

Optimal Online Bipartite Matching in Degree-2 Graphs

Amey Bhangale   

University of California, Riverside, CA, USA

Arghya Chakraborty   

Tata Institute of Fundamental Research, Mumbai, India

Prahladh Harsha   

Tata Institute of Fundamental Research, Mumbai, India

Abstract

Online bipartite matching is a classical problem in online algorithms and we know that both the deterministic fractional and randomized integral online matchings achieve the same competitive ratio of $1 - \frac{1}{e}$. In this work, we study classes of graphs where the online degree is restricted to 2. As expected, one can achieve a competitive ratio of better than $1 - \frac{1}{e}$ in both the deterministic fractional and randomized integral cases, but surprisingly, these ratios are not the same. It was already known that for fractional matching, a 0.75 competitive ratio algorithm is optimal. We show that the folklore HALF-HALF algorithm achieves a competitive ratio of $\eta \approx 0.717772 \dots$ and more surprisingly, show that this is optimal by giving a matching lower-bound. This yields a separation between the two problems: deterministic fractional and randomized integral, showing that it is impossible to obtain a perfect rounding scheme.

2012 ACM Subject Classification Theory of computation $\rightarrow$ Online algorithms

Keywords and phrases Online Algorithm, Bipartite matching

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2025.13

Funding Research of the second and third authors supported by the Department of Atomic Energy, Government of India, under Project Identification No. RTI4001.

Amey Bhangale: Supported by Hellman Fellowship award & NSF CAREER award 2440882.

Arghya Chakraborty: Supported in part by Prof. Mrinal Kumar's SERB grant CRG/2023/006433.

Prahladh Harsha: Supported in part by the Google India Research Award.



1 Introduction

The problem of Bipartite Matching involves a bipartite graph $G = (L, R, E)$ with vertices $L \cup R$ and edges $E \subseteq L \times R$. The objective is to find the largest sized matching in G - in other words, the largest subset $M \subseteq E$ such that no two edges in M share a common vertex. This problem has been well studied [25, 4] and polynomial time algorithms are known.

Online Bipartite Matching was first introduced by Karp, Vazirani, and Vazirani [24]. In the online version of the problem, the set L is known in advance, but the vertices in R arrive one by one. Each time a vertex $j \in R$ arrives (along with its edges), we must make an irreversible decision on whether to match j with one of its neighboring vertices $i \in N(j) \subseteq L$. In a fractional version, we can assign fractional weights to the edges satisfying the matching constraints. The competitive ratio of an online matching algorithm is simply the approximation ratio achieved by the algorithm, that is, the worst-case ratio between the cost of the matching found by the algorithm and the cost of an optimal matching. Online Bipartite Matching has found immense applications in real-world problems. AdWords, organ matching, and online dating are some of the few areas with direct applications.



© Amey Bhangale, Arghya Chakraborty, and Prahladh Harsha;
licensed under Creative Commons License CC-BY 4.0

36th International Symposium on Algorithms and Computation (ISAAC 2025).

Editors: Ho-Lin Chen, Wing-Kai Hon, and Meng-Tsung Tsai; Article No. 13; pp. 13:1–13:19

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The seminal work of Karp, Vazirani, and Vazirani showed that if an unweighted graph has an optimal matching of size n then no online algorithm can guarantee a matching of size larger than $(1 - \frac{1}{e}) \cdot n$, in expectation. They also gave an algorithm, called RANKING, that attains the same competitive ratio of $(1 - \frac{1}{e})$ on unweighted graphs.

Many variants of this problem have also been studied. The survey by Mehta [27] provides an excellent overview of the problem and its variants. Among the various generalizations studied, there is the edge arrival model [20], where the edges arrive in an online fashion instead of the vertices. Recently, there have been breakthroughs [16, 30, 21, 5] in the setting where the edges are weighted (one has to make additional assumptions here, as in this case, no competitive algorithm exists without an additional assumption). Other variants, like stochastic weights [23, 15], ad words [28, 22, 13], display ads [19, 10, 17], online sub-modular welfare maximization [26, 9] have been studied. Most of these edge-weighted settings require either a stochastic input [23, 22] or a free disposal model [18, 16, 30, 5].

For all these matching-related problems, one may consider the polytope of matchings. Randomized algorithms naturally give rise to a fractional matching in this polytope. Hence, the problem of fractional matching is of extreme importance and has also been well studied. The WATER-LEVEL Algorithm [10] is a well-studied deterministic fractional matching algorithm, attaining a competitive ratio of $(1 - \frac{1}{e})$, and this is optimal. Almost all problems related to matching have this feature, which is that the optimal competitive ratio for randomized integral algorithms is the same as the optimal competitive ratio for fractional matching algorithms. Indeed, randomized integral algorithms inherently produce a probability distribution over matchings, and this probability distribution may be viewed as a fractional matching.

However, given that the competitive ratio is the same for all these related problems, it is interesting to ask if the fractional and the randomized integral matchings are inherently the same problems. Hence, people have tried to obtain integral matchings as a randomized rounding of fractional matchings [8]. In fact, for the edge weighted case, all the known algorithms [16, 30, 21, 5], first produce a fractional matching and then use (randomized) Online Correlated Selection (OCS) algorithm to perform the task of online rounding. Therefore, it is interesting to ask whether the fractional and randomized integral matching problems are the same for other variants of matching.

Bounded degree graphs. All the candidate graphs that prove the lower bound of $(1 - \frac{1}{e})$ competitive ratio for randomized integral or fractional matching have the feature that they contain very high-degree vertices. It is natural to ask whether one can construct low-degree graphs to obtain the same lower bounds. There have been some works on this [2, 1, 11, 12, 29, 6, 3]. In this paper, we study graphs where the online vertices have a degree at most 2. We show that, indeed, algorithms can perform better in this particular setting. More specifically, we prove the following theorem.

► **Theorem 1 (Integral Matching).** *For any graph, G , with n online vertices and degree of online vertices at most 2, there exists an online randomized integral matching algorithm with competitive ratio η , and no online algorithm can attain a competitive ratio better than η ,*

$$\text{where } \eta := 1 - \sum_{i=1}^{\infty} \frac{1}{2^{2^i+i-1}} \approx 0.717772.$$

This is interesting, in contrast to the following known result:

► **Theorem 2** (Fractional Matching). *For any graph, G , with n online vertices and degree of online vertices at most 2, the Water-Level Algorithm (c.f, Algorithm 2) attains a competitive ratio of 0.75 and no online algorithm can obtain a better competitive ratio.*

The analysis of the fractional matching problem has been done in [6], but for the sake of completeness, we provide an easy proof in Section 5.

It is intriguing that the fractional and the randomized integral matchings seem to be different problems for this class of graphs, as the optimal competitive ratio differs for the two different settings.

We use the Primal-Dual analysis used in prior works [7, 16, 5] on online bipartite matching (described in detail in the following section) for our algorithm. Our main technical contribution is to set the primal-dual variables in the algorithm's analysis in a non-trivial manner, allowing us to prove the optimality of our algorithm.

1.1 Brief overview

In Section 2, we formally define some of the notations that we will use throughout the paper, the models that we look at, and the online primal-dual framework which is a commonly used tool in the analysis of online bipartite matching algorithms. Then in Section 4.1, we explicitly construct a graph, and then use Yao's principle to prove a lower bound for our problem. Finally, in Section 4.2 we use the online primal-dual framework to prove the tightness of the above bound, by analyzing a simple algorithm.

2 Preliminaries

Throughout this paper, we shall denote by $G(d_1, d_2)$ the set of graphs that have degree at most d_1 for the offline vertices and degree at most d_2 for the online vertices. Note that a bipartite graph on n online vertices and n offline vertices can be any graph from $G(n, n)$. We shall focus on $G(n, 2)$ graphs where the online vertices may have a degree at most 2.

We shall use L to refer to the set of offline vertices and R to refer to the set of online vertices. To make things easier to read, we shall use i, i_1, i_2 to refer to vertices in L (offline vertices) and j to refer to vertices in R (online vertices). We shall mostly consider an online vertex j , arriving with exactly two neighbors i_1 and i_2 , and this is without loss of generality as given by the following lemma.

► **Lemma 3.** *Online Bipartite Matching on graphs in $G(n, 2)$ is equivalent to graphs with degree of online vertices exactly 2. In other words, a c -competitive randomized algorithm for one implies a c -competitive randomized algorithm for the other.*

Proof. Since $G(n, 2)$ includes graphs with online degrees at most 2, an online algorithm might perform better on graphs with exactly degree 2. We show that these two cases are equivalent. We only need to show that if we have a γ -competitive ratio algorithm on graphs with an online degree of exactly 2, then there is a γ -competitive ratio algorithm on graphs with an online degree of at most 2.

Fix an arbitrary algorithm A' with γ -competitive ratio on graphs with an online degree of exactly 2. From this, we will construct an algorithm A with γ -competitive ratio on $G(n, 2)$. Fix any graph $G = (L, R, E) \in G(n, 2)$. We construct a new graph $G'_m = (L', R', E')$, parameterized by an integer m , where all online vertices have exactly degree 2. This graph G'_m consists of m copies of G , where the i -th copy is denoted by $G_i = (L_i, R_i, E_i)$ and an additional vertex ℓ^* to adjust the degree of every online vertex in G'_m to exactly two. More

13:4 Optimal Online Bipartite Matching in Degree-2 Graphs

precisely, the set of offline vertices L' is $L_1 \cup L_2 \cup \dots \cup L_m \cup \{\ell^*\}$ while the set of online vertices R' is $R_1 \cup R_2 \cup \dots \cup R_m$. The edge set E' is $E_1 \cup E_2 \cup \dots \cup E_m \cup \tilde{E}$ where the edges in $\tilde{E}$ are defined next.

The arrival sequence of the online vertices in G'_m is as follows: all the m copies of the first vertex in R , followed by all the m copies of the second vertex in R , and so on. Recall, each of the copies $G_i = (L_i, R_i, E_i)$ are isomorphic to the original graph $G = (L, R, E)$. The graph G'_m has an additional set $\tilde{E}$ of the edges as follows: For every $v \in R_1 \cup R_2 \cup \dots \cup R_m$ with degree 1, we add an edge (v, ℓ^*) to the graph G'_m .

Now, we run A' on G'_m for the first m online vertices, followed by the next m online vertices, and so on. The matching produced by A' can be viewed as m different matchings on G , with an additional vertex $\ell^* \in L$. However, at most one of those matchings may contain ℓ^* as part of the matching. The remaining $m - 1$ matchings must not contain ℓ^* and can be directly viewed as a matching on G . The algorithm A will pick one of these m matchings uniformly at random. If A happens to pick the matching with ℓ^* we give up and do not output any matching. Note that A can be generated from A' in an online fashion, by selecting one of the m matchings and simulating it. If this matching contains ℓ^* , we stop at that point and otherwise we output a feasible matching.

Now, let us bound the competitive ratio of A , assuming that the competitive ratio of A' is γ . Let us assume that G has an optimal matching of size n' , implying that G'_m will have an optimal matching of size at least $m \cdot n'$. Hence A' , with a competitive ratio of γ , should give us at least $\gamma \cdot m \cdot n'$ size, in expectation. However, in this matching produced, we might have ℓ^* , which we cannot use in algorithm A . By our construction, we have m collections of matchings of which one contains ℓ^* and the remaining $m - 1$ of them are used by A . Now, if we leave aside the particular matching that contains ℓ^* , we would still have a matching of size at least $\frac{\gamma \cdot m \cdot n' - n'}{m}$ in expectation, as produced by A' . The optimal matching in this set is of size at most n' . Hence the expected competitive ratio of A is at least $\frac{\gamma \cdot m \cdot n' - n'}{m \cdot n'} = \frac{\gamma \cdot m - 1}{m}$, which converges to γ as m increases. Hence, by choosing a large m , this conversion ensures that A attains the same competitive ratio as A' on G . ◀

2.1 Primal and Dual programs for bipartite matching

At this point, we would like to explain the primal and dual programs for online bipartite matching. We shall use $x_{i,j}$ to denote whether i is matched to j or not. So, we would want $x_{i,j}$ to be either 0 or 1, and the size of the matching produced would be the sum of $x_{i,j}$, which we would like to maximize. To ensure that it is a matching, we add the constraints that for each vertex, there must be at most one edge incident on the vertex.

We shall also use $x_i := \sum_{j \in R} x_{i,j}$ to denote whether vertex i is matched or not. We shall also relax this problem as follows :

$$\begin{aligned} & \text{Maximize } \sum_{i \in L} \sum_{j \in R} x_{i,j} = \sum_{i \in L} x_i \\ & \text{subject to :} \end{aligned}$$

$$\forall i \in L \quad \sum_{j \in \text{Nbd}(i)} x_{i,j} \leq 1$$

$$\forall j \in R \quad \sum_{i \in \text{Nbd}(j)} x_{i,j} \leq 1$$

$$x_{i,j} \geq 0$$

$$\begin{aligned} & \text{Minimize } \sum_{i \in L} \alpha_i + \sum_{j \in R} \beta_j \\ & \text{subject to :} \end{aligned}$$

$$\forall (i,j) \in E, \alpha_i + \beta_j \geq 1$$

$$\forall i \in L, j \in R, \alpha_i \geq 0 \text{ and } \beta_j \geq 0$$

We have relaxed the $x_{i,j}$'s from being $\{0,1\}$ valued to instead allow any fraction : $x_{i,j} \in [0,1]$. This means that the optimal value of the primal objective can only be larger now (since the graph is bipartite, the optimal value actually is the same), which means that the matching size obtained (the competitive ratio) will only be lower, compared to the primal optimal. Also note that if we are doing a fractional or a randomized matching, we would want $x_{i,j}$ to represent the fraction that i is matched to j or the probability of i being matched to j , ensuring that the expected size of matching is $\sum_{i \in L, j \in R} x_{i,j}$ and the change/increment in the primal value because of the vertex j is $\sum_{i \in L} x_{i,j} = \sum_{i \in L} \Delta x_i$.

The dual program represents the vertex cover problem, but we shall not use this fact in any way. Note the dual constraint $\alpha_i + \beta_j \geq 1$. If a dual solution is not feasible, meaning that the solution does not satisfy $\alpha_i + \beta_j \geq 1$ for all $(i,j) \in E$ but instead satisfies $\alpha_i + \beta_j \geq \gamma$ for some $\gamma < 1$, then we shall call this solution γ -feasible.

In the analysis of our online algorithm, we will keep updating the primal and dual solutions. We shall use ΔP and ΔD to denote the increase in the primal and dual objectives, respectively. Similarly, we shall use Δx_i and $\Delta \alpha_i$ to refer to the increments in the corresponding variables.

The main idea that we use in our algorithms is an online primal-dual method, similar to the one used in prior work [14, 16]. We always maintain feasible primal and approximately feasible dual solutions. Whenever primal increases by some amount, we increase the dual by the same amount, so the primal and dual always have the same value. The primal will be feasible, so we would obtain an optimal solution if the dual were also feasible. However, we will be unable to maintain a feasible dual solution; instead, we will maintain an approximate feasible solution that satisfies the dual constraints with a slack of γ , in other words, be γ -feasible.

More specifically, we will always maintain the following throughout :

- **Approximate Dual Feasibility:** For any $i \in L$ and $j \in R$, $\alpha_i + \beta_j \geq \gamma$
- **Reverse Weak Duality:** The objective value of the primal (P) and the objective value of the dual (D) satisfy that $P \geq D$.

This is important because of the following claim.

▷ **Claim 4.** If the primal solution is feasible and has value P , and if the dual solution has objective value at most P , while being γ -feasible, then the primal value P is at least γ fraction of the primal optimal.

Proof. Let M be the primal optimal. By strong duality, it is also the value of the dual optimal. Since the primal is feasible, $P \leq M$. Let the dual solution have value $D \leq P$. Since the dual solution is γ feasible, multiplying all the dual variables by $\frac{1}{\gamma}$ gives a feasible dual solution with value $\frac{D}{\gamma} \geq M$ because it is feasible.

$$\text{Hence } D \leq P \leq M \leq \frac{D}{\gamma} \implies \gamma \cdot M \leq D \leq P \implies \frac{P}{M} \geq \gamma. \quad \triangleleft$$

3 Techniques

While the algorithm that we suggest is extremely simple, the interesting part is the analysis. We use the online primal-dual method to prove that the algorithm is optimal. The primal variables are easy to update, given the algorithm. The tricky part is to update the dual variables: α_i and β_j . Indeed, there are other intuitive candidate algorithms for this problem, which we tried but were unable to analyse using the primal-dual approach.

On arrival of any vertex, $j \in R$, we try to increase the α_i 's as much as possible and β_j as little as possible while maintaining γ -competitive ratio. The key observation, in Lemma 9, is that given the primal variable of a vertex $i \in L$, one can determine a lower bound for the dual variable α_i . Given this observation, it is sufficient to show that as the probability of i being matched approaches 1, the dual variable α_i approaches γ .

4 Randomized Integral Matching

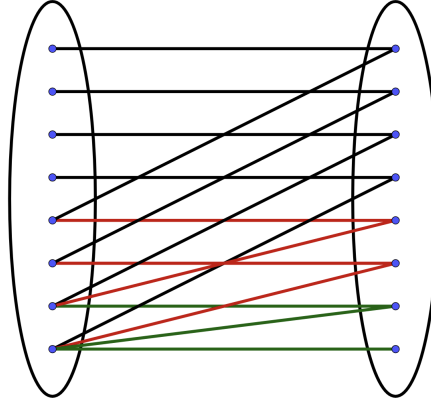
We start with a lower-bound construction first.

4.1 Lower Bound

Recall that $\eta := 1 - \sum_{i=1}^{\infty} \frac{1}{2^{2^i+i-1}}$. The following theorem shows that no online algorithm can achieve a competitive ratio better than η .

► **Theorem 5.** *For Online Integral Bipartite Matching, no online algorithm (deterministic or randomized) can attain a competitive ratio better than η .*

Proof. We will use Yao's principle to prove this. We shall first construct one particular graph G and then look at the uniform distribution over all graphs that are isomorphic to the described graph.



■ **Figure 1** The graph for the lower bound, with 8 online and offline vertices with online degree bounded by 2. The black edges correspond to online vertices in the first phase, the red ones correspond to the second phase and the green ones correspond to the third phase.

This graph will have $n = 2^k$ online and 2^k offline vertices for some integer k . We will describe the graph in phases. There will be $k = \log n$ phases in total. The online i -th vertex will always have an edge to the offline i -th vertex. Since the online vertices will have degree 2, it suffices to explain the other neighbor of the online vertex i . Figure 1 depicts this graph for $k = 3$, with $2^k = 8$ online and offline vertices.

This is where we will use the phases to describe the neighbors. In the first phase, there will be $\frac{n}{2}$ online vertices: 1 through $\frac{n}{2}$ and the i -th vertex would have the other edge to offline vertex $\frac{n}{2} + i$. In the second phase, there will be $\frac{n}{4}$ online vertices: $\frac{n}{2} + 1$ through $n - \frac{n}{4}$ and the $\frac{n}{2} + i$ -th vertex would have the other edge to offline vertex $n - \frac{n}{4} + i$. In the j -th phase, there will be $\frac{n}{2^j}$ online vertices: $n - \frac{n}{2^{j-1}} + 1$ through $n - \frac{n}{2^j}$ and the $n - \frac{n}{2^{j-1}} + i$ -th vertex would have the other edge to offline vertex $n - \frac{n}{2^j} + i$.

Notice that in the j -th phase, when an online vertex arrives, its neighbors' degree changes from $j - 1$ to j .

Now using this graph, G , let us construct a distribution over graphs. We shall consider a set of all graphs that can be obtained from G , by a permutation of the offline vertices, and then take a uniform distribution over these graphs. The idea is to use Yao's principle on this distribution and study the competitive ratio of deterministic algorithms. We shall compute the probability of offline vertices being matched, based on the phases. We shall consider any candidate deterministic algorithm for this.

Consider any offline vertex, u . If there is an online vertex, say v , in the j -th phase, with u as a neighbor of j , then we shall say that u "appeared" in the j -th phase. Notice that if u appeared in the j -phase, then u must have appeared in the phases $1, 2, \dots, j - 1$ too. For every offline vertex, u , there will be a unique j such that u appeared in the j -th phase and then did not appear in any of the phases $j + 1, j + 2, \dots$ again.

Let us first look at offline vertices that appear in the first phase and then never again. The probability that such a vertex is matched is $\frac{1}{2}$, and there are $\frac{n}{2}$ such vertices.

Next, for vertices that appear in the second phase and never after that, we shall compute the probability that they are not matched. Let us consider one such vertex, u . For u not to be matched, it has to be necessarily unmatched when it appeared in the first phase. This would happen with probability $\frac{1}{2}$. Also, when u appears in the second phase, the online vertex, say v , will have another neighbor, say i_1 . If i_1 happened to be matched earlier, in phase 1, then u will get matched in the second phase. Therefore, for u to be unmatched i_1 must be unmatched and then on the arrival of v the probability that the deterministic algorithm will leave u unmatched will be $\frac{1}{2}$. Hence the probability that u is unmatched is $\frac{1}{8}$.

For a vertex u that appears in the third phase and never thereafter, the probability that u is unmatched is $\frac{1}{2} \cdot \frac{1}{8} \cdot \frac{1}{8}$ because this is the case when u was unmatched even after phase two and i_1 , the competitor of u in phase three was also unmatched after phase two and in this case the probability of u being unmatched is $\frac{1}{2}$.

Similarly, for a vertex that appears in the i -th phase and then never thereafter, the probability that such a vertex is unmatched is $\frac{1}{2^{2^i-1}}$. Also, there are $\frac{n}{2^i}$ such vertices that appear in the i -th phase and then never thereafter.

Hence, the expected number of vertices unmatched equals

$$\sum_{i=1}^{\infty} \frac{n}{2^{2^i+i-1}}.$$

Therefore, the competitive ratio will be

$$1 - \sum_{i=1}^{\infty} \frac{1}{2^{2^i+i-1}} = \eta \approx 0.717772. \quad \blacktriangleleft$$

4.2 Upper Bound

In this section, we will show that the folklore HALF-HALF algorithm attains a competitive ratio of η . We begin by recalling the HALF-HALF algorithm, which can be described as follows. When an online vertex v comes, if it has only one unmatched neighbor, then match v with the unmatched neighbor. If both the neighbors of v are unmatched, then pick one uniformly at random and match it with v . The formal description of the algorithm is given in Algorithm 1.

■ **Algorithm 1** Half-Half Algorithm.

```

1: procedure  $A(j, S)$  ▷ Online vertex  $j$  with neighborhood  $S \subseteq L$ ,  $|S| \leq 2$ 
2:   if  $\forall i \in S$ ,  $i$  is already matched then
3:     Do nothing.
4:   else
5:     if There exists only one  $i \in S$ , such that  $i$  is unmatched then
6:       Match  $i$  to  $j$ .
7:     else
8:       We have  $S = \{i_1, i_2\}$  such that  $i_1$  and  $i_2$  are both unmatched
9:       Match  $j$  with  $i_1$  and  $i_2$  with probability half each.
10:    end if
11:  end if
12: end procedure

```

4.2.1 Analysis of the half-half algorithm

► **Theorem 6.** *The half-half algorithm attains a competitive ratio of η , where:*

$$\eta := 1 - \sum_{i=1}^{\infty} \frac{1}{2^{2^i+i-1}} = 1 - \frac{1}{4} - \frac{1}{32} - \frac{1}{2^{10}} - \frac{1}{2^{19}} - \dots \approx 0.717772.$$

Throughout the rest of the section, we are going to prove this theorem.

The plan is again to use the online Primal-Dual analysis. We will show that on the arrival of $j \in R$, with neighbors $i_1, i_2 \in L$, we can update the primal such that the expected size of the matching produced is at least the value of the primal. Simultaneously we will set up β_j and increment $\alpha_{i_1}, \alpha_{i_2}$ so that the dual conditions are η -feasible. We shall initialize by setting all the α_i 's to 0, and β_j will start off as 0 before we do the increments for the dual variables. We shall describe how to update the dual variables later. First, we will describe the updates to the primal variables.

Updating the Primal variables. According to the algorithm, we can change the primal variables according to the following rule. We always maintain the primal variables to take values of the form $1 - \frac{1}{2^p}$ where $p \in \{0, 1, 2, 3, \dots\}$. If an online vertex j comes with neighbors $\{i_1, i_2\}$, and the old values of x_{i_1} and x_{i_2} are as follows:

$$x_{i_1} = 1 - \frac{1}{2^{p_1}}, \quad x_{i_2} = 1 - \frac{1}{2^{p_2}},$$

then the updated values would be

$$x_{i_1} = x_{i_2} = 1 - \frac{1}{2^{p_1+p_2+1}}.$$

With this, the change in the primal value would be

$$\Delta P = 2 \left(1 - \frac{1}{2^{p_1+p_2+1}} \right) - \left(1 - \frac{1}{2^{p_1}} \right) - \left(1 - \frac{1}{2^{p_2}} \right).$$

► **Claim 7.** Changing the primal in this way ensures that primal variables are always of the form $1 - \frac{1}{2^p}$, and that the expected increment of matching size is at least the change in primal.

Proof. First of all, it is clear from these updates that the primal variables are always of the form $1 - \frac{1}{2^p}$. Had the events of i_1 and i_2 being unmatched been independent events, we could have said

$$\begin{aligned}\mathbb{P}[i_1 \text{ is unmatched after } j] &= \mathbb{P}[i_2 \text{ is unmatched after } j] \\ &= \frac{1}{2} \cdot \mathbb{P}[i_1 \text{ and } i_2 \text{ were unmatched before } j] = \frac{1}{2^{p_1+p_2+1}}.\end{aligned}$$

However, if the events are not independent, then there is only a negative correlation, i.e., given that i_1 is unmatched, the probability of i_2 being unmatched can only decrease. The proof is as follows. Conditioned on i_1 being unmatched, every online vertex, with i_1 as its neighbor, must have been matched to the offline vertex other than i_1 . Let us call this set of vertices S . Now, when these offline vertices from S , after being matched, were neighbors of some online vertex, say v , the online vertex v must have been matched to the other neighbor of v . We can update our S to include all such vertices that get matched in this manner. Continuing like this, the set S will consist of all vertices that are matched, conditioned on i_1 not being matched. If any online vertex had i_2 as its neighbor and a vertex from S as its other neighbor, i_2 must have been matched conditioned on i_1 not being matched. Otherwise, i_1 being matched and i_2 being matched are independent events. Hence,

$$\begin{aligned}\mathbb{P}[i_1 \text{ is unmatched after } j] &= \mathbb{P}[i_2 \text{ is unmatched after } j] \\ &= \frac{1}{2} \mathbb{P}[i_1 \text{ and } i_2 \text{ were unmatched before } j] \leq \frac{1}{2^{p_1+p_2+1}}.\end{aligned}$$

We have already seen that the change in primal is

$$\Delta P = 2 \left(1 - \frac{1}{2^{p_1+p_2+1}} \right) - \left(1 - \frac{1}{2^{p_1}} \right) - \left(1 - \frac{1}{2^{p_2}} \right) = \frac{1}{2^{p_1}} + \frac{1}{2^{p_2}} - \frac{2}{2^{p_1+p_2+1}}.$$

Now we can compare this to the expected increase in matching size,

$$\begin{aligned}\mathbb{E}[\text{Increment in size of matching}] &= \mathbb{P}[\text{Either } i_1 \text{ or } i_2 \text{ was unmatched on arrival of } j] \\ &= \mathbb{P}[i_1 \text{ was unmatched on arrival of } j] + \mathbb{P}[i_2 \text{ was unmatched on arrival of } j] - \\ &\quad \mathbb{P}[\text{Both } i_1 \text{ and } i_2 \text{ were unmatched on arrival of } j] \\ &\geq \frac{1}{2^{p_1}} + \frac{1}{2^{p_2}} - \frac{2}{2^{p_1+p_2+1}}.\end{aligned}$$

Thus,

$$\mathbb{E}[\text{Increment in size of matching}] \geq \Delta P,$$

and this proves the claim. $\triangleleft$

Updating the Dual variables. Now that we have described how the primal variables are updated, we shall describe how to update the dual variables. As mentioned earlier, we will always maintain $\Delta P \geq \Delta D$. So, when online vertex j with neighbors i_1, i_2 arrives, we shall first compute ΔP and then try to set up $\alpha_{i_1} = \alpha_{i_2}$. The goal is to set up the α 's as high as possible, preferably to η , and assign the remainder of ΔP to β_j if needed. This will ensure $\Delta P \geq \Delta D$. We would want $\alpha_{i_1} + \beta_j \geq \eta$ and also $\alpha_{i_2} + \beta_j \geq \eta$, while simultaneously maintaining $\beta_j \geq 0$. This would limit us from setting large α 's. It will be convenient to define the following notation.

► **Definition 8** ($\alpha_{(k)}$). We define the quantity $\alpha_{(k)}$ as follows: On arrival of the online vertex j with neighbors i_1 and i_2 , suppose the primal variables x_{i_1} and x_{i_2} take the value $1 - \frac{1}{2^k}$ after the update, then the minimum value that the dual variables α_{i_1} and α_{i_2} are set (while maintaining η -feasibility and $\beta_j \in [0, 1]$) is the quantity $\alpha_{(k)}$.

What we will show now is that it will always be possible for us to set the dual variables in order to maintain η -approximate dual feasibility.

We shall first show that for $1 \leq i \leq 3$, we can set up $\alpha_{(i)}$ to maintain approximate dual feasibility. After that, we will prove a general statement showing that we can assign appropriate dual variables for all x_i .

Setting $\alpha_{(1)}$. Let us first consider the case when the two neighbors i_1, i_2 had not appeared before. In this case, the half-half algorithm will set $x_{i_1} = x_{i_2} = \frac{1}{2}$. Observe that this is the only case when the x_i will be set to $\frac{1}{2}$, and hence the previous values of x_{i_1}, x_{i_2} is 0. So, $\Delta P = 1$. Let us set $\alpha_{i_1} = \alpha_{i_2} = \alpha_{(1)} = 1 - \eta$ in this case, and β_j to $2\eta - 1$. This will ensure that $\Delta D = 1 = \Delta P$ (recall, α_{i_1} and α_{i_2} were set to 0 at the beginning). Additionally, the dual constraints for the edges $\{i_1, j\}, \{i_2, j\}$ are η -satisfied.

Setting $\alpha_{(2)}$. Let the neighbors of j be i_1 and i_2 with $x_{i_1} = 1 - \frac{1}{2}$ and $x_{i_2} = 0$. Then the updated $x_{i_1} = x_{i_2} = 1 - \frac{1}{4}$ and notice that, up to symmetry, this is the only way for the primal variables to attain $x_i = 1 - \frac{1}{4}$. In this case, $\Delta P = 1$. Updating $\alpha_{i_1} = \alpha_{i_2} = \alpha_{(2)} = 2 - 2\eta$ means that the increment in $\alpha_{i_1} + \alpha_{i_2}$ is $(\alpha_{(2)} - \alpha_{(1)}) + (\alpha_{(2)} - 0) = ((2 - 2\eta) - (1 - \eta)) + ((2 - 2\eta) - 0)$ because for i_1 , with $x_{i_1} = 0$, we had $\alpha_{i_1} = 1 - \eta$ and for i_2 we had $\alpha_{i_2} = 0$ before the arrival of j . Hence the total increment in $\alpha_{i_1} + \alpha_{i_2}$ is $3 - 3\eta$. Therefore, assigning β_j to $\Delta P - \Delta\alpha_{i_1} - \Delta\alpha_{i_2} = 3\eta - 2$ ensures that β_j is positive and the increment in primal is equal to the increment in the dual. Also, for the edges, the dual constraints are η -satisfied since $(2 - 2\eta) + (3\eta - 2) = \eta$.

Setting $\alpha_{(3)}$. There are two cases in this case.

1. In the first case, let the neighbors of j be i_1 and i_2 with $x_{i_1} = x_{i_2} = 1 - \frac{1}{2}$ before the arrival of j . According to the change in primal values, both x_{i_1} and x_{i_2} would be set to $1 - \frac{1}{8}$ after the update. Therefore, $\Delta P = 2(1 - \frac{1}{8}) - \frac{1}{2} - \frac{1}{2} = \frac{3}{4}$. Also, α_{i_1} and α_{i_2} used to be $\alpha_{(1)} = 1 - \eta$ before j arrived. If we update α_{i_1} and α_{i_2} to $\alpha_{(3)}$, then we can set β_j to $\Delta P - (2\alpha_{(3)} - 2(1 - \eta))$ in order to ensure that the change in dual is at most the change in primal. However, after the update, the edge corresponding to $\{i_1, j\}$ must satisfy approximate dual feasibility. Hence, we must have $\beta_j + \alpha_{(3)} \geq \eta$, implying $\frac{3}{4} - \alpha_{(3)} + 2 - 2\eta \geq \eta$. Therefore $\alpha_{(3)} \leq \frac{11}{4} - 3\eta$.
2. In the second case, if the two neighbors i_1 and i_2 had $x_{i_1} = 1 - \frac{1}{4}$ and $x_{i_2} = 0$ (i.e. i_2 not having appeared before) even then both x_{i_1} and x_{i_2} would be set to $1 - \frac{1}{8}$ after the update. It can be checked that in this case, $\Delta P = 1$ and hence we can set $\alpha_{i_1} = \alpha_{i_2} = \eta$, while keeping $\beta_j = 0$.

Considering both these cases, we can ensure that $\alpha_{i_1}, \alpha_{i_2}$ will certainly be set to at least $\frac{11}{4} - 3\eta$ and hence $\alpha_{(3)} = \frac{11}{4} - 3\eta$.

Setting $\alpha_{(k)}$ for $k \geq 4$. As we have seen till now, we can set up the α variables so that approximate dual feasibility is satisfied.

The next lemma gives a strategy for the remaining cases to set the dual variables so that the Dual is η -feasible.

► **Lemma 9.** *For every $k \geq 4$, there is a fixed $\alpha_{(k)}$ such that whenever the primal variables are set to $1 - \frac{1}{2^k}$, one can set the value of the corresponding dual variables α such that $\alpha \geq \alpha_{(k)}$ while ensuring $\Delta D \leq \Delta P$, $\beta_j \in [0, 1]$ and the η -approximate dual feasibility. More specifically, the algorithm sets $\alpha_{(k)}$ to be η whenever k is not of the form $2^m - 1$ (for some integer m), otherwise*

$$\alpha_{(2^m-1)} \geq \sum_{i=1}^{m-1} 2^{m-1-i} \left(\frac{1}{2^{2^i-2}} - \frac{1}{2^{2(2^i-2)+2}} \right) + 2^{m-1} - (2^m - 1)\eta.$$

Before we prove this lemma, let's first see why this is enough to prove the main theorem.

Proof of Theorem 6. It is easy to see that proving this lemma completes the proof. This is because, after the arrival of j , i_1 and i_2 are more likely to be matched and hence x_{i_1} and x_{i_2} increase. We have seen that x_i may only take values of the form $1 - \frac{1}{2^k}$ for some integer k . For $k = 1$ to 3 , we have explicitly checked that we can always satisfy reverse weak duality and η -approximate dual feasibility. For any larger value of k , the Lemma 9 tells us that we will always be able to set up the dual variables while satisfying reverse weak duality and η -approximate deal feasibility. Hence, we shall always be able to update the dual variables in this manner, for all k and therefore the Half-half Algorithm would be η -competitive. ◀

We now prove Lemma 9.

Proof of Lemma 9. The proof will be by induction on k . Let us assume that $\alpha_{(k)}$ follows the lemma up to some $k < T$.

We will need to check the base cases for $k = 4, 5, 6, 7$ after which the proof follows by induction. The calculations are exactly like that for $k = 1, 2, 3$. We provide a table of values here for each k . This table describes how the dual variables are set for that particular k . Each row describes a different initial setting of x_{i_1} and x_{i_2} that can lead to $x_{i_1} = x_{i_2} = 1 - \frac{1}{2^k}$. The reader may check that these updates satisfy $\Delta P \geq \Delta D$ and η -approximate dual feasibility and satisfy Lemma 9.

For $k = 4$,

x_{i_1}	x_{i_2}	ΔP	Old α_{i_1}	Old α_{i_2}	New $\alpha_{i_1} = \alpha_{i_2}$	β_j
0	$1 - \frac{1}{8}$	1	0	$\frac{11}{4} - 3\eta$	η	0
$\frac{1}{2}$	$1 - \frac{1}{4}$	$\frac{1}{2} + \frac{1}{4} - \frac{1}{8}$	$1 - \eta$	$2 - 2\eta$	η	0

For $k = 5$,

x_{i_1}	x_{i_2}	ΔP	Old α_{i_1}	Old α_{i_2}	New $\alpha_{i_1} = \alpha_{i_2}$	β_j
0	$1 - \frac{1}{16}$	1	0	η	η	0
$\frac{1}{2}$	$1 - \frac{1}{8}$	$\frac{1}{2} + \frac{1}{8} - \frac{1}{16}$	$1 - \eta$	$\frac{11}{4} - 3\eta$	η	0
$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4} + \frac{1}{4} - \frac{1}{16}$	$2 - 2\eta$	$2 - 2\eta$	η	0

For $k = 6$,

x_{i_1}	x_{i_2}	ΔP	Old α_{i_1}	Old α_{i_2}	New $\alpha_{i_1} = \alpha_{i_2}$	β_j
0	$1 - \frac{1}{32}$	1	0	η	η	0
$\frac{1}{2}$	$1 - \frac{1}{16}$	$\frac{1}{2} + \frac{1}{16} - \frac{1}{32}$	$1 - \eta$	η	η	0
$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{4} + \frac{1}{8} - \frac{1}{32}$	$2 - 2\eta$	$\frac{11}{4} - 3\eta$	η	0

13:12 Optimal Online Bipartite Matching in Degree-2 Graphs

For $k = 7$,

x_{i_1}	x_{i_2}	ΔP	Old α_{i_1}	Old α_{i_2}	New $\alpha_{i_1} = \alpha_{i_2}$	β_j
0	$1 - \frac{1}{64}$	1	0	η	η	0
$\frac{1}{2}$	$1 - \frac{1}{32}$	$\frac{1}{2} + \frac{1}{32} - \frac{1}{64}$	$1 - \eta$	η	η	0
$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{4} + \frac{1}{16} - \frac{1}{64}$	$2 - 2\eta$	η	η	0
$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8} + \frac{1}{8} - \frac{1}{64}$	$\frac{11}{4} - 3\eta$	$\frac{11}{4} - 3\eta$	$\frac{367}{64} - 7\eta$	$8\eta - \frac{367}{64}$

For T , we can have three cases: (1) T is of the form $2^m - 1$ for some m . (2) T is of the form $2^{m_1} + 2^{m_2} - 1$, or (3) T is not of these forms.

Case 3. We note that

$$\alpha_{(T)} = \min_{k \in \{0, 1, 2, \dots, \frac{T-1}{2}\}} \left[\left(\frac{1}{2^k} + \frac{1}{2^{T-1-k}} - \frac{1}{2^{T-1}} \right) + \alpha_{(k)} + \alpha_{(T-1-k)} - \eta \right]$$

can be achieved by putting the entire ΔP into α without increasing the β . Now, considering the case we are in, we know that either $\alpha_{(k)}$ or $\alpha_{(T-1-k)}$ is η (inductive hypothesis). Since the other α already had value more than $\eta - (1 - x_i)$ (this can be seen from the fact that $\alpha_{(T)}$ converges to η as T tends to ∞ and the total value added to $\alpha_{(T)}$, at any point, is at most the increment in x_i) and observe that $\Delta P \geq (1 - x_i)$. Therefore, the alpha can be increased to η .

Case 1. In this case $T = 2^m - 1$ for some m .

$$\alpha_{(2^m-1)} = \min_{k \in \{0, 1, 2, \dots, 2^{m-1}-1\}} \left(\frac{1}{2^k} + \frac{1}{2^{2^m-k-2}} - \frac{1}{2^{2^m-2}} \right) + \alpha_{(k)} + \alpha_{(2^m-k-2)} - \eta$$

Note that the minimum occurs when $k = 2^{m-1} - 1$ (on top of this, we need to show that $\Delta P - \Delta\alpha \geq 0$, so that β can be set to a value in $[0, 1]$). In this case,

$$\begin{aligned} \alpha_{(2^m-1)} &= \left(\frac{1}{2^{2^{m-1}-1}} + \frac{1}{2^{2^m-2^{m-1}+1-2}} - \frac{1}{2^{2^m-2}} \right) + \alpha_{(2^{m-1}-1)} + \alpha_{(2^m-2^{m-1}+1-2)} - \eta \\ &= \sum_{i=1}^{m-1} 2^{m-1-i} \left(\frac{1}{2^{2^i-2}} - \frac{1}{2^{2(2^i-2)+2}} \right) + 2^{m-1} - (2^m - 1)\eta. \end{aligned}$$

as required. Now, we need to show that the following quantity is non-negative.

$$\begin{aligned} \Delta P - \Delta\alpha &= \left(\frac{1}{2^{2^{m-1}-1}} + \frac{1}{2^{2^m-2^{m-1}-1}} - \frac{1}{2^{2^m-2}} \right) - 2 \cdot (\alpha_{(2^m-1)} - \alpha_{(2^{m-1}-1)}) \\ &= \left(\frac{1}{2^{2^{m-1}-1}} + \frac{1}{2^{2^m-2^{m-1}-1}} - \frac{1}{2^{2^m-2}} \right) - 2 \cdot (\alpha_{(2^m-1)} - \alpha_{(2^{m-1}-1)}) \\ &= \left(\frac{1}{2^{2^{m-1}-2}} - \frac{1}{2^{2^m-2}} \right) - 2 \cdot (\alpha_{(2^m-1)} - \alpha_{(2^{m-1}-1)}). \end{aligned}$$

We have the following claim.

▷ **Claim 10.** The quantity $\left(\frac{1}{2^{2^{m-1}-2}} - \frac{1}{2^{2^m-2}} \right) - 2 \cdot (\alpha_{(2^m-1)} - \alpha_{(2^{m-1}-1)})$ is non-negative for all values of $m \geq 1$.

Proof. We first simplify the last term.

$$\begin{aligned}
& (\alpha_{(2^m-1)} - \alpha_{(2^{(m-1)}-1)}) \\
&= \left(\sum_{i=1}^{m-1} 2^{m-1-i} \left(\frac{1}{2^{2^i-2}} - \frac{1}{2^{2(2^i-2)+2}} \right) + 2^{m-1} - (2^m - 1)\eta \right) \\
&\quad - \left(\sum_{i=1}^{m-2} 2^{m-2-i} \left(\frac{1}{2^{2^i-2}} - \frac{1}{2^{2(2^i-2)+2}} \right) + 2^{m-2} - (2^{m-1} - 1)\eta \right) \\
&= \frac{1}{2} \cdot \left(\frac{1}{2^{2^{m-1}-2}} - \frac{1}{2^{2(2^{m-1}-2)+2}} \right) + (2^{m-1} - 2^{m-2}) - (2^m - 2^{m-1})\eta \\
&\quad + \sum_{i=1}^{m-1} (2^{m-1-i} - 2^{m-2-i}) \left(\frac{1}{2^{2^i-2}} - \frac{1}{2^{2(2^i-2)+2}} \right) \\
&= \frac{1}{2} \cdot \left(\frac{1}{2^{2^{m-1}-2}} - \frac{1}{2^{2(2^{m-1}-2)+2}} \right) + 2^{m-2} - 2^{m-1} \left(1 - \sum_{i=1}^{\infty} \frac{1}{2^{2^i+i-1}} \right) \\
&\quad + \sum_{i=1}^{m-1} 2^{m-2-i} \left(\frac{1}{2^{2^i-2}} - \frac{1}{2^{2(2^i-2)+2}} \right) \\
&= \frac{1}{2} \cdot \left(\frac{1}{2^{2^{m-1}-2}} - \frac{1}{2^{2(2^{m-1}-2)+2}} \right) - 2^{m-2} + 2^{m-2} \sum_{i=1}^{\infty} \frac{2}{2^{2^i+i-1}} \\
&\quad + 2^{m-2} \cdot \sum_{i=1}^{m-1} \frac{1}{2^i} \cdot \left(\frac{1}{2^{2^i-2}} - \frac{1}{2^{2(2^i-2)+2}} \right).
\end{aligned}$$

Now,

$$\begin{aligned}
2(\alpha_{(2^m-1)} - \alpha_{(2^{(m-1)}-1)}) &= \left(\frac{1}{2^{2^{m-1}-2}} - \frac{1}{2^{2(2^{m-1}-2)+2}} \right) - 2^{m-1} + 2^{m-1} \sum_{i=1}^{\infty} \frac{2}{2^{2^i+i-1}} \\
&\quad + 2^{m-1} \cdot \sum_{i=1}^{m-1} \frac{1}{2^i} \cdot \left(\frac{1}{2^{2^i-2}} - \frac{1}{2^{2(2^i-2)+2}} \right) \\
&= \left(\frac{1}{2^{2^{m-1}-2}} - \frac{1}{2^{2^{m-2}}} \right) - 2^{m-1} + 2^{m-1} \sum_{i=1}^{\infty} \frac{2}{2^{2^i+i-1}} \\
&\quad + 2^{m-1} \cdot \sum_{i=1}^{m-1} \frac{1}{2^i} \cdot \left(\frac{1}{2^{2^i-2}} - \frac{1}{2^{2(2^i-2)+2}} \right).
\end{aligned}$$

Therefore,

$$\begin{aligned}
\Delta P - \Delta \alpha &= 2^{m-1} - 2^{m-1} \sum_{i=1}^{\infty} \frac{2}{2^{2^i+i-1}} - 2^{m-1} \cdot \sum_{i=1}^{m-1} \frac{1}{2^i} \cdot \left(\frac{1}{2^{2^i-2}} - \frac{1}{2^{2(2^i-2)+2}} \right) \\
&= 2^{m-1} \left(1 - \sum_{i=1}^{\infty} \frac{2}{2^{2^i+i-1}} - \sum_{i=1}^{m-1} \frac{1}{2^i} \cdot \left(\frac{1}{2^{2^i-2}} - \frac{1}{2^{2(2^i-2)+2}} \right) \right) \\
&= 2^{m-1} \left(1 - \sum_{i=1}^{\infty} \frac{2}{2^{2^i+i-1}} - \sum_{i=1}^{m-1} \left(\frac{2}{2^{2^i+i-1}} - \frac{1}{2^{(2^i+1)+i-2}} \right) \right).
\end{aligned}$$

13:14 Optimal Online Bipartite Matching in Degree-2 Graphs

Now, we simplify the last summation as follows

$$\begin{aligned}
\sum_{i=1}^{m-1} \left(\frac{2}{2^{2^i+i-1}} - \frac{1}{2^{(2^{i+1}+i-2)}} \right) &= \sum_{i=1}^{m-1} \left(\frac{2}{2^{2^i+i-1}} - \frac{1}{2^{(2^{(i+1)}+(i+1)-3)}} \right) \\
&= \sum_{i=1}^{m-1} \left(\frac{2}{2^{2^i+i-1}} - \frac{4}{2^{(2^{(i+1)}+(i+1)-1)}} \right) \\
&= \frac{2}{2^2} - \sum_{i=2}^{m-1} \frac{2}{2^{2^i+i-1}} - \frac{4}{2^{(2^m+m-1)}}
\end{aligned}$$

Hence,

$$\begin{aligned}
\Delta P - \Delta \alpha &= 2^{m-1} \left(1 - \sum_{i=1}^{\infty} \frac{2}{2^{2^i+i-1}} - \sum_{i=1}^{m-1} \left(\frac{2}{2^{2^i+i-1}} - \frac{1}{2^{(2^{i+1}+i-2)}} \right) \right) \\
&= 2^{m-1} \left(1 - \sum_{i=1}^{\infty} \frac{2}{2^{2^i+i-1}} - \frac{2}{2^2} + \sum_{i=2}^{m-1} \frac{2}{2^{2^i+i-1}} + \frac{4}{2^{(2^m+m-1)}} \right) \\
&= 2^{m-1} \left(1 - \frac{2}{2^2} - \sum_{i=m}^{\infty} \frac{2}{2^{2^i+i-1}} - \frac{2}{2^2} + \frac{4}{2^{(2^m+m-1)}} \right) \\
&= 2^{m-1} \left(\frac{4}{2^{(2^m+m-1)}} - \sum_{i=m}^{\infty} \frac{2}{2^{2^i+i-1}} \right) \geq 0.
\end{aligned}$$

◁

Given the claim, we can see that $\Delta P - \Delta \alpha \geq 0$, allowing us to set $\beta_j \geq 0$. ◀

Case 2. In the case when T is of the form $2^{m_1} + 2^{m_2} - 1$ (where $m_1 > m_2$), we will try to find out $\alpha_{(T)}$ using $\alpha_{(2^{m_1}-1)}$ and $\alpha_{(2^{m_2}-1)}$. For all other instances in which we end up with $\alpha_{(T)}$, at least one of the offline vertices will already have α set to η so like case 1, we are in good shape. Hence we just need to consider the case when one offline vertex, say i_1 , had x_{i_1} at $1 - \frac{1}{2^{2^{m_1}-1}}$ and the other vertex, i_2 , had x_{i_2} at $1 - \frac{1}{2^{2^{m_2}-1}}$.

First of all, we note that

$$\Delta P = \frac{1}{2^{2^{m_1}-1}} + \frac{1}{2^{2^{m_2}-1}} - \frac{2}{2^{2^{m_1}+2^{m_2}-1}}.$$

The plan is to put all the ΔP into both the α values and confirm that indeed we can make both the α values equal to η , without incrementing the β .

This can be achieved iff ΔP is at least $2\eta - \alpha_{(2^{m_1}-1)} - \alpha_{(2^{m_2}-1)}$. The following claim precisely shows this.

▷ **Claim 11.** $\Delta P \geq 2\eta - \alpha_{(2^{m_1}-1)} - \alpha_{(2^{m_2}-1)}$.

Proof. We show that $\Delta P - 2\eta + \alpha_{(2^{m_1}-1)} + \alpha_{(2^{m_2}-1)}$ is non-negative as follows.

$$\begin{aligned}
&\Delta P - 2\eta + \alpha_{(2^{m_1}-1)} + \alpha_{(2^{m_2}-1)} \\
&= \frac{1}{2^{2^{m_1}-1}} + \frac{1}{2^{2^{m_2}-1}} - \frac{2}{2^{2^{m_1}+2^{m_2}-1}} + \sum_{i=1}^{m_1-1} 2^{m_1-1-i} \left(\frac{1}{2^{2^i-2}} - \frac{1}{2^{2(2^i-2)+2}} \right) + \\
&\quad \sum_{i=1}^{m_2-1} 2^{m_2-1-i} \left(\frac{1}{2^{2^i-2}} - \frac{1}{2^{2(2^i-2)+2}} \right) + 2^{m_1-1} + 2^{m_2-1} - (2^{m_1} + 2^{m_2})\eta
\end{aligned}$$

$$\begin{aligned}
&= \frac{1}{2^{2^{m_1}-1}} + \frac{1}{2^{2^{m_2}-1}} - \frac{2}{2^{2^{m_1}+2^{m_2}-1}} + 2^{m_1-1} + 2^{m_2-1} - (2^{m_1} + 2^{m_2})\eta + \\
&\quad 2^{m_1-1} \left(\frac{1}{2} - \sum_{i=2}^{m_1-1} \frac{2}{2^{2^i+i-1}} - \frac{4}{2^{(2^{m_1}+m_1-1)}} \right) + \\
&\quad 2^{m_2-1} \left(\frac{1}{2} - \sum_{i=2}^{m_1-1} \frac{2}{2^{2^i+i-1}} - \frac{4}{2^{(2^{m_1}+m_1-1)}} \right) \\
&= -\frac{1}{2^{2^{m_1}-1}} - \frac{1}{2^{2^{m_2}-1}} - \frac{2}{2^{2^{m_1}+2^{m_2}-1}} + \\
&\quad 2^{m_1-1} \left(1 - 2 + 2 \sum_{i=1}^{\infty} \frac{1}{2^{2^i+i-1}} + \frac{1}{2} - \sum_{i=2}^{m_1-1} \frac{2}{2^{2^i+i-1}} \right) + \\
&\quad 2^{m_2-1} \left(1 - 2 + 2 \sum_{i=1}^{\infty} \frac{1}{2^{2^i+i-1}} + \frac{1}{2} - \sum_{i=2}^{m_2-1} \frac{2}{2^{2^i+i-1}} \right) \\
&= -\frac{1}{2^{2^{m_1}-1}} - \frac{1}{2^{2^{m_2}-1}} - \frac{2}{2^{2^{m_1}+2^{m_2}-1}} + 2^{m_1-1} \left(\sum_{i=m_1}^{\infty} \frac{2}{2^{2^i+i-1}} \right) + \\
&\quad 2^{m_2-1} \left(\sum_{i=m_2}^{\infty} \frac{2}{2^{2^i+i-1}} \right) \\
&\geq 0
\end{aligned}$$

◁

5 Fractional Matching

In this section, we prove a lower bound on the competitive ratio of any algorithm for fractional matching and then give an online algorithm that achieves the best attainable competitive ratio.

5.1 Lower Bound

► **Theorem 12.** *No online algorithm can attain a competitive ratio better than 0.75 for online fractional bipartite matching.*

Proof. Consider a graph with two offline vertices i_1 and i_2 . The first online vertex will have edges to both i_1 and i_2 while the second online vertex will have an edge to only one of the vertices. The optimal matching is of size 2 but one can see that no online algorithm can attain a competitive ratio better than 0.75 using this graph. ◀

5.2 Upper Bound

For the fractional matching, we shall consider the water-level algorithm which was an optimal algorithm for the case without the degree bound.

13:16 Optimal Online Bipartite Matching in Degree-2 Graphs

■ **Algorithm 2** Water-Level Algorithm.

```

1: procedure  $A(j, S)$  ▷ Online vertex  $j$  with neighborhood  $S \subseteq L$ ,  $|S| \leq 2$ 
2:   Find  $\ell$  such that  $\sum_{i \in S} \max\{0, \ell - x_i\} = 1$ 
3:   if  $\ell \leq 1$  then
4:     Vertex  $j$  can be completely matched.
5:     For all  $i \in S$  with  $x_i < \ell$ , set  $x_i = \ell$ 
6:   else
7:     All the neighbors of  $j$  can be completely matched, but  $j$  can only be partially
       matched.
8:     For all  $i \in S$  set  $x_i = 1$ 
9:   end if
10: end procedure

```

We show that the same algorithm gives the optimal algorithm for the degree 2 case.

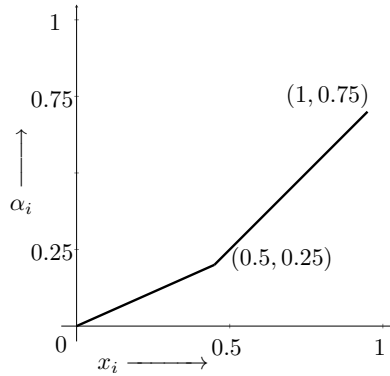
► **Theorem 13.** *Water-level algorithm attains a competitive ratio of 0.75 when the vertices have a degree at most 2.*

Proof. We shall use the primal-dual analysis for this. The main idea is to maintain the value of the fractional matching produced by water level and simultaneously maintain dual variables.

We already know how the primal variables are updated by water level, given the algorithm. We will now explain how the dual variables will be updated to maintain reverse weak duality and weak dual feasibility.

Let us define $x_i := \sum_{j \in R} x_{i,j}$. The variable x_i denotes how much the vertex i is matched. We will maintain the α_i 's such that, given x_i , we will know exactly what α_i is. Thus, we will know how much α_i increases by considering how much x_i was incremented by water level. Given the increments in α_i 's, we will know what value to set β_j because we will maintain that the primal and dual objectives will always be the same. So, the dual objective will be increased by the same amount as the primal, which is the amount of matching done by the water level algorithm on the arrival of online vertex j .

Hence, first, we will describe how to maintain the α_i 's.



For the first half fraction of the matching, for any vertex $i \in L$, if a fractional matching of size p is done between i and $j \in R$, then we shall increment α_i by $p/2$ and β_j by $p/2$. Whereas, if u is already matched more than 0.5, then for a matching of size x , we shall increment α_i by x , and we shall not increment β_j at all.

Firstly, note that we never decrease any dual variables. So if some dual constraint is satisfied at a given time then it will certainly be satisfied in the future. We shall now see that whenever a new vertex $j \in R$ arrives, both the newly introduced edges (i_1, j) and (i_2, j) satisfy approximate dual feasibility. This will complete the proof.

In order to observe the approximate dual feasibility, we shall look at a few cases based on how much i_1 and i_2 were matched right before the arrival of j .

1. i_1 and i_2 are completely unmatched.

Notice that in this case, the water-level algorithm will match j half to i_1 and a half to i_2 , and the primal increases by 1. Let us see how the dual variables are updated. Since x_{i_1} and x_{i_2} are both at 0.5 we will have $\alpha_{i_1} = \alpha_{i_2} = 0.25$ which adds 0.5 to the dual objective. Hence, we can add 0.5 to β_j .

Therefore $\alpha_{i_1} + \beta_j = \alpha_{i_2} + \beta_j = 0.75$

2. i_1 is x_{i_1} fraction matched and i_2 is completely unmatched

We will assume that i_1 is x_{i_1} fraction matched where $x_{i_1} \neq 0$. First, we note that the water-level algorithm maintains that if $x_{i_1} \neq 0$, then we must have $x_{i_1} \geq 0.5$. In other words, if a vertex $i \in L$ starts getting matched, it will be matched at least till the half-level mark. Hence we may assume $x_{i_1} \geq 0$.

In this case, water-level algorithm will first match x_{i_1} fraction of vertex i_2 to j and then $\frac{1-x_{i_1}}{2}$ to i_1 and $\frac{1-x_{i_1}}{2}$ to i_2 . This will mean that $\frac{1+x_{i_1}}{2}$ will be the updated value of x_{i_1} and x_{i_2} after vertex j is matched fractionally.

Hence, after the matching $\alpha_{i_1} = \alpha_{i_2} = 0.25 + \frac{x_{i_1}}{2}$ and $\beta_j = 0.25$. Therefore $\alpha_{i_1} + \beta_j = \alpha_{i_2} + \beta_j = 0.5 + \frac{x_{i_1}}{2} \geq 0.75$ since x_{i_1} was greater than $\frac{1}{2}$.

Note that the case where i_1 is completely unmatched but i_2 is partially matched is symmetric.

3. i_1 is x_{i_1} fraction matched and i_2 is x_{i_2} fraction matched

In this last case, when both i_1 and i_2 were partially matched to begin with (hence, at least half matched as stated earlier), we can see that the water-level algorithm would end up matching i_1 and i_2 to j - whatever fraction of i_1 and i_2 was still unmatched. After this, we will have $x_{i_1} = x_{i_2} = 1$ and therefore $\alpha_{i_1} = \alpha_{i_2} = 0.75$ and therefore $\alpha_{i_1} + \beta_j = \alpha_{i_2} + \beta_j = 0.75$. ◀

6 Conclusion

An interesting open problem is finding the right competitive ratios for the integral online bipartite matching for (online) degree d unweighted graphs. If we denote the competitive ratios by $\mathcal{F}(d)$, for fractional, and $\mathcal{I}(d)$, for integral, respectively, then we have shown that $\mathcal{I}(2) = 0.717772\dots$. Given that $\mathcal{F}(2) = 0.75$, it would be interesting to find the values of $\mathcal{I}(d)$ when $d \geq 3$ and compare them with $\mathcal{F}(d)$ especially given that we know both $\mathcal{I}(d)$ and $\mathcal{F}(d)$ approach $1 - \frac{1}{e}$ as d approaches infinity. We conjecture that $\mathcal{F}(d)$ and $\mathcal{I}(d)$ will be different for all $d \geq 2$.

It is also interesting to ask the more general question: given an algorithm that computes fractional solutions for a given class of graphs, how well can we round a fractional solution into a randomized integral solution?

References

- 1 Susanne Albers and Sebastian Schubert. Online ad allocation in bounded-degree graphs. In Kristoffer Arnsfelt Hansen, Tracy Xiao Liu, and Azarakhsh Malekian, editors, *Proc. 18th International Conference on Web and Internet Economics (WINE)*, volume 13778 of *LNCS*, pages 60–77. Springer, 2022. doi:10.1007/978-3-031-22832-2_4.

- 2 Susanne Albers and Sebastian Schubert. Tight bounds for online matching in bounded-degree graphs with vertex capacities. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *Proc. 30th Annual European Symp. of Algorithms (ESA)*, volume 244 of *LIPICs*, pages 4:1–4:16. Schloss Dagstuhl, 2022. doi:10.4230/LIPICs.ESA.2022.4.
- 3 Yossi Azar, Ilan Reuven Cohen, and Alan Roytman. Online lower bounds via duality. In Philip N. Klein, editor, *Proc. 28th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1038–1050, 2017. doi:10.1137/1.9781611974782.66.
- 4 Claude Berge. Two theorems in graph theory. *PNAS*, 43(9):842–844, 1957. doi:10.1073/pnas.43.9.842.
- 5 Guy Blanc and Moses Charikar. Multiway online correlated selection. In Jelani Nelson, editor, *Proc. 63rd IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 1277–1284, 2022. doi:10.1109/FOCS52979.2021.00124.
- 6 Niv Buchbinder, Kamal Jain, and Joseph (Seffi) Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In Lars Arge, Michael Hoffmann, and Emo Welzl, editors, *Proc. 15th Annual European Symp. of Algorithms (ESA)*, volume 4698 of *LNCS*, pages 253–264. Springer, 2007. doi:10.1007/978-3-540-75520-3_24.
- 7 Niv Buchbinder and Joseph (Seffi) Naor. The design of competitive online algorithms via a primal-dual approach. *Found. Trends Theor. Comput. Sci.*, 3(2-3):93–263, 2009. doi:10.1561/04000000024.
- 8 Niv Buchbinder, Joseph (Seffi) Naor, and David Wajc. Lossless online rounding for online bipartite matching (despite its impossibility). In Nikhil Bansal and Viswanath Nagarajan, editors, *Proc. 34th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 2030–2068, 2023. doi:10.1137/1.9781611977554.CH78.
- 9 Deeparnab Chakrabarty and Gagan Goel. On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and GAP. *SIAM J. Comput.*, 39(6):2189–2211, 2010. (Preliminary version in *49th FOCS*, 2008). doi:10.1137/080735503.
- 10 Denis Xavier Charles, Max Chickering, Nikhil R. Devanur, Kamal Jain, and Manan Sanghi. Fast algorithms for finding matchings in lopsided bipartite graphs with applications to display ads. In David C. Parkes, Chrysanthos Dellarocas, and Moshe Tennenholtz, editors, *Proc. 11th ACM Conf. Electronic Commerce (EC)*, pages 121–128. ACM, 2010. doi:10.1145/1807342.1807362.
- 11 Ilan Reuven Cohen and Binghui Peng. Primal-dual schemes for online matching in bounded degree graphs. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *Proc. 31st Annual European Symp. of Algorithms (ESA)*, volume 274 of *LIPICs*, pages 35:1–35:17. Schloss Dagstuhl, 2023. doi:10.4230/LIPICs.ESA.2023.35.
- 12 Ilan Reuven Cohen and David Wajc. Randomized online matching in regular graphs. In Artur Czumaj, editor, *Proc. 29th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 960–979, 2018. doi:10.1137/1.9781611975031.62.
- 13 Nikhil R. Devanur and Thomas P. Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In John Chuang, Lance Fortnow, and Pearl Pu, editors, *Proc. 10th ACM Conf. on Electronic Commerce (EC)*, pages 71–78. ACM, 2009. doi:10.1145/1566374.1566384.
- 14 Nikhil R. Devanur, Kamal Jain, and Robert D. Kleinberg. Randomized primal-dual analysis of RANKING for online bipartite matching. In Sanjeev Khanna, editor, *Proc. 24th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 101–107, 2013. doi:10.1137/1.9781611973105.7.
- 15 Tomer Ezra, Michal Feldman, Nick Gravin, and Zhihao Gavin Tang. Online stochastic max-weight matching: Prophet inequality for vertex and edge arrival models. In Péter Biró, Jason D. Hartline, Michael Ostrovsky, and Ariel D. Procaccia, editors, *Proc. 21st ACM Conf. Economics and Comput. (EC)*, pages 769–787. ACM, 2020. doi:10.1145/3391403.3399513.
- 16 Matthew Fahrbach, Zhiyi Huang, Runzhou Tao, and Morteza Zadimoghaddam. Edge-weighted online bipartite matching. *J. ACM*, 69(6):45:1–45:35, 2022. (Preliminary version in *61st FOCS*, 2020). doi:10.1145/3556971.

- 17 Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S. Mirrokni, and Clifford Stein. Online stochastic packing applied to display ad allocation. In Mark de Berg and Ulrich Meyer, editors, *Proc. 18th Annual European Symp. of Algorithms (ESA), Part I*, volume 6346 of *LNCS*, pages 182–194. Springer, 2010. doi:10.1007/978-3-642-15775-2_16.
- 18 Jon Feldman, Nitish Korula, Vahab S. Mirrokni, S. Muthukrishnan, and Martin Pál. Online ad assignment with free disposal. In Stefano Leonardi, editor, *Proc. 5th International Workshop on Internet and Network Economics (WINE)*, volume 5929 of *LNCS*, pages 374–385. Springer, 2009. doi:10.1007/978-3-642-10841-9_34.
- 19 Jon Feldman, Aranyak Mehta, Vahab S. Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating $1 - \frac{1}{e}$. In Daniel A. Spielman, editor, *Proc. 50th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 117–126, 2009. doi:10.1109/FOCS.2009.72.
- 20 Buddhima Gamlath, Michael Kapralov, Andreas Maggiori, Ola Svensson, and David Wajc. Online matching with general arrivals. In David Zuckerman, editor, *Proc. 60th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 26–37, 2019. doi:10.1109/FOCS.2019.00011.
- 21 Ruiquan Gao, Zhongtian He, Zhiyi Huang, Zipei Nie, Bijun Yuan, and Yan Zhong. Improved online correlated selection. In Nisheeth Vishnoi, editor, *Proc. 62nd IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 1265–1276, 2021. doi:10.1109/FOCS52979.2021.00123.
- 22 Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In Shang-Hua Teng, editor, *Proc. 19th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 982–991, 2008. URL: <http://dl.acm.org/citation.cfm?id=1347082.1347189>.
- 23 Bernhard Haeupler, Vahab S. Mirrokni, and Morteza Zadimoghaddam. Online stochastic weighted matching: Improved approximation algorithms. In Ning Chen, Edith Elkind, and Elias Koutsoupias, editors, *Proc. 7th International Workshop on Internet and Network Economics (WINE)*, volume 7090 of *LNCS*, pages 170–181. Springer, 2011. doi:10.1007/978-3-642-25510-6_15.
- 24 Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In Harriet Ortiz, editor, *Proc. 22nd ACM Symp. on Theory of Computing (STOC)*, pages 352–358, 1990. doi:10.1145/100216.100262.
- 25 Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. doi:10.1002/nav.3800020109.
- 26 Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. *Games Econ. Behav.*, 55(2):270–296, 2006. doi:10.1016/J.GEB.2005.02.006.
- 27 Aranyak Mehta. Online matching and ad allocation. *Found. Trends Theor. Comput. Sci.*, 8(4):265–368, 2013. doi:10.1561/04000000057.
- 28 Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5):22, 2007. (Preliminary version in *46th FOCS*, 2005). doi:10.1145/1284320.1284321.
- 29 Joseph (Seffi) Naor and David Wajc. Near-optimum online ad allocation for targeted advertising. *ACM Trans. Economics and Comput.*, 6(3-4):16:1–16:20, 2018. (Preliminary version in *16th EC*, 2015). doi:10.1145/3105447.
- 30 Yongho Shin and Hyung-Chan An. Making three out of two: Three-way online correlated selection. In Hee-Kap Ahn and Kunihiro Sadakane, editors, *Proc. 32nd International Symposium on Algorithms and Computation (ISAAC)*, volume 212 of *LIPICs*, pages 49:1–49:17. Schloss Dagstuhl, 2021. doi:10.4230/LIPICs.ISAAC.2021.49.