Online Algorithms for Bipartite Matching

Locke Cai* **Adam Zweiger*** **Anand John***

Abstract

This reading project explores the evolution of online algorithms for bipartite matching, focusing on three variants: unweighted, vertex-weighted, and edge-weighted scenarios. We analyze the techniques and competitive analyses presented in three seminal papers, culminating in an exploration of *Online Correlated Selection* (OCS), a recent breakthrough for edge-weighted matching. We simplify the key ideas and reconstruct the underlying lines of reasoning, and provide insight into how these algorithms were discovered. We also contrast the different techniques, exploring how and why ideas from one scenario may or may not be transferable to another.

1 Introduction

The online bipartite matching problem is a fundamental problem in discrete algorithms. Consider a bipartite graph $G = (L, R, E)$ consisting of a left set of vertices L and a right set of vertices R, with a set of edges E connecting vertices in L with vertices in R. Here, L is the set of offline vertices that are known to the algorithm beforehand, while R is the set of online vertices that arrive sequentially, from 1 to n. When an online vertex $j \in R$ arrives, the edges that involve j are revealed to our algorithm, which must make an irreversible (possibly randomized) decision to match j to some offline vertex $i \in L$ such that $(i, j) \in E$ or leave it unmatched. Each vertex must be involved in at most one match. In the unweighted, vertex-weighted, and edge-weighted versions of the problem, the goal is to design an algorithm with a good competitive ratio.

In the unweighted case, we seek to maximize the competitive ratio of the total number of matchings that are made. Specifically, we maximize the worst-case ratio over all possible graphs between the expected number of matchings made by our algorithm and the number of matchings made by the optimal offline algorithm. The work of Karp et al. (STOC 1990) introduced RANKING, a randomized algorithm that achieves a competitive ratio of $1 - 1/e$, which is provably optimal for this problem.

In the vertex-weighted case, the problem generalizes to maximizing the total weight of matched vertices in L. Each offline vertex $i \in L$ is assigned a weight $w_i \geq 0$, and the objective is to maximize the sum of weights of matched vertices. Aggarwal et al. (SODA 2011) extend the ideas of Ranking to this setting, introducing PERTURBED-GREEDY, which achieves the same optimal competitive ratio of $1 - 1/e$ by perturbing the weights of offline vertices using random multiplicative factors and selecting matches based on these perturbed weights.

Finally, in the edge-weighted case, the edges rather than vertices have weights. The goal is to maximize the total weight of matched edges, which can vary across different pairs of vertices. Here, we require an additional assumption called *free-disposal*. Under the free-disposal model, an online vertex may dispose of a previously matched edge to make room for a new heavier one. Equivalently, we can assign an offline vertex to multiple online vertices, but we only count the highest-weight edge. Note that in the unweighted and vertex-weighted problems, this assumption does not provide any benefit—an online vertex matching with a previously matched vertex is at most as good as keeping

*Equal contribution.

the old edge. However, this model is a natural assumption in the edge-weighted case, and necessary to get any competitive algorithm, which we will briefly show as well. In this case, Fahrbach et al. (FOCS 2020) overcome the long-standing $1/2$-competitive ratio barrier in this setting by introducing a novel technique called *Online Correlated Selection* (OCS). This achieves a competitive ratio of 0.5086 by negatively correlating decisions across edge pairs, using a combination of a primal-dual framework and a complementary cumulative distribution function (CCDF) viewpoint to analyze match probabilities and dual feasibility. We will take a look at a slightly simplified version of the algorithm that achieves a competitive ratio of 0.505.

We will explore the key ideas behind these algorithms and reconstruct the lines of thought that led to each of their discovery.

2 Unweighted Online Bipartite Matching

2.1 Deterministic Algorithms

We will begin by exploring the simplest problem setting—unweighted online bipartite matching.

Before we develop and analyze RANKING, let us first explore a simple deterministic greedy algorithm (GREEDY) to develop some intuition on the problem. GREEDY employs a simple strategy: as a new vertex v is revealed from R, if it can be matched with any vertex in L, then we match it to one of them with some deterministic strategy. Otherwise, we simply ignore v. The algorithm is formally described below:

Algorithm 1 GREEDY Algorithm for Online Bipartite Matching

Input: Offline vertices L, online vertices R (arrive sequentially), and edges E.
Initialize: $M = \emptyset$ (an initially empty matching)
for each online vertex $v \in R$ as it arrives **do**
  if $\exists u \in L$ such that $(u, v) \in E$ and u is unmatched in M **then**
    Match v to such an unmatched u deterministically
    Update $M = M \cup \{(u, v)\}$
  else
    Leave v unmatched
  end if
end for
Output: Final matching M

While the GREEDY algorithm is intuitive and simple, it yields a suboptimal competitive ratio of $1/2$. Consider the following adversarial example. We are given a graph where $L = \{u_1, u_2\}$, $R = \{v_1, v_2\}$, and $E = \{(u_1, v_1), (u_1, v_2), (u_2, v_2)\}$, as shown below. The adversary would first reveal v_2, thereby revealing (u_1, v_2) and (u_2, v_2). Now, our greedy algorithm can make 2 choices, either add (u_1, v_2) or (u_2, v_2) to M, and both choices are completely symmetrical from the perspective of the algorithm. Thus, since GREEDY is purely deterministic, the adversary can always re-indexed the graph such that GREEDY chooses to put (u_1, v_2) in M. Then, when v_1 is revealed, no more matches can be formed. Thus, GREEDY find a matching of size 1, but the optimal matching clearly has a size of 2.

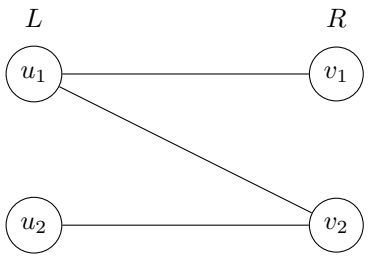


Figure 1: Adversarial example for the GREEDY algorithm.

To generalize this example to an arbitrarily large size, we can simply repeat the above "component" arbitrary number of times. For each component, 2 matches could be formed, but GREEDY would only be able to find 1. Thus, overall, GREEDY only achieves a competitive ratio of $1/2$.

In fact, any deterministic algorithm can only achieve competitive ratio of at most $1/2$. Consider Figure 1 again, and let's consider any deterministic algorithm A. Our adversary again reveal v_2 to A. If A choose to make a match, then, we can proceed with our above analysis and conclude that A would only find 1 match. If A chooses to not make a match, then A can also only make at most 1 match because at most 1 match can be formed when v_1 is revealed. Again, generalizing to arbitrarily large graphs, we can show that A is at most $1/2$ competitive.

2.2 The RANKING Algorithm

As shown from the above section, to overcome the $1/2$ competitive-ratio barrier, we must employ randomization. One such randomized algorithm is RANKING, which proceeds as follows. Prior to receiving any vertices from R, RANKING first generates a random permutation of all vertices in L, which we denote σ. Let $\sigma(v)$ be the index of v in σ. Then, we define the rank of v as $r(v) = |L| + 1 - \sigma(v)$. Intuitively, the rank decreases as we iterate σ from left to right. When a new vertex v_r arrives, just like GREEDY, we find all vertices in L that have not been matched and could be matched to v_r, and we match v_r to the highest ranked option if there is any option at all. The RANKING algorithm is formally described below:

Algorithm 2 RANKING Algorithm for Online Bipartite Matching

Input: Offline vertices L, online vertices R (arrive sequentially), and edges E.
Preprocessing:
Generate a random permutation σ of the offline vertices L.
Define the rank of each offline vertex $v \in L$ as $r(v) = |L| + 1 - \sigma(v)$.
Initialize: $M = \emptyset$ (an initially empty matching).
for each online vertex $v_r \in R$ as it arrives **do**
  Let $C = \{v_\ell \in L \mid (v_\ell, v_r) \in E \text{ and } v_\ell \text{ is unmatched in } M\}$.
  if $C \neq \emptyset$ **then**
    Match v_r to the unmatched vertex $v_\ell \in C$ with the highest rank $r(v_\ell)$.
    Update $M = M \cup \{(v_\ell, v_r)\}$.
  else
    Leave v_r unmatched.
  end if
end for
Output: Final matching M.

Overall, RANKING is very similar to GREEDY. In fact, the only difference is that RANKING chooses the vertex to match via a random permutation, but this key step ensures that RANKING does not make deterministic choices that can be exploited by the adversary. More concretely, consider Figure 1 again. When the adversary reveals v_2, it can no longer guarantee that RANKING chooses (u_1, v_2) as the match due to randomization. Thus, the same adversarial example would not work on RANKING.

In fact, RANKING achieves a provably optimal competitive ratio of $1 - 1/e$. To intuitively see why there exists a constant upper bound on the competitive ratio, recall that Yao's Minimax Principle allows us to bound the performance of randomized algorithms by bounding the performance of deterministic algorithms on a carefully chosen distribution of inputs. As we showed earlier, deterministic algorithms can be easily exploited to reach a low competitive ratio of $1/2$. Thus, it is reasonable that there exist a input distribution where all deterministic algorithms perform suboptimally, thus leading an upper bound on the competitive ratio of randomized algorithms.

However, we find the detailed proofs provided by Karp et al. for both achievability and optimality of $1 - 1/e$ unintuitive and uneducational, so we omit them for now. We will instead move on to discuss the more general problem of vertex-weighted online bipartite matching, where we will explain the competitive ratio of $1 - 1/e$.

3 Vertex-Weighted Online Bipartite Matching

In the vertex-weighted problem setting, a weight $w_v > 0$ is introduced for every vertex $v \in L$, and our goal is to maximize the total weight of all vertices in L that appears in the final matching. Just like the unweighted case, to yield a good competitive ratio, we must employ randomization. Naively applying RANKING would not yield a good competitive ratio as it does not consider the weights at all. Nonetheless, RANKING remains a good starting point for tackling the vertex-weighted version. In fact, it turns out, if we bias the rankings by the weights in a certain way, we can achieve the same optimal competitive ratio of $1 - 1/e$ as the unweighted case. Specifically, PERTURBED-GREEDY, introduced by Aggarwal et al, achieves this bound.

3.1 The PERTURBED-GREEDY Algorithm

PERTURBED-GREEDY is identify to RANKING except for their definition of the ranking function. In PERTURBED-GREEDY, we first define function $\psi(x) := 1 - e^{-(1-x)}$. Note that $\psi(\cdot)$ is a decreasing function. Then, for every $u \in L$, we sample x_u uniformly from $[0, 1]$. Finally, we define the rank function as $r(u) := \psi(x_u)w_u$ where w_u is the weight of vertex u. When a new vertex $v \in R$ arrives, just like RANKING, we find all vertices in L that could be matched with v, and we create a match with the one with the highest ranking. The formal algorithm is shown below:

Algorithm 3 PERTURBED-GREEDY Algorithm for Vertex-Weighted Online Bipartite Matching

Input: Offline vertices L, online vertices R (arrive sequentially), edges E, and weights $w_u > 0$ for all $u \in L$.
Preprocessing:
Define function $\psi(x) := 1 - e^{-(1-x)}$ for $x \in [0, 1]$.
for each $u \in L$ **do**
  Sample x_u uniformly from $[0, 1]$.
  Define rank $r(u) := \psi(x_u)w_u$, where w_u is the weight of vertex u.
end for
Initialize: $M = \emptyset$ (an initially empty matching).
for each online vertex $v_r \in R$ as it arrives **do**
  Let $C = \{u \in L \mid (u, v_r) \in E \text{ and } u \text{ is unmatched in } M\}$.
  if $C \neq \emptyset$ **then**
    Match v_r to the unmatched vertex $u \in C$ with the highest rank $r(u)$.
    Update $M = M \cup \{(u, v_r)\}$.
  else
    Leave v_r unmatched.
  end if
end for
Output: Final matching M.

Before we formally analyze the competitive-ratio of PERTURBED-GREEDY, we first note that PERTURBED-GREEDY is strictly a generalization of RANKING. Indeed, if all weights of $u \in L$ are identical, then all $r(u)$ essentially samples of $\psi(X)$ where $X \sim \text{Unif}([0, 1])$. Thus, all orderings of vertices in L are equally likely to be induced by $r(\cdot)$. As a result, the algorithm reduces to RANKING. Consequently, our competitive analysis in this setting also holds for the unweighted setting. We now give the proof outline for PERTURBED-GREEDY's competitive ratio of $1 - 1/e$.

3.2 Competitive Analysis of PERTURBED-GREEDY: Intuitions

Theorem 1. *PERTURBED-GREEDY achieves a competitive-ratio of* $1 - 1/e$ *for the vertex-weighted online bipartite matching problem.*

As we aim to provide an intuitive outline of the proof, we make a few simplifying assumptions. In our graph $G = (L, R, E)$, we assume that $|L| = |R|$ and that there exist a perfect matching. Next, we "quantize" our definition of our ranking function for ease of analysis. Specifically, for a fixed integer $k > 0$, we redefine $\psi(i) := 1 - (1 - \frac{1}{k})^{-(k-i+1)}$, and instead of uniformly sampling x_u from $[0, 1]$, we sample $\sigma(u)$ uniformly from the set $\{1, 2, ..., k\}$. Overall, our ranking function is

now $r(u) := \psi(\sigma(u))w_u$. For convenience, we denote $\sigma(u)$ as the "order" of u. Such quantization is permitted because as $k \to \infty$, we can show our quantized ranking function is equivalent to the original definition. In addition, note that conditioned on choice of σ, PERTURBED-GREEDY is deterministic.

In our analysis, we also assume the adversary chooses a fixed order π of revealing vertices in R. Note that it is sufficient to show that PERTURBED-GREEDY achieves a competitive-ratio of $1 - 1/e$ for all choice of π

Now, we begin by specifying a few necessary definitions.

Definition 3.1. *Q_t denotes the set of (σ, t, u) such that $\sigma(u) = t$ and u is matched by PERTURBED-GREEDY conditioned on σ. Formally:*

$$Q_t = \{(\sigma, t, u) : \sigma(u) = t \text{ and } u \text{ is matched}\}$$

Intuitively, such definitions allows us to easily express the expected contributions towards the total weight of the final matching from all vertices u with order t. Formally:

$$\text{Expected "gain" at order } t = x_t = \frac{\sum_{(\sigma,t,u)\in Q_t} w_u}{k^n}$$

Note that the expected total weight of the matching produced by PERTURBED-GREEDY would be $\sum_{t=1}^{k} x_t$.

Next, let M^* be any perfect matching on G, and for any $u \in L$, we denote u^* as the vertex that u is matched to in M^*. In addition, for ease of notation, for a σ, we define σ_u^i as a modified version of σ where $\sigma(u) = i$.

Now, we are ready to define a "charging map":

$$f(\sigma, t, u) = \{(\sigma_u^i, s, u') : 1 \le i \le k, \text{PERTURBED-GREEDY matches } u^* \text{ to } u' \text{ in } \sigma_u^i \text{ where } \sigma_u^i(u') = s\}$$

Intuitively, the charging map specifies what u^* would be matched to as we varies the order of u, and by definition, we get the following lemma:

Lemma 3.1. *If $(\rho, s, u') \in f(\sigma, t, u)$, then we must have $(\rho, s, u') \in Q_s$.*

Now, we present a crucial lemma that specifies how the behaviors of PERTURBED-GREEDY changes as we vary the order of a vertex $u \in L$.

Lemma 3.2. *If vertex $u \in L$ with order t and rank r_u is unmatched by PERTURBED-GREEDY, then for every $1 \le i \le k$, conditioned on σ_u^i, PERTURBED-GREEDY matches u^* to vertex u' such that $r_u \le r(u')$*

Proof. We consider two cases based on the relationship between i (the modified order) and t (the original order of u).

Case 1: $i \ge t$ (the order of u decreases): When $i \ge t$, u's order decreases relative to its original order in σ, which consequently decreases the rank of u.Since u was unmatched in the original ranking σ, it follows that u^* must have been matched in σ, otherwise it would be matched to u when it is revealed. Thus, under σ, at the time u^* arrives, it must have found a higher-ranked match than u because otherwise it would match to u. Thus, if we let u^* match to v under σ, then $r(v) \ge r_u$.

Under σ_u^i, when $u*$ is revealed, it would still match to v as the rank of u is now even lower. Thus, we have $r(u') \ge r_u$

Case 2: $i < t$ (the order of u increases): When $i < t$, u's order increases relative to its original order, so the rank of u also increased above r_u.

We consider three subcases:

- **Subcase 2.1: u is matched to u^*:** In this case, u^*'s match is preserved, and the lemma holds trivially since u^* is directly matched to u, whose rank has increased.

- **Subcase 2.2: u^* is matched before u get matched:** In this case, since u remains unmatched when u^* arrives, u^* would match to the same vertex conditioned on both σ and σ_u^i. Thus, following analysis from Case 1, we conclude that $r(u') \geq r_u$.
- **Subcase 2.3: u is matched before u^*:** If u is matched before u^*, it must have displaced some vertex u'' that was previously matched to the same online vertex. This displacement frees up u'', potentially increasing the options available for all vertices that come after, including u^*. Thus, the rank of u^*'s match can only increase when conditioned on σ_u^i. Thus, we can conclude that $r(u') \geq r_u$.

Conclusion: In all cases, modifying σ to σ_u^i ensures that u^* is matched to a vertex u' where $r(u') \geq r(u)$. This completes the proof. □ □

While Lemma 3.2 does not seem to be directly related to the competitiveness of PERTURBED-GREEDY, it establishes an important corollary.

Corollary 3.1. *For all $(\sigma, t.u) \in R_t$, $1 \leq t \leq k$, $f(\sigma, t, u)$ contains k values.*

Intuitively, this is because in our proof, if u is not matched under σ, then u^* is always matched for any σ_u^i.

In addition, Lemma 3.2 establishes that $r_u \leq$ average of $r(u')$. Combining with the above corollary, we get a concrete relation between rankings:

$$\psi(t)w_u \leq \frac{1}{k} \sum_{(\sigma_u^i, s, u') \in f(\sigma,t,u)} \psi(s)w_{u'} \tag{1}$$

However, equation (1) alone is not sufficient to analyze the competitiveness of PERTURBED-GREEDY as it does not involve x_t. Since x_t is defined based on Q_t, naturally, we would like to employ Lemma 3.1 in some way. In particular, if we could find a set S of (σ, u, t) such that $f(s_1)$ and $f(s_2)$ are disjoint for any $s_1 \neq s_2 \in S$, then we could get the following inequality

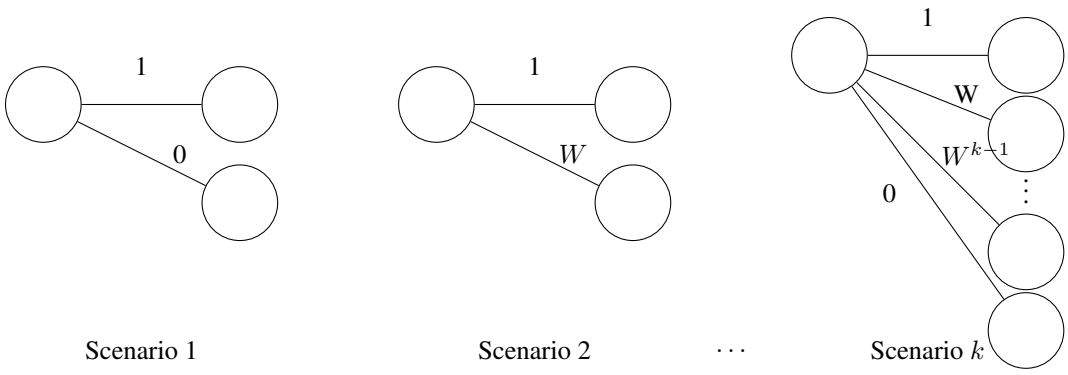
$$\sum_{(\sigma,t,u) \in S} \psi(t)w_u \leq \frac{1}{k} \sum_{t=1}^{k} \sum_{(\sigma,t,u) \in Q_t} \psi(t)w_u = \frac{1}{k} \sum_{t=1}^{k} x_t k^n \tag{2}$$

To complete the proof, we simply need to find a suitable S and perform appropriate algebraic manipulation. We omit the remaining proof details for clarity.

The proof that no algorithm can surpass this bound involves looking at the complete upper-triangular matrix and apply Yao's lemma, showing that the expected performance of any randomized algorithm is bounded by that of the best deterministic algorithm under a probabilistic input distribution. For deterministic algorithms, the analysis demonstrates that their expected performance is equivalent to RANDOM on this worst-case input, yielding a maximum expected match size of $(n(1 - 1/e)$, which aligns with the performance of the RANKING algorithm. Then, $1 - 1/e$ is shown to be a tight upper bound.

4 Edge-Weighted Online Bipartite Matching

Finally, in the edge-weighted online bipartite matching setting, we generalize the unweighted case in a different way: arbitrary nonnegative weights are assigned to *edges* rather than offline vertices. In this setting, we pose the additional assumption that we have *free-disposal*, meaning we can assign an offline vertex to multiple online vertices, but we only count the highest-weight edge. To see why this assumption is necessary to achieve any competitive algorithm, consider the series of scenarios below in which the online vertices are on the right and the offline vertices are on the left. It is easy to see that no deterministic algorithm can distinguish between the first two scenarios after just the first online vertex, thus the competitive ratio of any deterministic algorithm approaches 0 by taking W sufficiently large.



Even with randomized algorithms, we can consider a more generalized scenario k in which an adversary may stop a series of increasing weighted edges at any point, and in the limit possible scenarios k, no algorithm can be more than $O(1/W)$ competitive, which vanishes to 0 in the limit of W. With the assumption of free-disposal, all of these scenarios can be easily dealt with by taking every increased-weight edge, disposing of the previous ones.

For this problem, the absence of symmetry in edge weights and the need to account for the contribution of each offline vertex through its heaviest edge make the earlier techniques of RANKING and PERTURBED-GREEDY insufficient. These algorithms rely on vertex-level structures, which fail to generalize to edge-level dynamics. For example, we can't simply take a random priority ranking of the vertices and match online vertices to the highest ranked vertex since this doesn't factor in the edge weights. We can't perturb this ranking by the edge weights in advance because each offline vertex has a different set of edge weights with each online vertex.

Instead, we need to introduce a new idea called online correlated selection (OCS), which is the central innovation in achieving a competitive ratio greater than $1/2$. This idea proposed by Fahrbach et al. (FOCS 2020) takes a sequence of pairs of vertices and selects a single vertex from each one, negatively correlating decisions across pairs. This will be used in the main edge-weighted matching algorithm where we will look at sequences of pairs of offline vertices, and we will apply the method to negatively correlate decisions across time, leading to a higher competitive ratio.

4.1 Online Correlated Selection

Unlike independent random bits used in previous randomized approaches, OCS introduces carefully controlled negative correlation across edge selections. This improves the efficiency of the matching process by ensuring that offline vertices are not "overloaded" with matches while leaving other offline vertices basically unmatched.

OCS is a fairly general online method that can be applied for any sequence of pairs. The goal is to select a single element from each of a sequence of pairs of elements to maximize the probability that any given element is selected, especially those that appear more often.

In particular, we want to maximize γ such that if an element appears in $k \geq 1$ pairs, it is selected at least once with probability at least

$$1 - 2^{-k}(1-\gamma)^{k-1}.$$

The full definition of a γ-OCS is a bit more complicated, dealing with subsequences of consecutive pairs containing i, but is unnecessary to understand the key idea of this algorithm, so for the sake of clarity we will work with our simplified definition—what is called λ-semi-OCS.

The full version of OCS achieves a competitive ratio of 0.5086, breaking the $1/2$-barrier for the first time. For simplicity, we analyze a slightly weaker variant that still beats this barrier with a competitive ratio of 0.505, using a $1/16$-OCS.

Consider the following algorithm:

Algorithm 4 Online Correlated Selection (OCS)

Input: Sequence of pairs of elements $P = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$.
State Variables: $\tau_i \in \{\text{selected, not selected, unknown}\}$ for each element i.
Initialize $\tau_i = \text{unknown}\ \forall i$.
for each pair $(x, y) \in P$ **do**
With probability $1/2$, mark the pair as a **sender**:

1. Draw $z \in \{x, y\}$ uniformly at random.
2. Mark $\tau_z = \text{selected}$ and $\tau_{z'} = \text{unknown}$, where $z' \neq z$.

Otherwise (With probability $1/2$), mark the pair as a **receiver**:

1. Draw $z \in \{x, y\}$ uniformly at random.
2. **If** $\tau_z = \text{selected}$, select z' (the other element); **else if** $\tau_z = \text{not selected}$, select z.
3. **Otherwise** (if $\tau_z = \text{unknown}$), select z or z' uniformly at random.
4. Reset τ_x and τ_y to unknown.

end for
Output: Selected elements from the sequence.

It can be shown that the probability f_k that i is never selected after k pairs containing i is given by $f_k = \frac{1}{2} f_{k-1} - \frac{1}{4} \cdot \frac{1}{16} f_{k-2}$, with base cases $f_0 = 1$ and $f_1 = \frac{1}{2}$. This solves to $f_k = 2^{-k}(1-\gamma)^{k-1}$ for $\gamma = \frac{1}{16}$. Therefore, this algorithm is a $\frac{1}{16}$-semi-OCS. Proving this is essentially just casework on the state of the k-th pair and the two pairs before it.

4.2 Complementary Cumulative Distribution Function

The next key idea that is needed to understand the edge-weighted algorithm is the complementary cumulative distribution function (CCDF) of the heaviest edge weight matched to i. This will allow us to keep track of an offline vertex's current probability distribution over its edge weights. For an offline vertex $i \in L$ and some $w \geq 0$, we let $y_i(w)$ the be the probability that i is matched to some online vertex j such that $w_{ij} \geq w$. This is called the complementary cumulative distribution function because it is the complement (1 minus) the CDF of the distribution of weights that i might be matched to. The CCDF $y_i(w)$ is a non-increasing step function function of w, and is between 0 and 1. The expected weight of i's heaviest edge is the area under $y_i(w)$,

$$\int_0^\infty y_i(w)\, dw.$$

4.3 Online Primal-Dual Framework

The algorithm operates within an **online primal-dual framework**, where the primal and dual objectives are defined as follows:

Primal Variables: Let x_{ij} denote the probability that edge (i, j) is the heaviest edge matched to offline vertex i. We want to maximize the weight of the matching as well as ensure that the probability distributions for the vertices and edges are valid.

$$\max \sum_{(i,j)\in E} w_{ij} x_{ij} \quad \text{subject to} \quad \sum_{j \in R} x_{ij} \leq 1\ \forall i \in L, \quad \sum_{i \in L} x_{ij} \leq 1\ \forall j \in R, \quad x_{ij} \geq 0.$$

Dual Variables: Define α_i and β_j as the dual variables associated with offline and online vertices, respectively. α_i will represent the integral of the CCDF for a given i which as shown above is the expected value of it's maximum matched edge. β_j is more of a scalar indicator variable, going from 0 to a scalar value to indicate that the online vertex has arrived. The dual objective is:

$$\min \sum_{i \in L} \alpha_i + \sum_{j \in R} \beta_j \quad \text{subject to} \quad \alpha_i + \beta_j \geq w_{ij}\ \forall (i, j) \in E, \quad \alpha_i, \beta_j \geq 0.$$

An online algorithm achieves a competitive ratio Γ if it ensures:

1. **Approximate Dual Feasibility:** $\alpha_i + \beta_j \geq \Gamma \cdot w_{ij}$ for all edges (i, j).

2. **Reverse Weak Duality:** The dual objective satisfies $D \leq P$, where P is the primal objective.

By maintaining primal and dual feasibility under these conditions, the algorithm ensures that the primal objective is at least $\Gamma \cdot \text{OPT}$, where OPT is the offline optimal.

By keeping track of this, we're able to see when we make online decisions, what decision will be good since we know how far away from optimal we are by comparing the primal and the dual.

4.4 Implementation in the Edge-Weighted Setting

Now, we have the key components of the edge-weighted algorithm. The algorithm can be though of as a modified greedy algorithm. For each online vertex j, the algorithm identifies potential matches by computing an "offer" for each offline vertex i. Offers are determined based on the dual variables and the CCDF of the heaviest edge weight for i.

A pure greedy solution would just pick the one with the best offer. However, this solution is not competitive. What is done instead is that we threshold the offers based on some calculate value. If there are at least two vertices with acceptable offers, we pick two of them using our OCS subroutine. If there is only one, we just choose that one deterministically. Otherwise we leave j unmatched.

By combining a pseudo-greedy idea with the correlated matching from OCS that incentivizes all the edges to be matched if they're especially matched to low value edges, we get an algorithm that finally breaks the 0.5 competitive threshold.

Algorithm 5 Edge-Weighted Online Bipartite Matching (Primal-Dual Algorithm)

Input: Offline vertices L, online vertices R (arrive sequentially), edge weights w_{ij} for $(i, j) \in E$.
State Variables:
$k_i(w) \geq 0$: Number of randomized rounds in which offline vertex i was a candidate with edge weight $\geq w$.
$k_i(w) = \infty$: If i is chosen in a deterministic round with edge weight $\geq w$.
Initialize: Matching $M = \emptyset$, $k_i(w) = 0 \; \forall i \in L, \forall w > 0$.
for each online vertex $j \in R$ as it arrives **do**
  Step 1: Compute Dual Increments
  for each offline vertex $i \in L$ **do**
    Compute $\Delta_i^R \beta_j = \int_0^{w_{ij}} b(k_i(w))\, dw - \frac{1}{2} \int_{w_{ij}}^{\infty} \sum_{0 \leq \ell < k_i(w)} a(\ell)\, dw$.
    Compute $\Delta_i^D \beta_j = \kappa \cdot \Delta_i^R \beta_j = \kappa \cdot \Big(\int_0^{w_{ij}} b(k_i(w))\, dw - \frac{\kappa}{2} \int_{w_{ij}}^{\infty} \sum_{0 \leq \ell < k_i(w)} a(\ell)\, dw \Big)$.
  end for
  Step 2: Identify Candidate Vertices
  $i^* \leftarrow \arg\max_i \Delta_i^D \beta_j$ ▷ Candidate for deterministic match.
  $i_1, i_2 \leftarrow \arg\max_{i_1, i_2} (\Delta_{i_1}^R \beta_j + \Delta_{i_2}^R \beta_j)$ ▷ Candidates for randomized match.
  Step 3: Decide Matching Type
  if $\Delta_{i_1}^R \beta_j + \Delta_{i_2}^R \beta_j < 0$ and $\Delta_{i^*}^D \beta_j < 0$ **then**
    Leave j unmatched.
  else if $\Delta_{i_1}^R \beta_j + \Delta_{i_2}^R \beta_j \geq \Delta_{i^*}^D \beta_j$ **then**
    Match j to either i_1 or i_2 using the OCS algorithm.
    Update $M = M \cup \{(i, j)\}$ and update $k_i(w)$ values accordingly.
  else
    Match j deterministically to i^*.
    Update $M = M \cup \{(i^*, j)\}$ and update $k_i(w)$ values accordingly.
  end if
end for
Output: Final matching M.

The CCDF provides a probabilistic understanding of potential gains, the primal-dual framework enforces global constraints and ensures theoretical competitiveness, and OCS adds control over the randomness in the decision-making process. For example, when evaluating whether to match an online vertex j deterministically or randomly, the algorithm integrates CCDF values into the dual computations, compares deterministic and randomized offers using primal-dual updates, and applies

OCS only when the randomized option is superior. This make sure that each decision maximizes immediate gains while preserving long-term feasibility and competitiveness.

5 Future Directions

There remain many open problems that are brought up by these papers:

1. **Worst-Case Input for RANKING:** Is the complete upper-triangular matrix the worst-case input for the RANKING algorithm, minimizing its expected matching size? Proving this conjecture would get rid of lower order terms, showing that RANKING is the possible possible algorithm in the unweighted case.
2. **Improving Online Correlated Selection (OCS):** The current 0.1099-OCS is a breakthrough but not known to be optimal. Can we construct an OCS with a higher parameter, or identify the theoretical upper bound? A better OCS would directly improve competitive ratios.
3. **Competitive Ratio Bound with OCS:** Even with a perfect 1-OCS, the edge-weighted algorithm achieves at most a $5/9$ competitive ratio. Can we design other types of algorithms that are more than $1/2$-competitive that do not have such a limitation?
4. **Infeasibility of $1 - 1/e$ for Edge-Weighted Matching:** Can we prove that no algorithm can achieve a $1 - 1/e$ competitive ratio for edge-weighted online bipartite matching, or prove some other upper bound on the competitive-ratio?
5. **Practical and Approximate Applications:** How can these algorithms be adapted for real-world applications like online ad placement, where approximate solutions often suffice? What heuristics are useful in practice?

6 Conclusion

We explored the evolution of online algorithms for bipartite matching, analyzing techniques across unweighted, vertex-weighted, and edge-weighted variants, fom the foundational RANKING algorithm to the more sophisticated Online Correlated Selection (OCS). We demonstrated how symmetry, perturbation, and negative correlation underpin these techniques, highlighting the key ideas and how these methods relate to each other. Future work includes improving OCS, understanding worst-case inputs, and investigating the competitive ratio limits for edge-weighted matching.

References

- Karp, R., Vazirani, U., & Vazirani, V. (1990). An Optimal Algorithm for On-line Bipartite Matching. STOC.
- Aggarwal, G., Goel, G., Karande, C., & Mehta, A. (2011). Online Vertex-Weighted Bipartite Matching. SODA.
- Fahrbach, M., Huang, Z., Tao, R., & Zadimoghaddam, M. (2020). Edge-Weighted Online Bipartite Matching. FOCS.