Online Matching for Scheduling Problems

Marco Riedel*

Heinz Nixdorf Institute, University of Paderborn, Germany
E-mail: barcom@uni-paderborn.de

Abstract. In this work an alternative online variant of the matching problem in bipartite graphs is presented. It is motivated by a scheduling problem in an online environment. In such an environment, a task is unknown up to its disclosure. However, in that moment it is not necessary to take a decision on the service of that particular task. In reality, an online scheduler has to decide on how to use the current resources. Therefore, our problem is called *online request server matching* (ORSM). It differs substantially from the *online bipartite matching* problem of Karp *et al.* [KVV90]. Hence, the analysis of an optimal, deterministic online algorithm for the ORSM problem results in a smaller competitive ratio of 1.5.
We also introduce an extension to a weighted bipartite matching problem. A lower bound of $\frac{\sqrt{5}+1}{2} \approx 1.618$ and an upper bound of 2 is given for the competitive ratio.

1 Introduction

Motivation. The problem, which is investigated here, is motivated by online scheduling problems with deadlines. Such problems can be found in server systems for continuous data streams (e. g. video transmissions). Such a server consists of a set of memory modules (e. g. hard disks) to store a huge amount of data and to provide a high bandwidth. The latter is necessary to serve a large number of customers simultaneously. The memory modules are connected to a set of I/O-ports via a communication network inside the server system. Customers are connected to one of the I/O-ports in order to receive a demanded continuous data stream. It is clear that such a server is a real-time system and that the requests for data have hard deadlines. Data which violate their deadlines are not of interest anymore and can be omitted. Fortunately, the situation is relaxed a little bit because it seems to be acceptable when a small amount of the required data will not be delivered. A first question, which needs to be solved concerns the mapping of the data to the memory modules. To increase the usability of the system and to reduce the number of conflicts, the data or parts of it can be stored in more than one copy in different memory modules. Secondly, the access to memory modules and network resources has to be controlled. The difficulty of

* Supported by DFG-Graduiertenkolleg "Parallele Rechnernetzwerke in der Produktionstechnik", GRK 124/2-96. This work was partly supported by the EU ESPRIT Long Term Research Project 20244 (ALCOM-IT).

these problems is increased by the *online* nature of the environment. The continuous delivering of a data stream can be preempted by a customer or it can be restarted at a completely different time index. Additionally, customers can arrive to the system or leave it at any time. So, future demands of server resources are unknown. However, *scheduling* decisions (assign appropriate resources to the requests) have to be taken under this uncertainty.

It is not difficult to construct a formal model for such a server system including all dependencies. However, its analysis is a challenge. To start investigations we performed a process of abstractions and simplifications: A continuous data stream is divided into packages of roughly equal size. Under this assumption, a simple, discrete time model is used. Abstracting from additional restrictions that model a restricted communication network between memory modules and customers, the scheduling problem becomes a matching problem in a bipartite graph. The scheduler has to match requests for unit size data packages to appropriate server resources and time slots. However, these decisions does not need to take immediately when a request arrives. Instead, an online scheduler has to determine how to use the current resources.

In this work we start our studies with a very basic model for this problem and we call it *online request server matching*. It is highly idealized and therefore it is not able to represent the inherent structure of the initial problem. The ORSM model is a matching problem in a rather general bipartite graph in a new online variant. Nevertheless, the definition was selected in such a way, that an extension to more realistic models is possible. A development to a weighted matching problem is also discussed here and it is motivated by additional priorities of requests.

To investigate our online problems, we apply *competitive analysis*. Roughly speaking, it determines the worst case ratio (called *competitive ratio*) of the solution quality of an online algorithm and an optimal solution over all inputs. Nowadays, it is an established method and it counterbalances results which assume an input distribution. A comprehensive introduction to online problems and competitive analysis is the textbook by Borodin and El-Yaniv [BEY98].

Previous and Related Work. Over the last years a vast amount of publications and single results on online scheduling can be found. [Sga98] gives an encyclopedic survey. However, studies of scheduling models which allow deadlines are rare and cover very restricted cases only. None of them seems to be closely related to our model. The opposite is true for bipartite matching problems in online settings. The study of such problems was initiated by Karp, Vazirani and Vazirani in [KVV90]. Their model consists of a 'known' and an 'unknown' partition. Vertices of the unknown partition are revealed over time and an online algorithm can put at most one edge of a just revealed vertex in the online matching. The simple GREEDY algorithm is an optimal, 2-competitive, deterministic online algorithm for this problem. The key contribution of [KVV90] is the analysis of the optimal, randomized algorithm RANKING which is $\frac{e}{e-1} \approx 1.582$-competitive.

Various extensions and variants of this problem have been studied later. A comprehensive survey and further references can be found in [KP98].

An online matching problem in general graphs is the *roommates problem* [BR93]: Guests arrive by and by at a hotel for a conference. The hotel consists of double rooms only. Every guest has a list of acceptable roommates (it is assumed that these lists are symmetric; they represent the adjacency lists of vertices in an undirected graph). The manager must immediately assign a room to every guest and wants to minimize the number of occupied rooms. In [BR93] an optimal, deterministic, 1.5-competitive algorithm is shown. For a weighted extension of the problem a lower bound of 3, and a 4-competitive algorithm are given.

Results and Organization of the Material. In this work we investigate a different online variant of the bipartite matching problem. The precise description can be found in Sect. 2. Section 3 presents the results of the unweighted case. A lower bound and a matching upper bound are given. It includes a 1.5-competitive algorithm. The weighted extension of our model is studied in Sect. 4. It is shown, that the golden ratio ($\frac{\sqrt{5}+1}{2} \approx 1.618$) is a lower bound for the competitive ratio. A 2-competitive, deterministic online algorithm is presented as well as an outline of its analysis. Due to space limitations, the proof of the upper bound had to be omitted. A few concluding remarks in Sect. 5 complete this work.

2 The Model

A formal definition of the *online request server matching* problem (ORSM) and its weighted extension (wORSM) is now presented.

A bipartite graph $G := (R \cup S, E)$ represents the underlying structure of the problem. Both partitions R and S are totally ordered. We denote the vertices by $r_1, r_2, r_3, \ldots$ and by $s_1, s_2, s_3, \ldots$ with $r_i \in R$, $s_i \in S$, and the indices indicating the position within the order. We interpret this order as a discrete time model. The vertices of partition S represent a single resource called *server*. It is available for one unit each time step. Partition R is interpreted as a set of tasks. Such a task has a demand of one server unit to be completed. They are called *requests*, and every time step one of them might occur. An edge $\{r_i, s_j\}$ between a request vertex r_i and a server vertex s_j means that request r_i can be served in time step j. The set of edges $E \subset R \times S$ is constructed with the following restriction:

$$\{r_i, s_j\} \in E \;\Rightarrow\; i \leq j \;\; . \tag{1}$$

This means a request that occurs at time step i must not specify a possible service time in the past. Without this restriction the modelled scheduling problem does not make sense and no competitive online algorithm exists.

Now we have to specify how this model works online: When the system starts partition R is completely unknown[1]. In the beginning of a time step i the request

[1] When taking a close look this is not the truth. Every time step i a new vertex r_i is inserted but its set of incident edges is before unknown. For reasons of convenience the input process is interpreted in the introduced way.

r_i is revealed as input, i. e., vertex r_i and all of its edges are inserted into the previously known part of G. If no request appears, vertex r_i is isolated. After this input, an algorithm has to decide on how to use the server in the current time step i. It can add an edge incident to s_i to the online matching M. It is worth noting that due to restriction (1) *all* edges incident to s_i are known when this decision has to be taken. The online algorithm has the objective to maximize the cardinality of matching M, i. e. to serve as many requests as it can.

We want to emphasize the difference between the model in [KVV90] and ours. Karp *et al.* study an online version of the bipartite matching problem where the input of a time step is a vertex out of the 'unknown' partition and all of its incident edges. Then, an online algorithm has to select at most one of these *just revealed* edges for the online matching. Contrarily, the decision on a matching edge in the ORSM problem is taken by selecting one of the edges which are incident to a vertex of the *opposite* partition. Nevertheless, the ORSM problem can be interpreted as a special case of the online bipartite matching. Then, the server partition is the 'unknown' partition and whenever a decision has to be taken, all edges incident to that vertex are known. Furthermore, all additional information provided by the ORSM model is ignored. When the focus is shifted, the ORSM problem can also be recognized as a special case of the *roommates problem*. The precise relationship is shown in [Rie98].

So far, the graph G is unweighted. The objective changes when a weight function $w : E \to \mathbb{R}_+$ is added. In this case, a matching with maximal total weight has to be constructed. This version is called *online request server weighted matching* problem or in short wORSM problem.

In the discussions and proof of this work M_{ALG} is used in order to denote an online matching which is constructed by algorithm ALG, $|M|$ to denote the total weight (cardinality respectively) of matching M, and OPT denotes an optimal solution.

3 Analysis of the ORSM Problem

This section starts with a general lower bound for the competitive ratio of the ORSM problem for deterministic online algorithms. Thereafter, the optimal 1.5-competitive algorithm LMM is presented and analysed.

3.1 The Lower Bound

By applying the standard argument of an adversary strategy, we will show the following general lower bound:

Theorem 1. *Every deterministic online algorithm ALG for the ORSM problem has a competitive ratio of at least* 1.5.

Proof. The adversary strategy starts with the following input structure (see Fig. 1): $E = \{\{r_1, s_2\}, \{r_1, s_3\}, \{r_2, s_2\}, \{r_2, s_4\}\}$.

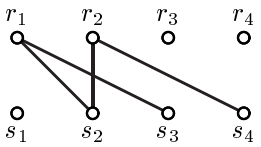


Fig. 1. Situation at time $t = 2$.

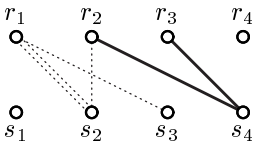


Fig. 2. $\{r_1, s_2\} \in M_{\mathsf{ALG}}$

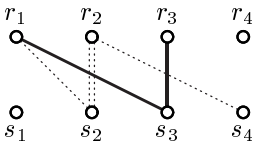


Fig. 3. $\{r_2, s_2\} \in M_{\mathsf{ALG}}$

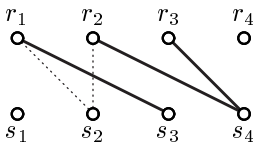


Fig. 4. $M_{\mathsf{ALG}} = \varnothing$

(the figures show the situations at time $t = 3$, according to different decisions of ALG and the reactions of the adversary; a double line is a matching edge; dotted lines are edges hich have been removed before $t = 3$)

An online algorithm ALG can react to this input at time $t = 2$ in three different ways:

Case 1 (Fig. 2): ALG puts edge $\{r_1, s_2\}$ to the online matching M_{ALG}. In the next step the adversary presents edge $\{r_3, s_4\}$. ALG cannot use the server vertex s_3. Therefore, $|M_{\mathsf{ALG}}| \leq 2$ whereas the optimal solution results in $|M_{\mathsf{OPT}}| = 3$.

Case 2 (Fig. 3): ALG puts edge $\{r_2, s_2\}$ to the online matching M_{ALG}. In the next step the adversary presents edge $\{r_3, s_3\}$. ALG cannot use s_4. Again $|M_{\mathsf{ALG}}| \leq 2$ and the maximum matching results in $|M_{\mathsf{OPT}}| = 3$.

Case 3 (Fig. 4): ALG decides not to match s_2. The adversary will present the input of Case 1 and $|M_{\mathsf{ALG}}| \leq 2$, $|M_{\mathsf{OPT}}| = 3$ holds. Alternatively, it is fairly obvious that ALG cannot take advantage of such a decision.

This strategy can be infinitely repeated every four time steps and this fact shows the ratio

$$\frac{|M_{\mathsf{OPT}}|}{|M_{\mathsf{ALG}}|} \geq \frac{3}{2} \ . \qquad \square$$

3.2 The Algorithm LMM

At a time step i the graph G representing the input of an online algorithm is known up to request vertex r_i. More precisely, the subgraph of G induced by the set $\{r_k |\ r_k \in R, 1 \leq k \leq i\} \cup S$ is known. Due to the irreversible decisions taken in former time steps all previous server vertices $s_1, \ldots, s_{i-1}$ and all hitherto matched request vertices cannot be rearranged anymore. It remains a vertex induced bipartite subgraph B_i of G with the vertex set V_i :
$V_i = \{r_k |\ r_k \in R, r_k \text{ not matched so far}, 1 \leq k \leq i\} \ \cup \ \{s_k |\ s_k \in S, i \leq k\}$.

Our online algorithm is called 'Local Maximum Matching' (LMM) because it constructs a maximum matching on every local subgraph B_i (denoted by $\mathcal{M}(B_i)$). This can be done by searching and processing 'augmenting paths', i. e. paths with unmatched end vertices and alternately sequent matching and non-matching edges (see e. g. [Edm65] or any comprehensive text book on algorithms for an explanation of how to do so). The exact function of LMM is:

1: **loop** {$\forall$ time steps i}
2: read input of time i and build up B_i
3: construct a maximum matching $\mathcal{M}(B_i)$ on B_i :
start with all matching edges of $\mathcal{M}(B_{i-1})$ which are edges in B_i ;
4: look for an augmenting path which starts at vertex r_i and perform the augmentation when found[2]
5: **if** s_i is matched in $\mathcal{M}(B_i)$ **then**
6: add the matching edge of s_i to the online matching M_{LMM}
7: **else if** s_i is not isolated in B_i **then** {all neighbours of s_i are matched in $\mathcal{M}(B_i)$}
8: add an arbitrary edge $\{s_i, r\}$ of B_i to the matching M_{LMM} and delete the matching edge of r in $\mathcal{M}(B_i)$
9: **end if**
10: **end loop**

Line 8 of this algorithm is essential and prefers the current server vertex s_i.

3.3 The Upper Bound

To analyse the performance of LMM three observations are needed:

1. After a request vertex r_i has been matched in B_i (in line 4 of LMM), it is in all following maximum matchings[3] up to the time step where its current matching edge is added to M_{LMM} (in line 6 or 8 of LMM).
2. If s_i is not isolated in B_i, then s_i is matched in M_{LMM} (lines 5 to 8 of LMM).
3. Let M be a matching such that no augmenting path has length less than $2\ell + 1$, then an maximum matching M_{OPT} holds $|M_{\mathsf{OPT}}| \leq \frac{\ell+1}{\ell}|M|$. (An augmenting path of length $2\ell + 1$ has ℓ matching edges in M and $\ell + 1$ in M_{OPT}.)

Theorem 2. *The deterministic online algorithm LMM is 1.5-competitive.*

Proof. It will be shown that no online matching M_{LMM} can be extended by augmenting paths of length one or three. Therefore, shortest augmenting paths for M_{LMM} must have a length of five. This fact combined with observation 3 completes the proof.

[2] Due to the maximum cardinality of $\mathcal{M}(B_{i-1})$, every augmenting path must have r_i at one end.

[3] See the copy process of line 3 and remember that augmentations change matching edges but do not remove vertices out of the matching.

Applying a complete distinction of cases, the non-existence of augmenting paths of length one and three in the online matching M_{LMM} is proven by contradiction.

Fig. 5. Structure of augmenting paths of length one and three.

Case 1: augmenting path of length one $\{s_i, r_a\} \notin M_{\mathsf{LMM}}$: Vertex r_a was never matched because r_a is not in M_{LMM} (reverse application of observation 1). So $\{s_i, r_a\}$ is in B_i and thus contradicts observation 2.

Case 2: augmenting path of length three and $i < j$: r_a was matched not until time j, which implies the edge $\{s_i, r_a\}$ was in B_i. This is a contradiction to observation 2.

Case 3: augmenting path of length three and $i > j$: At time j, request vertices r_a and r_b are not in M_{LMM} and so the whole path $\{\{s_i, r_a\}, \{r_a, s_j\}, \{s_j, r_b\}\}$ is in B_j. The case s_i is not in $\mathcal{M}(B_j)$ contradicts the optimality of $\mathcal{M}(B_j)$ because the path is an augmenting one (see Fig. 5). Therefore, at time j, s_i must be matched in $\mathcal{M}(B_j)$, i. e., there exists a request vertex r_c with $\{r_c, s_i\} \in \mathcal{M}(B_j)$. Later at time k ($j < k < i$), line 8 of LMM deletes the matching edge $\{r_c, s_i\}$ and adds $\{s_k, r_c\}$ to M_{LMM}.

Due to the definition of the ORSM problem, both edges $\{r_c, s_i\}$ and $\{r_c, s_k\}$ are known at time j. Now the above argument about s_i can be recursively applied to s_k and due to the finite structure of B_i, this fact contradicts the existence of an augmenting path of length three in M_{LMM}. □

4 The Weighted Model

Similar to Sect. 3, a lower bound for the wORSM problem is shown first. Then the algorithm wLMM is presented as well as an outline of its analysis.

4.1 A General Lower Bound

Let $\phi := \frac{\sqrt{5}+1}{2} \approx 1.618$ be the golden ratio.

Theorem 3. *Every deterministic online algorithm ALG for the wORSM problem has a competitive ratio of at least* $\phi = \frac{\sqrt{5}+1}{2}$.

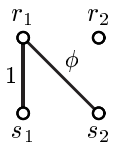


Fig. 6. Situation at time $t = 1$.

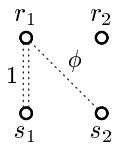


Fig. 7. $\{r_1, s_2\} \in M_{\mathsf{ALG}}$

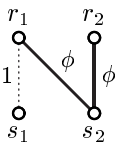


Fig. 8. $M_{\mathsf{ALG}} = \varnothing$

(Fig. 7 and 8 show the situations at time $t = 2$ after the different decisions and the reactions of the adversary; a double line is a matching edge; dotted lines are edges which have been removed before $t = 2$)

Proof. The adversary strategy starts with input edges $\{r_1, s_1\}$ and $\{r_1, s_2\}$ and their weights are $w(\{r_1, s_1\}) = 1$ and $w(\{r_1, s_2\}) = \phi$ as you can see in Fig. 6. ALG can react to this input at time $t = 1$ in two different ways:

Case 1 (Fig. 7): ALG adds edge $\{r_1, s_1\}$ to the weighted online matching M_{ALG}. Thereafter, the adversary does not present any new edge incident to s_2. So $|M_{\mathsf{ALG}}| = 1$ and $|M_{\mathsf{OPT}}| = \phi$ holds.

Case 2 (Fig. 8): ALG does not change the online matching M_{ALG}. The adversary presents edge $\{r_2, s_2\}$ with weight $w(\{r_2, s_2\}) = \phi$. Now ALG can construct a matching with weight $|M_{\mathsf{ALG}}| \leq \phi$ only, whereas it holds $|M_{\mathsf{OPT}}| = 1 + \phi$. The ratio of these two values is also ϕ.

Every two time steps the adversary can repeat this strategy up to infinity and this fact shows the lower bound of the competitive ratio of $\phi = \frac{\sqrt{5}+1}{2}$. □

4.2 The Algorithm wLMM

The algorithm wLMM works similarly to LMM. wLMM computes a maximum weighted matching on the local bipartite graph B_i. Furthermore, the algorithm works without the special preference of vertex s_i. The way LMM does it cannot help for the weighted problem because the weights of edges can differ by very small values.[4] The problems arising from this fact will be demonstrated in the following analysis. The formal description of wLMM is:

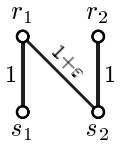
1: $i \Leftarrow 0$
2: **loop**
3: $\quad i \Leftarrow i + 1$
4: $\quad$ read input of time step i and build up B_i
5: $\quad$ construct a maximum weighted matching $\mathcal{M}(B_i)$ on B_i
6: $\quad$ **if** s_i is matched in $\mathcal{M}(B_i)$ **then**
7: $\quad\quad$ add the matching edge of s_i to the online weighted matching M_{wLMM}
8: $\quad$ **end if**
9: **end loop**

[4] Taking advantage of this possibility, an adversary can force such kind of 'clever' online algorithm to take the same decisions as wLMM. Therefore, the weights of the input edges need to be changed by a very small ε only.

The Lower Bound of wLMM

Theorem 4. *The competitive ratio of wLMM cannot be less than 2.*

Proof. A simple input structure is applied to show the theorem. This structure (which can be repeated arbitrarily often) is the following: $w(\{r_1, s_1\}) = 1$, $w(\{r_1, s_2\}) = 1 + \varepsilon$, and $w(\{r_2, s_2\}) = 1$



The online algorithm wLMM is too greedy and matches r_1 to s_2. Then it holds $|M_{\mathsf{wLMM}}| = 1 + \varepsilon$, whereas $|M_{\mathsf{OPT}}| = 2$. Therefore, this lower bound of the competitive ratio exceeds any fixed real number less than 2 for a proper small value of ε. □

The Upper Bound of wLMM

Due to strict space limitations the next theorem is presented without a proof. It can be found in [Rie98].

Theorem 5. *The deterministic online algorithm wLMM is 2-competitive.*

The proof is based on a technique developed in [BR93]. It extensively uses arguments of how dynamic changes of the local bipartite graph (insertion and deletion of edges) modify the structure and property of augmenting paths. The lemmata which are describing these characteristics are not difficult but rather lengthy and a bit technical. So we had to omit them and decide to omit the whole proof because its presentation is rather senseless without the preparatory explanations. However, this proof may be of minor interest only. Theorems 3 and 5 reveal a large gap in the analysis of the wORSM problem. We have the feeling that the upper bound can be improved. Therefore, an online algorithm must prefer the current server vertex, e. g. by increasing the weights of its edges. Unfortunately, such an algorithm causes different changes in the local bipartite graph and our proof collapses completely. Hence, a different proof technique needs to be developed.

5 Concluding Remarks

Very often the following question arises: Can a randomized algorithm perform 'better'? However, randomized online algorithms are studied in a different model of competitive analysis (weaker adversary models) and that was not the subject of our investigations. Nevertheless, a simple extension of the lower bound construction of Theorem 1 immediately shows a $\frac{6}{5} = 1.2$ lower bound for the competitive ratio of randomized online algorithms against the oblivious adversary. The

two input structures used by the adversary in Theorem 1 are chosen independently, uniformly at random to build up the input of the next four time steps. It is also not difficult to transform LMM into a randomized online algorithm. It selects one of the proper maximum cardinality matchings on B_i at random. However, we do not have an improved analysis yet. Note that a straight forward application of the RANKING algorithm [KVV90] cannot achieve an improved bound for the ORSM problem.

A different direction for further research is to study restricted model variants. It is possible to model additional concepts like parallel resources, lookahead, or individual time windows for the service of requests, by restricting the structure of the set of edges. Some of these model variants are more closely related to real world problems and a few decreased competitive ratios have been already discovered.

Acknowledgments

The author would like to thank Yossi Azar for a discussion and the comment that the ORSM problem can be regarded as a special case of the online bipartite matching problem.

References

[BEY98] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[BR93] Ethan Bernstein and Sridhar Rajagopalan. The roommates problem: Online matching on general graphs. Technical Report CSD-93-757, University of California, Berkeley, CS Department, 1993.

[Edm65] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal on Mathematics*, 7:449–467, 1965.

[KP98] Bala Kalyanasundaram and Kirk Pruhs. On-line network optimization problems. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms: The State of the Art*, LNCS 1442, pages 268–280. Springer-Verlag, Berlin-Heidelberg-New York, 1998.

[KVV90] 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*, pages 352–358, New York, 1990. ACM Press.

[Rie98] Marco Riedel. Online request server matching. Technical Report tr-ri-98-195, University of Paderborn, Germany, April 1998. available via: ftp://ftp.uni-paderborn.de/doc/techreports/Informatik/tr-ri-98-195.ps.Z

[Sga98] Jiří Sgall. On-line scheduling. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms: The State of the Art*, LNCS 1442, pages 196–231. Springer-Verlag, Berlin-Heidelberg-New York, 1998.