UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MATEMÁTICA

EMPAREJAMIENTOS EN LÍNEA EN GRAFOS BIPARTITOS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO

CHRISTIAN THOMAS VON BORRIES SEGOVIA

PROFESOR GUÍA:
JOSÉ SOTO SAN MARTÍN

PROFESOR CO-GUÍA:
JOSÉ VERSCHAE TANNENBAUM

MIEMBROS DE LA COMISIÓN:
JOSÉ SOTO SAN MARTÍN
JOSÉ VERSCHAE TANNENBAUM
JOSÉ CORREA HAEUSSLER

Este trabajo ha sido parcialmente financiado por Fondecyt de Iniciación 11130266:
"APPROXIMATION ALGORITHMS FOR INCREMENTAL SELECTION PROBLEMS"

SANTIAGO DE CHILE
OCTUBRE 2014

Resumen

El objetivo principal de esta memoria es estudiar generalizaciones del problema de emparejamientos en línea. En un artículo seminal Karp, Vazirani y Vazirani [12] estudiaron el siguiente problema de optimización: Dado un grafo bipartito $G = (L, R, E)$ del que el lado L es conocido y el lado R llega en línea, se busca maximizar el tamaño de un emparejamiento, bajo la condición de que solo se puede emparejar un vértice en el momento en el que llega. Karp, Vazirani y Vazirani encuentran un algoritmo que es una $(1 - \frac{1}{e})$-aproximación para el problema. En esta memoria se generaliza el problema al caso en el que un lado no está fijo, o sea que vértices de ambos lados pueden llegar en línea. Se estudian tres modelos: el modelo adversarial, el modelo de orden aleatorio y el modelo fuera de línea. Para el modelo adversarial se definen algoritmos locales y se demuestra que ninguno de ellos puede ser mejor que una $\frac{1}{2}$-aproximación. Para el modelo de orden aleatorio se encuentra un algoritmo cuya competitividad está en el intervalo $[0.696, 0.727]$. Finalmente, para el modelo fuera de línea se encuentra un algoritmo óptimo cuya competitividad es desconocida, pero se demuestra que está en el intervalo $[0.526, 0.591]$.

Summary

The main objective of this thesis is to study generalizations of the on-line matching problem. In a seminal paper, Karp, Vazirani and Vazirani [12] study the following optimisation problem: We are given a bipartite graph $G = (L, R, E)$ where the side L is known and the side R arrives on-line. We want to find a large matching, but we are only allowed to match vertices when they arrive. Karp, Vazirani and Vazirani find an algorithm with competitive ratio $1 - \frac{1}{e}$ for this problem. This thesis generalizes the problem by removing the assumption of one side being known beforehand. We study three models: the adversarial model, the random order model and the offline model. For the adversarial model, we define *local* algorithms and prove that none of them has a competitive ratio greater than $\frac{1}{2}$. For the random order model, we find an algorithm with competitivity in the interval $[0.696, 0.727]$. Finally, for the offline model, we find an optimal algorithm and prove its competitive ratio is in the interval $[0.526, 0.591]$.

Agradecimientos

Quiero expresar mi más profundo y sincero agradecimiento a mis padres y hermanos por la infancia que me brindaron, por todo su apoyo durante los años de estudio y una amistad con la que siempre puedo contar.

A mis compañeros de universidad, por hacer de mis años de estudio una experencia única y placentera, caracterizados por un grato ambiente de estudio. Yo la pasé bien y espero que ustedes también. Aprovecho de agradecerles a aquellos que fueron mis auxiliares y a los que fueron mis alumnos, ya que en ambas instancias aprendí mucho.

A los profesores y funcionarios del DIM por hacer de este departamento lo que es.

A José Soto y José Verschae, por su ayuda entregada en esta investigación y gran disponibilidad por ayudarme. Este trabajo no es mío, a pesar de lo que diga la portada, es fruto de todas las reuniones en las que estuvimos pensando y discutiendo.

A Nicole, por su compañía, cariño y apoyo.

A todos ellos, muchas gracias.

Contents

Introduction **1**

1 Introduction **1**

1.1 Preliminary Definitions and Results 1

1.1.1 On-line Algorithms and Previous Work in Bipartite On-line matching . 2

1.2 The Models 5

1.3 Our Contribution 6

2 On-line Matching: The Adversarial Model **7**

2.1 Previous Work 8

2.2 Local Algorithms 9

2.3 Hardness Results 13

2.3.1 Rounding 13

3 The Random Order Model **17**

3.1 Related Work 17

3.2 Lower Bound and the Unknown Distribution Model 19

3.3 Upper Bound 23

3.4 General Graphs 24

4 The Offline Model **32**

4.1 Definition 32

4.2 An Optimal Algorithm 34

4.3 Upper Bound 35

Conclusion **40**

Bibliography **43**

List of Figures

2.1 A bad example for any deterministic algorithm in the adversarial model. . . . 7
2.2 Graph used to bound $p(f, o)$. 10
2.3 Bomb graph. 11
2.4 Fractional matching on the subgraph induced by the first 4 arrived vertices. . . 15
2.5 Next two vertices if $\mathbb{P}(M = M_{1,2}) > 0$. 15
2.6 Fractional matching on the subgraph induced by the first 6 arrived vertices. . . 16
2.7 Fractional matching on the subgraph induced by the first 7 arrived vertices. . . 16

3.1 Illustration with vertices shown in order. 21
3.2 A path of length three. 23
3.3 G_5. 26

4.1 Input example. 33
4.2 Probability of choosing each edge. 33
4.3 I_1. 37
4.4 I_2. 37
4.5 I_3. 37
4.6 $I_k \to I_{k+1}$. 37
4.7 A visualization of the variables of the dual LP. 39
4.8 Feasible solution for the dual LP. 39

Chapter 1

Introduction

1.1 Preliminary Definitions and Results

We begin with some graph terminology and some classical textbook results we will use.

Definition 1.1 *A graph G is an ordered pair (V, E), where V is a finite set and $E \subseteq \binom{V}{2} = \{\{u, v\} \mid u, v \in V, u \neq v\}$.*

Definition 1.2 *In a graph $G = (V, E)$, V or $V(G)$ is called the vertex set and E or $E(G)$ is called the edge set. Elements of V are called vertices and elements of E are called edges. An edge $e = \{u, v\}$ will be written as $e = uv$.*

Definition 1.3 *Two edges e_1 and e_2 are called adjacent if $|e_1 \cap e_2|$=1. Similarly, two vertices v and u are called adjacent or neighbours if there exists an edge e such that $e = uv$. The set of neighbours of a vertex v will be denoted $N(v)$.*

Definition 1.4 *We say an edge e is incident to a vertex v if $v \in e$. The set of edges incident to a vertex v will be denoted $\delta(v)$.*

Definition 1.5 *A graph $G' = (V', E')$ is called a subgraph of $G = (V, E)$ if $V' \subseteq V$ and $E \subseteq E'$.*

Definition 1.6 *For graph $G = (V, E)$ and a subset of vertices V', we call $G[V']$ the induced subgraph by V' defined by $G[V'] = (V', E')$ where $E' = \{uv \in E \ : \ u \in V', v \in V'\}$. For a vertex set $\{v_1, v_2, \ldots, v_n\}$, we will shorten $G[\{v_1, v_2, \ldots, v_n\}]$ to $G[v_1, v_2, \ldots, v_n]$.*

Definition 1.7 *For an edge set $F \subseteq E$, we define $\chi(F)$ as a vector in $[0, 1]^E$, where $x_e = 1$ if $e \in F$ and $x_e = 0$ otherwise.*

Definition 1.8 *A graph $P = (V, E)$ is a path if there exists $n \in \mathbb{N}$ such that $V = \{v_0, v_1, \ldots, v_n\}$ and $E = \{v_0v_1, v_1v_2, \ldots, v_{n-1}v_n\}$. The length of a path is $n = |E(P)|$.*

Definition 1.9 *A graph $C = (V, E)$ is a cycle if there exists $n \in \mathbb{N}$ such that $V = \{v_1, \ldots, v_n\}$ and $E = \{v_1v_2, v_2v_3, \ldots, v_{n-1}v_n, v_nv_1\}$. The length of a cycle is $n = |E(C)|$.*

Definition 1.10 *A graph $G = (V, E)$ is called bipartite if V can be partitioned into two sets L and R, such that all edges are incident to both a vertex in L and a vertex in R. We write $G = (L, R, E)$ to denote the partitions of the vertex set when defining a bipartite graph.*

Theorem 1.11 *(König [1916]) [14] A graph is bipartite if and only if it contains no cycles of odd length.*

Definition 1.12 *A graph $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$ and $E = \binom{V}{2}$ is called a complete graph of n vertices and denoted K_n.*

Definition 1.13 *A bipartite graph $G = (L, R, E)$ where $L = \{v_1, \ldots, v_n\}$, $R = \{u_1, \ldots, u_m\}$ and $E = \{v_iu_j \ : \ 1 \leq i \leq n, 1 \leq j \leq m\}$ is called a complete bipartite graph and it is denoted $K_{n,m}$.*

Definition 1.14 *A matching of a graph $G = (V, E)$ is a subset of pairwise non-adjacent edges.*

Definition 1.15 *A vertex cover of a graph $G = (V, E)$ is a subset of vertices V' such that each edge of the graph is incident to some vertex of V'.*

Definition 1.16 *For a bipartite graph $G = (V, E)$, we define a fractional matching as a vector $(x_e)_{e \in E}$ satisfying*

$$x_e \in [0, 1] \ \forall e \in E, \qquad \sum_{e \in \delta(v)} x_e \leq 1 \ \forall v \in V.$$

The size of a fractional vertex cover is $\sum_{e \in E} x_e$.

Definition 1.17 *For a bipartite graph $G = (V, E)$, we define a fractional vertex cover as a vector $(x_v)_{v \in V}$ satisfying*

$$x_v \in [0, 1] \ \forall v \in V, \quad x_u + x_v \geq 1 \ \forall e \in E.$$

The size of a fractional vertex cover is $\sum_{v \in V} x_v$.

Theorem 1.18 *[14] For bipartite graphs, the linear programming (*LP*) problems of finding a maximum size fractional matching and a minimum size fractional vertex cover are integral and duals of each other.*

1.1.1 On-line Algorithms and Previous Work in Bipartite On-line matching

In classic optimisation problems studied within computer science and operations research, the input is given to the algorithm before it starts executing. This setting is realistic when full information is actually known in advance. On the other hand, in *on-line* optimisation problems the input is given to the algorithm in a piece-by-piece fashion, without having the entire input available from the start. Typically, the solution must maintain certain invariants and this forces the algorithm to act before the full input is known.

This thesis focuses on variants of the on-line matching problem in bipartite graphs. These are generalisations of the following problem studied by Karp, Vazirani and Vazirani [12] in a seminal paper. Suppose you are given a bipartite graph $G = (L, R, E)$ where only the left side of the graph is known beforehand. The right side arrives *on-line*, that is, vertices in R arrive in a preselected order, and the edges incident to a vertex in R are revealed to us only when the vertex arrives. The task is to decide, as each vertex in R arrives, to which vertex in L should we match it, so that the size of this matching is maximized. Decisions are permanent: once a vertex has been matched, this decision cannot be undone.

Since the algorithm has to act before the full input is known, typically the algorithm cannot achieve an optimal solution. Therefore, competitive analysis is the predominant framework for evaluating the performance of on-line algorithms, explained in the following definition.

Definition 1.19 *Let $\mathcal{A}$ be an algorithm for an on-line optimisation problem. For an input I, let $\mathbb{E}[A(I)]$ be the expected size of the solution found by the algorithm and OPT(I) the size of the optimal solution.*

We say that $\mathcal{A}$ achieves a competitivity or competitive ratio of α if, for all input I

$$\frac{\mathbb{E}[A(I)]}{OPT(I)} \geq \alpha$$

if the problem is a maximisation problem.

Analogously, we say that $\mathcal{A}$ achieves a competitivity *or* competitive ratio *of α if, for all input I*

$$\frac{\mathbb{E}[A(I)]}{OPT(I)} \leq \alpha$$

if the problem is a minimisation problem.

Example For the on-line matching problem, no deterministic algorithm can achieve a competitive ratio better than $\frac{1}{2}$. To prove this, let $\mathcal{A}$ be a deterministic algorithm and use the following input: Let $L = \{v_1, v_2\}$ and a vertex u_1 arriving on-line connected to both v_1 and v_2. $\mathcal{A}$ has to match u_1 (or its competitive ratio would be at most 0 on the input so far), so let u_2 arrive on-line connected to the matched vertex in L. $\mathcal{A}$ can not match u_2, so its competitive ratio is at most $\frac{1}{2}$.

There actually exists a deterministic algorithm with competitive ratio $\frac{1}{2}$, the GREEDY algorithm.

Algorithm 1 GREEDY

```
for $v$ arriving on-line do
  if $v$ has unmatched neighbours then
    Choose any unmatched neighbour $u$ amongst the vertices arrived so far
    Add $uv$ to the matching
  end if
end for
```

Theorem 1.20 *[13]* GREEDY *achieves a competitive ratio of* $\frac{1}{2}$.

This result extends to all models we will study, so we include a proof.

PROOF. Let M be the matching obtained by the algorithm and M^* a matching of maximum size. If v is any vertex in G and v is not matched in M, then v has no unmatched neighbours, therefore M is a maximal matching. Take any edge e in M^*. If e is not adjacent to any edge in M, we could add e to M, contradicting maximality. Define now:

$$\begin{aligned} f : M^* &\to M \\ e &\mapsto f(e) = e' \quad \text{where } e' \text{ is any edge in } M \text{ adjacent to } e. \end{aligned}$$

Since M^* is a matching, any vertex is incident to at most one edge in M^* and it follows that $|f^{-1}\{e'\}| \leq 2$ for all e' in M. Therefore,

$$\begin{aligned} |M^*| &= |\bigcup_{e' \in M} f^{-1}\{e'\}| \\ &\leq 2|M|. \end{aligned}$$

And thus $\frac{|M|}{|M^*|} \geq \frac{1}{2}$. This bound is tight, as seen in the previous example. □

Karp, Vazirani and Vazirani find an optimal (in the sense of competitive analysis) randomised algorithm called RANKING for the on-line matching problem. It achieves a competitive ratio of $1-\frac{1}{e}$, which they prove is the best possible competitive ratio amongst randomised algorithms for this problem.

Since then, many generalisations of this problem have been studied, such as b-matching, the AdWords problem and the vertex weighted version [17, 8, 1, 18]. Simpler proofs of the $1-\frac{1}{e}$ bound have also been found [4, 6]. Additionally, relaxations on the input have been studied, such as the random order model and the known distribution model [7, 3, 15, 11, 16]. A newer variant, important to this thesis, was the on-line fractional matching problem studied in an unpublished paper by Wang and Wong [19, see also [20]], which presents a 0.526-competitive algorithm for fractional bipartite matching in the case that the entire graph arrives on-line, instead of a side being fixed.

Another line of research studies approximation algorithms for the matching problem in general graphs. Algorithms studied so far are randomised variants of the GREEDY algorithm [2, 9, **?**, 5].

1.2 The Models

We study a generalisation of the on-line matching problem mentioned earlier: Instead of a side being fixed beforehand, vertices of the entire bipartite graph arrive on-line in any order. We study this problem in the *adversarial* model, the *random order* model and a model we call the *offline* model where the algorithm knows the graph and the order in which the vertices arrive, but we force the algorithm to maintain its competitivity in any prefix of the input, just like any on-line algorithm. We define these models below and highlight the relationships between them.

Definition 1.21 *In the* ***adversarial model****, input consists of a bipartite graph $G = (V, E)$ and a permutation π of its vertices, both initially unknown to the algorithm. The graph then arrives on-line, vertex by vertex in the order of π, starting from the empty graph. When vertex v arrives, it reveals to us which of the previously arrived vertices are its neighbours. The algorithm may only match the vertex to one of its previously revealed neighbours when it arrives. Also, the algorithm may not remove edges from the matching. The goal is to build a large matching.*

Definition 1.22 *In the* ***random order model****, input consists of a bipartite graph $G = (V, E)$. A permutation π is then sampled uniformly amongst the permutations of V. Both the permutation and the graph are unknown to the algorithm. The graph then arrives on-line and the algorithm has to construct a matching with the same restrictions as in the adversarial model.*

Definition 1.23 *In the* ***offline model****, a randomised algorithm with competitive ratio α is given a graph $G = (V, E)$ and a permutation of its vertices $\pi = v_1, \dots, v_n$. It then has to output a random matching M, with the restriction*

$$\frac{\mathbb{E}[M \cap G_k]}{OPT(G_k)} \geq \alpha \quad \forall k = 1, \dots, n \ , \tag{1.1}$$

where $G_k = G[v_1, \dots, v_k]$ is the graph induced by the first k vertices of π.

Lemma 1.24 *Let $\mathcal{A}$ be any algorithm in the adversarial model achieving a competitive ratio of α. Then $\mathcal{A}$ achieves a competitive ratio of α in the random order model.*

Proof. Let I be an input in the random order model and let $\mathcal{A}(I)$ be the random variable corresponding to the size of the matching output by the algorithm. Let $S(V)$ be the set of permutations of V and $\sigma \in S(V)$ be the random variable representing the permutation of V in which the vertices arrive. Using the law of total probability,

$$\mathbb{E}(\mathcal{A}(I)) = \sum_{\pi \in S(V)} \mathbb{E}(\mathcal{A}(I) | \sigma = \pi) \frac{1}{n!}.$$

Conditional to arriving in a fixed order, we can see an input in the random order model as an input in the adversarial model, so

$$\begin{aligned}\mathbb{E}(\mathcal{A}(I)) &\geq \sum_{i=1}^{n!} \alpha \mathrm{OPT}(I) \frac{1}{n!} \\ &\geq \alpha \mathrm{OPT}(I).\end{aligned}$$ □

Lemma 1.25 *Let $\mathcal{A}$ be any algorithm in the adversarial model achieving a competitive ratio of α. Then $\mathcal{A}$ achieves a competitive ratio of α in the offline model.*

Proof. Ignore the extra information given in the offline model and use $\mathcal{A}$. □

Corollary 1.26 *Let α in $[0, 1]$. If no algorithm can achieve a competitive ratio greater than α in the random order (resp. offline) model, then no algorithm can achieve a competitive ratio greater than α in the adversarial model.*

1.3 Our Contribution

We summarise our main results below.

- For the adversarial model, we define *local* algorithms, a class that contains many algorithms studied in the literature. We prove that all local algorithms have competitive ratio at most $\frac{1}{2}$, even when restricted to bipartite graphs.
- We show, with a counterexample, that there is no on-line rounding method that turns a fractional matching $(x_e)_{e \in E}$ into a random integral matching M with $\mathbb{E}[|M|] = \sum_{e \in E} x_e$. This complements the rounding method found for bipartite on-line vertex cover by Wang and Wong [19].
- We prove that in the random order model, the Ranking algorithm achieves exactly the same competitive ratio as Ranking when one of the sides is fixed beforehand and the other side arrives in a random order. This competitivity lies somewhere in the interval [0.696, 0.727].
- An upper bound of 0.875 for the competitivity of any algorithm in the random order model.
- We introduce the *offline* model, find an optimal algorithm for this model, and prove its competitivity lies in the interval $[0.526, \frac{\sqrt{5}+9}{19} \approx 0.591]$. The upper bound holds in the adversarial model, as per Corollary 1.26.

Chapter 2

On-line Matching: The Adversarial Model

Let $G = (L, R, E)$ be a bipartite graph and $v_1, v_2, \dots, v_n$ a permutation of its vertices. The entire vertex set arrives on-line in the order $v_1, v_2, \dots, v_n$. An on-line algorithm only sees the induced subgraph of vertices that have arrived so far and, whenever a vertex arrives, it has to choose whether to match it to one of its neighbours or leave it unmatched. Decisions are immediate and permanent: The only time when a vertex can be matched to another is when one of the two just arrived and once two vertices have been matched this cannot be undone. Since the algorithm has to be competitive for any input, one can imagine the entire input as being chosen by an adversary, hence the name of the model. This model is stricter than the one introduced by Karp et al., where one side is already fixed beforehand: we can simulate an instance where L is fixed beforehand by choosing an order where every vertex of L arrives before a vertex of R.

Any deterministic algorithm for this problem achieves a competitive ratio of at most $\frac{1}{2}$. This follows directly from the fact that any deterministic algorithm in the model where one side is fixed beforehand has a competitive ratio of at most $\frac{1}{2}$ [12], and our model generalizes this setting. This fact helps for an easier proof in this case, consider the graph in Figure 2.1, with vertices labelled by the order in which they arrive.

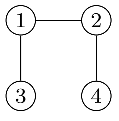

Figure 2.1: A bad example for any deterministic algorithm in the adversarial model.

Any deterministic algorithm with competitive ratio greater than $\frac{1}{2}$ has to pick the edge between vertices 1 and 2, because it has to be competitive in the graph consisting of a single edge. This matching is maximal, so the algorithm cannot add any more edges.

Moreover, the Greedy algorithm again achieves a competitive ratio of $\frac{1}{2}$ as shown in

Theorem 1.20, so the Greedy algorithm achieves the best possible competitive ratio for a deterministic algorithm.

The big question which motivates most of the work presented here is if there is a randomised algorithm which achieves a better competitive ratio than $\frac{1}{2}$ for this problem.

2.1 Previous Work

The on-line bipartite matching problem was first studied by Karp et al. [12] in the case where one side is fixed beforehand. As mentioned before, they devise an algorithm, called Ranking, that achieves a competitive ratio of $1 - \frac{1}{e}$ and no randomised algorithm can achieve a better competitive ratio than $1 - \frac{1}{e}$, so this algorithm is optimal. The algorithm is surprisingly simple and works as follows.

Algorithm 2 Ranking

```
Pick a random permutation of the known vertices in $L$, thereby assigning a random priority
to each vertex in $L$.
for $v$ arriving on-line do
  if $v$ has unmatched neighbours then
    $u \leftarrow$ the neighbour of $v$ with highest priority
    Add $uv$ to the matching
  end if
end for
```

The Ranking Algorithm has since then been modified to achieve a competitive ratio of $1 - \frac{1}{e}$ for generalisations of the previous problem, for example for the AdWords problem [17].

This thesis uses a recent result on on-line fractional matching by Wang and Wong [19, see also [20]]. In an unpublished paper, Wang and Wong present an algorithm with a competitive ratio of 0.526 for on-line fractional bipartite matching, in a model where the algorithm may only set values to an edge when the edge appears in the induced subgraph of arrived vertices. This is again with one side not necessarily fixed. Their algorithm is primal-dual. It simultaneously finds a fractional matching and a fractional vertex cover that are at most a factor of 1.901 away from each other. In particular, it achieves a competitive ratio of 0.526 for fractional on-line matching and a competitive ratio of 1.901 for fractional vertex cover. They also provide a randomised rounding method that returns (in an on-line fashion) an integral vertex cover with the same expected size as that of their fractional solution. Combining this, one gets an algorithm for the on-line vertex cover problem in bipartite graphs with competitive ratio better than 2. This proves a converse result to the following lemma.

Lemma 2.1 *[19] In the adversarial model, any randomised algorithm $\mathcal{A}$ for the on-line matching (resp. vertex cover) problem can be converted into a deterministic algorithm for the on-line fractional matching (resp. vertex cover) problem with the same competitive ratio.*

PROOF. Let $\mathcal{A}$ be any randomised algorithm for the on-line matching problem. Define a deterministic algorithm which simulates $\mathcal{A}$ and for an edge e sets $x_e = \mathbb{P}(e$ gets matched by $\mathcal{A})$. The same argument can be done for the on-line vertex cover problem. □

2.2 Local Algorithms

In this section we generalise both the RANKING and GREEDY algorithms, by defining a family of algorithms we call *local* algorithms, and prove that any algorithm in the family cannot be $(\frac{1}{2} + \varepsilon)$-competitive for any $\varepsilon > 0$. So far, the RANKING and variations of the GREEDY algorithms which randomise over some choices of the algorithm, are the most studied algorithms for on-line matching problems. Our result shows that if we want to break the $\frac{1}{2}$-barrier in the adversarial model, an algorithm needs to keep track of some global variable and make decisions based on it.

Definition 2.2 *We call an algorithm $\mathcal{A}$ a local algorithm if there exists a function $p : \mathbb{N}^2 \to [0,1]$, such that upon the arrival of a vertex v, $\mathbb{P}(v$ gets matched$)=p(f,o)$, where f is the number of free (unmatched) neighbours of v and o is the number of occupied (matched) neighbours of v. Additionally, every time $\mathcal{A}$ chooses to match the arriving vertex, this event is independent of previous choices of the algorithm.*

Note that no restriction is made on the choice on which vertex to match v to. For example, for any function $p : \mathbb{N}^2 \to [0,1]$, the following two algorithms are local algorithms, and were our main motivation for the definition:

Algorithm 3 RANDOMISED GREEDY

```
for $v$ arriving on-line do
  if $v$ has unmatched neighbours then
    $f \leftarrow$ amount of unmatched neighbours of $v$
    $o \leftarrow$ amount of matched neighbours of $v$
    $u \leftarrow$ an unmatched neighbour of $v$, picked uniformly at random
    match $u$ to $v$ with probability $p(f,o)$
  end if
end for
```

The second example tries to emulate the RANKING algorithm: Each vertex independently gets assigned a uniform random variable $\mathcal{U}[0,1]$, simulating its priority.

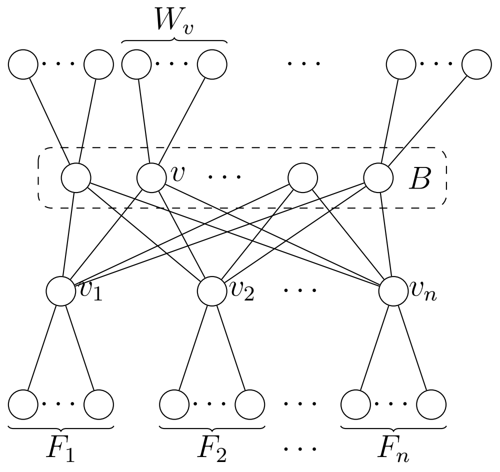

Figure 2.2: Graph used to bound $p(f, o)$.

Algorithm 4 Randomised Ranking

```
for $v$ arriving on-line do
    $x_v \leftarrow \mathcal{U}[0,1]$ (independently for each $v$)
    if $v$ has unmatched neighbours then
        $f \leftarrow$ amount of unmatched neighbours of $v$
        $o \leftarrow$ amount of matched neighbours of $v$
        $u \leftarrow \arg\min\{x_u \ : \ u \in N(v), u \text{ is unmatched}\}$
        match $u$ to $v$ with probability $p(f, o)$
    end if
end for
```

Lemma 2.3 *Suppose $\mathcal{A}$ is a local algorithm with competitivity $\frac{1}{2} + \varepsilon$ and let $p : \mathbb{N}^2 \to [0, 1]$ be the probability function it uses to match vertices. Then $p(f, o) \geq \frac{1}{2} + \varepsilon \forall f > 0$.*

Proof. Note that $p(1, 0) \geq \frac{1}{2} + \varepsilon > 0$ which follows by testing $\mathcal{A}$ on the input consisting of a single edge. For the first part of the proof, let $(f, o) \in \mathbb{N}^2$, $\varepsilon' > 0$ and $n \in \mathbb{N}$. Consider the following input I to the problem: First, a set B of o vertices arrive, without any edges. For each vertex v in B, add a set W_v of k vertices connected only to v. Since $p(1, 0) > 0$, we can take k big enough so that $\mathbb{P}(\forall\ v \in B\ v \text{ is matched}) \geq 1 - \varepsilon'$.

Afterwards, add n sets $(F_i)_{i=1}^n$ of f vertices each without any edges. Finally, add n vertices v_i, where each vertex v_i is connected to every vertex in F_i and to every vertex in B. (See Figure 2.2.)

For $f > 0$, this graph has a maximum matching of size $o + n$. Next we bound the expected value of the matching output by the algorithm. Note that with probability at least $1 - \varepsilon'$, every vertex v_i will have f unmatched neighbours and o matched neighbours. This helps to

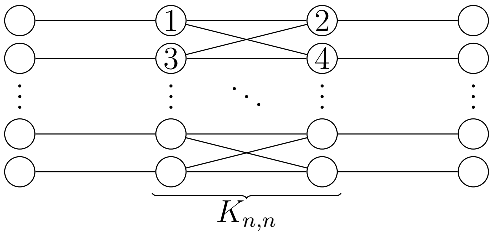

Figure 2.3: Bomb graph.

bound $\mathbb{E}[M(I)]$. In fact, define A as the event "all vertices in B got matched", then:

$$\begin{aligned}\mathbb{E}[|M(I)|] &= \mathbb{E}[M(I)|A]\mathbb{P}(A) + \mathbb{E}[M(I)|A^c]\mathbb{P}(A^c)\\ &\geq \mathbb{E}[M(I)|A](1-\varepsilon')\\ &\geq np(f,o)(1-\varepsilon')\end{aligned}$$

Similarly,

$$\begin{aligned}\mathbb{E}[|M(I)|] &= \mathbb{E}[M(I)|A]\mathbb{P}(E) + \mathbb{E}[M(I)|A^c]\mathbb{P}(A^c)\\ &\leq \mathbb{E}[M(I)|A] + \mathbb{E}[M(I)|A^c]\varepsilon'\\ &\leq np(f,o) + (o+n)\varepsilon'\end{aligned}$$

By taking the limit as $\varepsilon' \to 0$ and $n \to \infty$, we get

$$\lim_{\varepsilon' \to 0} \lim_{n \to \infty} \frac{\mathbb{E}[|M(I)|]}{o+n} = p(f,o)$$

Therefore, $p(f,o) \geq \frac{1}{2} + \varepsilon$ if $f > 0$. □

Theorem 2.4 *The competitive ratio of any local algorithm for the on-line matching problem is at most* $\frac{1}{2}$.

Proof. We proceed by contradiction. Let $\mathcal{A}$ be a local algorithm with competitivity $\frac{1}{2} + \varepsilon$ and $p : \mathbb{N}^2 \to [0,1]$ be the probability function it uses to match vertices. For an input I let $M(I)$ be the random variable defined as the matching obtained by $\mathcal{A}$ on input I and $\mathrm{OPT}(I)$ the size of the maximum matching in the graph given by I.

Consider the input I_n consisting of the complete bipartite graph $K_{n,n}$ and add to every single vertex of $K_{n,n}$ a neighbour (this graph is sometimes called the "bomb-graph" and has been used before to find upper bounds for matching algorithms [9]). The order of arrival will be as follows: First, vertices from $K_{n,n}$ arrive, alternating the side. Then, the rest of the vertices arrive, in any order. Note that the graph has a perfect matching, so $OPT(I_n) = 2n$. (See Figure 2.3.)

Our next step will be proving that for big n, $\mathcal{A}$ matches many vertices of the $K_{n,n}$ half of the input. To see this, we define the random variable X_i as the amount of unmatched vertices so far amongst the first i arrived vertices of $K_{n,n}$. Therefore, $i - X_i$ is the amount of matched vertices amongst the first i arrived vertices, half of which are on the left side of the graph with the other half on the right side. Therefore the ith vertex arriving always has (before the algorithm matches it) $\frac{i-X_i}{2}$ matched neighbours. If i is odd, the ith vertex arrives on the left side and has a total of $\lfloor \frac{i}{2} \rfloor$ neighbours. Otherwise, if i is even, it has a total of $\frac{i}{2}$ neigbhours. In both cases, the ith vertex has $\lfloor \frac{i}{2} \rfloor - \frac{i-X_i}{2}$ unmatched neighbours.

Because of the symmetry of the graph, if a vertex gets matched it does not matter which vertex the algoritm matches it to. Since the choice of matching each vertex is independent of previous choices of the algorithm, the stochastic process (X_i) is a (not necessarily time-homogeneous) Markov chain, with transition probabilities given by:

$$\mathbb{P}(X_i = k+1 | X_{i-1} = k) = 1 - p\left(\left\lfloor \frac{i}{2} \right\rfloor - \frac{i-k}{2}, \frac{i-k}{2}\right)$$
$$\mathbb{P}(X_i = k-1 | X_{i-1} = k) = p\left(\left\lfloor \frac{i}{2} \right\rfloor - \frac{i-k}{2}, \frac{i-k}{2}\right)$$

Note that $p(0, o) = 0$, because if a vertex does not have unmatched neighbours, it cannot be matched.

Now let $(U_i)_{i\in\mathbb{N}}$ be i.i.d. random variables with uniform distribution $\mathcal{U}(0,1)$. We define another Markov chain X'_i by defining $X'_0 = 0$ and for $i \geq 1$

$$X'_i = X'_{i-1} + \mathbb{1}_{\{U_i > p(\lfloor \frac{i}{2} \rfloor - \frac{i-X'_{i-1}}{2}, \frac{i-X'_{i-1}}{2})\}} - \mathbb{1}_{\{U_i \leq p(\lfloor \frac{i}{2} \rfloor - \frac{i-X'_{i-1}}{2}, \frac{i-X'_{i-1}}{2})\}}$$

X'_i is then also a Markov chain with transition probabilities given by:

$$\mathbb{P}(X'_i = k+1 | X'_{i-1} = k) = 1 - p\left(\left\lfloor \frac{i}{2} \right\rfloor - \frac{i-k}{2}, \frac{i-k}{2}\right)$$
$$\mathbb{P}(X'_i = k-1 | X'_{i-1} = k) = p\left(\left\lfloor \frac{i}{2} \right\rfloor - \frac{i-k}{2}, \frac{i-k}{2}\right)$$

So X'_i has the same transition probabilities as X_i and therefore both have the same distribution. Specifically, we have that $\mathbb{E}[X_i] = \mathbb{E}[X'_i]$.

Define now a new random variable $Y_0 = 0$ and for $i \geq 1$, let

$$Y_i = Y_{i-1} + \mathbb{1}_{\{Y_{i-1}=0\}} + \mathbb{1}_{\{Y_{i-1}\neq 0\}}(\mathbb{1}_{\{U_i > \frac{1}{2}\}} - \mathbb{1}_{\{U_i \leq \frac{1}{2}\}}).$$

Y_i is then a time-homogeneous Markov chain. In fact, Y_i is a symmetric walk in $\mathbb{N}$.

By Lemma 2.3, $p(f, o) \geq \frac{1}{2} + \varepsilon$ for $f > 0$. Using this fact we can prove by induction that $X'_i \leq Y_i$. The base case is $X'_0 = Y_0 = 0$. Suppose then that $X'_{i-1} \leq Y_{i-1}$. If $X'_i = X'_{i-1} - 1$, then $X'_i \leq Y_i$ because Y_i can only decrease by at most 1. On the other hand, if $X'_i = X'_{i-1} + 1$, then $U_i > p(\lfloor \frac{i}{2} \rfloor - \frac{i-X'_i}{2}, \frac{i-X'_i}{2}) > \frac{1}{2}$, so $Y_i = Y_{i-1} + 1$. Therefore $X'_i \leq Y'_i$ for all i.

Let $(Z_i)_{i\in\mathbb{N}}$ be independent Bernoulli random variables with parameter $\frac{1}{2}$. Then $2Z_i - 1 \in \{-1, 1\}$ can be seen as the step of a random walk in $\mathbb{Z}$. Thus, Y_i has the same distribution as

$|2\sum_{k=1}^{i} Z_k - i|$. By the central limit theorem, $\frac{2\sum_{k=1}^{i} Z_k - i}{\sqrt{i}} = 2\sqrt{i}(\frac{\sum_{k=1}^{i} Z_k}{i} - \frac{1}{2})$ converges in distribution to a standard normal distribution $\mathcal{N}(0,1)$. Therefore, $\frac{Y_i}{\sqrt{i}}$ converges in distribution to its absolute value $|\mathcal{N}(0,1)|$, so $\lim_{i\to\infty} \mathbb{E}[\frac{Y_i}{\sqrt{i}}] = \sqrt{\frac{2}{\pi}}$.

Since $n - \frac{X_n}{2}$ is the amount of edges matched in $K_{n,n}$ and the amount of edges matched by the vertices arriving later can be at most X_n, we have:

$$\begin{aligned}\mathbb{E}[M(I_n)] &\leq \mathbb{E}\Big[n - \frac{X_n}{2}\Big] + \mathbb{E}[X_n] \\ &\leq n + \frac{E[Y_n]}{2}\end{aligned}$$

Thus,

$$\begin{aligned}\lim_{n\to\infty} \frac{\mathbb{E}[M(I_n)]}{\mathrm{OPT}(I_n)} &\leq \frac{1}{2} + \lim_{n\to\infty} \frac{E[Y_n]}{4n} \\ &\leq \frac{1}{2} + \lim_{n\to\infty} \frac{E[Y_n]}{\sqrt{n}} \cdot \lim_{n\to\infty} \frac{1}{4\sqrt{n}} = \frac{1}{2}\end{aligned}$$

Hence, $\inf_{I \text{ input for } \mathcal{A}} \frac{\mathbb{E}[M(I)]}{\mathrm{OPT}(I)} \leq \frac{1}{2}$, so $\mathcal{A}$ cannot have competitive ratio greater than $\frac{1}{2}$. $\square$

2.3 Hardness Results

Our main result on the hardness of this problem is an upper bound on the competitive ratio of any on-line algorithm:

Theorem 2.5 *No on-line algorithm has a competitive ratio greater than* $\frac{\sqrt{5}+9}{19} \approx 0.591$.

Previously the best upper bound for this problem was 0.625 [19]. Both numbers are below $1 - \frac{1}{e} \approx 0.632$ and both bounds carry over to the on-line fractional matching problem. To prove the $\frac{\sqrt{5}+9}{19} \approx 0.591$ bound, we relax the model to a model where the algorithm has full information of the input, but still has to be competitive at the time of every vertex arrival. This model, which we call the *offline* model, will be studied in chapter 4, where we also prove the $\frac{\sqrt{5}+9}{19} \approx 0.591$ bound.

2.3.1 Rounding

There exists an algorithm for the on-line fractional matching problem with competitive ratio greater than $\frac{1}{2}$, even if one of the sides of the bipartite graph is not fixed beforehand [19]. Therefore, a method to round such a fractional matching $(x_e)_{e\in E}$ to a random integral matching M with $\mathbb{E}[|M|] = \sum_{e\in E} x_e$ in an on-line setting would result in an algorithm with competitive ratio greater than $\frac{1}{2}$ for the on-line matching problem studied in this chapter.

In this section we show that there exists a fractional matching $x = (x_e)_{e \in E}$ which cannot be rounded on-line while maintaining $\mathbb{P}(e \text{ gets matched}) = x_e$. This contrasts the method to round vertex cover found by Wang and Wong.

Theorem 2.6 *[19] For a bipartite graph $G = (L, R, E)$, any on-line assignment of fractional vertex cover values $(y_v)_{v \in V}$ can be rounded to an integral vertex cover V' with $\mathbb{E}[V'] = \sum_{v \in V} y_v$.*

Proof. Before the first on-line vertex arrives, sample $t \in [0, 1]$ uniformly at random. For a vertex $v \in L$ arriving on-line, add it to V' if $y_v \geq t$. Otherwise, if $v \in R$, add v to V' if $y_v \geq 1 - t$. This gives a valid vertex cover, because for any edge $e = uv$, $y_u + y_v \geq 1$ implies that for any t either $y_u \geq t$ or $y_v \geq 1 - t$. Finally, $\mathbb{E}[|V'|] = \sum_{v \in L} \mathbb{P}(v \text{ gets added to } V') + \sum_{v \in R} \mathbb{P}(v \text{ gets added to } V') = \sum_{v \in V} y_v$. □

Note that the rounding method has to know if $v \in L$ or $v \in R$ in order to work. While in practical applications this is usually available information, in theory the algorithm may not be able to make that distinction. It is an open problem if such a rounding for vertex cover can be done without that information. Also note that for all $v \in V$ $\mathbb{P}(v \in V') = y_v$. Based on that, we define a rounding method for the on-line matching problem as follows.

Definition 2.7 *Suppose a graph G arrives on-line and each edge arrives with a value x_e, where $(x_e)_{e \in E}$ is a fractional matching. Let v_k be the k-th arriving vertex. A rounding method for the on-line matching problem is an algorithm that outputs, at the arrival of each vertex v_k, a random matching M_k such that:*

- *For all edges e arrived so far, $\mathbb{P}(e \in M_k) = x_e$.*
- *$M_{k-1} \subseteq M_k$.*

If a rounding method for the on-line matching problem exists, we could use it to create an algorithm with competitive ratio greater than $\frac{1}{2}$ as follows: Simulate the fractional matching algorithm by Wang and Wong and output the matching given by the rounding method.

But no rounding method for the on-line matching problem exists, even if the algorithm knows the side of each arriving vertex.

Theorem 2.8 *There exists an on-line assignment of fractional matching values with no rounding method.*

Proof. We proceed by contradiction and suppose such a rounding method exists. Consider two pairs of vertices arriving on-line, where each pair is connected, so we have two edges e_1 and e_2. Let x be the fractional matching $x_{e_1} = x_{e_2} = \frac{1}{2}$. (See Figure 2.4.)

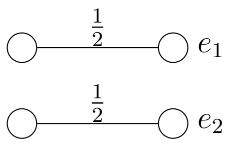

Figure 2.4: Fractional matching on the subgraph induced by the first 4 arrived vertices.

Define M as the random set consisting of the matching output by the rounding method. We define the matching $M_{i_1,i_2,\ldots,i_k} = \{e_{i_1}, e_{i_2}, \ldots, e_{i_k}\}$. Then we have, for each edge e,

$$\begin{aligned} x_e =&\mathbb{P}(e \in M) \\ =&\mathbb{P}(e \in M_1 | M = M_1)\mathbb{P}(M = M_1) + \mathbb{P}(e \in M_2 | M = M_2)\mathbb{P}(M = M_2) + \\ &\mathbb{P}(e \in M_{1,2} | M_{1,2} = M)\mathbb{P}(M = M_{1,2}). \end{aligned}$$

So we get that $\frac{1}{2} = \mathbb{P}(M = M_1) + \mathbb{P}(M = M_{1,2}) = \mathbb{P}(M = M_2) + \mathbb{P}(M = M_{1,2})$. Using this and writing the previous identity vectorially, we obtain

$$\begin{aligned} x &= \mathbb{P}(M = M_1)\chi(M_1) + \mathbb{P}(M = M_2)\chi(M_2) + \mathbb{P}(M = M_{1,2})\chi(M_{1,2}) \\ &= (\frac{1}{2} - \mathbb{P}(M = M_{1,2}))\chi(M_1) + (\frac{1}{2} - \mathbb{P}(M = M_{1,2}))\chi(M_2) + \mathbb{P}(M = M_{1,2})\chi(M_{1,2}). \end{aligned}$$

Suppose $\mathbb{P}(M = M_{1,2}) > 0$. In that case, assume a vertex arrives connected to one endpoint of e_1 and to one endpoint of e_2, with the fractional matching valued $\frac{1}{2}$ on each new edge, like in Figure 2.5.

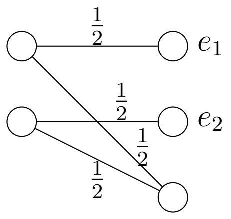

Figure 2.5: Next two vertices if $\mathbb{P}(M = M_{1,2}) > 0$.

Let f_1 and f_2 be the new edges adjacent to e_1 and e_2 respectively. Since $\mathbb{P}(f_1 \in M) = \mathbb{P}(f_2 \in M)\frac{1}{2}$ and the events are disjoint, $\mathbb{P}(f_1 \in M \text{ or } f_2 \in M) = 1$. But none of those edges can be added to $M_{1,2}$, which has to occur with positive probability, a contradiction. Therefore, $\mathbb{P}(M = M_{1,2}) = 0$ and

$$x = \frac{1}{2}\mathbb{P}(M = M_1)\chi(M_1) + \frac{1}{2}\mathbb{P}(M = M_2)\chi(M_2).$$

Next, suppose a new pair of connected vertices arrives. Call the new edge e_3 and let $x_{e_3} = \frac{1}{2}$ (see Figure 2.6).

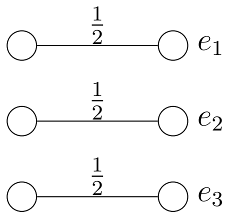

Figure 2.6: Fractional matching on the subgraph induced by the first 6 arrived vertices.

The rounding method would have to choose between adding e_3 to either the matching M_1 or M_2, so $\mathbb{P}(M = M_{2,3}) + \mathbb{P}(M = M_{1,3}) = \frac{1}{2}$ and so

$$\begin{aligned} x =& (\frac{1}{2} - \mathbb{P}(M = M_{1,3}))\chi(M_1) + \mathbb{P}(M = M_{1,3})\chi(M_{1,3}) + \\ & (\frac{1}{2} - \mathbb{P}(M = M_{2,3}))\chi(M_2) + \mathbb{P}(M = M_{2,3})\chi(M_{2,3}). \end{aligned}$$

If the next vertex arrives connected to one endpoint of e_1 and to one endpoint of e_2, with the fractional matching valued $\frac{1}{2}$ on each new edge (see Figure 2.7), then again with probability 1 one of those new edges has to be in M. But because none of these edges can be added to $M_{2,3}$, necessarily $\mathbb{P}(M = M_{2,3}) = 0$.

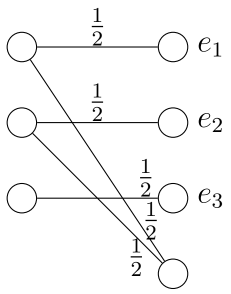

Figure 2.7: Fractional matching on the subgraph induced by the first 7 arrived vertices.

Repeating the previous argument but with a vertex connected to one endpoint of e_1 and one of e_3 yields $\mathbb{P}(M = M_{1,3}) = 0$. But then $0 = \mathbb{P}(M = M_{1,3}) = 1 - \mathbb{P}(M = M_{2,3}) = 1$, a contradiction. □

This result does not discard the approach of finding a big fractional matching and rounding it into an integral matching. But if one wants to round a fractional matching, the resulting integral matching might have to be smaller. This suggests there exists a gap between the best possible competitive ratio for both models.

Chapter 3

The Random Order Model

In the random order model, there is a bipartite graph $G(V, E)$ which the algorithm does not know. Vertices of this graph arrive in the order induced by a random, uniformly chosen permutation $v_1, v_2, \ldots, v_n$ of V. Similarly to other models introduced so far, the algorithm learns which vertices v_i are neighbours of v_k for $i < k$ when vertex v_k arrives. Equivalently, when vertex v_k arrives, the induced graph $G[v_1, \ldots, v_k]$ is revealed to the algorithm. Upon arrival of v_k the algorithm may only add an edge $v_i v_k$ to the matching, for $i < k$. Note that, in this whole chapter, the probability will be taken in regard to both the randomness of the algorithm and the randomness of the input.

The algorithm does not know the size of the input either. This information could eventually help for sampling techniques used by the algorithm.

3.1 Related Work

A very similar model has been studied by Goel and Mehta [8], where one side of the bipartite graph is fixed and the other side arrives on-line, but in a random order. In the case when one side is fixed and with consistent tie-breaking of the fixed side, the Greedy algorithm is equivalent to the Ranking algorithm in the adversarial model, because the order in which the vertices arrive induces a ranking on them. Therefore the Greedy algorithm has a competitive ratio of $1 - \frac{1}{e}$ in the random order model with one fixed side. The model was then further analyzed by Mahdian and Yan [15] and Karande, Mehta and Tripathy [11]. Both groups independently study the Ranking algorithm in this model. This algorithm is the same algorithm as in the adversarial model, and since it achieves a competitivity of $1 - \frac{1}{e}$ in the adversarial model it does at least as well in the random order model. This bound has been improved: it is known that the competitivity of this algorithm lies in the interval $[0.696, 0.727]$ [15, 11].

Another line of related research studies randomised variants of the Greedy algorithm, giving randomised approximation algorithms for the matching problem in general graphs. In these variants the graph does not arrive on-line, and the main goal is to develop fast algorithms with good approximation ratios. The following two algorithms have been studied for general graphs:

Algorithm 5 Random-Greedy

```
Choose a random permutation of the vertices $v_1, \dots, v_n$
for $i = 1, \dots, n$ do
    if $v_i$ has unmatched neighbours then
        Choose an unmatched neighbour $u$ at random
        Add $uv$ to the matching
    end if
end for
```

The Ranking algorithm, which takes its name from its variant in on-line bipartite matching:

Algorithm 6 Ranking

```
Choose a random permutation of the vertices $v_1, \dots, v_n$
for $i = 1, \dots, n$ do
    if $v_i$ has unmatched neighbours then
        Choose the unmatched neighbour $u$ which comes first in the permutation
        Add $uv$ to the matching
    end if
end for
```

From now on in this chapter we always refer to this algorithm as Ranking, despite the fact the algorithm by Karp, Vazirani and Vazirani has the same name.

Since both the Ranking and Random-Greedy algorithms output maximal matchings, their competitivity is at least $\frac{1}{2}$ (see the proof of Theorem 1.20). It has been shown that both algorithms have a competitive ratio strictly greater than $\frac{1}{2}$, even on general graphs. We summarize the process on the competitive ratio of these algorithms in Table3.1:

Algorithm	Year	Lower Bound	Upper Bound
Random-Greedy	1995 [2]	$\frac{1}{2} + \frac{1}{400000}$	–
	2012 [9]	$\frac{1}{2} + \frac{1}{256}$	$\frac{2}{3}$
Ranking	2012 [?]	–[1]	0.75
	2013 [5]	$\frac{2(5-\sqrt{7})}{9} \approx 0.523$	0.724 (experimental bound)

Table 3.1: Competitive ratios of algorithms for matching in general graphs.

None of these algorithms actually work in our random order model, because in both algorithms a vertex could get matched to a vertex that has not arrived yet.

[1]The paper seems to have an incorrect proof of a lower bound of 0.56

3.2 Lower Bound and the Unknown Distribution Model

An interesting motivation to introduce the random order model comes from the *unknown distribution model*. In the unknown distribution model, arriving vertices are sampled from a probability distribution unknown to the algorithm. In other words, there is a base graph $\hat{G} = (\hat{V}, \hat{E})$ and a probability vector over $\hat{V}$, both unknown to the algorithm. Vertices are then sampled from $\hat{G}$ and arrive on-line, adding edges to pairs that were originally connected in $\hat{G}$. If a vertex is sampled more than once, copies of it arrive on-line. This can be used to model real-life on-line matching situations where some assumption can be made about the on-line arrivals. For example, if we want to match blood donations to compatible patients, a vertex in $\hat{G}$ would represent the blood type and if the vertex is a donor or a patient. It is reasonable to assume that arrival times are independent and follow an exponential distribution, which fixes the probability distribution of which vertex arrives next.

Since there is randomness in the input, this time we have to extend the definition of the competitive ratio.

Definition 3.1 *Let $\mathcal{A}$ be an algorithm for the on-line matching problem in the unknown distribution model. For an input I, let $\mathbb{E}[\mathcal{A}(I)]$ be the expected size of the solution found by the algorithm and $OPT(I)$ the size of the optimal solution. We say $\mathcal{A}$ achieves a competitive ratio α if $\frac{\mathbb{E}[M(\mathcal{A})]}{\mathbb{E}[OPT(I)]} \geq \alpha$ for all input I.*

Karante et al. [11] study the case where one side of the on-line graph is already fixed and known to the algorithm. We will assume that the graph is initially empty and vertices of any side can arrive on-line. When one side is fixed, the random order model is more strict than the unknown distribution model, because any algorithm in the random order model can also be used in the unknown distribution model [11]. This holds for the model where both sides arrive on-line as well, as shown in the following lemma.

Lemma 3.2 *If an on-line algorithm $\mathcal{A}$ for the random order model has a competitive ratio of α, then the same algorithm achieves a competitive ratio of at least α in the unknown distribution model.*

Proof. Consider an instance in the unknown distribution model, consisting of n vertices sampled from a base graph $\hat{G}$. For a vertex v in G, define $\hat{v}$ as the vertex in $\hat{G}$ it was sampled from. Let P be the set of all possible vertex sequences in the problem instance. Define the following equivalence relation $\sim$ over P: $v_1, \ldots, v_n \sim u_1, \ldots, u_n$ if the multisets $\{\hat{v_1}, \ldots, \hat{v_n}\}$ and $\{\hat{u_1}, \ldots, \hat{u_n}\}$ are equal. $P/$ will denote the set of equivalence classes of . Each equivalence class can be seen as the permutations of one sequence of vertex arrivals. Therefore, members of each equivalence class have the same probability to be sampled and the resulting graph is the same for each sequence in the class. For a sequence of vertices $v_1, \ldots, v_n$, call $[v_1, \ldots, v_n]$ its equivalence class, $M(v_1, \ldots, v_n)$ the matching output by the algorithm with that vertex sequence as an input and $OPT([v_1, \ldots, v_n])$ the size of the maximum matching in the graph induced by $[v_1, \ldots, v_n]$ (which is well defined since the resulting graph is the same for all members in the equivalence class). Then

$$\begin{aligned}\mathbb{E}[|M|] &= \sum_{(v_1,\ldots,v_n)\in P} |M(v_1,\ldots,v_n)|\mathbb{P}(v_1,\ldots,v_n \text{ gets sampled}) \\ &= \sum_{[v_1,\ldots,v_n]\in P/\sim} \ \sum_{(u_1,\ldots,u_n)\in[v_1,\ldots,v_n]} |M(u_1,\ldots,u_n)|\mathbb{P}(u_1,\ldots,u_n \text{ gets sampled}) \\ &= \sum_{[v_1,\ldots,v_n]\in P/\sim} \mathbb{P}(v_1,\ldots,v_n \text{ gets sampled})n! \sum_{(u_1,\ldots,u_n)\in[v_1,\ldots,v_n]} |M(u_1,\ldots,u_n)|\frac{1}{n!}.\end{aligned}$$

We identify each equivalence class in $P/\sim$ as an instance in the random order model, because it is the set of all permutations of vertices of the graph induced by $[v_1,\ldots,v_k]$. Thus the second sum is the expected size of the matching in the random order model, which we know is greater than $\alpha OPT([v_1,\ldots,v_n])$ and we can bound

$$\begin{aligned}\mathbb{E}[|M|] &\geq \sum_{[v_1,\ldots,v_n]\in P/\sim} \mathbb{P}(v_1,\ldots,v_n \text{ gets sampled})n!\alpha OPT([v_1,\ldots,v_n]) \\ &\geq \alpha \sum_{v_1,\ldots,v_n\in P} \mathbb{P}(v_1,\ldots,v_n \text{ gets sampled})OPT([v_1,\ldots,v_n]) \\ &\geq \alpha\mathbb{E}[OPT].\end{aligned}$$

Which proves the lemma. □

We now try to find an algorithm with good competitivity for the random order model. Since the Greedy algorithm achieves a competitivity of $\frac{1}{2}$ in this model, we are interested in finding algorithms with greater competitivity. With this in mind, we will analyse the performance of the following algorithm for the random order model.

Algorithm 7 Shuffle

```
for $v$ arriving on-line do
  if $v$ has unmatched neighbours then
    $u \leftarrow$ the unmatched neighbour of $v$ which arrived first
    Add $uv$ to the matching
  end if
end for
```

This algorithm is a natural adaptation of the Ranking algorithm for the on-line random order model. We cannot use Ranking in the random order model, because it needs to know all neighbours of each vertex a priori. But Ranking and Shuffle are actually the same algorithm, as shown by the following theorem.

Definition 3.3 *For $G = (V, E)$ a (not necessarily bipartite) graph, we define Ranking(π) (resp. Shuffle(π)) as the matching obtained when running* Ranking *(resp.* Shuffle*) with π as the permutation of vertices.*

Theorem 3.4 *Shuffle(π)=Ranking(π).*

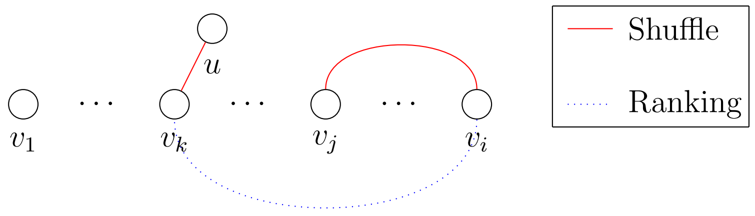

Figure 3.1: Illustration with vertices shown in order.

Proof. Let v_i be the ith vertex in the order induced by π and proceed by induction in $G[v_1, \ldots, v_i]$, the graph Shuffle sees. Since Shuffle$(\pi) \cap G[v_1, \ldots, v_i]$ is always a maximal matching, it suffices to prove that Shuffle$(\pi) \cap G[v_1, \ldots, v_i] \subseteq$ Ranking$(\pi) \cap G[v_1, \ldots, v_i]$.

Suppose v_i arrives and gets matched by the Shuffle algorithm to a vertex v_j in $G[v_1, \ldots, v_i]$. We will prove that Ranking matches v_j to v_i when inspecting the free neighbours of v_j. For that we have to prove two things: v_i is an unmatched vertex when Ranking inspects the free neighbours of v_j and v_i is the first vertex (in the order of π) with that property.

For the sake of contradiction, suppose v_i has already been matched by the Ranking algorithm when Ranking inspects the neighbours of v_j. Then v_i is already matched by the Ranking algorithm to a vertex which Ranking inspected earlier than v_j, say v_k with $k < j$. If v_k were unmatched by the Shuffle algorithm in $G[v_1, v_2, \ldots, v_{i-1}]$, then Shuffle would not have matched v_i to v_j, so v_k must be matched by Shuffle to a vertex u in $\{v_1, v_2, \ldots, v_{i-1}\}$. (See Figure 3.1 for clarity.) By the induction hypothesis, the edge $v_k u$ is in both Ranking$(\pi) \cap G[v_1, \ldots, v_{i-1}]$ and in Shuffle$(\pi) \cap G[v_1, \ldots, v_{i-1}]$. But then v_k is matched to both v_i and to u in Shuffle$(\pi) \cap G[v_1, \ldots, v_{i-1}]$, a contradiction.

Now, if v_i is not the first unmatched neighbour of v_j when Ranking inspects the neighbours of v_j, then v_j would get matched to a vertex in $\{v_1, \ldots, v_{i-1}\}$. By the induction hypothesis, this edge would also be in Shuffle$(\pi) \cap G[v_1, \ldots, v_{i-1}]$. But then Shuffle could not match v_i to v_j, contradicting our initial assumption. □

Corollary 3.5 *The competitive ratios of* Shuffle *and* Ranking *coincide.*

Because of the previous theorem and corollary, we will no longer differentiate the Shuffle and the Ranking algorithms, and just say the Ranking algorithm.

Next we prove that, in the random order model, the Ranking algorithm has exactly the same competitive ratio as the Ranking algorithm in the random order model where one side is fixed, which has already been studied [11, 15].

Lemma 3.6 *Let $G = (V, E)$ be a (not necessarily bipartite) graph and $\pi = v_1 v_2 \ldots v_n$ a permutation of its vertices. Suppose that for some i one of the following holds:*

- *The edge $v_i v_{i+1}$ exists and it is not contained in a triangle.*
- *The distance between v_i and v_{i+1} is greater than 2.*

Then we can exchange the order of v_i and v_{i+1} in π and still obtain the same matching when running Ranking.

Proof. It suffices to show that both v_i and v_{i+1} get matched the same way in both cases, since then for the other vertices the algorithm proceeds identically. Call ρ the permutation obtained if we exchange the order of v_i and v_{i+1} in π. Since the algorithm always outputs maximal matchings, it suffices to show that Ranking(π) $\subseteq$ Ranking(ρ).

For the first case, suppose v_i and v_{i+1} are neighbours but not contained in a triangle. We proceed by cases:

- Suppose v_i gets matched to v_{i+1} in Ranking(π). That means v_i does not have unmatched neighbours upon its arrival in π and the only unmatched neighbour of v_{i+1} upon its arrival is v_i. Therefore, if we switch their order, when v_{i+1} arrives it has no unmatched neighbours, and when v_i arrives its only unmatched neighbour is v_{i+1} and they get matched.
- Suppose now v_i gets matched to a vertex u, different to v_{i+1}. Therefore u arrived before v_{i+1} in π. The vertex u cannot be a neighbour of v_{i+1}, otherwise v_i and v_{i+1} would be contained in a triangle. If we switch the order, v_{i+1} cannot be matched to u and u will still be the earliest neighbour of v_i upon its arrival, thus uv_i is in Ranking(ρ).
- Finally, suppose v_{i+1} gets matched to a vertex u different to v_i. This case is the same as the previous case, but with the roles of π and ρ reversed.

Suppose now the distance between v_i and v_{i+1} is greater than two. If v_i gets matched to a vertex u, u is not a neighbour of v_{i+1}. So, when switching the order of v_i and v_{i+1}, u will still be the neighbour of v_i that arrived earliest, and so uv_i is still in the matching. The case of v_{i+1} getting matched is again the same, but with the roles of π and ρ reversed. □

Corollary 3.7 *If $G = (L, R, E)$ is a bipartite graph, and π a permutation of its vertices, define π_L as the permutation induced by π on L and π_R as the permutation induced by π on R. Define Ranking(π_L, π_R) as the matching obtained when the permutation is π_L followed by π_R. Then Ranking(π)=Ranking(π_L, π_R).*

Proof. Suppose $\pi = v_1, \ldots, v_n$ and for some i, $v_i \in R$ but $v_{i+1} \in L$. If v_i and v_{i+1} are neighbours they are not in a triangle, because G is bipartite. If they are not neighbours, their distance is at least 3, since they are in opposite sides of a bipartite graph. Thus, by the previous lemma we can exchange the order of v_i and v_{i+1} in π and still obtain the same matching. We can apply this as many times as necessary so that the permutation consists only of vertices in L before vertices in R. □

We summarize the previous results in the following theorem.

Theorem 3.8 *In the random order model for bipartite graphs, where the initial set of vertices is empty, the* Ranking *algorithm achieves the same competitive ratio as when one of the sides is fixed initially.*

This competitive ratio is known to lie somewhere in the interval [0.696, 0.727] [11, 15]. Hence, using Lemma 3.2 we get the following corollary for the unknown distribution model.

Corollary 3.9 RANKING *is 0.696-competitive in the unknown distribution model.*

3.3 Upper Bound

Our goal in this section is to find an upper bound on the competitive ratio of any algorithm in the random order model. We prove that any algorithm can only get a matching of expected value at most $\frac{7}{4}$ on a path of size three. This translates into the following upper bound:

Lemma 3.10 *No algorithm can achieve a competitivity greater than* 0.875 *in the random order model.*

PROOF. Let $\mathcal{A}$ be an algorithm for the on-line matching problem in the random order model. Let us test $\mathcal{A}$ on a path of length three with vertex set $\{v_1, v_2, v_3, v_4\}$ (see Figure 3.2).

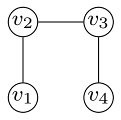

Figure 3.2: A path of length three.

Define M as the matching output by the algorithm and $p = \mathbb{P}(\mathcal{A}$ adds the first edge it sees to $M \mid$ the first two arriving vertices are connected). By symmetry, we can assume that the first vertex arriving is either v_1 or v_2. Then:

$$\begin{aligned}\mathbb{E}[|M|] =& \mathbb{E}[|M| \mid v_1 \text{ arrives first and } v_2 \text{ second}]\frac{1}{6} + \mathbb{E}[|M| \mid v_1 \text{ arrives first and } v_3 \text{ second}]\frac{1}{6} + \\ & \mathbb{E}[|M| \mid v_1 \text{ arrives first and } v_4 \text{ second}]\frac{1}{6} + \mathbb{E}[|M| \mid v_2 \text{ arrives first and } v_1 \text{ second}]\frac{1}{6} + \\ & \mathbb{E}[|M| \mid v_2 \text{ arrives first and } v_3 \text{ second}]\frac{1}{6} + \mathbb{E}[|M| \mid v_2 \text{ arrives first and } v_4 \text{ second}]\frac{1}{6}\end{aligned}$$

Next we calculate each of those conditional expectations:

- If v_1 arrives first and v_2 second, then $\mathcal{A}$ adds that edge to the matching with probability p. If it adds that edge, the matching cannot be larger than 2. Otherwise the matching can only be the edge v_2v_3, so

$$\mathbb{E}[|M| \mid v_1 \text{ arrives first and } v_2 \text{ second}] \leq 2p + (1-p) = p + 1.$$

  The same reasoning holds when v_2 arrives first and v_1 second.

- If v_1 arrives first and v_3 second, there are two cases: With probability $\frac{1}{2}$ the next vertex arriving is v_4, in which case we will bound the matching output by the algorithm by the optimal one. Otherwise, v_2 arrives next and the algorithm would see a path of length two. Let q be the probability of choosing the edge incident to the vertex that arrived first when the algorithm sees a path of length 2. Then

$$\begin{aligned}\mathbb{E}[|M| \mid v_1 \text{ arrives first and } v_3 \text{ second}] &\leq \frac{1}{2} \cdot 2 + \frac{1}{2}(2q + (1-q)) \\ &= \frac{1}{2}(3+q)\end{aligned}$$

  Repeating the same reasoning when v_2 arrives first and v_4 second gives:

$$\begin{aligned}\mathbb{E}[|M| \mid v_2 \text{ arrives first and } v_4 \text{ second}] &\leq \frac{1}{2} \cdot 2 + \frac{1}{2}(q + 2(1-q)) \\ &= \frac{1}{2}(3+(1-q))\end{aligned}$$

  So $\mathbb{E}[|M| \mid v_1 \text{ arrives first and } v_3 \text{ second}] + \mathbb{E}[|M| \mid v_2 \text{ arrives first and } v_4 \text{ second}] \leq \frac{7}{2}$.
- We will bound $\mathbb{E}[|M| \mid v_1 \text{ arrives first and } v_4 \text{ second}] \leq 2$.
- If v_2 arrives first and v_3 second, the algorithm adds v_2v_3 to the matching with probability p. Otherwise it might output the optimal matching, so

$$\mathbb{E}[|M| \mid v_2 \text{ arrives first and } v_3 \text{ second}] \leq p + 2(1-p) = 2 - p$$

Therefore

$$\begin{aligned}\mathbb{E}[|M|] &\leq (p+1)\frac{1}{6} + \frac{7}{2} \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + (p+1)\frac{1}{6} + (2-p) \cdot \frac{1}{6} \\ &= \frac{p}{6} + \frac{19}{12} \\ &\leq \frac{1}{6} + \frac{19}{12} \\ &= \frac{7}{4}.\end{aligned}$$

Thus, $\frac{\mathbb{E}[|M|]}{2} \leq \frac{7}{8} = 0.875$. □

It is interesting to note that the Ranking algorithm actually outputs a matching of average size 1.75 in this input.

3.4 General Graphs

The fact that the Ranking and Shuffle algorithms have the same competitive ratio, implies another corollary for the on-line matching problem in the random order model, but for general graphs. In Table 3.1 we can see the best known competitive ratio for the Ranking algorithm in general graphs, which is greater than $\frac{1}{2}$, so there exists an algorithm with a competitive ratio greater than $\frac{1}{2}$ for the on-line matching problem in the random order model for general graphs.

Corollary 3.11 *The* Shuffle *algorithm achieves a competitivity of at least* $\frac{2(5-\sqrt{7})}{9} \approx 0.523$ *in general graphs.*

The Random-Greedy algorithm, on the other hand, cannot be used for the on-line matching problem in the random order model. A natural adaptation for this, is the following algorithm.

Algorithm 8 Random-Past-Greedy

```
Choose a random permutation of the vertices $v_1, \ldots, v_n$
for $i = 1, \ldots, n$ do
  Let $H = G[v_1, \ldots, v_i]$
  if $v_i$ has unmatched neighbours in $H$ then
    Choose an unmatched neighbour $u$ in $H$ uniformly at random
    Add $uv$ to the matching
  end if
end for
```

The Random-Past-Greedy algorithm always outputs a maximal matching, so it achieves a competitivity of $\frac{1}{2}$ (see the proof of Theorem 1.20). Its similarity to the Random-Greedy algorithm suggests the algorithm has a greater competitivity than $\frac{1}{2}$. We will now analyse the performance of the Random-Past-Greedy algorithm on a family of graphs which we believe is a hard instance for the algorithm.

Lemma 3.12 *The competitivity of the* Random-Past-Greedy *algorithm is at most* $\frac{239}{324} \approx 0.738$.

Proof. For a positive even integer n, consider the graph G_n obtained by taking the complete graph K_n and adding n new vertices, each connected to a different vertex of K_n. We write $V = C \cup P$, where C (clique) is the complete subgraph of n vertices and P (pendant) are the adjacent vertices. Figure 3.3 shows G_5.

This graph contains a perfect matching of size n. Intuitively speaking, the graph also contains many small maximal matchings, and since the algorithm always outputs a maximal matching we believe this family of graph to be a good family to search for upper bounds on the competitivity of the algorithm. For a vertex v in the clique, we name v^* its neighbour in P. To induce a uniform permutation, assume each vertex v in the graph will arrive at a certain time T_v in the interval $(0, 1)$, each with independent uniform distribution $\mathcal{U}(0, 1)$. This will help in the analysis. We will freely use that for $x \in [0, 1]$ $\mathbb{P}(T_v \leq x) = x$.

Define the random variable M as the matching obtained by the Random-Past-Greedy algorithm. Since $|M| = \sum_{e \in E} \mathbb{1}_{\{e \in M\}}$, we have

$$\mathbb{E}[|M|] = \sum_{\{v,u\} \subseteq C} \mathbb{P}(vu \in M) + \sum_{v \in C} \mathbb{P}(vv^* \in M).$$

Due to the symmetry of the graph the probabilities in the sums are equal and can be rewritten

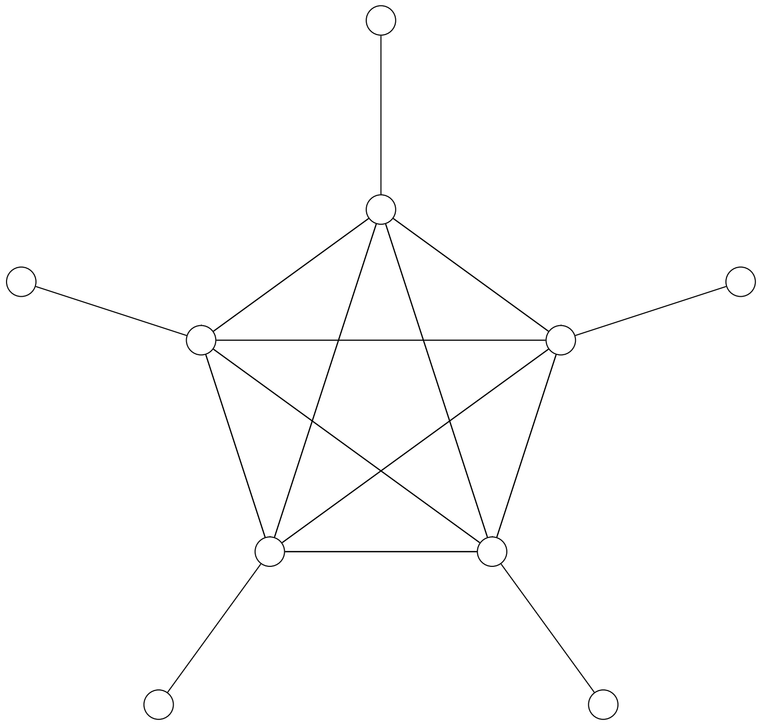

Figure 3.3: G_5.

by fixing v and u in C:

$$\mathbb{E}[|M|] = \frac{n(n-1)}{2}\mathbb{P}(vu \in M) + n\mathbb{P}(vv^* \in M).$$

Also note that if v does not get matched to v^*, then it gets matched to one of its neighbours in the complete graph, so we have:

$$1 - \mathbb{P}(vv^* \in M) = (n-1)\mathbb{P}(vu \in M)$$

So we can write $\mathbb{E}[|M|]$ in terms of $\mathbb{P}(vv^* \in M)$ as follows:

$$\mathbb{E}[|M|] = \frac{n}{2}(1 + \mathbb{P}(vv^* \in M)) \tag{3.1}$$

Therefore, bounds on $\mathbb{P}(vv^* \in M)$ translate to bounds in $\mathbb{E}[|M|]$.

For $t \in (0,1)$, define the random variable $C(t)$ as the amount of unmatched vertices in the clique among the vertices arrived strictly before t. Note that, since the matching is always maximal, $C(t) \in \{0,1\}$.

We define the following events (note they are disjoint):

$$A_1 := T_{v^*} > T_v,\ vv^* \in M.$$
$$A_2 := T_{v^*} < T_v,\ C(T_v) = 0.$$
$$A_3 := T_{v^*} < T_v,\ C(T_v) = 1.$$

Then we have:

$$\mathbb{P}(vv^* \in M) = \mathbb{P}(A_1) + \mathbb{P}(A_2) + \frac{1}{2}\mathbb{P}(A_3). \tag{3.2}$$

This is because if $T_{v^*} < T_v$ and there is no unmatched vertex in the clique, then when v arrives v gets matched to v^*. But if there is an unmatched vertex in the clique, then v gets matched to v^* with probability $\frac{1}{2}$ since the algorithm chooses a random neighbour between v^* and the vertex already in the clique.

The event A_1 is highly unlikely. In fact, we will prove that $\mathbb{P}(A_1) = o(1)$. Remember that T_v and T_{v^*} follow an uniform distribution. Therefore, conditioning, we get

$$\mathbb{P}(A_1) = \int_0^1 \int_{t_v}^1 \mathbb{P}(A_1 | T_v = t_v, T_{v^*} = t_{v^*}) dt_{v^*} dt_v. \tag{3.3}$$

Because the matching output is always maximal, v getting matched to v^* is equivalent to any other vertex w of the clique not getting matched to v. Therefore

$$A_1 = T_v < T_{v^*}, \forall w \in C \setminus \{v\} \ w \text{ does not get matched to } v \ . \tag{3.4}$$

Given that $T_v < T_{v^*}$, if no vertex $w \in C \setminus \{v\}$ got matched to v, then each w must fulfil one of the following conditions:

- $T_w \notin (T_v, T_{v^*})$.
- $T_w \in (T_v, T_{v^*})$, $T_{w^*} < T_w$ and the algorithm matches w to w^* when w arrives.

Therefore,

$$\begin{aligned}
\mathbb{P}(A_1 | T_v = t_v, T_{v^*} = t_{v^*}) &\leq \mathbb{P}\Big(\bigcap_{w \in C \setminus \{v\}} \big[T_w \notin (T_v, T_{v^*}) \cup T_{w^*} < T_w\big] | T_v = t_v, T_{v^*} = t_{v*}\Big) \\
&= \mathbb{P}\Big(\bigcap_{w \in C \setminus \{v\}} \big[T_w \notin (t_v, t_{v^*}) \cup T_{w^*} < T_w\big]\Big)
\end{aligned}$$

And since the arrival times are independent, we can write the last line as a product, by fixing a $w \in C \setminus \{v\}$:

$$\begin{aligned}
\mathbb{P}(A_1 | T_v = t_v, T_{v^*} = t_{v^*}) &\leq \mathbb{P}\Big(T_w \notin (t_v, t_{v^*}) \cup T_{w^*} < T_w\Big)^{n-1} \\
&= \Big(\int_0^1 \mathbb{P}(T_w \notin (t_v, t_{v^*}) \cup T_{w^*} < T_w) | T_w = t_w) dt_w \Big)^{n-1} \\
&= \Big(\int_0^1 \mathbb{P}(t_w \notin (t_v, t_{v^*}) \cup T_{w^*} < t_w) dt_w \Big)^{n-1} \\
&= \Big(1 - (t_{v^*} - t_v) + \int_{t_v}^{t_{v^*}} \mathbb{P}(T_{w^*} < t_w) dt_w \Big)^{n-1} \\
&= \Big(1 - (t_{v^*} - t_v) + \frac{t_{v^*}^2 - t_v^2}{2} \Big)^{n-1} \\
&= \Big(1 + (t_{v^*} - t_v)(\frac{t_{v^*} + t_v}{2} - 1) \Big)^{n-1}
\end{aligned}$$

So using this in 3.3:

$$\mathbb{P}(A_1) = \int_0^1 \int_{t_v}^1 \Big(1 + (t_{v^*} - t_v)(\frac{t_{v^*} + t_v}{2} - 1)\Big)^{n-1} dt_{v^*} dt_v.$$

Now, for $t_{v^*} \neq 1 \neq t_v$, $1 + (t_{v^*} - t_v)(\frac{t_{v^*}+t_v}{2} - 1) < 1$. Therefore, $\lim_{n\to\infty} \Big(1 + (t_{v^*} - t_v)(\frac{t_{v^*}+t_v}{2} - 1)\Big)^{n-1} = 0$ almost everywhere. Thus, by the dominated convergence theorem

$$\mathbb{P}(A_1) = \int_0^1 \int_{t_v}^1 \Big(1 + (t_{v^*} - t_v)(\frac{t_{v^*} + t_v}{2} - 1)\Big)^{n-1} dt_{v^*} dt_v = o(1).$$

We now calculate $\mathbb{P}(A_2)$ and $\mathbb{P}(A_3)$. Define v_k as the kth vertex arriving in C. Then, by the law of total probability:

$$\mathbb{P}(A_2) = \frac{1}{n} \sum_{k=1}^{n} \mathbb{P}(T_{v^*} < T_v, C(T_v) = 0 | v_k = v).$$

Note that conditional to v arriving at position k the events $T_{v^*} < T_v$ and $C(T_v) = 0$ are independent, since we can determine $C(T_v)$ with the arrival times of the $k-1$ vertices of the clique that arrived before v and the arrival times of their neighbours in P. By defining

$$\begin{aligned} p_k &= \mathbb{P}(C(T_v) = 0 | v = v_k) \\ &= \mathbb{P}(C(T_{v_k}) = 0) \end{aligned}$$

we rewrite the equation as

$$\mathbb{P}(A_2) = \frac{1}{n} \sum_{k=1}^{n} \mathbb{P}(T_{v^*} < T_v | v_k = v) p_k$$

Because the arrival times T_v induce a uniform permutation, we can calculate $\mathbb{P}(T_{v^*} < T_v | v_k = v)$ by sampling $(T_u)_{u\in C}$ first and then T_{v^*}. With this reasoning, $\mathbb{P}(T_{v^*} < T_v | v_k = v) = \mathbb{P}(v^*$ is among the first k vertices in a permutation of $n+1$ vertices$) = \frac{k}{n+1}$. Therefore,

$$\mathbb{P}(A_2) = \frac{1}{n} \sum_{k=1}^{n} \frac{k}{n+1} p_k.$$

Repeating the previous argument for A_3 we get

$$\mathbb{P}(A_3) = \frac{1}{n} \sum_{k=1}^{n} \frac{k}{n+1} (1 - p_k).$$

And using this in equation 3.2

$$\mathbb{P}(vv^* \in M) = \frac{1}{n} \sum_{k=1}^{n} \frac{k}{n+1} \Big(\frac{p_k}{2} + \frac{1}{2}\Big) + o(1) \tag{3.5}$$

$$= \frac{1}{4} + \frac{1}{2n(n+1)} \sum_{k=1}^{n} k p_k + o(1). \tag{3.6}$$

Using that $0 \leq p_k \leq 1$ and equation 3.1 we get $\frac{5}{8}n \leq \mathbb{E}(M) \leq \frac{3}{4}n + o(n)$. To get a better estimate, we will try to determine better bounds for p_k.

Note that $p_1 = 1$. We now find a recurrence equation for p_k, by conditioning on the events $C(T_{v_{k-1}}) = 0$ and $C(T_{v_{k-1}}) = 1$.

$$p_k = \mathbb{P}\big(C(T_{v_k}) = 0 | C(T_{v_{k-1}}) = 0\big)p_{k-1} + \mathbb{P}\big(C(T_{v_k}) = 0 | C(T_{v_{k-1}}) = 1\big)(1 - p_{k-1}) \tag{3.7}$$

Given that $C(T_{v_{k-1}}) = 0$, $C(T_{v_k}) = 0$ is equivalent to v_{k-1} getting matched to v^*_{k-1}, which happens if and only if v^*_{k-1} arrives before v_k.

$$\begin{aligned}\mathbb{P}\big(C(T_{v_k}) = 0 | C(T_{v_{k-1}}) = 0\big) &= \mathbb{P}(v^*_{k-1} \text{ arrives before } v_k)\\ &= \frac{k}{n+1}\end{aligned}$$

On the other hand, given that there is an unmatched vertex w in the clique when v_{k-1} arrives, then $C(T_{v_k}) = 0$ if and only if both v_{k-1} and w get matched. This happens if and only if one of the following disjoint events holds:

- v^*_{k-1} arrives after v_{k-1}, so w gets matched to v_{k-1}. The probability of this event is the same as the probability of a fixed object not being among the first $k-1$ in an uniform permutation of $n+1$ objects. This probability is equal to $1 - \frac{k-1}{n+1}$.
- v^*_{k-1} arrives before v_{k-1} and the algorithm matches v_{k-1} to w. The probability of the algorithm matching v_{k-1} to w in this case is $\frac{1}{2}$, and is an independent choice made by the algorithm. By the same reasoning as above, the probability of v^*_{k-1} arriving before v_{k-1} is $\frac{k-1}{n+1}$.
- v^*_{k-1} arrives before v_{k-1} and when v_{k-1} arrives the algorithm randomly matches v_{k-1} to v^*_{k-1}. Then w^* arrives between v_{k-1} and v_k, so w gets matched to w^*. By similar reasoning to the previous events, this event has probability $\frac{k-1}{n+1} \cdot \frac{1}{n+1} = \frac{k-1}{(n+1)^2}$.

So we have the following recurrence equation:

$$p_k = \frac{k}{n+1}p_{k-1} + \Big[1 - \frac{k-1}{n+1} + \frac{1}{2}\frac{k-1}{n+1} + \frac{k-1}{(n+1)^2}\Big](1 - p_{k-1}).$$

Which we can rewrite as

$$p_k = \Big[\frac{3k-1}{2(n+1)} - \frac{k-1}{(n+1)^2} - 1\Big]p_{k-1} + 1 - \frac{k-1}{2(n+1)} + \frac{k-1}{(n+1)^2}.$$

This recurrence equation has the following solution (which we will not use in this proof, but is listed here for possible future use):

$$p_k = \sum_{i=1}^{k} \Big[\prod_{j=i}^{k-1} \Big(\frac{3j+2}{2(n+1)} - \frac{j+1}{(n+1)^2} - 1\Big)\Big]\Big(1 - \frac{i-1}{2(n+1)} + \frac{i-1}{(n+1)^2}\Big)$$

Note that $k \leq \frac{2n+3}{3}$ if and only if $\frac{3k-1}{2(n+1)} - 1 \leq 0$. Therefore, for $k \leq \frac{2n+3}{3}$ the first part of the recurrence equation is negative and we can bound p_k for $k \leq \frac{2n+3}{3}$:

$$\begin{aligned}
p_k &= \Big[\frac{3k-1}{2(n+1)} - \frac{k-1}{(n+1)^2} - 1\Big]p_{k-1} + 1 - \frac{k-1}{2(n+1)} + \frac{k-1}{(n+1)^2} \\
&\leq 1 - \frac{k-1}{2(n+1)} + \frac{k-1}{(n+1)^2}(1 - p_{k-1}) \\
&\leq 1 - \frac{k-1}{2(n+1)} + \frac{k-1}{(n+1)^2}.
\end{aligned}$$

For $k > \frac{2n+3}{3}$ we will simply bound $p_k \leq 1$. Thus,

$$\begin{aligned}
\sum_{k=1}^{n} kp_k \leq& \sum_{k=1}^{\lfloor\frac{2n+3}{3}\rfloor} k\Big(1 - \frac{k-1}{2(n+1)} + \frac{k-1}{(n+1)^2}\Big) + \sum_{k=\lfloor\frac{2n+3}{3}\rfloor+1}^{n} kp_k \\
\leq& \frac{\lfloor\frac{2n+3}{3}\rfloor(\lfloor\frac{2n+3}{3}\rfloor+1)}{2} - \frac{\lfloor\frac{2n+2}{3}\rfloor(\lfloor\frac{2n+2}{3}\rfloor+1)(2\lfloor\frac{2n+2}{3}\rfloor+1)}{12(n+1)} + \\
& \frac{n(n+1)}{2} - \frac{\lfloor\frac{2n+3}{3}\rfloor(\lfloor\frac{2n+3}{3}\rfloor+1)}{2} + o(n^2) \\
\leq& \frac{1}{2}n^2 - \frac{4}{81}n^2 + o(n^2).
\end{aligned}$$

Therefore, in the limit and using equation 3.6

$$\limsup_{n\to\infty} \mathbb{P}(vv^* \in M) \leq \frac{1}{4} + \frac{1}{4} - \frac{2}{81}$$

And with that and equation 3.1 we can find an upper bound for the competitive ratio

$$\limsup_{n\to\infty} \frac{\mathbb{E}[|M|]}{n} \leq \frac{1}{2}(1 + \frac{1}{2} - \frac{2}{81}) = \frac{239}{324} \approx 0.738$$

We do not believe the above upper bound is tight, even for this family of graphs. In fact, numerical results suggest the above limit to be close to 0.714. □

We can use the same method used above to find a lower bound for the performance of the algorithm in this family of graphs. For $k \geq \frac{2n+3}{3}$, $\frac{3k-1}{2(n+1)} - 1 \geq 0$, so we can bound p_k for $k \geq \frac{2n+3}{3}$ similarly to before:

$$\begin{aligned}
p_k &= \Big[\frac{3k-1}{2(n+1)} - \frac{k-1}{(n+1)^2} - 1\Big]p_{k-1} + 1 - \frac{k-1}{2(n+1)} + \frac{k-1}{(n+1)^2} \\
&\geq 1 - \frac{k-1}{2(n+1)} + \frac{k-1}{(n+1)^2}(1 - p_k) \\
&\geq 1 - \frac{k-1}{2(n+1)}.
\end{aligned}$$

And so

$$\begin{aligned}\sum_{k=1}^{n} kp_k \geq& \sum_{k=1}^{\lfloor\frac{2n+3}{3}\rfloor} kp_k + \sum_{k=\lfloor\frac{2n+3}{3}\rfloor+1}^{n} \Big(k - \frac{k^2-k}{2(n+1)}\Big) \\ \geq& \sum_{k=\lfloor\frac{2n+3}{3}\rfloor+1}^{n} \Big(k - \frac{k^2}{2(n+1)}\Big) \\ \geq& \frac{1}{2}n^2 - \frac{2}{9}n^2 - \frac{1}{6}n^2 + \frac{4}{81}n^2 + o(n^2)\end{aligned}$$

Taking the limit and using equation 3.6

$$\liminf_{n\to\infty} \mathbb{P}(vv^* \in M) \geq \frac{1}{4} + \frac{1}{4} - \frac{1}{9} - \frac{1}{12} + \frac{2}{81} = \frac{107}{324}.$$

And so, using equation 3.1, we finish with

$$\liminf_{n\to\infty} \frac{\mathbb{E}[|M|]}{n} \geq \frac{1}{2}(1 + \frac{107}{324}) = \frac{431}{648} \approx 0.665.$$

To summarize, the competitive ratio of RANDOM-PAST-GREEDY when restricted to this family of graphs lies somewhere in the interval $[0.665, 0.738]$. The lower bound is quite far from $\frac{1}{2}$, so we believe the competitive ratio of this algorithm for any graph to be bounded away from $\frac{1}{2}$. Proving this algorithm's competitivity is $\frac{1}{2} + \varepsilon$ for general graphs for some $\varepsilon > 0$ is left as an open problem.

Chapter 4

The Offline Model

In this section we again relax the adversarial model. We consider the *offline model*, where the algorithm knows the input beforehand (both the graph and the order of arrival of vertices), but still needs to be competitive upon the arrival of every vertex. We present an optimal algorithm for this model, a lower bound of 0.526 for the competitivity of this algorithm and find an upper bound on the hardness of this problem of $\frac{\sqrt{5}+9}{19} \approx 0.591$.

4.1 Definition

To motivate the definition, suppose we have an algorithm for the adversarial model with competitive ratio α. Let G_i be the graph induced by the first i arrived vertices, $\mathrm{OPT}(G_i)$ the maximum size of the matching in G_i, and M_i the matching constructed so far by the algorithm. Since the algorithm does not know the size of the input, for every i we must have

$$\frac{\mathbb{E}[|M_i|]}{\mathrm{OPT}(G_i)} \geq \alpha.$$

In the offline model, we will require an algorithm to do exactly the same, but drop the lack of information assumption of the algorithm: It will know every move done by the adversary.

Definition 4.1 *In the offline model, a randomised algorithm with competitive ratio α is given a graph $G = (V, E)$ and a permutation of its vertices $\pi = v_1, \dots, v_n$. It then has to output a random matching M, with the restriction*

$$\frac{\mathbb{E}[|M \cap G_k|]}{OPT(G_k)} \geq \alpha \quad \forall k = 1, \dots, n \ , \tag{4.1}$$

where $G_k = G[v_1, \dots, v_k]$ is the graph induced by the first k vertices of π.

Since we know an algorithm in the adversarial model with competitive ratio $\frac{1}{2}$, which is simply the Greedy algorithm, this algorithm achieves the same competitive ratio in the offline model. We are interested in knowing if we can do better in this model.

Example Suppose the input is the following path of length three, where the labels of the vertices indicate the order in the permutation:

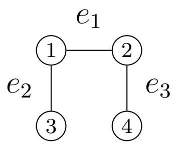

Figure 4.1: Input example.

Again, no deterministic algorithm can achieve a competitive ratio better than $\frac{1}{2}$ on this input. We want to find the best answer for this input, that is, we want to maximize α subject to $\frac{\mathbb{E}[|M \cap G_k|]}{\mathrm{OPT}(G_k)} \geq \alpha$ for all k. Without loss of generality, an optimal algorithm always outputs a maximal matching, since otherwise we can add edges to the matching output without lowering the competitivity. Let p be the probability that the algorithm chooses the matching $\{e_1\}$, then, since there are no other maximal matchings, $1-p$ is the probability that the algorithm chooses $\{e_2, e_3\}$.

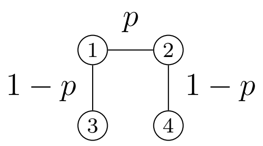

Figure 4.2: Probability of choosing each edge.

We can now write $\frac{\mathbb{E}[|M \cap G_k|]}{\mathrm{OPT}(G_k)} \geq \alpha$ for all k:

$$\begin{aligned} p &\geq \alpha \\ p + (1-p) &\geq \alpha \\ p + 2(1-p) &\geq 2\alpha \end{aligned}$$

Ignore the second inequality since it is implied by the first. We write the problem as an LP:

$$\begin{array}{lrl} \max & & \alpha \\ \text{s.t.} & p & \geq \alpha \\ & 2-p & \geq 2\alpha \\ & 0 \leq p \leq 1 & \end{array}$$

Since the linear problem has two dimensions, the optimal solution will be a vertex in the polytope defined by two restriction equalities. Because the GREEDY algorithm achieves a competitive ratio of $\frac{1}{2}$, we know that $p \geq \alpha \geq \frac{1}{2} > 0$. If $p = 1$, we get $\alpha = \frac{1}{2}$. Finally, if $0 \neq p \neq 1$, α is maximized when $p = \alpha$ and $2 - p = 2\alpha$. Solving this, we get $p = \frac{1}{3}$ and $\alpha = \frac{2}{3}$, which is therefore optimal.

Hence, no algorithm can have a better competitive ratio than $\frac{2}{3}$ for the offline matching problem. We will improve this bound later in this chapter.

4.2 An Optimal Algorithm

Generally, any on-line randomised algorithm for matching can be used to find a fractional matching x with $\sum_{e\in E} x_e = \mathbb{E}[|M|]$, where M is the random matching output by the algorithm (see Lemma 2.1). In this model, the converse is true, as shown by the following lemma.

Lemma 4.2 *In the offline model, any deterministic algorithm that finds a fractional matching satisfying equation 4.1 can be turned into a randomised algorithm without changing the competitive ratio.*

Proof. Let x be the fractional matching output by the algorithm. Then x is a convex combination of integral matchings $x = \sum \lambda_i \chi(M_i)$. We get a randomised algorithm as follows: Define the random variable M to be the matching M_i with probability λ_i, then for any k

$$\mathbb{E}[|M \cap G_k|] = \sum \lambda_i |M_i \cap G_k| = \sum_{e\in G_k} x_e \geq \alpha \mathrm{OPT}(G_k)$$

so this randomised algorithm has the same competitive ratio. $\square$

We can thus restrict ourselves to finding a fractional matching x such that $\sum_{e\in G_k} x_e \geq \alpha\mathrm{OPT}(G_k)$ for all k. Remember that since the graph is bipartite, the matching polytope is given by $\{(x_e)_{e\in E} | x_e \in [0,1]\ \forall e \in E,\quad \sum_{e\in\delta(v)} x_e \leq 1\ \forall v \in V\}$. Therefore, an optimal algorithm for the problem is:

Algorithm 9

```
Compute OPT($G_k$) for $k = 1, \dots, n$.
Solve the following LP:
$$\begin{array}{lll} \max & \alpha & \\ \text{s.t.} & \sum_{e\in G_k} x_e \geq \mathrm{OPT}(G_k)\alpha & \forall k = 1,\dots,|V| \\ & \sum_{e\in\delta(v)} x_e \leq 1 & \forall v \in V \\ & x_e \geq 0 & \forall e \in E \end{array}$$
Decompose the solution $x$ of the previous LP as a convex combination of matchings $x = \sum \lambda_i \chi(M_i)$.
return the matching $M_i$ with probability $\lambda_i$.
```

The correctness and optimality follow from Lemma 4.2. The above algorithm is polynomial, since decomposing a fractional matching into a convex combination of integral matchings can be done in polynomial time [10, Theorem 6.5.11].

Despite knowing the algorithm is optimal, we do not know its competitivity. Because the Greedy algorithm achieves a competitive ratio of $\frac{1}{2}$, so does the optimal algorithm. Thanks to an unpublished paper of Wang and Wong [19, see also [20]] , we can slightly improve this to 0.526. Wang and Wong present an algorithm for on-line fractional matching in bipartite graphs with a competitivity of 0.526. This result holds in an adversarial model where vertices can arrive from either side of the graph. This model is stronger than our offline model, due to Lemma 4.2. We use this fact in the following proof.

Corollary 4.3 *The optimal algorithm achieves a competitivity of 0.526.*

Proof. We can simulate the vertices arriving on-line and use Wang and Wong's algorithm to find a fractional matching $(x_e)_{e\in E}$. This fractional matching can then be rounded as in Lemma 4.2, so this algorithm achieves a competitivity of 0.526 as well. The competitive ratio of the optimal algorithm is greater or equal than the competitive ratio of any other, which completes the proof. □

4.3 Upper Bound

We are interested in finding a tight competitivity bound for the optimal algorithm. We do not believe 0.526 is a tight bound. In this section we present an upper bound of $\frac{\sqrt{5}+9}{19}$ on the competitive ratio of the optimal algorithm for the offline model. For this, we define an input and bound the solution of the LP defined previously. Before that, we prove two lemmas which give us properties one can assume about a worst-case input.

Lemma 4.4 *For the purpose of analyzing the competitive ratio of an algorithm for the offline model, one can assume without loss of generality that the input for this problem is such that $OPT(G_k) = \frac{k}{2}$ for even k.*

Proof. Let $\pi = v_1, v_2, \ldots, v_n$ be the permutation in the input and suppose π does not satisfy the lemma. Let k be the first even number such that $\mathrm{OPT}(G_k) < \frac{k}{2}$. Then necessarily $\mathrm{OPT}(G_k) = \frac{k}{2} - 1$. We proceed inductively on k. Define i as the first index when $\mathrm{OPT}(G_i) = \frac{k}{2}$. Let M_k be an optimal matching in G_k. Since $\mathrm{OPT}(G_k) < \mathrm{OPT}(G_i)$, there exists an augmenting path in G_i for M_k. Let u and v be the endpoints of this augmenting path. Change π by inserting u and v right after v_{k-2}. If there exists a fractional matching x which achieves a competitivity of α on the new permutation, then x also achieves it on the old permutation as well, so this only makes the input harder.

After repeating this enough times, one is left with a permutation $\pi' = v'_1, v'_2, \ldots, v'_n$ with $\mathrm{OPT}(G_k) = \frac{k}{2}$ except for the last vertices where the size of the optimum matching does not increase. Let k' be te last even number such that $\mathrm{OPT}(G_{k'}) = \frac{k'}{2}$. Then, for $k > k'$ the inequality $\mathbb{E}[|M \cap G_k|] \geq \alpha\mathrm{OPT}(G_k) = \mathrm{OPT}(G'_k)$ is already implied by the inequality $\mathbb{E}[|M \cap G_{k'}|] \geq \alpha\mathrm{OPT}(G_{k'}) = \frac{k'}{2}$, so the algorithm can ignore the vertices after $v_{k'}$. □

With the previous lemma, the following algorithm has the same competitivity as the optimal algorithm for the offline model.

Algorithm 10

```
Change the permutation of the vertices as in Lemma 4.4.
Solve the following LP:
```

$$\begin{array}{lll} \max & \alpha & \\ \text{s.t.} & \sum_{e \in G_k} x_e \geq \frac{k}{2}\alpha & \forall k = 2, 4, \dots, n \\ & \sum_{e \in \delta(v)} x_e \leq 1 & \forall v \in V \\ & x_e \geq 0 & \forall e \in E \end{array}$$

```
Decompose the solution $x$ of the previous LP as a convex combination of matchings $x = \sum \lambda_i \chi(M_i)$.
return the matching $M_i$ with probability $\lambda_i$.
```

Since finding an augmenting path for a matching can be done in polynomial time, changing the permutation as in Lemma 4.4 can be done in polynomial time, so the above algorithm is still polynomial. While this algorithm has the same competitive ratio as the optimal algorithm presented before, it will usually give worse solutions. This is because Lemma 4.4 changes the input for a harder input, by changing and ignoring some of it. The algorithm only helps for theoretical analysis.

The next lemma tells us that we can take a worst-case input graph which is somewhat sparse. In fact, the worst example we know so far, which we will present later, is a tree.

Lemma 4.5 *Let e be an edge of the graph and $(M_i)_{i=1}^n$ matchings such that $M_i \subseteq E(G_i)$ and $OPT(G_i) = |M_i|$. For the purpose of analyzing the competitive ratio of an algorithm for the offline model, one can assume without loss of generality that there exists an i such that $e \in M_i$.*

Proof. If e is not in any M_i, remove e from the graph. □

The previous lemmas tell us something about the structure of the worst case input for this problem, which inspired the following upper bound for the hardness in the offline model.

Theorem 4.6 *No algorithm for the offline model achieves a competitivity greater than $\frac{\sqrt{5}+9}{19} \approx 0.591$.*

Proof. We will inductively build a family of inputs $(I_k)_{k \in \mathbb{N}}$. In general, the graph in I_k will have $4k$ vertices. Figure **??** shows I_1, I_2 and I_3, with vertices labelled by their arrival order.

In general, for $k \geq 3$, we build I_{k+1} from I_k by adding 4 new vertices. This is specified in Figure 4.6, with vertices again labelled by the order in which they arrive:

An easy way to understand this input is noting that each graph has a perfect matching. We then add four more edges which generate two alternating paths of size three. So this

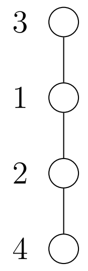

Figure 4.3: I_1.

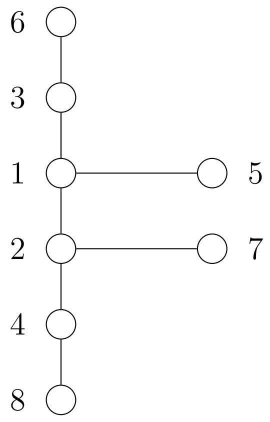

Figure 4.4: I_2.

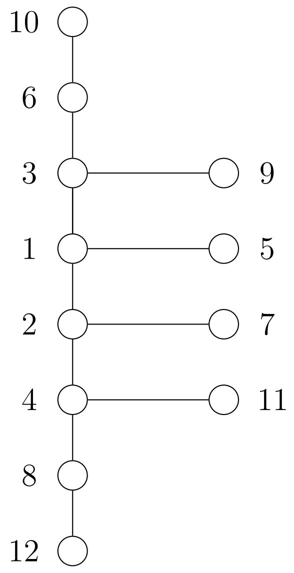

Figure 4.5: I_3.

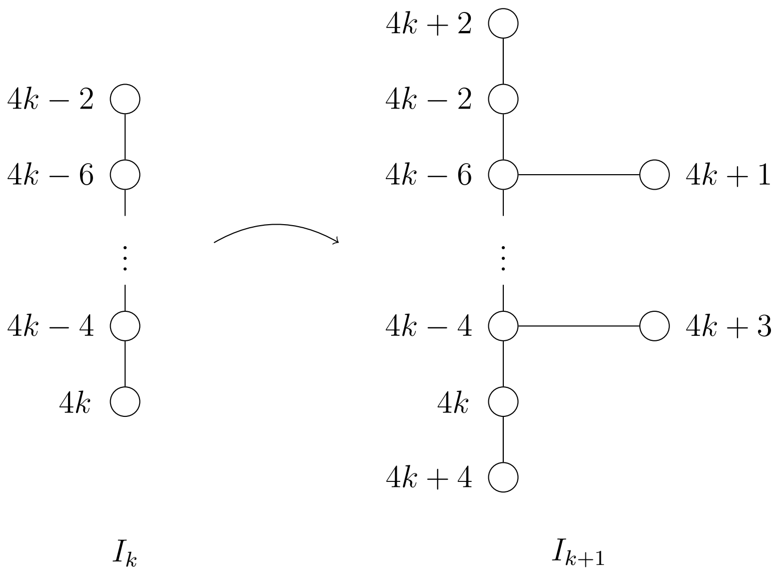

Figure 4.6: $I_k \to I_{k+1}$.

graph and vertex ordering satisfies that, for even i, $\mathrm{OPT}(G_i) = \frac{i}{2}$, as in Lemma 4.4.

Fix $k \in \mathbb{N}$. Let $v_1, v_2, \ldots, v_{4k}$ be the vertices in their arrival order. To upper bound the solution of the LP of the optimal algorithm for I_k, we will find a feasible solution of the dual LP. For a vertex v_i, define $t_{v_i} = \lfloor \frac{i}{2} \rfloor$. If we imagine the vertices arriving in pairs such that every time a pair arrives the optimal matching increases by one, t_{v_i} is exactly the time of the pair in which vertex v_i arrives. After changing variables and normalizing, the dual LP is equivalent to

$$\begin{array}{lll} \min & \sum_{v\in V} y_v & \\ \text{s.t.} & y_{v_i} + y_{v_j} \geq w_{\max\{t_{v_i}, t_{v_j}\}} & \forall e = v_iv_j \in E \\ & \sum_{i=1}^{2k} w_i \geq 1 & \\ & w_1 \geq w_2 \geq \ldots \geq w_{2k} \geq 0 & \\ & y_v \geq 0 & \forall v \in V. \end{array}$$

Figure 4.7 should give a good mental picture of this problem. Vertices are labelled by their arrival order and an edge v_iv_j is labelled with $w_{\max\{t_{v_i}, t_{v_j}\}}$.

Note that any optimal solution of this LP satisfies $\sum_{i=1}^{2k} w_i = 1$, because otherwise we can divide all variables by $\sum_{i=1}^{2k} w_i$ and obtain a smaller solution. With that in mind, we first propose a feasible solution which we will normalize later. Let f_i be the ith Fibonacci number with the usual convention $f_0 = 0$ and $f_1 = 1$ and take

$$\begin{array}{llll} y_{v_{4(k-i)}} & = y_{v_{4(k-i)-2}} & = f_i & \forall i = 0, \ldots, k-2 \\ y_{v_4} & = y_{v_3} & = f_{k-1} & \\ y_{v_1} & = y_{v_2} & = f_k & \\ y_{v_i} & & = 0 & \text{otherwise.} \end{array}$$

And

$$\begin{array}{llll} w_{2(k-i)} & = w_{2(k-i)-1} & = f_{i+2} & \forall i = 0, \ldots, 2k-2 \\ w_2 & & = f_{k+1} & \\ w_1 & & = 2f_k & \end{array}$$

We show these values on the graph in Figure 4.8.

Let us see if this is feasible. The variables w_i are decreasing and their sum is greater than 1, because $w_{2k} = 1$. Checking the edge restrictions is easy, using that $f_i + f_{i+1} = f_{i+2}$ and Figure 4.8, one sees that all edge restrictions are satisfied with equality.

Using that $\sum_{i=0}^{n} f_i = f_{n+2} - 1$, which is a classical induction exercise, we can calculate $\sum_{v\in V} y_v$:

$$\sum_{v\in V} y_v = 2\sum_{i=0}^{k} f_i = 2f_{k+2} - 2.$$

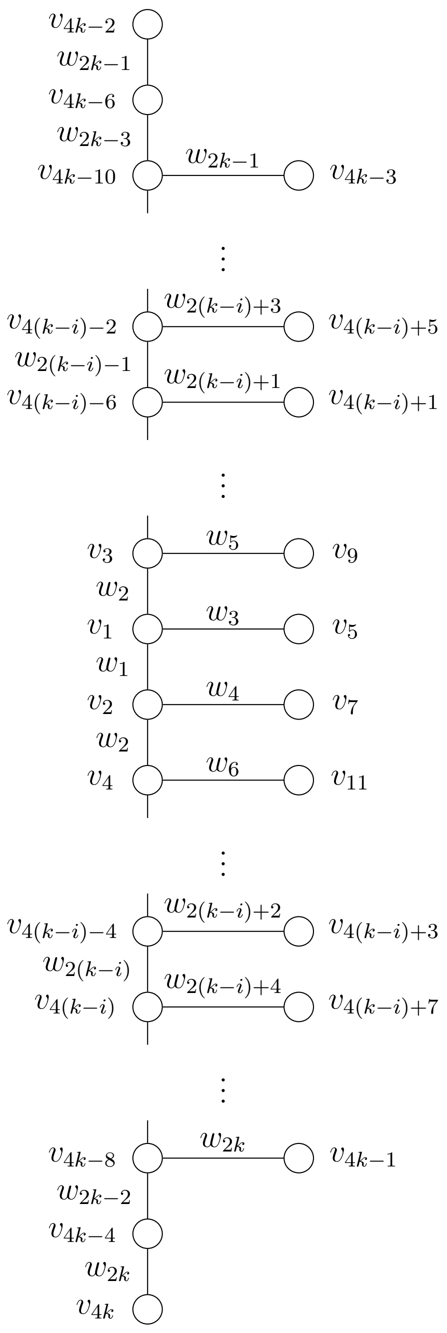

Figure 4.7: A visualization of the variables of the dual LP.

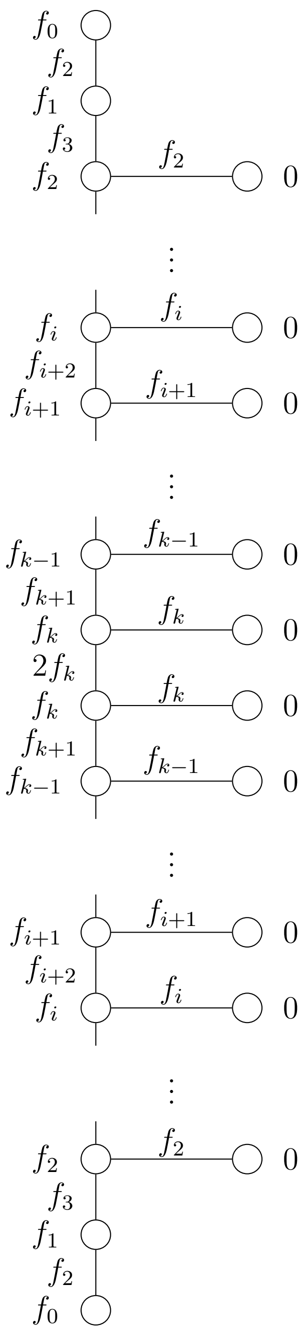

Figure 4.8: Feasible solution for the dual LP.

Next we normalize this feasible solution by defining $S = \sum_{i=1}^{2k} w_i$ and

$$\hat{y}_v = \frac{y_v}{S}.$$
$$\hat{w}_i = \frac{w_i}{S}.$$

The pair $(\hat{y}, \hat{w})$ is still feasible, as commented above. Let us calculate S:

$$S = \sum_{i=1}^{2k} w_i = 2f_k + f_{k+1} + \sum_{i=2}^{k} f_i = f_k + f_{k+2} + 2(f_{k+2} - 2) = 3f_{k+2} + f_k - 4.$$

Therefore,

$$\sum_{v \in V} \hat{y}_v = \frac{\sum_{v \in V} y_v}{S} = \frac{2f_{k+2} - 2}{3f_{k+2} + f_k - 4}.$$

We can calculate the limit using that $f_k = \frac{\phi^k - (-\phi)^{-k}}{\sqrt{5}}$, where ϕ is the golden ratio:

$$\lim_{n \to \infty} \sum_{v \in V} \hat{y}_v = \lim_{k \to \infty} \frac{2f_{k+2} - 2}{3f_{k+2} + f_k - 4} = \frac{\phi^2}{\phi^2 + \phi + 1} = \frac{\sqrt{5} + 9}{19}.$$

Therefore the competitivity of the optimal algorithm can not be greater than $\frac{\sqrt{5}+9}{19}$. $\square$

Since the on-line fractional matching model studied by Wang and Wong is stronger than the offline model studied by us, we get the following corollaries.

Corollary 4.7 *No algorithm achieves a competitivity greater than* $\frac{\sqrt{5}+9}{19} \approx 0.591$ *in the on-line fractional matching model.*

The adversarial model studied by us is also stronger (see Corollary 1.26), so this upper bound carries over to the adversarial model.

Corollary 4.8 *No algorithm achieves a competitivity greater than* $\frac{\sqrt{5}+9}{19} \approx 0.591$ *in the adversarial model.*

Previously the best upper bound known for both the adversarial model and the on-line fractional matching model was 0.6252 [19]. There still is a big gap between 0.526 and 0.591. Improving these bounds and hopefully determining tight bounds is left as an open problem.

Conclusion

Most algorithms studied in the framework of online matching fall in the category of what we defined as *local* algorithms. Discarding this family of algorithms for the adversarial model means that we have to do something different to beat the $\frac{1}{2}$-barrier. From the proof itself we can gather that no algorithm should actually perform too well on an instance. That is, for any algorithm $\frac{\mathbb{E}[|M|]}{\mathrm{OPT}(I)}$ should be bounded away from 1. This suggests that the probability of matching an arrived vertex should depend on $\mathbb{E}(|M|)$ and $\mathrm{OPT}(I)$ and that an algorithm might have to keep track of these variables.

We had two main motivation for studying the random order model. One was its relationship with the adversarial model. In that regard, we find that the random order model is substantially easier than even the offline model. We find that the Shuffle algorithm is at least 0.696-competitive in this model, while our best upper bound for the adversarial model is about 0.591. The proof links the Shuffle and both versions of the Ranking algorithms: the one used in general graphs and the one used in online bipartite matching. This gives a new insight to the Ranking algorithm when applied to bipartite graphs: The matching output only depends on the order amongst the vertices of each side. Our other motivation was that we consider the unknown distribution model a good model for applications. Our results show that the Shuffle algorithm is at least 0.696-competitive in the unknown distribution model as well.

We beat the $\frac{1}{2}$-barrier in the offline model, showing that the optimal algorithm is at least 0.526-competitive. The hardness of the offline model shows how much power the adversary has now: Even if we have access beforehand to the adversary's moves, our competitivity is bounded by at least 0.591. In other words, the hardness does not come from the lack of information of the graph and its arrival sequence, but rather from the fact that we must be competitive at every step. This is the best upper bound for the adversarial model at the time of writing this: The previous best upper bound was approximately 0.625 [19].

The main open problem is if an algorithm exists with competitivity greater than $\frac{1}{2}$ for the adversarial model. Algorithms studied so far in this subject have been local algorithms, inspired by the Ranking and Greedy algorithms. But this paradigm needs to be broken if we want to design an algorithm with competitivity greater than $\frac{1}{2}$ for this model. It would be extremely interesting if no algorithm with competitivity greater than $\frac{1}{2}$ exists - especially because there exists one for the fractional matching model. If this happens to be the case, it would be interesting to pinpoint why one can be done while the other one cannot and apply this to other problems. It is also still open if the dual problem, online vertex cover, admits an algorithm with competitivity smaller than 2 if the algorithm does not know to which side of the bipartite graph each arriving vertex belongs to.

In the random order model, we believe there is much room for improvement. Sampling techniques could be used so that the algorithm could learn about the general structure of the graph and apply this information to future arrivals. This also holds for the random order model with one fixed side. The random order model studied by us should be weaker, yet the best algorithm known for both models has the same competitive ratio on both. So our result suggests there is even more improvement available for the random order model with one fixed side.

We conjecture that the competitive ratio of the Shuffle algorithm is greater than $\frac{1}{2}$ for general graphs. So far, no algorithm with competitive ratio greater than $\frac{1}{2}$ exists for the random order model in general graphs, so proving this conjecture would effectively prove that one can break the $\frac{1}{2}$-barrier in the random order model as well.

A very interesting problem is left open for the offline model: What is the tight competitive ratio of the optimal algorithm for the offline model? We only know it lies somewhere in the interval $[0.526, 0.591]$. A good interpretation of the dual problem used in the proof of Theorem 4.6 might help, as it could help design a primal-dual algorithm.

Further models which could be studied are the known distribution model, also called the stochastic model, which has already been studied in the case where one side is fixed [7].

Bibliography

[1] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1253–1264. SIAM, 2011.

[2] Jonathan Aronson, Martin E. Dyer, Alan M. Frieze, and Stephen Suen. Randomized greedy matching II. *Random Struct. Algorithms*, 6(1):55–74, 1995.

[3] Bahman Bahmani and Michael Kapralov. Improved bounds for online stochastic matching. In Mark de Berg and Ulrich Meyer, editors, *Algorithms - ESA 2010, 18th Annual European Symposium, Liverpool, UK, September 6-8, 2010. Proceedings, Part I*, pages 170–181. Springer, 2010.

[4] Benjamin E. Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *SIGACT News*, 39(1):80–87, 2008.

[5] T.-H. Hubert Chan, Fei Chen, Xiaowei Wu, and Zhichao Zhao. Ranking on arbitrary graphs: Rematch via continuous LP with monotone and boundary condition constraints. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1112–1122. SIAM, 2014.

[6] Nikhil R. Devanur, Kamal Jain, and Robert D. Kleinberg. Randomized primal-dual analysis of RANKING for online bipartite matching. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 101–107. SIAM, 2013.

[7] Jon Feldman, Aranyak Mehta, Vahab S. Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 117–126. IEEE Computer Society, 2009.

[8] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In Shang-Hua Teng, editor, *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco,*

California, USA, January 20-22, 2008, pages 982–991. SIAM, 2008.

[9] Gagan Goel and Pushkar Tripathi. Matching with our eyes closed. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 718–727. IEEE Computer Society, 2012.

[10] Martin Grötschel, Lászlo Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer-Verlag, New York, 1988.

[11] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 587–596. ACM, 2011.

[12] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, STOC 1990, Baltimore, Maryland, USA,May 13-17 1990,*, pages 352–358. ACM, 1990.

[13] Bernhard Korte and Dirk Hausmann. An analysis of the greedy algorithm for independence systems. In *Annals of Discrete Mathematics*, volume 2, pages 65–74, 1978.

[14] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms.* Springer Publishing Company, Incorporated, 4th edition, 2007.

[15] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing LPs. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 597–606. ACM, 2011.

[16] Vahideh H. Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1285–1294. SIAM, 2011.

[17] Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5), 2007.

[18] Rad Niazadeh and Robert D. Kleinberg. A unified approach to online allocation algorithms via randomized dual fitting. *CoRR*, abs/1308.5444, 2013.

[19] Yajun Wang and Sam Chiu-wai Wong. Online vertex cover and matching: Beating the greedy algorithm. *CoRR*, abs/1305.1694, 2013.

[20] Sam Chiu-wai Wong. Competitive algorithms for online matching and vertex cover problems. Master's thesis, Massachusetts Institute of Technology, 2013.