertex of rank $i$ to a vertex of rank $j$, such that $i \leq j$. This edge will distribute its primal weight between its end-points. The exact values are discussed in the proof of the claim below.

- 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 $0$ or $\epsilon$ or $2\epsilon$ of its primal charge is transferred to the rank $i$ vertex as required, and rest is transferred to the rank $j$ vertex.
- If $e$ belongs to two matchings, then at most $3\epsilon$ of its primal charge is transferred to the rank $i$ vertex as required, and rest is transferred to the rank $j$ vertex.
- If $e$ belongs to three or four matchings, then its entire primal charge is transferred to the rank $j$ vertex.

Claim 1. The dual constraint for each edge $e \equiv (u, v)$ is satisfied at least $\frac{15}{28}$ in expectation, i.e. $\mathbb{E}[y_u + y_v] \geq \frac{15}{28}$.

Proof. Consider an edge $e \equiv (u, v)$ from vertex $u$ of rank $i$ to vertex $v$ of rank $j$, such that $i \leq j$. We will show that $y_u + y_v \geq 2 + \epsilon$ for such an edge. The value of $\epsilon$ is chosen later on to prove the above claim. The claim is proved by considering the following six cases. The proof is by induction on $< j, i >$.

1. Suppose $e$ does not belong to any matching. Then it must be a leaf edge. Hence, $i = 0$. There must be 4 edges incident on $v$ besides $e$, each belonging to a distinct matching. Of these 4, at least 3 edges $e_1$, $e_2$, and $e_3$, must be from lower ranked vertices to the rank $j$ vertex $v$. The edges $e_1$, $e_2$, and $e_3$, each transfer $1 - 2\epsilon$ to $y_v$. Therefore, $y_u + y_v \geq 3 - 6\epsilon \geq 2 + \epsilon$.

2. Suppose $e$ is a “bad” edge that belongs to a single matching. Since no “internal” edge can be a “bad” edge, $i = 0$. This implies (Lemma I) that, there is an edge $e_1$ from a rank $j - 1$ vertex to $v$, which belongs to a single matching, and there is an edge $e_2$, from $v$ to a higher ranked vertex, which also belongs to a single matching. The edge $e$ transfers a charge of 1 to $y_v$. If $e_1$ transfers a charge of 1 (or $1 - \epsilon$) to $y_v$, then $e_2$ transfers $\epsilon$ (or $2\epsilon$ respectively) to $y_v$. In either case, $y_u + y_v = 2 + \epsilon$.

3. Suppose $e$ is not a “bad” edge, and it belongs to a single matching.

- $i = 0$. There are two sub cases.
  - There is an edge $e_1$ from some rank $j - 1$ vertex to $v$ which belongs to 2 matchings, or there are two other edges $e_2$ and $e_3$ from some lower rank vertices to $v$, each belonging to separate matchings. The edge $e$ transfers a charge of 1 to $y_v$. Either $e_1$ transfers a charge of at least $1 - 3\epsilon$ to $y_v$, or $e_2$ and $e_3$ transfer a charge of at least $1 - 2\epsilon$ each, to $y_v$. In either case, $y_u + y_v \geq 3 - 4\epsilon \geq 2 + \epsilon$.
  - There is one edge $e_1$, from a rank $j - 1$ vertex to $v$, which belongs to single matching, and there is one edge $e_2$, from $v$ to an higher ranked vertex, which belongs to 2 matchings. The edge $e$ transfers a charge of 1 to $y_v$. If $e_1$ transfers a charge of 1 (or $1 - \epsilon$ or $1 - 2\epsilon$) to $y_v$, then $e_2$ transfers $\epsilon$ (or $2\epsilon$ or $3\epsilon$ respectively) to $y_v$. In either case, $y_u + y_v = 2 + \epsilon$.

- $i > 0$. There are two sub cases.
  - There are at least two edges $e_1$ and $e_2$ from lower rank vertices to $u$, and one edge $e_3$ from $v$ to a higher ranked vertex. All these edges belong to single matching (not necessarily the same).
  - There is one edge $e_4$ from a vertex of lower rank to $u$, at least one edge $e_5$ from a lower ranked vertex to $v$, and one edge $e_6$ from $v$ to a vertex of higher rank. All these edges belong to a single matching (not necessarily the same).

The edge $e$ transfers a charge of 1 among $y_u$ and $y_v$. If $e_1$ and $e_2$ transfer a charge of at least $1 - 2\epsilon$ each, to $y_v$, then $y_u + y_v \geq 3 - 4\epsilon \geq 2 + \epsilon$. Similarly, if $e_4$ transfers a charge of at least $1 - 2\epsilon$ to $y_u$, and $e_5$ transfers a charge of at least $1 - 2\epsilon$, to $y_v$, then $y_u + y_v \geq 3 - 4\epsilon \geq 2 + \epsilon$.
4. Suppose $e$ is a “bad” edge that belongs to two matchings. Then $i = 0$ from our observations that no internal edge can be “bad”. This implies that there is an edge $e_1$ from $v$ to a vertex of higher rank, which belongs to a single matching. The edge $e$ transfers a charge of 2 to $y_v$, and edge $e_1$ transfers a charge of $\epsilon$ to $y_v$. Thus, $y_u + y_v = 2 + \epsilon$.

5. 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 lower ranked vertex to $u$, which belongs to at least one matching, or there is an edge from some lower rank vertex to $v$ that belongs to at least one matching, or there is an edge from $v$ to some higher rank vertex which belongs to two matchings. The edge $e$ transfers a charge of 2 among $y_u$ and $y_v$. Either of the neighboring edges transfer a charge of $\epsilon$ to $y_u$ or $y_v$ (depending on which vertex it is incident), to give $y_u + y_v \geq 2 + \epsilon$.

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

From the above conditions, the best value for the competitive ratio is obtained when $\epsilon = \frac{1}{7}$. So, $\mathbb{E}[y_u + y_v] \geq \frac{15}{28}$.

The above claim implies that the competitive ratio of the algorithm is at most $\frac{28}{15}$.

4 Lower Bounds

4.1 Lower Bound for MWM

In this section, we prove a lower bound on the competitive ratio of a natural class of randomized algorithms in the online preemptive model for MWM. The algorithms in this class, which we call local algorithms, have the property that their decision about whether to accept or reject a new edge is completely determined by the weights of the new edge and the conflicting edges in the matching maintained by the algorithm. Indeed, the optimal deterministic algorithm by McGregor [5] is a local algorithm. The notion of locality can be extended to randomized algorithms as well. In case of randomized local algorithms, the event that a new edge is accepted is independent of all such previous events, given the current matching maintained by the algorithm. Furthermore, the probability of this event is completely determined by the weight of the new edge and the conflicting edges in the matching maintained by the algorithm. Given that the optimal $(3 + 2\sqrt{2})$-competitive deterministic algorithm for MWM is a local algorithm, it is natural to ask whether randomized local algorithms can beat the deterministic lower bound of $(3 + 2\sqrt{2})$ by Varadaraja [7]. We answer this question in the negative, and prove the following theorem.

**Theorem 4.** No randomized local algorithm for the MWM problem can have a competitive ratio less than $\alpha = 3 + 2\sqrt{2} \approx 5.828$.

Note that the randomized algorithm by Epstein et. al. [2] does not fall in this category, since the decision of accepting or rejecting a new edge is also dependent on the outcome of the coins tossed at the beginning of the run of the algorithm. (For details, see Section 3 of [2].) In order to prove Theorem 4, we will crucially use the following lemma, which is a consequence of Section 4 of [7].

**Lemma 5.** If there exists an infinite sequence $(x_n)_{n \in \mathbb{N}}$ of positive real numbers such that for all $n$, $\beta x_n \geq \sum_{i=1}^{n+1} x_i + x_{n+1}$, then $\beta \geq 3 + 2\sqrt{2}$.

4.1.1 Characterization of local randomized algorithms

Suppose, for a contradiction, that there exists a randomized local algorithm $A$ with a competitive ratio $\beta < \alpha = 3 + 2\sqrt{2}$, $\beta \geq 1$. Define the constant $\gamma$ to be

$$\gamma = \frac{\beta (1 - \frac{1}{\alpha})}{(1 - \frac{\beta}{\alpha})} = \frac{\beta (\alpha - 1)}{\alpha - \beta} \geq 1 > \frac{1}{\alpha}$$

For $i = 0, 1, 2$, if $w$ is the weight of a new edge and it has $i$ conflicting edges, in the current matching, of weights $w_1, \ldots, w_i$, then $f_i(w_1, \ldots, w_i, w)$ gives the probability of switching to the new edge. The behavior of $A$ is completely described using the three functions. The following lemmas are easily proven.
Lemma 6. For every $w > 0$, $f_0(w) > 1/\alpha$.

Proof. If not, then a single edge of weight $w$ results in algorithm’s expected cost $w f_0(w) \leq w/\alpha < w/\beta$, whereas the optimum is $w$. This contradicts $\beta$-competitiveness. 

Lemma 7. For every $w_1$ and $w \leq w_1/\alpha$, $f_1(w_1, w) = 0$.

Proof. If $f_1(w_1, w) > 0$ for some $w_1$ and $w$ such that $w \leq w_1/\alpha$, then the adversary’s input is a star, with a single edge of weight $w_1$ followed by a large number $n$ of edges of weight $w$. Regardless of whether the first edge of weight $w_1$ is accepted or not, the algorithm holds an edge of weight $w$, with probability approaching 1 as $n \to \infty$, in the end. The optimum is $w_1 \geq \alpha w > \beta w$, thus, contradicting $\beta$-competitiveness.

Lemma 8. For every $w_1$, and $w \geq \gamma w_1$, $f_1(w_1, w) \geq 1/\alpha$.

Proof. Suppose $f_1(w_1, w) < 1/\alpha$ for some $w_1$ and $w$. This is a contradiction.

Lemma 9. For every $\delta \in (0, 1/\alpha)$, $e > 0$, and $w_1$, there exist $x$ and $y$ such that $f_1(w_1, x) \geq \delta$, $f_1(w_1, y) \leq \delta$, $x - y \leq \epsilon$, and $w_1/\alpha \leq y \leq x \leq \gamma w_1$.

Proof. By Lemma 7, $f_1(w_1, w_1/\alpha) = 0$, and by Lemma 8, $f_1(w_1, \gamma w_1) \geq 1/\alpha$. Take a finite sequence of points, increasing from $w_1/\alpha$ to $\gamma w_1$, such that for any two consecutive points in the sequence, the difference is at most $\epsilon$, and observe the value of $f_1(w_1, z)$ at each such point $z$. Since $f_1(w_1, w_1/\alpha) < \delta$ and $f_1(w_1, \gamma w_1) > \delta$, there must exist two consecutive points in the sequence, say $y$ and $x$, such that $f_1(w_1, y) \leq \delta$ and $f_1(w_1, x) \geq \delta$. Furthermore, $x - y \leq \epsilon$ and $w_1/\alpha \leq y \leq x \leq \gamma w_1$, by construction.

4.1.2 The adversarial input

The adversarial input is parameterized by four parameters: $\delta \in (0, 1/\alpha)$, $e > 0$, $m$, and $n$, where $m$ and $n$ determine the graph and $\delta$ and $e$ determine the weights of its edges.

Define the infinite sequences $(x_i)_{i \in \mathbb{N}}$ and $(y_i)_{i \in \mathbb{N}}$, as functions of $\epsilon$ and $\delta$, as follows. $x_1 = 1$, and for all $i$, having defined $x_i$, let $x_{i+1}$ and $y_i$ be such that $f_1(x_i, x_{i+1}) \geq \delta$, $f_1(x_i, y_i) \leq \delta$, $x_{i+1} - y_i \leq \epsilon$, and $x_{i+1}/\alpha \leq y_i \leq x_{i+1} \leq \gamma x_i$. Lemma 8 ensures that such $x_{i+1}$ and $y_i$ exist. Furthermore, by induction on $i$, it is easy to see that for all $i$,

$$1/\alpha^i \leq y_i \leq x_{i+1} \leq \gamma^i \tag{1}$$

These sequences will be the weights of the edges in the input graph.

Given $m$ and $n$, the input graph contains several layers of vertices, namely $A_1, A_2, \ldots, A_{n+1}, A_{n+2}$ and $B_1, B_2, \ldots, B_{n+1}$; each layer containing $m$ vertices. The vertices in the layer $A_i$ are named $a_{i1}, a_{i2}, \ldots, a_{im}$, and those in layer $B_i$ are named analogously. We have a complete bipartite graph $K_i$ between layer $A_i$ and $A_{i+1}$ and an edge between $a_{ij}$ and $b_{ij}$ for every $i, j$ (that is, a matching $M_i$ between $A_i$ and $B_i$).

For $i = 1$ to $n$, the edges $\{(a_{ij}, a_{ij+1}) | 1 \leq j, j' \leq m\}$ in the complete bipartite graph between $A_i$ and $A_{i+1}$, have weight $x_i$, and the edges $\{(a_{ij}', b_{ij+1}) | 1 \leq j \leq m\}$, in the matching between $A_i$ and $B_i$, have weight $y_i$. The edges in the complete graph $K_{n+1}$ have weight $x_n$, and those in the matching $M_{n+1}$ have weight $y_n$. Note that any weight $x_i$, $y_i$ depends on $\epsilon$ and $\delta$, but is independent of $m$ and $n$. Clearly, the weight of the maximum weight matching in this graph is bounded from below by the weight of the matching $\bigcup_{i=1}^{n+1} M_i$. Since $y_i \geq x_{i+1} - \epsilon$, we have

$$\text{OPT} \geq m \left( \sum_{i=1}^{n} y_i + y_n \right) \geq m \left( \sum_{i=2}^{n+1} x_i + x_{n+1} - (n + 1)\epsilon \right) \tag{2}$$

9
The edges of the graph are revealed in $n + 1$ phases. In the $i$th phase, the edges in $K_i \cup M_i$ are revealed as follows. The phase is divided into $m$ sub phases. In the $j$th sub phase of the $i$th phase, edges incident on $a_j^i$ are revealed, in the order $(a_j^i, a_{j+1}^i), (a_j^i, a_{j+1}^i), \ldots, (a_j^i, a_m^i), (a_j^i, b_j^i).

4.1.3 Analysis of lower bound

The overall idea of bounding the weight of the algorithm’s matching is as follows. In each phase $i$, we will prove that as many as $m - O(1)$ edges of $K_i$ and only $\delta m + O(1)$ edges of $M_i$ are picked by the algorithm. Furthermore, in the $i + 1$st phase, since $m - O(1)$ edges from $K_{i+1}$ are picked, all but $O(1)$ edges of the edges picked from $K_i$ are preempted out. Thus, the algorithm ends up with $\delta m + O(1)$ edges from each $M_i$, and $O(1)$ edges from each $K_i$, except possibly $K_n$ and $K_{n+1}$. The algorithm can end up with at most $m$ edges from $K_n \cup K_{n+1}$, since the size of the maximum matching in $K_n \cup K_{n+1}$ is $m$. Thus, the weight of the algorithm’s matching is at most $m x_n$ plus a quantity that can be neglected for large $m$ and small $\delta$.

Let $X_i$ (resp. $Y_i$) be the set of edges of $K_i$ (resp. $M_i$) held by the algorithm at the end of input. Let us estimate the expected sizes of these sets.

**Lemma 10.** For every $i, j$, the probability that $a_j^i$ is not matched to any vertex in $A_{i+1}$, in the $j$th sub phase of the $i$th phase, just before the edge $(a_j^i, b_j^i)$ is revealed, is at most $(1 - \delta)^{m-j+1}$. 

**Proof.** Consider the $j$th sub phase of the $i$th phase, in which, the edges $(a_j^i, a_{j+1}^i), (a_j^i, a_{j+2}^i), \ldots, (a_j^i, a_m^i), (a_j^i, b_j^i)$ are revealed. Before this sub phase, the number of unmatched vertices in $A_{i+1}$ must be at least $m - j + 1$. Call this set $A'$. If $a_j^i$ was matched at the end of phase $i - 1$, then the weight of edge incident on $a_j^i$, at the beginning of the current phase, is $x_{i-1}$. For each vertex $a_j^{i+1} \in A'$, given that $a_j^i$ did not get matched to any of $a_{j+1}^i, a_{j+2}^i, \ldots, a_{j-1}^i$, the probability that $a_j^i$ gets matched to $a_j^{i+1}$ is $f_i(x_{i-1}, x_i) \geq \delta$. Thus, the probability of $a_j^i$ not being matched to any vertex in $A' \subseteq A_{i+1}$, in the current sub phase, is at most $(1 - \delta)^{m-j+1}$. Note that this argument applies even if $a_j^i$ was not matched at the beginning of the current phase, due to Lemma 8 and since $\delta < 1/\alpha < f_0(x_i)$. \qed

**Lemma 11.** For all $i = 1$ to $n$

$E[|Y_i|] \leq \delta m + \frac{1 - \delta}{\delta}$

**Proof.** First, observe that the sequence in which the edges are revealed ensures that no edge adjacent to any edge $e \in M_i$ appears after $e$. Thus, if $e$ is picked when it is revealed, it is never preempted, and is maintained till the end of input. Hence, $Y_i$ is also the set of edges of $M_i$ that were picked as soon as they were revealed.

When the edge $(a_j^i, b_j^i)$ is given, the algorithm picks it with probability at most $\delta$ (since $f_i(x_i, y_i) \leq \delta$) if $a_j^i$ was matched to some vertex in $A_{i+1}$. By Lemma 10, the probability of $a_j^i$ not being matched to any vertex in $A_{i+1}$ is at most $(1 - \delta)^{m-j+1}$. Thus, the probability that the edge $(a_j^i, b_j^i)$ appears in $Y_i$ is at most $\delta + (1 - \delta)^{m-j+1}$. Hence, $E[|Y_i|] \leq \delta m + \sum_{j=1}^{m}(1 - \delta)^{m-j+1} \leq \delta m + (1 - \delta)/\delta$. \qed

**Lemma 12.** For all $i = 1$ to $n - 1$

$E[|X_i|] \leq \frac{1 - \delta}{\delta}$

**Proof.** Consider the set $A'$ of all vertices $a_{j+1}^{i+1}$, which remain matched to some vertex in $A_i$ at the end of input. Then clearly, $|A'| = |X_i|$. Let us find the probability that a vertex $a_{j+1}^{i+1}$ appears in $A'$. For this to happen, it is necessary that $a_{j+1}^{i+1}$ not be matched to any vertex in $A_{i+2}$, in the $j$th sub phase of the $i + 1$st phase. By lemma 11, this happens with probability at most $(1 - \delta)^{m-j+1}$. Thus, $E[|X_i|] = \sum_{j=1}^{m}(1 - \delta)^{m-j+1} \leq (1 - \delta)/\delta$. \qed

**Lemma 13.** For every $j$, the probability that $a_j^{n+1}$ is not matched to any vertex in $A_n \cup A_{n+2}$, in the $j$th sub phase of the $n + 1$st phase, just before the edge $(a_j^{n+1}, b_j^{n+1})$ is revealed, is at most $(1 - \delta)^{m-j+1}$. 

10
Proof. This proof is analogous to the proof of Lemma 14. If \( a_{j}^{n+1} \) was matched to some vertex in \( A_n \) at the end of the \( n^{th} \) phase, then it will continue to remain matched to some vertex in \( A_n \cup A_{n+2} \), until the edge \( (a_{j}^{n+1}, b_{j}^{n+1}) \) is revealed. Otherwise \( a_{j}^{n+1} \) will get matched to some vertex in \( A_{n+2} \) with probability at least \( 1 - (1 - \delta)^{m-j+1} \), and remain unmatched with probability at most \( (1 - \delta)^{m-j+1} \). \( \square \)

**Lemma 14.**

\[
E[|Y_{n+1}|] \leq \delta m + \frac{1 - \delta}{\delta}
\]

**Proof.** This proof is analogous to the proof of Lemma 14. Again, the sequence in which the edges are revealed ensures that no edge adjacent to any edge \( e \) in any \( M_{n+1} \) appears after \( e \). Thus, if \( e \) is picked when it is revealed, it is never preempted. Hence, \( Y_i \) is also the set of edges of \( M_i \) that were picked as soon as they were revealed.

When the edge \( (a_{j}^{n+1}, b_{j}^{n+1}) \) is given, the algorithm picks it with probability at most \( \delta \) (since \( f_i(x_n, y_n) \leq \delta \)) if \( a_{j}^{n+1} \) was matched to some vertex in \( A_n \cup A_{n+1} \). Thus, the probability that the edge \( (a_{j}', b_{j}') \) appears in \( Y_{n+1} \) is at most \( \delta + (1 - \delta)^{m-j+1} \). Hence, \( E[|Y_{n+1}|] \leq \delta m + \sum_{j=1}^{m} (1 - \delta)^{m-j+1} \leq \delta m + (1 - \delta)/\delta \). \( \square \)

We are now ready to prove Theorem 4. The expected weight of matching held by \( \mathcal{A} \) is

\[
E[\text{ALG}] \leq \sum_{i=1}^{n} y_i E[|Y_i|] + y_n E[|Y_{n+1}|] + \sum_{i=1}^{n-1} x_i E[|X_i|] + x_n E[|X_n \cup X_{n+1}|]
\]

Using Lemmas 11, 13, 12 and the facts that \( y_i \leq x_{i+1} \) for all \( i \) and \( E[|X_n \cup X_{n+1}|] \leq m \) (since \( X_n \cup X_{n+1} \) is a matching in \( K_n \cup K_{n+1} \)), we have

\[
E[\text{ALG}] \leq \left( \delta m + \frac{1 - \delta}{\delta} \right) \left( \sum_{i=2}^{n+1} x_i + x_{n+1} \right) + \frac{1 - \delta}{\delta} \sum_{i=1}^{n-1} x_i + mx_n
\]

Since the algorithm is \( \beta \)-competitive, for all \( n, m, \delta \) and \( \epsilon \) we must have \( E[\text{ALG}] \geq \text{OPT} / \beta \). From the above and equation (2), we must have

\[
\left( \delta m + \frac{1 - \delta}{\delta} \right) \left( \sum_{i=2}^{n+1} x_i + x_{n+1} \right) + \frac{1 - \delta}{\delta} \sum_{i=1}^{n-1} x_i + mx_n \geq \frac{m}{\beta} \left( \sum_{i=2}^{n+1} x_i + x_{n+1} - (n+1) \epsilon \right)
\]

Since the above holds for arbitrarily large \( m \), ignoring the terms independent of \( m \) (recall that \( x_i \)'s are functions of \( \epsilon \) and \( \delta \) only), we have for all \( \delta \) and \( \epsilon \),

\[
\delta \left( \sum_{i=2}^{n+1} x_i + x_{n+1} \right) + x_n \geq \frac{1}{\beta} \left( \sum_{i=2}^{n+1} x_i + x_{n+1} - (n+1) \epsilon \right)
\]

that is,

\[
x_n \geq \frac{1}{\beta} \left( \sum_{i=2}^{n+1} x_i + x_{n+1} - (n+1) \epsilon \right) - \delta \left( \sum_{i=2}^{n+1} x_i + x_{n+1} \right)
\]

Taking limit inferior as \( \delta \to 0 \) in the above inequality, and noting that limit inferior is super-additive we get for all \( \epsilon \),

\[
\liminf_{\delta \to 0} x_n \geq \frac{1}{\beta} \left( \sum_{i=2}^{n+1} \liminf_{\delta \to 0} x_i + \liminf_{\delta \to 0} x_{n+1} - (n+1) \epsilon \right) - \limsup_{\delta \to 0} \delta \left( \sum_{i=2}^{n+1} x_i + x_{n+1} \right)
\]

Recall that \( x_i \)'s are functions of \( \epsilon \) and \( \delta \), and that from equation (1), \( 1/\alpha' \leq x_{i+1} \leq \gamma' \), where the bounds are independent of \( \delta \). Thus, all the limits in the above inequality exist. Moreover, \( \lim_{\delta \to 0} \delta \left( \sum_{i=2}^{n+1} x_i + x_{n+1} \right) \) exists and is 0, for all \( \epsilon \). This implies \( \limsup_{\delta \to 0} \delta \left( \sum_{i=2}^{n+1} x_i + x_{n+1} \right) = 0 \) and we get for all \( \epsilon \),

\[
\liminf_{\delta \to 0} x_n \geq \frac{1}{\beta} \left( \sum_{i=2}^{n+1} \liminf_{\delta \to 0} x_i + \liminf_{\delta \to 0} x_{n+1} - (n+1) \epsilon \right)
\]
Again, taking limit inferior as $\epsilon \to 0$, and using super-additivity,
\[
\liminf_{\epsilon \to 0} \liminf_{\delta \to 0} x_n \geq \frac{1}{\beta} \sum_{i=2}^{n+1} \liminf_{\epsilon \to 0} \liminf_{\delta \to 0} x_i + \liminf_{\epsilon \to 0} \liminf_{\delta \to 0} x_{n+1}
\]
Note that the above holds for all $n$. Finally, let $\overline{x_n} = \liminf_{\epsilon \to 0} \liminf_{\delta \to 0} x_{n+1}$. Then we have the infinite sequence $(\overline{x_n})_{n \in \mathbb{N}}$ such that for all $n$, $\delta \overline{x_n} \geq \sum_{i=1}^{n+1} x_i + \overline{x_{n+1}}$. Thus, by Lemma 9 we have $\beta \geq 3 + 2\sqrt{2}$.

4.2 Lower Bound for $\theta$ structured graphs

Recall that an edge weighted graph is said to be $\theta$-structured if the weights of the edges are powers of $\theta$. The following bound applies to any deterministic algorithm for MWM on $\theta$-structured graphs.

**Theorem 5.** No deterministic algorithm can have a competitive ratio less than $2 + \frac{2}{\theta - 2}$ for MWM on $\theta$-structured graphs, for $\theta \geq 4$.

The overall idea of the adversarial strategy is as follows. The input graph is a tree whose edges are partitioned into $n+1$ layers which are numbered 0 through $n$ from bottom to top. Every edge in layer $i$ has weight $\theta^i$. The edges are revealed bottom-up. The edges in layer $i$ are given in such a manner that all the edges in layer $i-1$ held by the algorithm will be preempted. This ensures that in the end, the algorithm’s matching contains only one edge, whereas the adversary’s matching contains $2^{n-i}$ edges from layer $i$, for each $i$.

Let $A$ be any deterministic algorithm for maximum matching in the online preemptive model. The adversarial strategy uses a recursive function, which takes $n \in \mathbb{N}$ as a parameter. For a given $n$, this recursive function, given by Algorithm 3, constructs a tree with $n+1$ layers by giving weighted edges to the algorithm in an online manner, and returns the tree, the adversary’s matching in the tree, and a vertex from the tree.

Let us prove a couple of properties about the behavior of the algorithm and the adversary, when the online input is generated by the call $\text{MakeTree}(n)$.

**Lemma 15.** Suppose that the call $\text{MakeTree}(n)$ returns $(T', M', v')$. Then
1. $M'$ is a matching in $T'$.
2. $M'$ does not cover the vertex $v'$.
3. The weight of $M'$ is $\sum_{i=0}^{n} \theta^i 2^{n-i} = (\theta^{n+1} - 2^{n+1})/(\theta - 2)$.

**Proof.** By induction on $n$. For $n = 0$, the claim is obvious from the description of $\text{MakeTree}$. Assume that the claim holds for $n - 1$, and consider the call $\text{MakeTree}(n)$, which returns $(T', M', v')$. Then, by induction hypothesis, the two recursive calls must have returned $(T_1, M_1, v_1)$ and $(T_2, M_2, v_2)$ satisfying the conditions of the lemma. Suppose the algorithm replaced $(v_1, v_2)$ by $(v, v_1)$ in its matching. Since $v_1$ and $v_2$ were respectively left uncovered by $M_1$ and $M_2$, $M = M_1 \cup M_2 \cup \{(v_1, v_2)\}$ is a matching in $T$, and $M'$ does not cover $v' = v$. The case when the algorithm did not replace $(v_1, v_2)$ by $(v, v_1)$ is analogous. In either case, the additional edge in $M'$, apart from edges in $M_1$ and $M_2$ has weight $\theta^n$, and $M_1, M_2$ themselves have weight $\sum_{i=0}^{n-1} \theta^i 2^{n-1-i}$, by induction hypothesis. Thus, the weight of $M'$ is $\theta^n + 2 \sum_{i=0}^{n-1} \theta^i 2^{n-1-i} = \sum_{i=0}^{n} \theta^i 2^{n-i}$.

**Lemma 16.** When the call $\text{MakeTree}(n)$ returns $(T, M, v)$, the algorithm’s matching contains exactly one edge from $T$. This edge is incident on $v$ and has weight $\theta^n$.

**Proof.** By induction on $n$. Again, the claim is obvious for $n = 0$. Assume that the claim holds for $n - 1$, and consider the call $\text{MakeTree}(n)$, which returns $(T', M', v')$. At the end of the two recursive calls which return $(T_1, M_1, v_1)$ and $(T_2, M_2, v_2)$. The algorithm will have exactly one edge $e_1$ from $T_1$ incident on $v_1$, and one edge $e_2$ from $T_2$ incident on $v_2$, by induction hypothesis. If the algorithm does not pick the next edge $(v_1, v_2)$, then the tree is discarded. If the algorithm picks that edge, then it must preempt $e_1$ and $e_2$. Therefore, if the algorithm replaces $(v_1, v_2)$ by $(v, v_1)$ in its matching, then $v' = v$. Otherwise, if the algorithm keeps $(v_1, v_2)$, then $v' = v_2$. In either case, the algorithm is left with exactly one edge, which is incident on $v'$, and which has weight $\theta^n$. 

12
Algorithm 4 MakeTree($n$)

1: if $n = 0$ then
2:   while true do
3:     Take fresh vertices $v, v_1, v_2$, and give the edges $(v_1, v_2), (v, v_1)$ with weight 1.
4:     $T := \{(v_1, v_2), (v, v_1)\}$.
5:     if algorithm picks $(v_1, v_2)$ then
6:       return $(T, \{(v, v_1)\}, v_2)$
7:     else if algorithm picks $(v, v_1)$ then
8:       return $(T, \{(v_1, v_2)\}, v)$
9:     else
10:        Discard $T$ and retry.
11:   end if
12: end while
13: else
14:   while true do
15:     $(T_1, M_1, v_1) :=$ MakeTree($n - 1$)
16:     $(T_2, M_2, v_2) :=$ MakeTree($n - 1$)
17:     Give the edge $(v_1, v_2)$ with weight $\theta^n$.
18:     if algorithm picks $(v_1, v_2)$ then
19:       Take a fresh vertex $v$, and give the edge $(v, v_1)$ with weight $\theta^n$.
20:       $T := T_1 \cup T_2 \cup \{(v_1, v_2), (v, v_1)\}$.
21:       if algorithm replaces $(v_1, v_2)$ by $(v, v_1)$ then
22:         return $(T, M_1 \cup M_2 \cup \{(v_1, v_2)\}, v)$.
23:       else
24:         return $(T, M_1 \cup M_2 \cup \{(v, v_1)\}, v_2)$
25:       end if
26:     else
27:       {algorithm does not pick $(v_1, v_2)$}
28:       Discard the constructed tree and retry.
29:     end if
30:   end while
31: end if
The adversary’s strategy is given by Algorithm 5, where $n \geq 1$ is a parameter.

**Lemma 17.** When a tree $T$ is discarded in a call to $\text{MAKE TREE}(n)$ or $\text{ADV}(n)$, $\text{ALG}(T) \leq \left( 2 + \frac{2}{\theta - 2} \right) \cdot \text{ADV}(T)$, where $\text{ALG}(T)$ and $\text{ADV}(T)$ are respectively the total weights of the edges of the algorithm’s and the adversary’s matchings, in $T$.

**Proof.** For $n \geq 1$, consider the two calls to $\text{MAKE TREE}$, which returned $(T_1, M_1, v_1)$ and $(T_2, M_2, v_2)$ before the edge $(v_1, v_2)$ is revealed. By Lemma 10, the algorithm had exactly one edge in each of $T_1$ and $T_2$, and this edge had weight $\theta^{n-1}$. The tree was discarded because the algorithm did not pick the edge $(v_1, v_2)$. Thus, $\text{ALG}(T) = 2\theta^{n-1}$. On the other hand, the adversary picks the matching $M_1 \cup M_2 \cup \{(v_1, v_2)\}$, which, by Lemma 10 has weight $\text{ADV}(T) = \theta^n + 2 \sum_{i=0}^{n-1} \theta^i 2^{n-1-i} = \sum_{i=0}^{n} \theta^i 2^n - i = \left( \theta^{n+1} - 2^n \right) / (\theta - 2)$. Thus,

$$\frac{\text{ADV}(T)}{\text{ALG}(T)} = \frac{\theta^{n+1} - 2^n}{2^n \theta - 2} = \frac{\theta}{2} \cdot \frac{1 - \left( \frac{2}{\theta} \right)^{n+1}}{1 - \frac{2}{\theta}} \geq \frac{\theta}{2} \cdot \frac{1 - \left( \frac{2}{\theta} \right)^2}{1 - \frac{2}{\theta}} = \frac{\theta}{2} \left( 1 + \frac{2}{\theta} \right) = \frac{\theta}{2} + 1 \geq \left( 2 + \frac{2}{\theta - 2} \right)$$

The last inequality follows from the fact that $\theta \geq 4$. Finally, note that when the discard happens in a call to $\text{MAKE TREE}(0)$, $\text{ALG}(T) = 0$ and $\text{ADV}(T) = 1$.

Now we are ready to prove Theorem 5.

**Proof of Theorem 5.** For $n \geq 1$, give the adversarial input by calling $\text{ADV}(n)$. If the call does not terminate, then an unbounded number of trees are discarded, and by Lemma 17 a lower bound of $\left( 2 + \frac{2}{\theta - 2} \right)$ is forced on each discarded tree. If the call terminates, then suppose $T$ is the final tree constructed. Let $(T_1, M_1, v_1)$ and $(T_2, M_2, v_2)$ be returned by the two calls to $\text{MAKE TREE}(n - 1)$. By the description of $\text{ADV}$ and Lemma 16 it is clear that the algorithm holds only one edge of $T$ in the end, and this edge has weight $\theta^n = \text{ALG}(T)$. On the other hand, the adversary’s matching contains $M_1$ and $M_2$, and two edges of weight $\theta^n$, where by Lemma 15, the weight of $M_1$ and $M_2$ is $(\theta^n - 2^n) / (\theta - 2)$ each. Thus, $\text{ADV}(T) = 2\theta^n + 2(\theta^n - 2^n) / (\theta - 2)$. Therefore,

$$\frac{\text{ADV}(T)}{\text{OPT}(T)} = 2 + \frac{2(\theta^n - 2^n)}{\theta^n (\theta - 2)} = 2 + \frac{2(1 - \frac{2}{\theta})^n}{\theta - 2}$$

14
This approaches \(2 + \frac{2}{\sqrt[2]{2}}\) as \(n \to \infty\). Furthermore, this lower bound is also forced on the trees discarded during the execution of \(\text{ADV}(n)\). Thus, the algorithm can not have a competitive ratio less than \(2 + \frac{2}{\sqrt[2]{2}}\). ⊓⊔

### 5 Randomized Algorithm for Paths

When the input graph is restricted to be a collection of paths, then every new edge that arrives connects two (possibly empty) paths. Our algorithm consists of several cases, depending on the lengths of the two paths.

**Algorithm 6 Randomized Algorithm for Paths**

1. \(M = \emptyset\). (\(M\) is the matching stored by the algorithm.)
2. **for** each new edge \(e\) **do**
3. \(\text{Let } L_1 \geq L_2 \text{ be the lengths of the two (possibly empty) paths } P_1, P_2 \text{ that } e \text{ connects.}\
4. **if** \(L_1 > 0\) (resp. \(L_2 > 0\)), let \(e_1\) (resp. \(e_2\)) be the edge on \(P_1\) (resp. \(P_2\)) adjacent to \(e\).
5. **if** \(e\) is a disjoint edge \(\{L_1 = L_2 = 0\}\) **then**
   \(M = M \cup \{e\}\).
6. **else if** \(e\) is revealed on a disjoint edge \(e_1\) \(\{L_1 = 1, L_2 = 0. e_1 \in M\}\) **then**
   \(M = M \setminus \{e_1\} \cup \{e\}\).
7. **else if** \(e\) is revealed on a end point of path of length \(> 1\) \(\{L_1 > 1, L_2 = 0\}\) **then**
   \(M = M \setminus \{e\}\).
8. **else if** \(e\) joins two disjoint edges \(\{L_1 = L_2 = 1. e_1, e_2 \in M\}\) **then**
   \(M = M \setminus \{e_1, e_2\} \cup \{e\}\).
9. **else if** \(e\) joins a path and a disjoint edge \(\{L_1 > 1, L_2 = 1. e_2 \in M\}\) **then**
   \(M = M \setminus \{e_2\} \cup \{e\}\).
10. **else if** \(e\) joins two paths of length \(> 1\) \(\{L_1 > 1, L_2 > 1\}\) **then**
    \(M = M \setminus \{e\}\).
11. **end if**
12. **end for**

Following simple observations can be made by looking at the algorithm:

- All isolated edges belong to \(M\) with probability one.
- The end vertex of any path of length \(> 1\) is covered by \(M\) with probability \(\frac{1}{2}\), and this is independent of the end vertex of any other path being covered.
- For a path of length 2, 3, or 4, each maximal matching is present in \(M\) with probability \(\frac{1}{2}\).

**Theorem 6.** The randomized algorithm for finding MCM on path graphs is \(\frac{4}{3}\)-competitive.

Theorem 6 can be proved using the following lemma.

**Lemma 18.** For any (maximal) path \(P\) of length \(n > 0\),

- if \(n\) is even then \(\mathbb{E}[|M \cap P|] \geq (3/4)(n/2) + 1/4 = p_0(n)\) (say).
- if \(n\) is odd then \(\mathbb{E}[|M \cap P|] \geq (3/4)(n/2) + 3/8 = p_1(n)\) (say).

**Proof.** For \(n = 1\) and \(n = 2\), \(\mathbb{E}[|M \cap P|] = 1\), and for \(n = 3\), \(\mathbb{E}[|M \cap P|] = 3/2\). Thus the lemma holds when \(n \leq 3\). We will induct on the number of edges in the input. (Case \(n = 1\) covers the base case.) Suppose the lemma is true before the arrival of the new edge \(e\). We prove that the lemma holds even after \(e\) has been processed. We may assume that the length \(n\) of the new path \(P\) resulting from addition of \(e\) is at least 4.
1. If \( n \) is even and \( L_2 = 0 \), (therefore \( L_1 \) is odd, and \( L_1 \geq 3 \), \( \Pr[e_1 \notin M] = \frac{1}{4} \). Therefore, \( e \) is added to \( M \) with probability \( \frac{1}{2} \).

\[ \mathbb{E}[|M \cap P|] \geq p_0(n - 1) + \frac{1}{2} \geq p_0(n) \]

2. If \( n \) is even, \( L_1 = n - 2 \) and \( L_2 = 1 \).

\[ \mathbb{E}[|M \cap P|] \geq p_0(n - 2) + 1 = \frac{3}{4} \left( \frac{n - 2}{2} \right) + 1 + 1 \geq p_0(n) \]

3. If \( n \) is even, and \( L_2 > 1 \), where \( n = L_1 + L_2 + 1 \), \( L_1 \) is even, and \( L_2 \) is odd. \( \Pr[e_1 \notin M, e_2 \notin M] = \frac{1}{4} \)

\[ \mathbb{E}[|M \cap P|] \geq p_0(L_1) + p_1(L_2) + \frac{1}{4} = \frac{3}{4} \left( \frac{L_1 + L_2 + 1}{2} \right) + \frac{1}{4} + \frac{3}{8} - \frac{3}{8} + \frac{1}{4} \geq p_0(n) \]

4. If \( n \) is odd, and \( L_2 = 0 \), (therefore \( L_1 \) is even, and \( L_1 \geq 3 \), \( \Pr[e_1 \notin M] = \frac{1}{4} \). Therefore, \( e \) is added to \( M \) with probability \( \frac{1}{2} \).

\[ \mathbb{E}[|M \cap P|] \geq p_0(n - 1) + \frac{1}{2} = p_1(n) \]

5. If \( n \) is odd, \( L_1 = n - 2 \) and \( L_2 = 1 \).

\[ \mathbb{E}[|M \cap P|] \geq p_1(n - 2) + 1 = \frac{3}{4} \left( \frac{n - 2}{2} \right) + \frac{3}{8} + 1 \geq p_1(n) \]

6. If \( n \) is odd, and \( L_2 > 1 \), where \( n = L_1 + L_2 + 1 \), \( L_1 \) is even, and \( L_2 \) is even. \( \Pr[e_1 \notin M, e_2 \notin M] = \frac{1}{4} \)

\[ \mathbb{E}[|M \cap P|] \geq p_0(L_1) + p_0(L_2) + \frac{1}{4} = \frac{3}{4} \left( \frac{L_1 + L_2 + 1}{2} \right) + \frac{1}{4} + \frac{1}{4} - \frac{3}{8} + \frac{1}{4} = p_1(n) \]

7. If \( n \) is odd, and \( L_2 > 1 \), where \( n = L_1 + L_2 + 1 \), \( L_1 \) is odd, and \( L_2 \) is odd. \( \Pr[e_1 \notin M, e_2 \notin M] = \frac{1}{4} \)

\[ \mathbb{E}[|M \cap P|] \geq p_1(L_1) + p_1(L_2) + \frac{1}{4} = \frac{3}{4} \left( \frac{L_1 + L_2 + 1}{2} \right) + \frac{3}{8} - \frac{3}{8} + \frac{1}{4} \geq p_1(n) \]

This completes the induction and hence implies a \( \frac{3}{4} \)-competitive ratio for this algorithm. \( \square \)
Acknowledgment

The first author thanks Sagar Kale for reviewing Section 4.1, correcting several mistakes and helping us improve the presentation substantially.

References

[1] Niv Buchbinder and Joseph Naor. The design of competitive online algorithms via a primal-dual approach. Foundations and Trends in Theoretical Computer Science, 3(2-3):93–263, 2009.

[2] Leah Epstein, Asaf Levin, Danny Segev, and Oren Weimann. Improved bounds for online preemptive matching. In STACS, pages 389–399, 2013.

[3] 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.

[4] 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.

[5] Andrew McGregor. 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, pages 170–181, Berlin, Heidelberg, 2005. Springer-Verlag.

[6] Matthias Poloczek and Mario Szegedy. Randomized greedy algorithms for the maximum matching problem with new analysis. In 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012, pages 708–717, 2012.

[7] Ashwinkumar Badanidiyuru Varadaraja. Buyback problem - approximate matroid intersection with cancellation costs. In ICALP (1), pages 379–390, 2011.

[8] Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In Foundations of Computer Science, 1977., 18th Annual Symposium on, pages 222–227, Oct 1977.

Appendices

A Randomized Algorithm for Paths

Algorithm 7 Barely Random Algorithm for Paths

1. The algorithm maintains two matchings: $M_1$ and $M_2$.

2. On receipt of an edge $e$, the processing happens in two phases.
   (a) The augment phase. Here, the new edge $e$ is added to each $M_i$ such that there is no edge in $M_i$ sharing an end point with $e$.
   (b) The switching phase. Edge $e$ is added to $M_2$ and the conflicting edge is discarded, provided it decreases the quantity $|M_1 \cap M_2|$.

3. Output a matching $M_i$ with probability $\frac{1}{2}$. 

17
Claim 2. Dual constraint for each edge is satisfied at least \( \frac{2}{3}rd \) in expectation.

\( M_1 \) and \( M_2 \) are valid matchings and hence correspond to valid primal solutions. For each edge \( e \equiv (u, v) \) in some matching \( M_i \), we distribute a charge of \( x_e = 1 \) amongst dual variables \( y_u \) and \( y_v \) of its vertices. We prove that for each edge \( e \), \( y_u + y_v \geq \frac{2}{3} \). Thus, \( E[y_u + y_v] \geq \frac{2}{3} \). Hence, this algorithm has a competitive ratio \( \frac{2}{3} \). All the dual variables are initialized to 0. Suppose \( e \equiv (u, v) \in M_i \) for some \( i \in [2] \). Then distribution of primal charge \( x_e \) amongst dual variables \( y_u \) and \( y_v \) is done as follows. If there is an edge incident on \( u \) which does not belong to any matching, and there is an edge incident on \( v \) which does belong to some matching, then \( e \) transfer a primal charge of \( \frac{2}{3} \) to \( y_u \) and rest is transferred to \( y_v \). Else, the primal charge of \( e \) is transferred equally amongst \( y_u \) and \( y_v \).

We look at three cases and prove that \( y_u + y_v \geq \frac{4}{3} \) for each edge \( e \equiv (u, v) \).

1. The edge \( e \equiv (u, v) \) is not present in any matching.
   (a) If there are no edges on both its end points in the input graph, then this edge has to be covered by both \( M_1 \) and \( M_2 \). So this case is not possible.
   (b) If there is no edge on one end point (say \( u \)) in the input graph, then \( e \) has to belong to belong to some matching. So, this case is not possible.
   (c) If there are two edges incident on end points of \( e \) in the input graph, then each of them has to be covered by some matching. So, \( y_u + y_v \geq 2 \cdot \frac{2}{3} = \frac{4}{3} \).

2. The edge \( e \equiv (u, v) \) is present in a single matching.
   (a) If there are no edges on both its end points in the input graph, then this edge has to be covered by both \( M_1 \) and \( M_2 \). So this case is not possible.
   (b) If there is no edge on one end point (say \( u \)) in the input graph, then edge on the other end point must be covered by the other matching. Otherwise, edge \( e \) would have been covered by both matchings. So, \( y_u + y_v \geq 1 + \frac{1}{3} = \frac{4}{3} \).
   (c) If there are two edges incident on end points of \( e \) in the input graph, then at least one of them has to be covered by other matching. Else, edge \( e \) would have been covered by both matchings. So, \( y_u + y_v \geq 1 + \frac{1}{3} = \frac{4}{3} \).

3. The edge \( e \equiv (u, v) \) is present in both the matchings, then \( y_u + y_v = 2 \geq \frac{4}{3} \).

This proves the above claim. The corollary of the above claim is that we have a \( \frac{2}{3} \)-competitive randomized algorithm for the MCM on paths.

**Proof.** (of the second part of theorem [1]) Suppose \( U \) is the set of matchings used by a barely random algorithm \( A \). Following input is given to this algorithm. Reveal two edges \( x_1 \) and \( y_1 \) such that they share an end point. Let \( S \) be the set of matchings to which \( x_1 \) is added, and \( S' \) be the set of matchings to which \( y_1 \) is added. Here, \( U = S \cup S' \). Now give two more edges \( x_2 \) and \( y_2 \) disjoint from the previous edges, such that \( x_2 \) and \( y_2 \) share an end point. Wlog, \( x_2 \) will be added to set of matchings \( S \), and \( y_2 \) will be added to set of matchings \( S' \). Give an edge between \( y_2 \) and \( x_1 \). Continue the input similarly for \( i > 2 \). It can be seen that expected increase in the size of optimum matching is \( \frac{1}{3} \), whereas increase in the size of matching held by the algorithm is 1. Thus, we get a lower bound \( \frac{3}{2} \) on the competitive ratio of any barely random algorithm.

**B**

**Proof.** (of lemma [4]) Consider an edge \( (u, v) \) revealed at \( u \).

1. When revealed it is not put in any matching. This means that there are four covered edges incident on \( u \). (Call an edge covered if it belongs to some matching.) This situation cannot change as more edges are revealed. Thus the edge will remain covered by four matchings, and can never become a bad edge.
2. When revealed it is put in one matching. This means that there are three matching edges on at least two covered edges incident on \( u \). If there were three covered edges incident on \( u \) then they remain covered edges. So suppose otherwise. Then there are two covered edges of which one is in two matchings. Hence there will always be three matching edges covering \( u \). If an edge is revealed at \( v \) then there will be four matching edges covering the given edge. The edge may become bad if \( v \) stays a leaf and if one of the matchings on the edge with two of them, switches.

3. When revealed it is put in two matchings. Then there are two matching edges at \( u \) and at least one covered edge. If there are two covered edges, they remain so. Of the two copies of the edge in matchings, one may switch to a new edge but will always remain adjacent to this edge. Hence there will always be three matching edges covering \( u \). If an edge is revealed at \( v \) then there will be four matching edges covering the given edge. The edge may become bad if \( v \) stays a leaf and if one of the matchings on the edge with two of them, switches.

4. When revealed it is put in three matchings. Then there is one covered edge at \( u \). If one more edge is now revealed on \( u \), then we are back to case 3. If a new edge is revealed on \( v \), it replaces \((u,v)\) in one of the matchings. Now, even if more edges are revealed on either side of \((u,v)\), it continues to be covered by four matchings.

5. When revealed it is put in four matchings. If a new edge is revealed either on \( u \) or \( v \), then this case reduces to case 2.

This completes the proof of the first part of lemma.

For the second part of lemma, consider a leaf edge present on each of the vertices \( p, q, \) and \( r \). Suppose the leaf edge incident on \( q \) is bad. When this edge was revealed, there must have been some edge incident on \( q \), either \((p,q)\) or \((q,r)\), which belonged to two matchings. Wlog, assume \((p,q)\) belonged to two matchings. Then for a matching to switch out this edge, there need to be three edges incident on \( p \), and hence the leaf edge incident on \( p \) cannot be a bad edge.

\[ \square \]

\section*{C}

Consider input graph as a 4-regular tree with large number of vertices, and an extra edge on every vertex other than the leaf vertices. Every edge other than the extra edges will belong some matching. For every edge that belongs to some matching, there will be one edge on each of its end points which does not belong to any matching. If the rule for distributing primal charge among dual variables is similar to one described in section 3.1, then for each edge belonging to some matching will transfer its primal charge equally amongst both its end points. For each edge which does not belong to any matching, \( y_u + y_v = 2 \), which will imply only a competitive ratio of 2. We wish to get a competitive ratio better than 2. So we need some other idea.