Online Matching in Regular Bipartite Graphs

Lali Barrière[1], Janosch Fuchs[2], Xavier Muñoz[1], Walter Unger[2]
[1] Departament de Matemàtiques.UPC
[2] Lehrstuhl für Informatik I. RWTH-Aachen

August 10, 2018

Abstract

In an online problem, the input is revealed one piece at a time. In every time step, the online algorithm has to produce a part of the output, based on the partial knowledge of the input. Such decisions are irrevocable, and thus online algorithms usually lead to nonoptimal solutions. The impact of the partial knowledge depends strongly on the problem. If the algorithm is allowed to read binary information about the future, the amount of bits read that allow the algorithm to solve the problem optimally is the so-called advice complexity. The quality of an online algorithm is measured by its competitive ratio, which compares its performance to that of an optimal offline algorithm.

In this paper we study online bipartite matchings focusing on the particular case of bipartite matchings in regular graphs. We give tight upper and lower bounds on the competitive ratio of the online deterministic bipartite matching problem. The competitive ratio turns out to be asymptotically equal to the known randomized competitive ratio. Afterwards, we present an upper and lower bound for the advice complexity of the online deterministic bipartite matching problem.

1 Introduction

In an online problem the input is revealed one piece at a time. In every time step, the online algorithm has to produce a part of the output, based on the partial knowledge of the input. Such decisions are irrevocable, and thus online algorithms usually lead to nonoptimal solutions. Online algorithms respond to situations in which decisions have to be made without having the information of future requests. It can also respond to the case of distributed systems in which servers may take decisions without knowing the requests that arrive to other servers. Some online problems are scheduling, paging, routing, financial management, and other optimization problems. See [1, 2] for more information about online algorithms.

The quality of an online algorithm is measured by its competitive ratio, which compares its performance to that of an optimal offline algorithm. Given a minimization problem, an online algorithm A is c-competitive if for every instance σ we have

$$C_A(\sigma) \leq c \cdot C_{OPT}(\sigma) + \alpha, \tag{1}$$

where C_A and C_{OPT} denote the cost or profit of the algorithm A, and of an optimal algorithm, respectively.

Conversely given a maximization problem, an online algorithm A is c-competitive if for every instance σ we have

$$c \cdot C_A(\sigma) + \alpha \geq C_{OPT}(\sigma). \tag{2}$$

The analysis is done in terms of an adversary (or worst case). It turns out that deterministic algorithms often confer too much power to the adversary, which is supposed to know our decisions in advance. For such cases, deterministic online algorithms are not competitive, and randomization or advice is considered. In an online randomized algorithm, part of the decisions are taken probabilistically, so that the adversary cannot know the choice of the algorithm in advance, and the impact of worst case instances is reduced. The competitive ratio of a randomized online algorithm is defined by taking expectations on the performance of the (probabilistic) algorithm (see [2] for more details.)

An online algorithm with advice can read advice bits before decision making. The advice bits were previously written by an oracle with full knowledge of the input. Thus, the algorithm communicates with an entity that is as powerful as the adversary. The number of bits read until the end is called the advice complexity of an online algorithm with advice.

1.1 Online Bipartite Matching

In the online bipartite matching problem we are given a bipartite graph, $G = (U \cup V, E)$, and we assume that it contains a perfect matching. The online algorithm processes the input as follows. The vertices in U are *known*; the vertices in V arrive *one at a time*, together with their *list of adjacent vertices*, and each must be matched (or not matched) exactly at the time it arrives. The goal is to obtain a matching as close as possible to the optimal one.

First results on this problem are due to the seminal paper by Karp, Vazirani, and Vazirani [3]. Since then, several variants of the problem have been studied, some of them motivated by the following applications: adwords [4], matching in metric space [5], weighted matching in metric spaces [6], matching in the real line [7].

An interesting related problem is that of maintaining a matching with a minimum number of switches [8]. In this case, the partial solution is improved step by step.

The competitive ratio of the online deterministic problem is 2 [3], as any algorithm that always matches a vertex when possible constructs a maximal matching. Also, given any deterministic algorithm, it is easy to construct an input that forces the algorithm to find a matching of the size of no more than half of the optimum.

It is also proven in [3] that choosing uniformly at random every time a vertex arrives does not provide a significant improvement. Indeed, the expected size of the matching is bounded by $\frac{n}{2} + O(\log n)$.

Several randomized algorithms for online bipartite matching have been proposed. The first known is an algorithm called RANKING [3]. It initially ranks the known vertices, so that each time a new vertex arrives it is matched to the highest ranked available vertex that is incident to it. The competitive ratio is proven to be $1 + \frac{1}{e-1}$, where e is Euler's number. In [9] the authors revisit the proof and give a simpler one.

Other works on randomized algorithms for this problem are [10] and [11]. In [10], the authors compare the adversarial model with the random arrivals model, showing that in the former one the competitive ratio is 1.437. In [11], the stochastic bipartite matching is discussed, showing a competitive ratio of at most 1.44. It is also shown that no online algorithm can produce a $(1+\varepsilon)$-approximation with ε arbitrarily small.

This paper deals with the online matching problem for regular bipartite graphs, i.e., the graph presented by the adversary is a regular bipartite graph with known degree k and in which both vertex sets have the same cardinality. In Section 2, the competitive ratio for online deterministic algorithms is determined. Section 3 is devoted to the advice complexity for optimality, i.e., determining the number of advice bits that is necessary for solving the online matching problem for regular bipartite graphs optimally. A preliminary version of the results in Section 2 were presented in a poster session at SIROCCO 2015. Some of the results in Section 2 are also presented without proofs in [12].

2 Online Matching in Regular Bipartite Graphs

Let $G = (U \cup V, E)$ be a k–regular bipartite graph, $|U|= |V|= n$. The setting of the problem is as follows:

- Vertices in U are *known* before the algorithm starts.
- The adversary shows vertices in V *one at a time* together with their *list of adjacent vertices*. In each time step the algorithm must decide whether an edge will be in the matching or not.

The goal is to obtain a matching as large as possible. Recall that all regular bipartite graphs have a perfect matching (see [13],chapter 2), and therefore, an optimum matching will contain n edges.

Definition 1 *At any time step, the* revealed degree *of a vertex $u \in U$ is the number of edges adjacent to u that have already been presented by the adversary.*

2.1 Lower Bound

In the following proposition we give a lower bound for the competitive ratio, $c(k)$, of the online matching problem in k-regular bipartite graphs. The proof is based on the construction of an adversary, such that no algorithm can output a matching with more than $\frac{1}{c(k)} \cdot n$ edges.

Proposition 2.1 *The competitive ratio for any deterministic online algorithm for the matching problem for k-regular bipartite graphs, $c(k)$, is at least*

$$\frac{1}{1-\left(\frac{k-1}{k}\right)^k}. \tag{3}$$

Proof. Let us define an adversary presenting a k-regular bipartite graph with sets U and V, with k^k vertices each. The algorithm runs in a sequence of k iterations, each one divided in two phases, as follows.

We say that a vertex is ACTIVE if its revealed degree is smaller than k. The number of vertices that are ACTIVE at the beginning of iteration i is denoted by n_i. Let us also define $d_i = k - i + 1$. At the beginning of iteration 1, all the vertices in U are declared ACTIVE, $n_1 = n$, and $d_1 = k$. Notice that every active vertex has revealed degree $0 = k - d_1$.

Let us describe **Phase 1** and **Phase 2** of iteration i.

Phase 1: Matching phase. The adversary presents vertices in the set V in such a way that at the end of this phase every ACTIVE vertex in U has appeared exactly once. Therefore, every ACTIVE vertex is adjacent to exactly one of the vertices of V that have been presented in this phase. At the end of this phase any algorithm will have chosen $\frac{n_i}{k}$ edges in the matching. We will label the vertices in U to which the edges in the matching are incident as MATCHED.

Phase 2: Non-matching phase. After phase 1 we can label the ACTIVE vertices as $u_{\ell,j}$, with $1 \leq \ell \leq \frac{n_i}{k}$ and $1 \leq j \leq k$, in such a way that for every ℓ, $u_{\ell,1}$ is the one that became MATCHED.

The adversary now presents vertices that are adjacent to vertices MATCHED in phase 1, as long as possible: the first $d_i - 1$ vertices are all adjacent to the first k MATCHED vertices, $u_{1,1}, \ldots, u_{k,1}$; the next $d_i - 1$ vertices are all adjacent to the next k MATCHED vertices, $u_{k+1,1}, \ldots, u_{2k,1}$; and so on, until the $\frac{n_i}{k}$ MATCHED vertices have a revealed degree of k. Or in other words, for every MATCHED vertex in U, the adversary presents $d_i - 1$ adjacent vertices. The total number of vertices presented in this phase is $(d_i - 1)\frac{n_i}{k^2}$ and the algorithm has not been able to choose any edge for the matching.

At the beginning of iteration i, ACTIVE vertices are the vertices of U that are not MATCHED at the end of iteration $i-1$, and every ACTIVE vertex has been presented only once in the previous iteration. Therefore, the parameters n_i and d_i satisfy:

- $n_i = n_{i-1} - \dfrac{n_{i-1}}{k}$.
- $d_i = k - i + 1 = d_{i-1} - 1$ is exactly the number of edges (in the final graph) incident to an ACTIVE vertex that are not yet presented.

Notice that, at the beginning of iteration k, all the MATCHED vertices have revealed degree k. Also $n_k = k(k-1)^{k-1}$ and $d_k = 1$. After Phase 1 of iteration k, all vertices have revealed degree k. Thus, there are not ACTIVE vertices left, and the algorithm stops without performing Phase 2.

Figure 1 shows the 3 iterations of the algorithm with this adversary in the case $k = 3$.

With this input sequence, since $\frac{n_i}{k}$ vertices become matched in iteration i, we have: $n_1 = n$ and $n_i = \frac{n_{i-1}}{k}(k-1)$, $2 \leq i \leq k$. The the number of vertices matched in iteration i is

$$\frac{n_i}{k} = k^{k-i}(k-1)^{i-1}, \quad 1 \leq i \leq k.$$

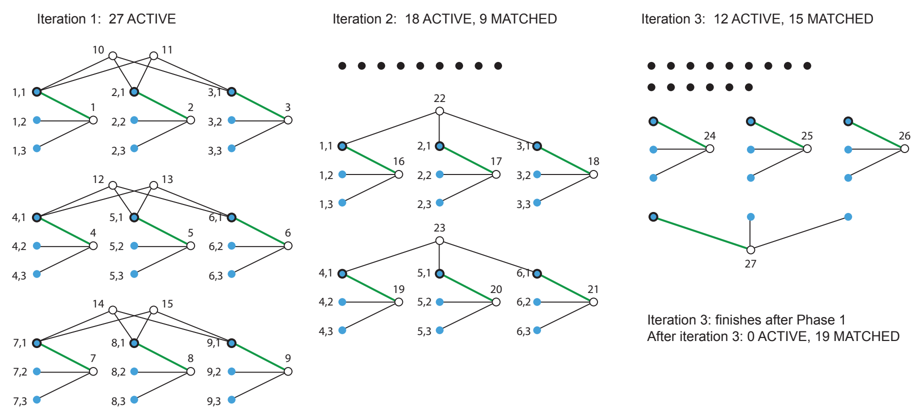
Iteration 1: 27 ACTIVE
Iteration 2: 18 ACTIVE, 9 MATCHED
Iteration 3: 12 ACTIVE, 15 MATCHED
Iteration 3: finishes after Phase 1
After iteration 3: 0 ACTIVE, 19 MATCHED

Figure 1: The adversary in the case of $k = 3$, $n = 27$. The blue vertices are the vertices of U, outlined in black if they become MATCHED. Their labels are set after phase 1 of their corresponding iteration. The vertices of V, arriving one step at a time, are white and numbered in arrival order. The green edges are the edges chosen by the algorithm. In each iteration, the vertices of V presented in phase 2 are those adjacent to already MATCHED vertices.

Hence, the total number m of matched vertices obtained by any algorithm is given by:

$$\begin{aligned} m &\leq k^{k-1} + k^{k-2} \cdot (k-1) + k^{k-3} \cdot (k-1)^2 + \ldots + k \cdot (k-1)^{k-2} + (k-1)^{k-1} \\ &= k^k \left(1 - \left(\frac{k-1}{k}\right)^k\right). \end{aligned} \tag{4}$$

Since any k-regular bipartite graph has a perfect matching, the bound holds. □

2.2 Upper Bound for the Competitive Ratio

Let us consider the following online deterministic algorithm.

Algorithm MAXIMUM REVEALED DEGREE. Let us sort the vertices in U with any ordering. In any time step, the adversary presents a new vertex $v \in V$ with adjacency list $L = \{u_0, \ldots u_{k-1}\} \subset U$. The algorithm will (or will not) match the vertex v to a vertex in L according to the following rules:

1. If all vertices in L have been matched in some previous time step, then the algorithm cannot choose any vertex.
2. Otherwise, the algorithm takes a vertex among the set of non-matched vertices in L with maximum revealed degree, i.e., vertices that have appeared the most in adjacency lists of the vertices of V presented so far.
3. If the previous rule leads to ambiguity, the algorithm takes the smallest tied vertex according to the initially decided ordering.

Proposition 2.2 *For all values of k, the competitive ratio of the algorithm MAXIMUM REVEALED DEGREE satisfies*

$$c_{MRD}(k) \le \frac{1}{1 - \left(\frac{k-1}{k}\right)^k}. \tag{5}$$

Proof. Let us consider an adversary presenting the vertices of V to our algorithm. We can assume w.l.o.g. that the vertices of V that become matched at the end of the algorithm are presented before the vertices that remain unmatched after the algorithm stops.

After the last matched vertex is presented, we are left with a set $M \subset V$ of matched vertices and every vertex in U is either matched or has revealed degree k.

For $1 \le i \le k$, let us define N_i, ℓ_i, and a_i as follows:

- Let N_i be the set of vertices in U matched exactly when the i-th edge adjacent to it appears. Let $n_i = |N_i|$.
- For every $u \in N_i$, let $e_u = \{u, v_u\}$ be its i-th adjacent edge (the one selected by the algorithm). Let W_u be the set of vertices that are adjacent to v_u and are already matched at the time vertex v_u is presented. Let

  $$\ell_i = \sum_{u \in N_i} |W_u|$$

  Notice that ℓ_i is the number of already matched vertices in U which are used to produce new vertices in N_i. Notice also that each vertex is counted as many times as it is used.
- Let a_i be the number of vertices that could have been used to produce a new vertex in N_i, but are not. Notice that $a_k = 0$, since $a_k \ne 0$ would imply that other vertices could be matched, increasing n_k.

Let us recall that because of the behavior of our algorithm, to produce a vertex $u \in N_i$ the only vertices that can be used are those with revealed degree at most i (taking into account the step in which u is matched), and those that have been matched in a previous time step and have revealed degree smaller than k.

Let us define $a_0 = 0$. We observe that:

$$n_i = \frac{1}{k}(n - \sum_{j=1}^{i-1} n_j + \ell_i - a_i + a_{i-1}), \quad 1 \le i \le k \tag{6}$$

Notice that a_{i-1} has to be added to the computation of n_i. The reason is that if a vertex could be used to produce a vertex in N_{i-1} but is not used, it can be used once more to produce a vertex in N_i.

In order to simplify the notation let us define

$$s_i = \frac{\ell_i - a_i + a_{i-1}}{k}, \quad 1 \le i \le k. \tag{7}$$

Next we prove by induction that the expression (6) can be rewritten as

$$n_i = \frac{n}{k}\left(\frac{k-1}{k}\right)^{i-1} + s_i - \frac{1}{k}\sum_{j=1}^{i-1}\left(\frac{k-1}{k}\right)^{i-j-1} s_j. \tag{8}$$

Since $n_1 = \frac{1}{k}(n + \ell_1 - a_1 + a_0) = \frac{n}{k} + s_1$, the case $i = 1$ trivially holds.

Let us now assume that the equation (8) holds for $i < k$. Then, from (6) we have:

$$n_{i+1} - n_i = -\frac{n_i}{k} + s_{i+1} - s_i$$

which is equivalent to

$$n_{i+1} = \frac{k-1}{k} n_i + s_{i+1} - s_i$$

The equation (8) gives us

$$n_{i+1} = \frac{k-1}{k}\left(\frac{n}{k}\left(\frac{k-1}{k}\right)^{i-1} + s_i - \frac{1}{k}\sum_{j=1}^{i-1}\left(\frac{k-1}{k}\right)^{i-j-1} s_j\right) + s_{i+1} - s_i =$$

$$= \frac{n}{k}\left(\frac{k-1}{k}\right)^{i} + s_{i+1} - \frac{1}{k}\sum_{j=1}^{i-1}\left(\frac{k-1}{k}\right)^{i-j-1} s_j - \frac{1}{k}s_i =$$

$$= \frac{n}{k}\left(\frac{k-1}{k}\right)^{i} + s_{i+1} - \frac{1}{k}\sum_{j=1}^{i}\left(\frac{k-1}{k}\right)^{i-j} s_j$$

This completes the induction proof of the validity of Equation (8).

Taking into account that the number of matched vertices at the end of the algorithm is $|M| = n_1 + n_2 + \ldots + n_k$, by manipulating the previous expressions we have

$$|M| = n\left(1 - \left(\frac{k-1}{k}\right)^k\right) + s_1\left(\frac{k-1}{k}\right)^{k-1} + s_2\left(\frac{k-1}{k}\right)^{k-2} + \ldots + s_k, \quad (9)$$

which implies

$$|M| \geq n\left(1 - \left(\frac{k-1}{k}\right)^k\right) + \left(\frac{k-1}{k}\right)^{k-1}\sum_{i=1}^{k} s_i. \quad (10)$$

From the definition of s_i we get

$$k \cdot \sum_{i=1}^{k} s_i = \sum_{i=1}^{k}(\ell_i - a_i + a_{i-1}) = \sum_{i=1}^{k}\ell_i - \sum_{i=1}^{k}a_i + \sum_{i=0}^{k-1}a_i = \sum_{i=1}^{k}\ell_i - a_k + a_0. \quad (11)$$

By taking into account that $a_k = a_0 = 0$, we can assure that $\sum_{i=1}^{k} s_i \geq 0$. We can therefore conclude that

$$|M| \geq n\left(1 - \left(\frac{k-1}{k}\right)^k\right). \quad (12)$$

□

As a direct consequence of Proposition 2.1 and Proposition 2.2 we have the following.

Theorem 2.3 *The competitive ratio of the online bipartite regular matching is*

$$c(k) = \frac{1}{1 - \left(\frac{k-1}{k}\right)^k}. \tag{13}$$

3 Advice Complexity

The difficulty for online algorithms that do not have any information about the future is the lack of crucial information regarding the problem. The classical online scenario does not allow to analyze the amount of crucial information for a problem, which would help to classify the complexity of online problems in a different way.

Therefore, the online setting was extended with advice by Dobrev et al. [14]. Then, it was improved and refined by Emek et al. [15] and Böckenhauer et al. [16]. The algorithm can read advice bits from an infinite advice tape which were written by an oracle with the same knowledge as the adversary. The algorithm can read any amount of bits at any time during computation. The amount of bits that were read until the end of calculation is the advice complexity of the algorithm.

Sometimes, the tradeoff between advice bits and competitive ratio stands in focus of research. In the following, instead, we are interested in lower and upper bounds for the number of advice bits that are necessary to compute an optimal solution for the online bipartite matching problem.

The advice complexity for the matching problem in general bipartite graphs is studied in [17]. In this paper it is proven that the number of advice bits that are necessary and sufficient for an optimal matching in a general bipartite graph is $\lceil \log_2(n!) \rceil$. In this section we find upper and lower bounds for the number of advice bits required for an optimal matching in a k-regular bipartite graph.

3.1 The k-ary Decision Tree: an Upper Bound on the Advice Complexity

For any given algorithm, we construct the k-ary decision tree that simulates all possible advice strings:

- The first vertex presented by the adversary is the root of the tree.
- At each vertex, we generate k children, according to the k possible edges our algorithm can be advised to choose. We call this set of edges a *claw*.
- The adversary then presents a new vertex, possibly different for every vertex on the same level. Again, the k possible edge choices generate k children in the new level of the k-ary tree.
- After n levels, the algorithm stops with at most k^n different leaves.

Every path from the root to the leaves gives us a possible matching. Thus, each leaf represents and identifies a chain of decisions for the algorithm.

Proposition 3.1 *The number of advice bits needed to optimally solve online matching on regular bipartite graphs satisfies:*

$$\#bits \leq n \log k. \tag{14}$$

Proof. The number of leaves of the tree is k^n. Every leaf could be the encoding of a matching. We need $\log k^n = n \log k$ bits to encode the whole set of leaves.

Once the leaves can be identified, we know which choices lead us to an optimal matching.

Thus, $n \log k$ bits are sufficient to optimally solve the problem. □

3.2 Working with a Particular Adversary: Lower Bound on the Advice Complexity

We want to give a lower bound on the number of advice bits, showing that there exists an adversary such that, for any algorithm, the set of all possible advices needs this number of bits to be encoded.

Let us recall that we are interested in optimal solutions and therefore the advice should be such that a perfect matching is obtained, i.e., such that one edge is chosen at each round. The next Lemma shows a condition on the number of advice bits in order to ensure a perfect matching

Lemma 3.2 *Let the adversary present a k-regular bipartite graph. If the advice is such that the edges in the matching are not determined univocally in k different time steps, then the algorithm could end up with a non-perfect matching.*

Proof. Let us assume that there are k different choices for which the advice is not enough, i.e., the algorithm may choose a wrong edge/vertex (since there will be at least two edges with the same advice). If there are k wrong choices, the adversary may present a vertex incident to exactly those vertices that were chosen wrong, and therefore avoiding a perfect matching. □

Proposition 3.3 *The number of advice bits needed to optimally solve online matching on k-regular bipartite graphs satisfies:*

$$\#bits \geq \frac{n}{2} \log k - k. \tag{15}$$

Proof. Let us consider the class of adversaries presenting the vertices of a k-regular bipartite graph with $n = rk$ vertices in k phases. At each phase a set of r independent vertices are sequentially presented, i.e., no two different vertices have a common neighbor.

Due to Lemma 3.2, the difference between the advice required to guarantee a perfect matching and advice required to determine every choice univocally is less or equal than k, i.e., one bit per choice in k different choices.

Let us count the number of bits required to determine every choice:

In phase 1 it is clear that $\log k$ bits are required for each vertex presented. Therefore, $r \log k$ bits are necessary for the first phase.

Before starting phase i, $(i-1)r$ edges will have been selected, provided that the advice is producing a perfect matching. Therefore, there are $(i-1)r$ vertices that cannot be selected during phase i. Those vertices will be distributed among the neighbors of the different vertices presented in this phase. Let $a_{i,j}$ be the number of vertices adjacent to vertex j that are free to be chosen during phase i, i.e., those that are not matched during the previous rounds.

The number of required advice bits during phase i is given by

$$\sum_{j=1}^{r} \log a_{i,j} = \log \prod_{j=1}^{r} a_{i,j}, \tag{16}$$

subject to

$$\sum_{j=1}^{r} a_{i,j} = (k-i+1)r \quad , \quad a_{i,j} \in \{1,..,k\}. \tag{17}$$

The condition $a_{i,j} \geq 1$ is given by the fact that the advice will allow the algorithm to compute a perfect matching.

By using Lagrange multipliers, it is not difficult to prove that the distribution of values $a_{i,j}$ that minimizes the product $\prod_{j=1}^{r} a_{i,j}$ subject to $\sum_{j=1}^{r} a_{i,j} = (k-i+1)r$, $a_{i,j} \in \{1,..,k\}$ is obtained by maximizing the number of elements $a_{i,j}$ that are minimized, i.e., $a_{i,j} = 1$.

In phase i, the maximum number of presented vertices with $k-1$ neighbors that are already matched in previous rounds, (i.e., such that $a_{i,j} = 1$), is given by $\lfloor \frac{(i-1)r}{k-1} \rfloor$. For the remaining $\lceil \frac{(k-i)r}{k-1} \rceil$ presented vertices none of their neighbors is already matched, i.e., $a_{i,j} = k$.

Notice that, $\lfloor \frac{(i-1)r}{k-1} \rfloor + \lceil \frac{(k-i)r}{k-1} \rceil$ is either r or $r-1$, and therefore one of the presented vertices may have some of the neighbors already matched.

Therefore, the number of advice bits required during phase i is, at least,

$$\log \prod_{j=1}^{r} a_{i,j} \geq \log k^{\frac{(k-i)r}{k-1}}. \tag{18}$$

By adding up the advice bits in all phases, we have:

$$\sum_{i=1}^{k} \log k^{\frac{(k-i)r}{k-1}} = \frac{r \log k}{k-1} \sum_{i=1}^{k} (k-i) = \frac{n}{2} \log k. \tag{19}$$

Finally, according to Lemma 3.2, the number of required bits to guarantee a perfect matching satisfies

$$\#bits \geq \frac{n}{2} \log k - k. \tag{20}$$

□

4 Conclusions

We have studied two online models for the bipartite matching problem in regular graphs: First, the online deterministic setting, where complexity is measured by the competitive ratio, and second, the online deterministic setting with advice, where the advice complexity is defined as the number of advice bits that are necessary and sufficient to ensure an optimal matching.

We proved that the competitive ratio for the online deterministic bipartite matching problem for regular graphs is $c(k) = \frac{1}{1-\left(\frac{k-1}{k}\right)^k}$, where k is the degree.

Observe that, if k is large, the competitive ratio grows asymptotically to the competitive ratio given by randomized algorithms for general bipartite graphs in [3], i.e.,

$$\lim_{k\to\infty}\left(\frac{1}{1-\left(\frac{k-1}{k}\right)^k}\right)=\frac{e}{e-1}. \tag{21}$$

This result is a bit surprising and we wonder if there is an intrinsic explanation for this.

We also proved that the advice complexity for the online deterministic bipartite matching in regular graphs for optimality satisfies:

$$\frac{n}{2}\log k - k \le \#bits \le n\log k. \tag{22}$$

Acknowledgements

This research is supported in part by the *Ministerio de Ciencia e Innovción*, Spain, and the European Regional Development Fund under project MTM2014-60127-P, and the *Catalan Research Council* under project 2014SGR1147.

The authors would like to thank Rastislav Kralovic and Juraj Hromkovic for their useful discussions and comments.

References

[1] Allan Borodin and Ran El-Yaniv, *Online Computation and Competitive Analysis* (Cambridge University Press, 1998).

[2] Denis Komm, *An Introduction to Online Computation* (Springer, 2016).

[3] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani, An Optimal Algorithm for On-line Bipartite Matching, in *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 352–358, 1990.

[4] Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani, AdWords and generalized online matching, *Journal of the ACM*, **54**(5):1–19, 2007.

[5] Adam Meyerson, Akash Nanavati, and Laura Poplawski, Randomized online algorithms for minimum metric bipartite matching, in *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 954–959, 2006.

[6] Bala Kalyanasundaram and Kirk Pruhs, Online weighted matching, *Journal of Algorithms*, **14**(3):478–488, 1993.

[7] Bernhard Fuchs, Winfried Hochstättler, and Walter Kern, Online matching on a line, *Electronic Notes in Discrete Mathematics*, **13**:49–51, 2003.

[8] Kamalika Chaudhuri, Constantinos Daskalakis, Robert D. Kleinberg, and Henry Lin, Online bipartite perfect matching with augmentations, in *Proceedings of IEEE INFOCOM*, pp. 1044–1052, 2009.

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

[10] Mohammad Mahdian and Qiqi Yan, Online bipartite matching with random arrivals: An approach based on strongly factor-revealing LPs, in *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 597–605, 2011.

[11] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S. Muthukrishnan, Online stochastic matching: Beating $1 - 1/e$, in *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 117–126, 2009.

[12] Joseph Naor, David Wajc, Near-Optimum Online Ad Allocation for Targeted Advertising, in *Proceedings of the Sixteenth ACM Conference on Economics and Computation, EC'15*, pp.131–148, 2015.

[13] Reinhard Diestel, *Graph Theory, 3rd ed.* (Springer, 2005).

[14] Stefan Dobrev, Rastislav Královič, and Dana Pardubská, How much information about the future is needed?, in *SOFSEM 2008: Theory and Practice of Computer Science*, Lecture Notes in Computer Science, vol. 4910, pp. 247–258, 2008.

[15] Yuval Emek, Pierre Fraigniaud, Amos Korman, and Adi Rosén, Online computation with advice, in *Automata, Languages and Programming (ICALP 2009)*, Lecture Notes in Computer Science, vol. 5555, pp. 427–438, 2009.

[16] Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Královič, Richard Královič, and Tobias Mömke, On the advice complexity of online problems, *Algorithms and Computation (ISAAC 2009)*, Lecture Notes in Computer Science, vol. 5878, pp. 331–340, 2009.

[17] Shuichi Miyazaki, On the advice complexity of online bipartite matching and online stable marriage, *Information Processing Letters*, **114**(12):714–717, 2014.