A Formal Analysis of RANKING

Mohammad Abdulaziz
King's College London, United Kingdom
Technische Universität München, Germany

Christoph Madlener
Technische Universität München, Germany

Abstract

We describe a formal correctness proof of RANKING, an online algorithm for online bipartite matching. An outcome of our formalisation is that it shows that there is a gap in all combinatorial proofs of the algorithm. Filling that gap constituted the majority of the effort which went into this work. This is despite the algorithm being one of the most studied algorithms and a central result in theoretical computer science. This gap is an example of difficulties in formalising graphical arguments which are ubiquitous in the theory of computing.

2012 ACM Subject Classification Theory of computation; Mathematics of computing

Keywords and phrases Matching Theory, Formalized Mathematics, Online Algorithms

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding *Mohammad Abdulaziz*: Part of this work was funded by DFG Koselleck Grant NI 491/16-1

1 Introduction

Matching is a classical problem in computer science, operations research, graph theory, and combinatorial optimisation. In short, in this problem, given an undirected graph, one tries to compute a subset of the edges of this graph, s.t. no two edges are incident on the same vertex. This subset is usually optimised w.r.t. a given objective, e.g. matching cardinality, sum of weights of edges in the matching, etc. An important special case of matching problems is maximum cardinality matching in bipartite graphs. It is one of the first problems to be addressed in combinatorial optimistation, where, for instance, the Hungarian method was invented in 1955 to solve it in the edge-weighted setting [1]. The online version of that problem, i.e. the version in which one of the parties of the graphs arrive online, one vertex at a time, along with its incident edges, has received special attention. This is because the problem can model many economic situations, most-notably Google's Adwords market [15].

The most basic version of online bipartite matching is the one where vertices and edges have no weights. That problem was studied by Karp, Vazirani, and Vazirani (henceforth, KVV) [14], where they devised the so-called RANKING algorithm. In that paper, KVV showed that their algorithm can solve the online problem with a *competitive ratio*, i.e. the average case ratio of the online algorithm's solution quality compared to the best offline algorithm, of $1 - 1/e$. They also showed that this ratio is the best possible for any randomised online bipartite matching algorithm. The analysis of the RANKING algorithm been continuously studied, where authors have mainly tried to simplify the algorithm's original correctness proof, i.e. the proof that it achieves a $1-1/e$ competitive ratio [10, 4, 5, 7, 20, 16]. This is because the algorithm's analysis, which can be divided into a probabilistic and a combinatorial part, is considered to be "extremely difficult" [19] by the algorithms community, despite the algorithm itself being very simple.

In this paper we formalise an analysis of the algorithm by Birnbaum and Mathieu [4] (henceforth, BM) in Isabelle/HOL [17]. BM claim to present the first simple proof of the algorithm's competitive ratio. Indeed, the paper's title is "Online bipartite matching made simple", and it is the last attempt at a simple combinatorial proof for the algorithm, as later works focused on primal-dual analyses of the algorithm.

Our most striking finding is that there is a "gap" in the proof, where there was one lemma whose proof was "a simple structural observation" by the authors. Formalising the proof of this lemma constitutes the majority of the effort that went into the work we describe here as well as the majority of the volume of the formal proof scripts. There are also other interesting aspects, from a formalisation perspective, of that proof. For instance, it combines graph theoretic, probabilistic, and graphical arguments. It also requires modelling and reasoning about online algorithms.

The rest of the paper is structured as follows. We first describe the algorithm and how we model it in Isabelle/HOL. Then we discuss the the probabilistic part of the proof and its formalisation. We then discuss the combinatorial part of the proof, where we describe the main findings of this work, namely, **1.** the first complete proof that covers the gap in the proof by BM, as well as other combinatorial proofs of the algorithm, and **2.** a significantly simpler proof of a lemma needed by BM to facilitate the algorithm's probabilistic analysis. Lastly, we discuss a part of the proof usually glossed over by other authors, which is lifting the analysis to obtain an asymptotic statement on the competitive ratio.

Isabelle/HOL Isabelle/HOL [18] is a theorem prover based on Higher-Order Logic. Roughly speaking, Higher-Order Logic can be seen as a combination of functional programming with logic. Isabelle's syntax is a variation of Standard ML combined with (almost) standard mathematical notation. Function application is written infix, and functions are usually curried (i.e., function f applied to arguments $x_1 \ \ldots \ x_n$ is written as $f \ x_1 \ \ldots \ x_n$ instead of the standard notation $f(x_1, \ \ldots \ , x_n)$). In Isabelle/HOL, *SOME* is the Hilbert choice, and *THE* is the definite description operator.

Availability Our formalisation is in the supplementary material and will be available online in case of acceptance. Throughout the paper, and in the appendix, we added excerpts from the formalisation representing important definitions and theorem statements to aid in linking the informal description in the paper and the formal proof scripts.

2 Basic Definitions and Notation

We denote a list of elements as $[x_1, x_2, \ldots, x_n]$. In the rest of this paper, we only consider lists with distinct elements. We say element x_i has rank i[1] in the list $[x_1, x_2, \ldots, x_i, \ldots, x_n]$. We overload the membership, subset, union and intersection set operations to lists. For a list vs, of length n, and an element $v \in vs$, let, for $1 \leq i \leq n$, $vs[v \mapsto i]$ denote the list with the same elements as vs, where v has rank i, the elements of rank less than i remain unchanged and the rank of the elements of rank at least i is increased by 1. Also, let $vs(v)$ denote the rank of v in vs and $vs[i]$ the element of rank i in vs. For a list vs, $v\#vs$ denotes the list vs but with the vertex v appended to its head. A permutation of a finite set s is a list whose elements are exactly the elements of s.

An edge is a set of vertices with size 2. A graph $\mathcal{G}$ is a set of edges. The set of vertices of a graph $\mathcal{G}$, denoted by $\mathcal{V}(\mathcal{G})$, is $\bigcup_{e \in \mathcal{G}} e$. For a vertex v, $N_{\mathcal{G}}(v)$ denotes $\{u \mid \{v, u\} \in \mathcal{G}\}$. We say a graph $\mathcal{G}$ is bipartite w.r.t. to two sets of vertices V and U (henceforth, the left and right party) iff **1.** $\mathcal{V}(\mathcal{G}) \subseteq (V \cup U)$, **2.** for any $\{v, u\} \in \mathcal{G}$, we have that $\{v, u\} \not\subseteq V$ and $\{v, u\} \not\subseteq U$. A set of edges $\mathcal{M}$ is a matching iff $\forall e \neq e' \in \mathcal{M}.\ e \cap e' = \emptyset$. For a matching $\mathcal{M}$

[1] In the formalisation we use index, which is the same as the rank less one.

and a vertex v, if there is u s.t. $\{v, u\} \in \mathcal{M}$, we say u is the partner of v, denoted by $\mathcal{M}(v)$. We use $\mathcal{G} - E$ to denote the edges in $\mathcal{G}$ that are not in E, and, for a set of vertices V, $\mathcal{G} \setminus V$ denotes $\mathcal{G} \cap \{e \mid e \cap V = \emptyset\}$, i.e. the graph with edges incident to vertices in V removed.

In many cases, a matching is a subset of a graph, in which case we call it a matching w.r.t. the graph. For a graph $\mathcal{G}$, a matching $\mathcal{M}$ w.r.t $\mathcal{G}$ is a maximum cardinality matching, aka maximum matching, w.r.t. $\mathcal{G}$ iff for any matching $\mathcal{M}'$ w.r.t. $\mathcal{G}$, we have that $|\mathcal{M}'| \leq |\mathcal{M}|$. A matching $\mathcal{M}$ w.r.t. $\mathcal{G}$ is a perfect matching w.r.t. $\mathcal{G}$ iff $\mathcal{V}(\mathcal{M}) = \mathcal{V}(\mathcal{G})$. A matching $\mathcal{M}$ w.r.t. $\mathcal{G}$ is a maximal matching w.r.t. $\mathcal{G}$ iff $\forall e \in \mathcal{G}.\ e \cap \mathcal{V}(\mathcal{M}) \neq \emptyset$.

A discrete probability space P is defined by a countable sample space Ω_P and a probability mass function (PMF) $\mathbb{P}_P : \Omega_P \to [0, 1]$ assigning a probability to each sample, where $\sum_{\omega \in \Omega_P} \mathbb{P}_P(\omega) = 1$. The PMF is lifted naturally to events (sets of samples) as $\mathbb{P}_P(E) = \sum_{\omega \in E} \mathbb{P}_P(E)$ for $E \subseteq \Omega_P$. The expectation of a random variable $X : \Omega_P \to \mathbb{R}$ is denoted $\mathbb{E}_{\omega \sim P}[X(\omega)]$. For a set B and a non-empty, finite subset $A \subseteq B$, $\mathcal{U}_B(A)$ is the discrete uniform distribution, i.e. $\Omega_{\mathcal{U}_B(A)} = B$ and $\mathbb{P}_{\mathcal{U}_B(A)}(a) = \frac{1}{|A|}$ if $a \in A$ and $\mathbb{P}_{\mathcal{U}(A)}(b) = 0$ if $b \notin A$. If $A = B$ we simply write $\mathcal{U}(A)$ for $\mathcal{U}_A(A)$.

We model randomised algorithms as probability distributions over the results of the algorithm. The Giry Monad [9] allows to compose random experiments in an elegant manner and is used to express randomised algorithms. The return operator gives a distribution which places probability 1 on a single sample ω, i.e. $\mathbb{P}_{\mathsf{return}(\omega)}(x)$ is 1, if $x = \omega$, and 0, otherwise.

Composition of experiments is achieved via the bind operator (written infix as $\ggg=$). Intuitively, $P \ggg= Q$ randomly chooses a sample ω according to P and then returns a value chosen randomly according to the distribution $Q(\omega)$. For additional legibility, we use Haskell-like **do**-notation for bind and return. This notation can be desugared recursively as follows:

$$\textbf{do}\{\ x \leftarrow P;\ \ stmts\ \} \equiv P \ggg= (\lambda x.\ stmts).$$

In Isabelle/HOL, we base our work on a simple formalisation of undirected graphs by Abdulaziz et al. [2], which was introduced in the context of the verification of Edmonds' blossom matching algorithm. The types of graphs and edges as well as the notion a matching in this formalisation are shown in Listing 7. We use this formalisation because of its simplicity, and the fact that it has a rich library on matchings and other related notions, as we will discuss later. However, we will not further discuss the merits of this representation as it is outside of the scope of this work. Interested readers should consult the original paper [2].

Probability theory in Isabelle/HOL is based on a general formalisation of measure theory by Hölzl [11]. In the formalisation, $\mathcal{U}(A)$ is denoted *pmf_of_set* A, and return is denoted *return_pmf*. The meanings of other Isabelle/HOL notations used in the rest of the paper should be self-explanatory.

3 RANKING

Given a bipartite graph $\mathcal{G}$, whose left and right parties are V and U, the ranking algorithm takes as an offline input V, and a sequence π as an online input, where vertices, along with their adjacent edges, arrive one-by-one. As an example, consider Fig. 1a, showing a graph whose left party, i.e. the offline vertices, is $\{v_4, v_2, v_6, v_1, v_5, v_3\}$. The right party, i.e. the online vertices, arrive in the order $[u_1, u_2, u_3, u_4, u_5]$. The first step in the algorithm is that it randomly permutes the offline input. In our example, this is shown in Fig. 1b. Then, vertices from the right party of the graph arrive one-by-one. The most important thing to note about that is that, for every arriving vertex u, the algorithm adds the edge connecting u and the

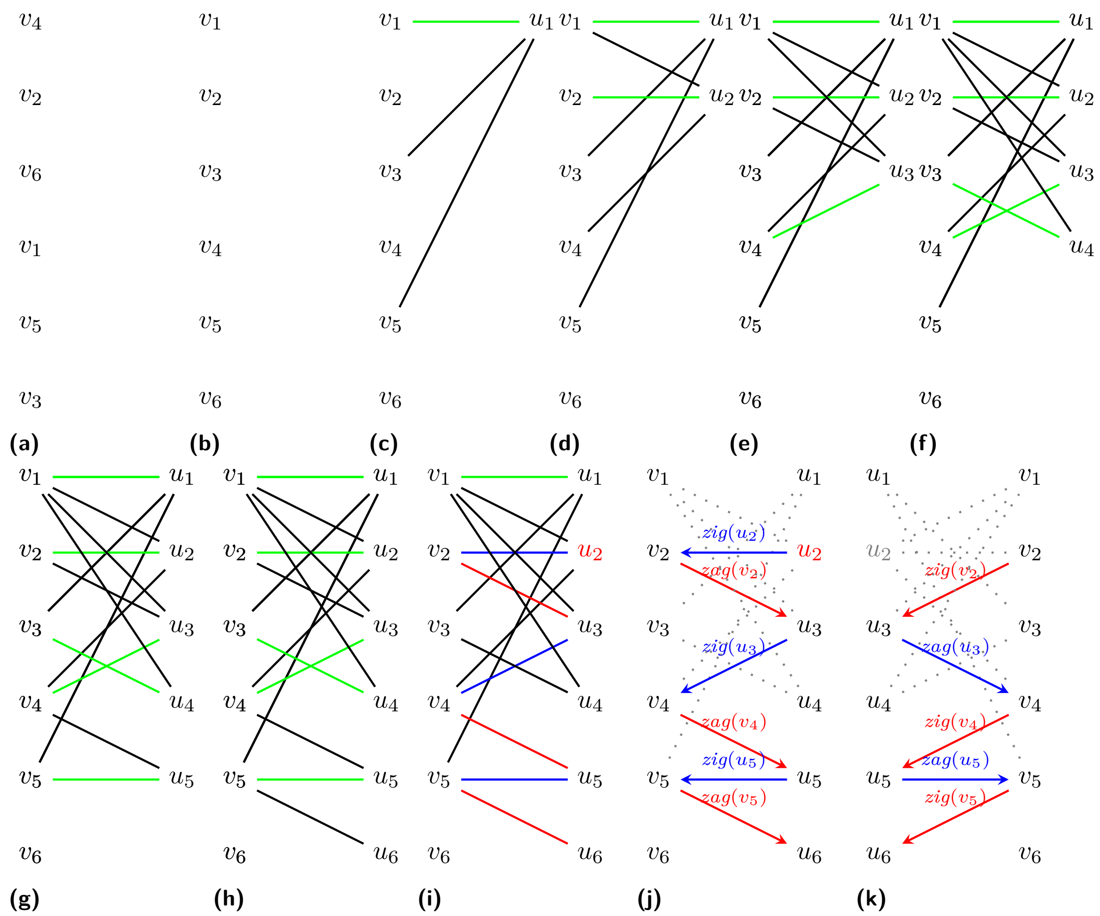

Figure 1 The steps of computing a matching using *online-match*, and what happens when an online vertex is removed from the input.

offline unmatched vertex with the minimum rank, if any such edge exists. In our example, we have the ranking $[v_1, v_2, v_3, v_4, v_5, v_6]$, of the offline vertices. Fig. 1c shows the state of the matching after the arrival of u_1: it has three edges connecting it to the offline vertices v_1, v_3, and v_5. The edge connecting it to v_1 is added to the matching, as it is unmatched and has the lowest rank among them. Then, the other vertices on the online side arrive based on the order given earlier, and the matching is updated, as shown in Fig. 1d-1g, and the final matching computed by the algorithm is the one represented by the green edges in Fig. 1h.

As should be clear by now, the algorithm's description and, accordingly, modeling is a simple task. The pseudo-code is in Algorithm 1. In Isabelle/HOL, we model the algorithm as shown in Listing 1. The first two functions are recursive on lists. The first function, *step*, is recursive on the list of the offline vertices, where, given a graph G, a vertex u from the online side, the list of offline vertices, and the matching, it adds to the matching the first edge it finds that connects u and an offline vertex v. The function does the recursion on the list, assuming the list is ordered according to the ranking of the offline vertices, with the head of the list being the vertex with the lowest rank. The second function, *online_match'*, is recursive on the on the list of online vertices, where the list is ordered according to the arrival order of those vertices, where the head of the list is the earliest arriving vertex. For each vertex in the list, *online_match'* tries to match it to an offline vertex using *step*. The

Algorithm 1 Pseudo-code of *RANKING*

function *online-match(G, π, σ)* **begin**
  $\mathcal{M} \leftarrow \emptyset$
  for *every arriving vertex u in π* **do**
    if $\exists v \in (N_{\mathcal{G}}(u) - \mathcal{V}(\mathcal{M}))$ **then** $\mathcal{M} \leftarrow \mathcal{M} \cup \{\{\mathsf{argmin}_{v\in(N_{\mathcal{G}}(u)-\mathcal{V}(\mathcal{M}))}\sigma(v), u\}\}$
  return $\mathcal{M}$
end
function *RANKING(G, π)* **begin**
  $\sigma \leftarrow$ a random permutation of V
  return *online-match*$(\mathcal{G}, \pi, \sigma)$
end

Listing 1: Modelling *RANKING* in Isabelle/HOL.

```
fun step :: "'a graph ⇒ 'a ⇒ 'a list ⇒ 'a graph ⇒ 'a graph" where
  "step _ _ [] M = M"
| "step G u (v#vs) M = (
      if v ∉ Vs M ∧ u ∉ Vs M ∧ {u,v} ∈ G
      then insert {u,v} M
      else step G u vs M
    )"

fun online_match' :: "'a graph ⇒ 'a list ⇒ 'a list ⇒ 'a graph ⇒ 'a graph" where
  "online_match' _ [] _ M = M"
| "online_match' G (u#us) σ M = online_match' G us σ (step G u σ M)"

abbreviation "online_match G π σ ≡ online_match' G π σ {}"

definition "ranking ≡
  do {
    σ ← pmf_of_set (permutations_of_set V);
    return_pmf (online_match G π σ)
  }"
```

other main function, *ranking*, chooses a permutation of the offline vertices and passes it to *online-match*.

We note that we avoid devising an involved way to model and reason about online computation, and only model it simply as a list of inputs and a step function that operates on each online input. This is because the algorithm description itself is simple. The primary focus of our work here is the formalisation of the correctness argument, the mathematical part of which is the main challenge.

3.1 Competitive Ratio of *RANKING*

The goal of this work is formalise the analysis of *RANKING*'s competitiveness. In general, for online algorithms solving optimisation problems, the analysis focuses on the quality of their outputs in comparison with the quality of the output of the best offline algorithm, i.e. an algorithm which has access to the entire input before it starts computing its output. The outcome of such an analysis is referred to as the *competitive ratio* of the respective online algorithm. In the case of bipartite matchings, the best offline algorithms, like the Hopcroft-Karp algorithm [12], can compute maximum cardinality matching for bipartite graphs. Thus, for *RANKING*, the natural way to analyse it is by showing that the size of the matching it computes maintains a certain ratio if compared to the size of the maximum matching of the input graph. Furthermore, since *RANKING* is a randomised algorithm, it is natural that this relationship is in expectation. More precisely, for *RANKING*, we have the following relation, which was first shown by KVV: for any given graph and arrival orders, the ratio between the expected size of the matching computed by *RANKING* and the size of the maximum matching is $1 - 1/e$. The expectation ranges over the different permutations of the offline side.

4 Competitiveness for Bi-Partite Graphs with Perfect Matchings

In the following, let $\mathcal{G}$ be a bipartite graph w.r.t. V and U, s.t. $\mathcal{M}$ is a perfect matching w.r.t. $\mathcal{G}$, and $|\mathcal{M}| = n$. Let π be an arrival order for U and let $\mathcal{S}(A)$ denote the set of all permutations of a finite set A.[2]

The algorithm can be modelled as the following Giry monad

$$\mathit{RANKING}(\mathcal{G}, \pi) \equiv \textbf{do } \{\ \sigma \leftarrow \mathcal{U}(\mathcal{S}(V));\ \mathsf{return}\ \mathit{online\text{-}match}(\mathcal{G}, \pi, \sigma)\ \}.$$

In the following, we describe our formal proof of the analysis of the competitive ratio for instances with perfect matching. This formal proof closely follows the one by BM. However, we highlight the differences to the original one as they arise.

We need the following lemma ([4, Lemma 5]) before the main result can be shown.

▶ **Lemma 1.** *Let x_t denote the probability over the random permutations of V that the vertex of rank t is matched by the algorithm, for $1 \leq t \leq n$. Then $1 - x_t \leq (1/n) \sum_{1 \leq s \leq t} x_s$.*

Let $v \in V$ be the vertex of rank t for some fixed permutation σ of V. The intuition behind this bound is that v only remains unmatched if its partner $\mathcal{M}(v)$ in the perfect matching is matched to a vertex ranked lower in π. Since v is a random vertex (when drawing a permutation), so is $\mathcal{M}(v)$. The right-hand-side is supposed to be the probability that $\mathcal{M}(v)$ is matched to a vertex arriving before v (since the sum is the expected number of vertices matched to vertices of rank at most t). This intuitive idea does not work due to the dependence of $\mathcal{M}(v)$ and the set of vertices matched to vertices of rank at most t. The correct argument avoids this dependence. However, this requires a stronger statement on what happens with $\mathcal{M}(v)$ if v stays unmatched, captured in the following lemma ([4, Lemma 4]), whose proof we discuss in the next section.

▶ **Lemma 2.** *Let $v \in V$, u denote $\mathcal{M}(v)$, and $\sigma \in \mathcal{S}(V)$. If v is not matched by online-match$(\mathcal{G}, \sigma, \pi)$ to u, then, for all $1 \leq i \leq n$, u is matched by online-match$(\mathcal{G}, \sigma[v \mapsto i], \pi)$ to a $v_i \in V$ s.t. $\sigma[v \mapsto i](v_i) \leq \sigma(v)$.*

Before presenting the proof of Lemma 1, we need to consider how to formally define x_t. It cannot be stated as a probability in the distribution $\mathit{RANKING}(\mathcal{G}, \pi)$. There is no way to refer to the "vertex of rank t in the permutation σ" since $\mathit{RANKING}(\mathcal{G}, \pi)$ is a distribution over subgraphs of $\mathcal{G}$ and the random permutations used to obtain them are not accessible. The solution is to explicitly define the Bernoulli distribution capturing the notion of the vertex of rank t being matched.

$$\mathbb{I}_t \equiv \textbf{do } \{\ \sigma \leftarrow \mathcal{U}(\mathcal{S}(V));\ \textbf{let } R = \mathit{online\text{-}match}(\mathcal{G}, \pi, \sigma);\ \mathsf{return}\ (\sigma[t] \in \mathcal{V}(R))\ \}$$

Then, $1 - x_t$ corresponds to the probability $\mathbb{P}_{\mathbb{I}_t}(\mathsf{False})$.

A key step to achieve the independence of the involved events revolves around not only drawing a random permutation, but also drawing a random vertex and moving it to rank

[2] In the formalisation $\mathcal{S}(A)$ is written *permutations_of_set A*.

$\mathbb{I}'_t \equiv \textbf{do}\ \{$
$\quad \sigma \leftarrow \mathcal{U}(\mathcal{S}(V));$
$\quad v \leftarrow \mathcal{U}(V);$
$\quad \textbf{let}\ R = \textit{online-match}(\mathcal{G}, \pi, \sigma[v \mapsto t]);$
$\quad \mathsf{return}\ (v \in \mathcal{V}(R))$
$\}$

(a) In addition to a random permutation $\sigma \in \mathcal{S}(V)$, a random vertex $v \in V$ is drawn and moved to rank t.

$\mathbb{I}''_t \equiv \textbf{do}\ \{$
$\quad \sigma \leftarrow \mathcal{U}(\mathcal{S}(V));$
$\quad v \leftarrow \mathcal{U}(V);$
$\quad \textbf{let}\ R = \textit{online-match}(\mathcal{G}, \pi, \sigma);$
$\quad \mathsf{return}\ (\mathcal{M}(v) \in \mathcal{V}(R) \wedge \sigma(R(\mathcal{M}(v)) \leq t))$
$\}$

(b) Distribution describing the probability that the partner $\mathcal{M}(v) \in U$ of a random vertex $v \in V$ is matched to a vertex of rank at most t.

Figure 2 Two Bernoulli distributions used in the proof of Lemma 1

t. This is reflected in the distribution $\mathbb{I}'_t$, given in Fig. 2a. This deceptively simple change ensures the independence of the drawn permutation, i.e. σ, and the actual partner in the perfect matching of the vertex of rank t, i.e. $\mathcal{M}(\sigma[v \mapsto t][t])$ which is the same as $\mathcal{M}(v)$. There is an aspect that is glossed over in the original proof and is intuitively clear: simply drawing a random permutation uniformly at random and the modified way where a random vertex is put at rank t are equivalent. This is shown explicitly in the formal proof.

The final distribution we present here, $\mathbb{I}''_t$ in Fig. 2b, captures the probability that the partner $\mathcal{M}(v)$ of a random $v \in V$ is matched to a vertex of rank at most t.[3]

Proof of Lemma 1. The first step follows from the fact that the permutation σ, in both $\mathbb{I}_t$ and $\mathbb{I}'_t$, and the vertex v are all drawn from uniform distributions.

$$\mathbb{P}_{\mathbb{I}_t}(\mathsf{False}) = \mathbb{P}_{\mathbb{I}'_t}(\mathsf{False})$$

By Lemma 2, if $v \in V$ is unmatched in *online-match*$(\mathcal{G}, \pi, \sigma[v \mapsto t])$, then, $\mathcal{M}(v)$ is matched to a vertex of rank at most t in *online-match*$(\mathcal{G}, \pi, \sigma)$ (by using $\sigma[v \mapsto t][v \mapsto \sigma(v)] = \sigma$).

$$\leq \mathbb{P}_{\mathbb{I}''_t}(\mathsf{True})$$

Then, the process of drawing a random $v \in V$ and considering $\mathcal{M}(v)$ in $\mathbb{I}''_t$ can be replaced with drawing a random $u \in U$ directly, using the bijection induced by $\mathcal{M}$. This describes the probability that a random $u \in U$ is matched to a vertex of rank at most t. That probability, in turn, is exactly the expected size of the set of online vertices matched to vertices of rank at most t. Formally, these two steps are performed by defining two more Bernoulli distributions capturing the involved concepts. Their definitions are omitted here. Let $\mathbb{I}^*_t$ be the distribution for the set of online vertices matched to vertices of rank at most t.

$$= \frac{1}{n} \mathbb{E}_{O \sim \mathbb{I}^*_t}[|O|]$$

The final step is to express the expected size of the set of online vertices matched to vertices of rank at most t as a sum of the probabilities that the offline vertices of rank up to t are matched. This completes the argument.

$$= \frac{1}{n} \sum_{s=1}^{t} \mathbb{P}_{\mathbb{I}_s}(\mathsf{True})$$

[3] The formalisations of the different distributions are in Listing 8.

◀

Then, we proceed to the main result of this section.

▶ **Theorem 1.** *The competitive ratio of RANKING for instances with a perfect matching of size n is at least $1-(1-\frac{1}{n+1})^n$, i.e. $1-(1-\frac{1}{n+1})^n \leq \frac{\mathbb{E}_{R\sim RANKING(\mathcal{G},\pi)}[|R|]}{n}$.*

Proof. The expected size of the matching produced by *RANKING*$(\mathcal{G},\pi)$ can be rewritten as a sum of the probabilities of the vertices of some rank getting matched.

$$\frac{\mathbb{E}_{R\sim RANKING(\mathcal{G},\pi)}\left[|R|\right]}{n} = \frac{1}{n}\sum_{s=1}^{n}\mathbb{P}_{\mathbb{I}_s}(\mathsf{True})$$

The bound obtained on $\mathbb{P}_{\mathbb{I}_s}(\mathsf{False})$ for $1 \leq s \leq n$ in Lemma 1 can be used to bound the sum. This requires a fact on sums provable by induction on n, followed by algebraic manipulation.

$$\geq \frac{1}{n}\sum_{s=1}^{n}\left(1-\frac{1}{n+1}\right)^{s}$$

More algebraic manipulation yields the final result.

$$= 1-\left(1-\frac{1}{n+1}\right)^{n}$$

◀

5 Lifting the Competitiveness to General Bi-Partite Graphs

Until now, we have shown that *RANKING* satisfies the desired competitive ratio for graphs with a perfect matching. Also, until now, our formalisation closely follows BM's proof. However, in all previous graph-theoretic expositions of the correctness proof of this algorithm [10, 4, 14], as opposed to linear programming-based expositions [5, 7, 20], the authors would stop at the current point, stating, or implicitly assuming, that it is obvious to see how the analysis of *RANKING* for bipartite graphs with perfect matchings extends to general bipartite graphs. The central argument is as follows: it is easy to see that, for a fixed permutation of the offline vertices, if we remove a vertex from a bipartite graph that does not occur in a maximum matching of that graph, then *online-match* will compute a matching that is either one edge smaller or of the same size as the matching *online-match* would compute, given the original graph.

Indeed, BM, who are the authors who give the most detailed account of the graph-theoretic correctness proof of this algorithm, state, as a proof for this fact [4, Lemma 2], that "it is an easy structural observation". In a sense they are correct: in our example, illustrated in Fig. 1, if we remove u_2, it is easy to see that *online-match*'s output size will be only one less than on the original graph. This is because all the matching edges will "cascade" down. This is illustrated in Fig. 1i, showing the blue edges being replaced with the red edges. In this section we mainly formalise this argument. We also formalise another easier, but no less crucial, graph-theoretic part of the proof by BM [4, Lemma 4]. This lemma is used in the probabilistic part of the proof, as stated earlier. In our formalisation we significantly simplified the proof. Before we do so, however, we introduce some necessary background and notions related to paths.

5.1 Alternating Paths, Augmenting Paths, and Berge's Lemma

A list of vertices $[v_1, v_2, \ldots, v_n]$ is a path w.r.t. a graph $\mathcal{G}$ iff $\{v_i, v_{i+1}\} \in \mathcal{G}$ for $1 \leq i < n$. Note: a path $[v_1 v_2 \ldots v_n]$ is always a simple path as we only consider distinct lists. A list of vertices $[v_1, v_2, \ldots, v_n]$ is an alternating path w.r.t. a set of edges E iff for some E' **1.** $E' = E$ or $E' \cap E = \emptyset$, **2.** $\{v_i, v_{i+1}\} \in E'$ holds for all even numbers i, where $1 \leq i < n$, and **3.** $\{v_i, v_{i+1}\} \notin E'$ holds for all odd numbers i, where $1 \leq i \leq n$. We call a list of vertices $[v_1, v_2, \ldots, v_n]$ an augmenting path w.r.t. a matching $\mathcal{M}$ iff $[v_1, v_2, \ldots, v_n]$ is an alternating path w.r.t. $\mathcal{M}$ and $v_1, v_n \notin \mathcal{V}(\mathcal{M})$. If $\mathcal{M}$ is a matching w.r.t. a graph $\mathcal{G}$, we call the path an augmenting path w.r.t. to the pair $\langle \mathcal{G}, \mathcal{M} \rangle$. Also, for two sets s and t, $s \oplus t$ denotes the symmetric difference of the two sets.

A central result in matching theory is Berge's lemma, which gives an algorithmically useful characterisation of a maximum cardiniality matching.

▶ **Theorem 2** (Berge's Lemma)**.** *For a graph $\mathcal{G}$, a matching $\mathcal{M}$ is maximum w.r.t. $\mathcal{G}$ iff there is not an augmenting path γ w.r.t. $\langle \mathcal{G}, \mathcal{M} \rangle$.*

We use a formalisation of the above concepts and Berge's Lemma by Abdulaziz et al. [2]. For completeness, the most important parts of this formalisation are demonstrated in Listing 9. Nonetheless, interested readers should refer to the original paper [2].

5.2 *online-match*'s Behaviour after Removing a Vertex

Now that we have all the necessary machinery, we can discuss the formalisation of the correctness of *RANKING* for general bipartite graphs. The central claim to show is stated in the following lemma, which is a restatement of Lemma 2 in BM's paper. It states what happens to the result of *online-match* when a vertex is removed from the graph.

▶ **Lemma 3.** *Let $\mathcal{G}$ be a bipartite graph w.r.t. the lists σ and π. Consider a vertex $u \in \pi$. Let $\mathcal{H}$ be $G \setminus \{u\}$. We have that either online-match$(\mathcal{G}, \pi, \sigma)$ = online-match$(\mathcal{H}, \pi, \sigma)$ or online-match$(\mathcal{G}, \pi, \sigma) \oplus$ online-match$(\mathcal{H}, \pi, \sigma)$ can be ordered into an alternating path w.r.t. online-match$(\mathcal{G}, \pi, \sigma)$ and w.r.t. online-match$(\mathcal{H}, \pi, \sigma)$, and that path starts at v.*

The above lemma was never proved by any of the previous expositions of the combinatorial argument for the algorithm's correctness. BM's exposition is an exception, where there is at least a graphical example, showing what happens when we remove a vertex before running *online-match*. A version of that graphical argument can be seen in Fig. 1.Fig. 1h shows the matching computed by the algorithm on the original graph, and Fig. 1i shows the difference in the computed matching if a vertex from the online side of the graph is removed. [4] As shown, when the vertex is removed, the matched edges "cascade downwards", where the original matching edges, shown in blue, are replaced with the red edges. The statement of the lemma states that the symmetric difference between the two computed matchings is always an alternating path, w.r.t. both the old and the new matchings, if there is any difference. When looking at the graphical illustration this is obvious. However, when formalising that argument, many challenges manifest themselves.

The first challenge is the characterisation of the path that constitutes the difference between the two matchings. This characterisation has to, among other things, make formal

[4] The the lemma above is stated for an online vertex being removed, while in the formalisation an offline vertex is removed. This highlights an important concept in many of the proofs: the interchangeability of the offline and online vertices for fixed orders σ and π.

Listing 2: Formalising *shifts-to* in Isabelle/HOL

```
definition "shifts-to G M u v v' π σ ≡
 u ∈ set π ∧ v' ∈ set σ ∧ index σ v < index σ v' ∧ {u,v'} ∈ G
 ∧ (∄u'. index π u' < index π u ∧ {u',v'} ∈ M) ∧
 (∀v''. (index σ v < index σ v'' ∧ index σ v'' < index σ v')
 ⟶ ({u,v''} ∉ G ∨ (∃u'. index π u' < index π u ∧ {u',v''} ∈ M)))
```

Listing 3: Formalising zig-zag and their termination relation in Isabelle/HOL.

```
function zig :: "'a graph ⇒ 'a graph ⇒ 'a ⇒ 'a list ⇒ 'a list ⇒ 'a list"
and zag :: "'a graph ⇒ 'a graph ⇒ 'a ⇒ 'a list ⇒ 'a list ⇒ 'a list" where
  proper-zig: "zig G M v π σ = v # (
                      if ∃u. {u,v} ∈ M
                      then zag G M (THE u. {u,v} ∈ M) π σ
                      else [])" if "matching M"
| no-matching-zig: "zig _ M v _ _ = [v]" if "¬matching M"

| proper-zag: "zag G M u π σ = u # (if ∃v. {u,v} ∈ M
                        then
                        (let v = THE v. {u,v} ∈ M in (
                          if ∃v'. shifts-to G M u v v' π σ
                          then zig G M (THE v'. shifts-to G M u v v' π σ) π σ
                          else [])
                        )
                        else []
                      )" if "matching M"
| no-matching-zag: "zag _ M v _ _ = [v]" if "¬matching M"
```

proofs by induction manageable. To do so, we had to formulate this characterisation *not* recursively on the given bipartite graph, i.e. the given bipartite graph should not change across different recursive calls. Otherwise, proving anything about the path would involve a complicated induction on the given bipartite graph.

To define that path, we first introduce a concept relating two vertices on the online side. We state v *shifts-to* v' iff **1.** v occurs before v' in the offline permutation σ, **2.** v is matched to some u, **3.** v' is not matched to any vertex that occurs before u in π, and **4.** any vertex $v'' \in N_{\mathcal{G}}(u)$ occurring between v and v' in σ is matched by *online-match* to a vertex occurring before u in the arrival order π. Intuitively, this means that, if v is removed from the graph, then v' would be matched to u by *online-match.* Our formalisation of this definition can found in Listing 2. Note: the omitted arguments in the text, $\mathcal{G}$, $\mathcal{M}$, π, and σ are usually clear from the context.

Now that we are done with the definition of *shifts-to*, we are ready to describe our characterisation of the path whose edges form the symmetric difference of the two matchings computed by *online-match.* We characterise it using the following functions:

$$zig(\mathcal{G},\mathcal{M},v,\pi,\sigma) \equiv \begin{cases} v\#zag(\mathcal{G},\mathcal{M},u,\pi,\sigma) & \text{if } \{v,u\} \in \mathcal{M} \\ [v] & \text{otherwise} \end{cases}$$

$$zag(\mathcal{G},\mathcal{M},u,\pi,\sigma) \equiv \begin{cases} u\#zag(\mathcal{G},\mathcal{M},v',\pi,\sigma) & \text{if } \{v,u\} \in \mathcal{M}\text{, for some } v\text{, and } v \textit{ shifts-to } v' \\ [u] & \text{otherwise} \end{cases}$$

As the names of the functions indicate, the path zig-zags between the online and the offline sides of the graph, going down the online ordering. This is indicated in Fig. 1j. The formalisation of *zig-zag* is given in Listing 3. Note that the formalisation has extra cases for when the second argument is not a matching: this is to ensure termination, which is not straightforward, as the definite descriptions are not well-defined in these cases. The

Listing 4: Formalising the specification of *online-match*'s output in Isabelle/HOL.

```
definition ranking-matching :: "'a graph ⇒ 'a graph ⇒ 'a list ⇒ 'a list ⇒ bool" where
  "ranking-matching G M π σ ≡ graph-matching G M ∧
    bipartite G (set π) (set σ) ∧ maximal-matching G M ∧
    (∀u v v'. ({u,v}∈M ∧ {u,v'}∈G ∧ index σ v' < index σ v) ⟶
      (∃u'. {u',v'}∈M ∧ index π u' < index π u)) ∧
    (∀u v u'. ({u,v}∈M ∧ {u',v}∈G ∧ index π u' < index π u) ⟶
      (∃v'. {u',v'}∈M ∧ index σ v' < index σ v))"
```

termination relation encodes the intuition that, while zig-zagging, the path also goes down the ordering of online vertices. More formally, because this is a mutually recursive function, we have to provide an order that relates the argument passed to recursive calls of *zag* from *zig* and the other way around. For evaluating *zig*$(\mathcal{G}, \mathcal{M}, v, \pi, \sigma)$, we need a call to *zag*$(\mathcal{G}, \mathcal{M}, u, \pi, \sigma)$, in which case the relation holds iff v and u satisfy **1.** $\{v, u\} \in$ *online-match*$(\mathcal{G}, \pi, \sigma)$ and **2.** if there is v', s.t. v *shifts-to* v', then $\sigma(v) < \sigma(v')$. For evaluating *zag*$(\mathcal{G}, \mathcal{M}, u, \pi, \sigma)$, we need a call to *zig*$(\mathcal{G}, \mathcal{M}, v', \pi, \sigma)$, in which case the relation holds iff u and v' satisfy **1.** there is v s.t. $\{v, u\} \in$ *online-match*$(\mathcal{G}, \pi, \sigma)$ and **2.** v *shifts-to* v' and $\sigma(v) < \sigma(v')$.

Another challenge for formalising the proof of Lemma 3 is devising a non-recursive characterisation of the properties of the matching computed by *online-match*, which would be enough for proving the lemma, yet more abstract than the actual specification of the algorithm. This characterisation can be intuitively described as follows: $\mathcal{M}$ is a *ranking-matching* w.r.t. $\mathcal{G}$, σ, and π iff **1.** $\mathcal{G}$ is bipartite w.r.t. σ and π, **2.** $\mathcal{M}$ is a maximal matching w.r.t. $\mathcal{G}$, **3.** every vertex from $u \in \pi$ is matched to the unmatched vertex from σ at u's arrival, to which it is connected, with the lowest rank in σ, and **4.** no vertex from σ "refuses" to be matched. The formal specification is given in Listing 4. It should be clear that the following properties hold for *ranking-matching*.

▶ **Proposition 1.** *Let $\mathcal{G}$ be a bipartite graph w.r.t. σ and π. We have that* **1.** *online-match$(\mathcal{G}, \pi, \sigma)$ is a ranking-matching w.r.t. $\mathcal{G}$, σ, and π,* **2.** *if $\mathcal{M}$ is a ranking-matching w.r.t. $\mathcal{G}$, σ, and π, then it is a ranking-matching w.r.t. $\mathcal{G}$, π, and σ, and* **3.** *if $\mathcal{M}$ and $\mathcal{M}'$ are both ranking-matchings w.r.t. $\mathcal{G}$, σ, and π, then $\mathcal{M} = \mathcal{M}'$.*

This specification of *online-match* makes our proofs about *online-match* much simpler, as it allows us to gloss over many of the computational details of the algorithm. In particulate, it allows us to avoid nested inductions, especially when using the I.H. of Lemma 3.

Now that we have characterised the difference between the matchings computed by *online-match* before and after removing a vertex, as well as the main properties satisfied by matchings computed by *online-match*, we are ready to present the proof that the competitiveness for bipartite graphs with perfect matchings lifts to general bipartite graphs. There are two main ideas to our proof. The first one is that we show that the output of *zig*, for some online vertex u, which is matched to an offline vertex v, stays the same when offline vertices are removed from the graph and the matching, if those offline vertices are all ranked lower than v. Graphically, this is clear. For instance, in Fig. 1j, if we remove the vertex v_1 from the graph and the matching, the result of *zig* applied to u_2, w.r.t. to the modified graph and matching, will be the same as its output w.r.t. the old graph and matching.

▶ **Lemma 4.** *Let $\mathcal{G}$ be a bipartite graph w.r.t. σ and π. Consider a vertex $u \in \pi$, s.t. there is v, where $\{v, u\} \in \mathcal{M}$. Consider a set of vertices $U' \subseteq \pi$, s.t. for all $u' \in U'$ we have that $\pi(u') < \pi(u)$. Let $\mathcal{M}$ be a ranking-matching w.r.t. $\mathcal{G}$, π, and σ. We have that zig$(\mathcal{G}, \mathcal{M}, u, \sigma, \pi) =$ zig$(\mathcal{G} \setminus U', \mathcal{M} \setminus U', u, \sigma, \pi)$ and zag$(\mathcal{G}, \mathcal{M}, v, \sigma, \pi) =$ zag$(\mathcal{G} \setminus U', \mathcal{M} \setminus U', v, \pi, \sigma)$.*

We do not prove this lemma here: the proof depends on an involved case analysis of the behaviour of *shifts-to*, and we describe below similar case analyses, which convey the difficulty of translating such obvious graphical arguments into proofs. Interested readers, however, should refer to the accompanying formal proof.

The second idea is that we exploit the symmetry between the online and the offline vertices. This is encoded in the following relationship between *zig* and *zag*.

▶ **Lemma 5.** *Let $\mathcal{G}$ be a bipartite graph w.r.t. σ and π. Consider a vertex $u \in \pi$. Let $\mathcal{H}$ be $G \setminus \{u\}$. Let $\mathcal{M}$ be a ranking-matching w.r.t. $\mathcal{G}$, π, and σ, and $\mathcal{M}'$ be a ranking-matching w.r.t. $\mathcal{H}$, σ, and π. Let v be a vertex s.t. $\{v, u\} \in \mathcal{M}$. We have that $zig(\mathcal{H}, \mathcal{M}', v, \pi, \sigma) = zag(\mathcal{G}, \mathcal{M}, v, \sigma, \pi)$.*

Before we discuss the proof, we first show a graphical argument of why the lemma holds. Fig. 1j and 1k show an example of how *zig* and *zag* would return the same list of vertices if invoked on the same vertex once on the offline side, and another time on the online side. In the first configuration, $zag(\mathcal{G}, \mathcal{M}, v_2, \sigma, \pi)$ chooses u_3, because in $\mathcal{M}$, we have that v_2 is matched to u_2, and u_2 *shifts-to* u_3. Then the rest of the recursive calls proceed as shown in the figure. When the online and offline sides are flipped, as shown in Fig. 1k, $zig(\mathcal{H}, \mathcal{M}', v_2, \pi, \sigma)$, where $\mathcal{H}$ denotes $\mathcal{G} \setminus \{u_2\}$, will also choose u_3 because, this time, it will be matched to v_2 in $\mathcal{M}'$, which is a *ranking-matching* for $\mathcal{H}$. As we will see in the proof, this graphical argument is much shorter than the corresponding textual proof, let alone the formal proof.

Proof. Our proof is by strong induction on the index of v. Let all the variable names in the I.H. be barred, e.g. the graph is $\overline{\mathcal{G}}$. Our proof is done by case analysis. We consider 3 cases: **1.** we have vertices u', v', s.t. $\{v, u'\} \in \mathcal{M}'$ and $\{u', v'\} \in \mathcal{M}$, **2.** we have a vertex u', s.t. $\{v, u'\} \in \mathcal{M}'$ and there is no v' s.t. $\{u', v'\} \in \mathcal{M}$, and **3.** there is no vertex u', s.t. $\{v, u'\} \in \mathcal{M}'$.

We focus on the first case, as that is the one where we employ the I.H. To apply the I.H., we use the following assignments of the quantified variables. $\overline{\mathcal{G}} \mapsto \mathcal{G} \setminus \{u, v\}$, $\overline{\pi} \mapsto \pi$, $\overline{\sigma} \mapsto \sigma$, $\overline{u} \mapsto u'$, $\overline{v} \mapsto v'$, $\overline{\mathcal{M}} \mapsto \mathcal{M} \setminus \{u, v\}$, and $\overline{\mathcal{M}'} \mapsto \mathcal{M}' \setminus \{v, u'\}$. From the I.H., we get $zig(\overline{\mathcal{H}}, \overline{\mathcal{M}}', \overline{v}, \overline{\pi}, \overline{\sigma}) = zag(\overline{\mathcal{G}}, \overline{\mathcal{M}}, \overline{u}, \overline{\sigma}, \overline{\pi})$. This proof is then finished by Lemma 4. ◀

We are now ready to prove a lemma that immediately implies Lemma 3.

▶ **Lemma 6.** *Let $\mathcal{G}$ be a bipartite graph w.r.t. σ and π. Consider a vertex $u \in \pi$. Let $\mathcal{H}$ be $G \setminus \{u\}$. Let $\mathcal{M}$ be a ranking-matching w.r.t. $\mathcal{G}$, σ, and π, and $\mathcal{M}'$ be a ranking-matching w.r.t. $\mathcal{H}$, σ, and π. We have that $\mathcal{M} \oplus \mathcal{M}' = zig(\mathcal{G}, \mathcal{M}, u, \sigma, \pi)$[5] or $\mathcal{M} = \mathcal{M}'$.*

Proof. Our proof is by strong induction on $|\mathcal{G}|$. Again, let all the variable names in the I.H. be barred. We consider two cases, either $u \notin \mathcal{V}(\mathcal{M})$ or $u \in \mathcal{V}(\mathcal{M})$. In the former case, the lemma follows immediately, since *online-match* will compute the same matching.

For the second case, we instantiate the I.H. as follows: $\overline{\mathcal{G}} \mapsto \mathcal{G} \setminus \{u\}$, $\overline{\mathcal{M}} \mapsto \mathcal{M}'$, $\overline{\mathcal{M}'} \mapsto \mathcal{M} \setminus \{v, u\}$, $\overline{\pi} \mapsto \sigma$, $\overline{\sigma} \mapsto \pi$, and $\overline{u} \mapsto v$, where v is some vertex s.t. $\{v, u\} \in \mathcal{M}$, which must exist since $u \in \mathcal{V}(\mathcal{M})$.[6] To show that the I.H. is usable in this case, we need to show that: **1.** $\overline{\mathcal{M}}$ is a *ranking-matching* w.r.t. $\overline{\mathcal{G}}$, $\overline{\pi}$, and $\overline{\sigma}$, and **2.** $\overline{\mathcal{M}'}$ is a *ranking-matching* w.r.t. $\overline{\mathcal{H}}$, $\overline{\pi}$, and $\overline{\sigma}$. The first requirement follows from the assumption that $\mathcal{M}'$ is *ranking-matching*

[5] We abuse the notation: although $zig(\mathcal{G}, \mathcal{M}, u, \sigma, \pi)$ is the list of vertices in the path, we use it here to denote the edges in the path.

[6] The instantiation of $\overline{\mathcal{H}}$ follows implicitly from the other instantiations.

w.r.t. $\mathcal{H}$, σ, and π, and the fact that *ranking-matching* is commutative w.r.t. the left and right parties of the given graph. The second requirement follows from a property of *ranking-matching*, which we do not prove here, stating that for any $\mathcal{M}$ that is a *ranking-matching* w.r.t. $\mathcal{G}$, σ, and π, and for any $e \in \mathcal{M}$, $\mathcal{M} - \{e\}$ is a *ranking-matching* w.r.t. $\mathcal{G} \setminus e$, σ, and π.

Then, from the I.H. and since we know that $v \in \mathcal{V}(\mathcal{M})$, we have that either **1.** $\overline{\mathcal{M}} = \overline{\mathcal{M}}'$ or **2.** $\overline{\mathcal{M}} \oplus \overline{\mathcal{M}'} = zig(\overline{\mathcal{G}}, \overline{\mathcal{M}}, \overline{u}, \overline{\sigma}, \overline{\pi})$. In the former case, we have that $\mathcal{M}' = \mathcal{M} \setminus \{u, v\}$, so v was not matched to anything in the graph, after removing u. This means that there is no u' for v s.t. u *shifts-to* u', which means that $zig(\mathcal{G}, \mathcal{M}, u, \sigma, \pi) = [u, v]$. From that, we have the lemma proved for this case, since $\mathcal{M} \oplus \mathcal{M}' = \{v, u\}$.

In the second case, we have that $\overline{\mathcal{M}} \oplus \overline{\mathcal{M}'} = zig(\mathcal{G} \setminus \{u\}, \mathcal{M}', v, \pi, \sigma)$. From Lemma 5, we have $zig(\mathcal{G} \setminus \{v\}, \mathcal{M}', v, \pi, \sigma) = zag(\mathcal{G}, \mathcal{M}, v, \sigma, \pi)$. From the definition of *zig* and since $\{u, v\} \in \mathcal{M}$, the lemma follows for this case. ◀

Proof of Lemma 3. Lemma 3 follows immediately from Lemma 6 lemma and from Proposition 1. ◀

5.3 Finishing the Proof

The next step in our proof is to generalise the previous analysis to address the case when the removed vertex is from the offline side of the graph. Although this is not considered by any of the previous expositions, this generalisation is crucial for proving the competitive ratio for general bipartite graphs, i.e. graphs that do not have a perfect matching.

▶ **Lemma 7.** *Let $\mathcal{G}$ be a bipartite graph w.r.t. σ and π. Consider a vertex $v \in \sigma$. Let $\mathcal{H}$ be $G \setminus \{v\}$. Let $\mathcal{M}$ be a ranking-matching w.r.t. $\mathcal{G}$, σ, and π, and $\mathcal{M}'$ be a ranking-matching w.r.t. $\mathcal{H}$, σ, and π. We have that $\mathcal{M} \oplus \mathcal{M}' = zig(\mathcal{G}, \mathcal{M}, v, \pi, \sigma)$ or $\mathcal{M} = \mathcal{M}'$.*

The proof of this lemma is very similar to that of Lemma 3. However, we are able to reuse all our lemmas that exploit the symmetry of the offline and online sides of the graphs, so there is not much redundancy in our proofs.

Until now, we have primarily focused on the *structural* difference between matchings computed by *online-match* before and after removing a vertex from the original graph. The next step in the proof is to use that to reason about the competitiveness of *online-match* for general bipartite graphs. The first step is proving the following lemma.

▶ **Lemma 8.** *Let $\mathcal{G}$ be a bipartite graph w.r.t. σ and π. Consider a vertex x. Let $\mathcal{H}$ be $G \setminus \{x\}$. Let $\mathcal{M}$ be a ranking-matching w.r.t. $\mathcal{G}$, σ, and π, and $\mathcal{M}'$ be a ranking-matching w.r.t. $\mathcal{H}$, σ, and π. We have that $|\mathcal{M}'| \leq |\mathcal{M}|$.*

Proof. Our proof is by case analysis. The first case is when $x \notin \mathcal{V}(\mathcal{M})$. In this case we will have that $\mathcal{M} = \mathcal{M}'$, which finishes our proof.

The second case is when $x \in \mathcal{V}(\mathcal{M})$. In this case, we have two sub-cases: either $x \in \pi$ or $x \in \sigma$. We only describe the first case here and the second is symmetric. Our proof is by contradiction, i.e. assuming $|\mathcal{M}'| > |\mathcal{M}|$. From Lemma 6, we have that $\mathcal{M} \oplus \mathcal{M}' = zig(\mathcal{G}, \mathcal{M}, u, \sigma, \pi)$. Also note that, from Berge's lemma, we will have that a subsequence of $zig(\mathcal{G}, \mathcal{M}, u, \sigma, \pi)$ is an augmenting path w.r.t. $\langle \mathcal{G}, \mathcal{M} \rangle$. We know from the definition of an augmenting path that, both, its first and last vertices are not in the matching it augments. Accordingly, we have that the first and last vertices of that subsequence of $zig(\mathcal{G}, \mathcal{M}, u, \sigma, \pi)$ are not in $\mathcal{M}$. This is a contradiction, because all vertices in $zig(\mathcal{G}, \mathcal{M}, u, \sigma, \pi)$, except possibly the last one, are in $\mathcal{V}(\mathcal{M})$. ◀

Listing 5: Formalising the specification of *make-perfect*'s output in Isabelle/HOL.

```
function make−perfect−matching :: "'a graph ⇒ 'a graph ⇒ 'a graph" where
  "make−perfect−matching G M = (
     if (∃x. x ∈ Vs G ∧ x ∉ Vs M)
     then make−perfect−matching (G \ {SOME x. x ∈ Vs G ∧ x ∉ Vs M}) M
     else G
  )
  " if "finite G"
| "make−perfect−matching G M = G" if "infinite G"
```

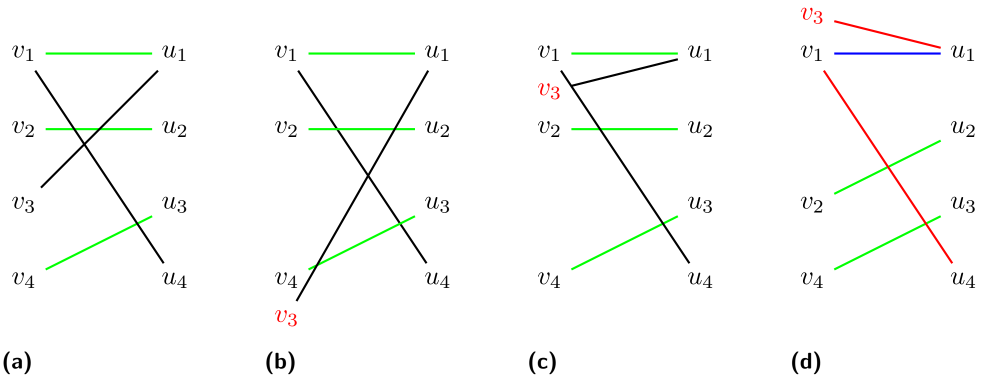

Figure 3 Illustrating Lemma 2, where $v = v_3$, and $\mathcal{M}(v_3) = u_1$. Initially (3a), v_3 is unmatched. Moving it further down in the ranking (3b) does not change the partner of u_1. Moving v_3 up in the ranking can either (3c) also leave u_1 untouched, or (3d) change the partner of u_1.

Lastly, we show that, given a bipartite graph $\mathcal{G}$ and a maximum cardinality matching $\mathcal{M}$ for that graph, we can recursively remove the vertices that do not occur in $\mathcal{M}$. To do that we define a recursive function, *make-perfect*, to remove these vertices and then prove the following lemma by computation induction, using the computation induction principle corresponding to *make-perfect*. Listing 5 shows the formalisation of that function.

▶ **Lemma 9.** *Let $\mathcal{G}$ be a bipartite graph w.r.t. σ and π. Let $\mathcal{M}$ be a ranking-matching w.r.t. $\mathcal{G}$, σ, and π, and $\mathcal{M}'$ be a ranking-matching w.r.t. make-perfect$(\mathcal{G}, \mathcal{M})$, σ, and π. We have that $|\mathcal{M}'| \leq |\mathcal{M}|$.*

This last lemma leads to the final theorem below.

▶ **Theorem 3.** *Let $\mathcal{G}$ be a bipartite graph w.r.t. σ and π. Let $\mathcal{M}$ be a maximum cardinality matching for $\mathcal{G}$. We have that $1 - (1 - \frac{1}{|\mathcal{M}|+1})^{|\mathcal{M}|} \leq \mathbb{E}_{R \sim RANKING(\mathcal{G},\pi)}[|R|]/|\mathcal{M}|$.*

Proof. This follows immediately from Lemma 9, Theorem 1, and the fact that the size of a maximum cardinality matching for *make-perfect*$(\mathcal{G}, \mathcal{M})$ is the same as the size of $\mathcal{M}$, if $\mathcal{M}$ is a maximum cardinality matching for $\mathcal{G}$. ◀

5.4 Proving Lemma 2

Until now we have not discussed how we formalised Lemma 2 – we believe it better fits here as its proof is a combinatorial argument. Graphically, Fig. 3 shows some instances of Lemma 2 for $v = v_3$ and $\mathcal{M}(v_3) = u_1$. No matter where v_3 is put, u_1 is always matched to a vertex of rank at most 3. BM prove this Lemma by stating that the difference, if any, between the matchings computed by *online-match* before and after moving the offline vertex is also an alternating path, where the ranks of the offline vertices traversed by that path increase.

Listing 6: The formalisation of Theorem 4

```
abbreviation matching−instance−nat :: "nat ⇒ (nat × nat) graph" where
  "matching−instance−nat n ≡ {{(0,k),(Suc 0,k)} |k. k < n}"

definition ranking−instances−nat :: "nat ⇒ (nat × nat) graph set" where
  "ranking−instances−nat n ≡ {G. max−card−matching G (matching−instance−nat n) ∧
     finite G ∧ G ⊆ {{(0,k),(Suc 0,l)} |k l. k < 2∗n ∧ l < 2∗n}}"

definition arrival−orders :: "(nat × nat) graph ⇒ (nat × nat) list set" where
  "arrival−orders G ≡ permutations−of−set {(Suc 0,l) |l. ∃k. {(0,k),(Suc 0,l)} ∈ G}"

definition offline−vertices :: "(nat × nat) graph ⇒ (nat × nat) set" where
  "offline−vertices G ≡ {(0, k) |k. ∃l. {(0, k),(Suc 0, l)} ∈ G}"

definition comp−ratio−nat where
  "comp−ratio−nat n ≡
     Min {Min {measure−pmf.expectation
                 (wf−ranking.ranking−prob G π (offline−vertices G)) card
                    / card (matching−instance−nat n)
                  |π. π ∈ arrival−orders G}
                    | G. G ∈ ranking−instances−nat n}"

theorem comp−ratio−limit ':
  assumes "convergent comp−ratio−nat"
  shows "1 − exp(−1) ≤ (lim comp−ratio−nat)"
```

Again, like other combinatorial parts of the analysis, graphically this is clearly evident: Fig. 3d shows the difference between *online-match*$(\mathcal{G}, \pi, \sigma)$ and *online-match*$(\mathcal{G}, \pi, \sigma[v_3 \mapsto 1])$. The blue edge was removed from the original matching, and the two red edges are added instead. The three edges form an alternating path w.r.t. to the original matching.

However, to formalise this argument would be as difficult as for Lemma 3. Indeed, we found out that there is no reason to construct the entire difference between the two matchings just to reason about the rank of the vertex v_i to which u is matched in *online-match*$(\mathcal{G}, \pi, \sigma[v \mapsto i])$. With this approach, the lemma follows almost immediately from the specification *ranking-matching*. Hence, the formal proof is much shorter than BM's approach.

6 The Competitive Ratio in the Limit

BM claim that the competitive ratio tends to $1-1/e$ if the matching's size tends to infinity. The main complication of showing that is to show that the competitive ratio converges, which they do not address at all. We formalised the following.

▶ **Theorem 4.** *Let $\mathcal{M}_n$ denote $\{\{(0,k),(1,k)\} \mid 1 \le k \le n\}$. Let Γ_n denote graphs in the power set of $\{\{(0,k),(1,l)\} \mid 1 \le k, l \le 2n\}$ and that have $\mathcal{M}_n$ as a maximum cardinality matching. Let π_n denote $\mathcal{S}(\{(1,k) \mid 1 \le k \le 2n\})$. If $\mathcal{Q}_n$ converges, then $\mathcal{Q}_n$ tends to $1-1/e$ as n tends to ∞, where $\mathcal{Q}_n$ denotes $\mathsf{min}_{(\mathcal{G},\pi)\in\Gamma_n\times\pi_n}\mathbb{E}_{R\sim RANKING(\mathcal{G},\pi)}[|R|]/|\mathcal{M}_n|$.*

We only prove the limit for a specific set of bipartite graphs, namely, Γ_n. We conjecture that Γ_n is isomorphic to the set of all bipartite graphs with maximum cardinality matchings of size n. Despite it being trivial, it was impressive that the part of the proof of this lemma which pertains to arithmetic manipulation was almost completely automated using Eberl's tool [6]. The other part of the proof was to show that Γ_n is finite, which was tedious.

The more interesting part would be to show that $\mathcal{Q}_n$ converges. In BM, they do not prove that, yet they do not have it as an assumption in their theorem statement. One way to show that this assumption holds is to use the theorem by KVV showing that no online algorithm for bipartite matching has a better competitive ratio that $1-1/e$. However, formalising that theorem is beyond the scope of our project.

7 Discussion

KVV's paper on online bipartite matching was a seminal result in the theory of online algorithms and matching. Its interesting theoretical properties, together with the emergence of online matching markets have inspired a lot of generalisations to other settings, e.g. for weighted vertices [3], online bipartite b-matching [13], the AdWords market [15], which models the multi-billion dollars industry online advertising industry, and general graphs [8], which models applications like ride-sharing. All of this means an improved understanding of the theory of online-matching, and especially RANKING, is of great interest.

Indeed, as stated earlier, multiple authors studied the analysis of RANKING. We mention here the most relevant five approaches: **1.** Goel and A. Mehta [10], tried to simplify the proof and fill in a "hole" in KVV's original proof, in particular in the proof of Lemma 6 in KVV's original paper, **2.** Birnbaum and C. Mathieu [4] also provided a simple, primarily combinatorial, proof for RANKING, **3.** Devanur, Jain, and Kleinberg [5] whose main contribution was to model the algorithm as a primal-dual algorithm, in an attempt to unify the approaches for analysing the unweighted, vertex-weighted, and the AdWords problem, **4.** Eden, Feldman, Fiat, and Segal [7], who tried to simplify the proof by using approaches from theory of economics, and finally **5.** Vazirani [20], who tried to simplify the proof of RANKING, in an attempt to use RANKING, or a generalisation of it, to solve AdWords. However, despite all of these attempts, the proof of RANKING's correctness is still considered difficult to understand, e.g. Vazirani's latest trial to generalize it had a critical non-obvious flaw in the combinatorial part of the analysis [20], which took months of reviewing to find out.

We believe this formalisation serves two purposes. First, it is yet another attempt to further the understanding of this algorithm's analysis. From that perspective, our work achieved two things. **1.** It further clarified the complexity of the combinatorial argument underlying the analysis of this algorithm by providing a detailed proof for how one could generalise the competitiveness of the algorithm from bipartite graphs with perfect matchings to general bipartite graphs. We note that this part of the analysis is analogous to the "no-surpassing property" in Vazirani's work [20], which is where his attempt to generalise RANKING to AdWords fell apart, further confirming our findings regarding the complexity of this part of the analysis. **2.** We significantly simplified the analysis of the consequences of changing the ranking of an offline vertex.

Another outcome of this project is interesting from a formalisation perspective. It further confirmed the previously reported observation that it is particularly hard to formalise graphical or geometric arguments and concepts. E.g. verbally, let alone formally, encoding the intuition behind *shifts-to*, which is a primarily graphical concept, is extremely cumbersome. We hypothesise that this is an inherent complexity in graphical concepts and arguments which manifests itself when the graphical argument is put into prose.

One point which we believe would particularly benefit from further study is that of modelling online computation. In its full generality, online computation is computation where the algorithm has access only to parts of the input, which arrive serially, but not the whole input. The way we model our algorithm is ad-hoc and does not capture that essence of online computation in its full generality. It remains an interesting question how can one model online computation, more generally. In addition to the theoretical interest, a satisfactory answer to that question is essential if one is to show that the competitive ratio of RANKING is optimal for online algorithms, which is a main result of KVV.

References

1 The Hungarian method for the assignment problem - Kuhn - 1955 - Naval Research Logistics Quarterly - Wiley Online Library. https://onlinelibrary.wiley.com/doi/10.1002/nav.3800020109.

2 Mohammad Abdulaziz, Kurt Mehlhorn, and Tobias Nipkow. Trustworthy graph algorithms (invited paper). In *MFCS*, 2019.

3 Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online Vertex-Weighted Bipartite Matching and Single-bid Budgeted Allocations. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, 2011.

4 Benjamin E. Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *SIGACT News*, 2008.

5 Nikhil R. Devanur, Kamal Jain, and Robert D. Kleinberg. Randomized Primal-Dual Analysis of RANKING for Online Bipartite Matching. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, January 2013.

6 Manuel Eberl. Verified Real Asymptotics in Isabelle/HOL. In *Proceedings of the 2019 on International Symposium on Symbolic and Algebraic Computation, ISSAC 2019, Beijing, China, July 15-18, 2019*, 2019.

7 Alon Eden, Michal Feldman, Amos Fiat, and Kineret Segal. An Economics-Based Analysis of RANKING for Online Bipartite Matching. In *Symposium on Simplicity in Algorithms (SOSA)*, January 2021.

8 Buddhima Gamlath, Michael Kapralov, Andreas Maggiori, Ola Svensson, and David Wajc. Online Matching with General Arrivals, April 2019. γarXiv:1904.08255.

9 Michele Giry. A categorical approach to probability theory. In *Categorical Aspects of Topology and Analysis*, 1982.

10 Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to Adwords. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, 2008.

11 Johannes Hölzl. *Construction and Stochastic Applications of Measure Spaces in Higher-Order Logic*. PhD thesis, Technical University Munich, 2013.

12 John E. Hopcroft and Richard M. Karp. An n\(\hat{}\mbox5/2\) Algorithm for Maximum Matchings in Bipartite Graphs. *SIAM J. Comput.*, 1973.

13 Bala Kalyanasundaram and Kirk R Pruhs. An optimal deterministic algorithm for online b-matching.

14 R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing - STOC '90*, 1990.

15 Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. AdWords and generalized online matching. *J. ACM*, 2007.

16 Milena Mihail and Thorben Tröbst. Online Matching with High Probability, December 2021. γarXiv:2112.07228.

17 Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*. 2002.

18 Lawrence C Paulson. *Isabelle: A Generic Theorem Prover*. 1994.

19 Vijay V. Vazirani. Online Bipartite Matching and Adwords, February 2022. γarXiv:2107.10777.

20 Vijay V. Vazirani. Online Bipartite Matching and Adwords (Invited Talk). In *47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022, August 22-26, 2022, Vienna, Austria*, 2022.

Appendix: Isabelle/HOL Listings

Listing 7: The formalisation of graphs and matching we use in Isabelle/HOL

```
locale graph-defs =
  fixes E :: "'a set set"

abbreviation "graph-invar E ≡ (∀e∈E. ∃u v. e = {u, v} ∧ u ≠ v) ∧ finite (Vs E)"

locale graph-abs =
  graph-defs +
  assumes graph: "graph-invar E"

definition matching where
  "matching M ⟷ (∀e1 ∈ M. ∀e2 ∈ M. e1 ≠ e2 ⟶ e1 ∩ e2 = {})"
```

Listing 8: Formalising the different distributions we need in our proof.

```
abbreviation rank−matched :: "nat ⇒ bool pmf" where
  "rank−matched t ≡
    do {
      σ ← pmf−of−set (permutations−of−set V);
      let R = online−match G π σ;
      return−pmf (σ ! t ∈ Vs R)
    }"

definition matched−before :: "nat ⇒ bool pmf" where
  "matched−before t ≡
    do {
      σ ← pmf−of−set (permutations−of−set V);
      v ← pmf−of−set V;
      let R = online−match G π σ;
      let u = (THE u. {u,v} ∈ M);
      return−pmf (u ∈ Vs R ∧ index σ (THE v. {u,v} ∈ R) ≤ t)
    }"

lemma matched−before−uniform−u: "matched−before t = do
    {
      σ ← pmf−of−set (permutations−of−set V);
      u ← pmf−of−set (set π);
      let R = online−match G π σ;
      return−pmf (u ∈ Vs R ∧ index σ (THE v. {u,v} ∈ R) ≤ t)
    }"

abbreviation "matched−before−t−set t ≡
  do {
    σ ← pmf−of−set (permutations−of−set V);
    let R = online−match G π σ;
    return−pmf {u ∈ set π. u ∈ Vs R ∧ index σ (THE v. {u,v} ∈ R) ≤ t}
  }"
```

$\mathbb{I}_t$ is formalised as `rank_matched`, $\mathbb{I}_t''$ is formalised as `matched_before`, `matched_before_uniform_u` is the formal statement showing that the distribution of a randomly chosen online vertex is matched to an offline vertex of rank at most t is the same as $\mathbb{I}_t''$, and $\mathbb{I}_t^*$ is formalised as `matched_before_t_set`.

Listing 9: The definitions of paths and augmenting paths and Berge's lemma as formalised in Isabelle/HOL.

```
context fixes G :: "'a set set" begin
inductive path where
  path0: "path []" |
  path1: "v ∈ Vs G ⟹ path [v]" |
  path2: "{v,v'} ∈ G ⟹ path (v'#vs) ⟹ path (v#v'#vs)"
end

inductive alt−list where
"alt−list P1 P2 []" |
"P1 x ⟹ alt−list P2 P1 l ⟹ alt−list P1 P2 (x#l)"

definition augmenting−path where
  "augmenting−path M p ≡
    alt−list (λe. e ∉ M) (λe. e ∈ M) (edges−of−path p)
    ∧ (length p ≥ 2) ∧ hd p ∉ Vs M ∧ last p ∉ Vs M"

abbreviation "augpath E M p ≡ path E p ∧ distinct p ∧ augmenting−path M p"

lemma Berge−1:
  assumes finite: "finite M" "finite M'" and
    matchings: "matching M" "matching M'" and
    lt−matching: "card M < card M'" and
    doubleton−neq−edges: "∀e∈(M ⊕ M').∃u v. e = {u,v} ∧ u ≠ v" "∀e∈M. ∃u v. e = {u,
        v} ∧ u ≠ v"
  shows "∃p. augmenting−path M p ∧ path (M ⊕ M') p ∧ distinct p"
```

Listing 10: Formal statement of Lemma 1.

```
lemma rank-t-unmatched-prob-bound:
  "t < card V ⟹
     1 - measure-pmf.prob (rank-matched t) {True} ≤
        1 / (card V) * (\<Sum>s≤t. measure-pmf.prob (rank-matched s) {True})"
```

Listing 11: Formal statement of Lemma 4.

```
lemma
  assumes "X ⊆ set π"
  assumes "bipartite M (set π) (set σ)"
  assumes "matching M"
  shows
   remove-online-vertices-zig-zig-eq:
     "v ∈ set σ ⟹
        ∀x ∈ X. ((∃v'. {x,v'} ∈ M) ⟶ index σ (THE v'. {x,v'} ∈ M) < index σ v)
          ⟹
                 zig (G \ X) (M \ X) v π σ = zig G M v π σ" and
   remove-online-vertices-zag-zag-eq:
     "u ∈ set π ⟹
     ((∃v. {u,v} ∈ M ⟹
           ∀x ∈ X. ((∃v. {x,v} ∈ M) ⟶
                 index σ (THE v. {x,v} ∈ M) < index σ (THE v. {u,v} ∈ M)))) ⟹
         zag (G \ X) (M \ X) u π σ = zag G M u π σ"
```

Listing 12: Formal statement of Lemma 5.

```
lemma\<^marker><tag important> ranking-matching-zig-zag-eq:
  assumes "{u,x} ∈ M"
  assumes "x ∈ set σ"
  assumes "ranking-matching G M π σ"
  assumes "ranking-matching (G \ {x}) M' σ π"
  shows "zig (G \ {x}) M' u σ π = zag G M u π σ"
```

Listing 13: Formal statement of Lemma 6.

```
lemma remove-offline-vertex-diff-is-zig:
  assumes "ranking-matching G M π σ"
  assumes "ranking-matching (G \ {x}) M' π σ"
  assumes "x ∈ set σ"
  shows "M ⊕ M' = set (edges-of-path (zig G M x π σ))"
```

Listing 14: Formal statement of Lemma 9.

```
lemma ranking-matching-card-leq-on-perfect-matching-graph:
  assumes "ranking-matching G M π σ" "ranking-matching (make-perfect-matching G N)
      M' π σ"
  shows "card M' ≤ card M"
```

Listing 15: The formalisation of Theorem 3.

```
lemma comp-ratio-no-limit:
  "measure-pmf.expectation ranking-prob card / (card V) ≥ 1 - (1 - 1/(card V + 1)) ^
       (card V)"
```