Online Request Server Matching*

Marco Riedel†
Graduate College at the
Heinz Nixdorf Institut
University of Paderborn, Germany
barcom@hni.uni-paderborn.de

April 1998

Abstract

In this report we study a new variant of an online bipartite matching problem, which can be interpreted as a scheduling problem. Then, we have one resource, called *server*, which is available for one unit in every step of a discrete time model. At most one task with unit demand can occur per time step. It is called *request*, and specifies a set of time steps when the serving is accepted. These times must not be situated in the past and neither need to be consecutive nor include the time of the arrival of the request. After the arrival of a request in time step i, an online algorithm has to decide which request will be served in step i. This decision has to be made without knowledge of future requests. The objective is a maximization of the number of served requests.

It is obvious, how to model this scheduling problem with a matching problem in a bipartite graph. However, this online bipartite matching problem differs considerably from the model in [KVV90]. So we call our model *online request server matching* (ORSM). Indeed, it is a special case of the *roommates problem*, that was introduced in [BR93].

To investigate the ORSM problem, we use competitive analysis. A lower bound of the competitive ratio of 1.5 is shown, and an optimal 1.5-competitive online algorithm is presented. Additionally, a weighted variant of the ORSM problem (wORSM) is investigated. Here, every edge has a weight and the objective is the online construction of a matching with maximal overall weight. For the wORSM problem a general lower bound of 1.618 for the competitive ratio is shown. A presented online algorithm is proven to be 2-competitive.

A few concluding remarks and a discussion of open problems complete this report.

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

Contents

1 Introduction **3**
1.1 Competitive Analysis 4
1.2 Previous Work 6

2 The Model **8**
2.1 Our Model in Comparison with Models out of Literature 9
2.2 Definitions and Notations 10

3 The Online Request Server Matching Problem **12**
3.1 The Lower Bound 12
3.2 The Algorithm LMM 14
3.3 The Upper Bound 15

4 The Online Request Server Weighted Matching Problem **16**
4.1 The Lower Bound 16
4.2 The Algorithm wLMM 17
4.3 The Performance of wLMM 18
4.4 The Algorithm PHI 31

5 Open Problems **32**

1 Introduction

In this report we study a model which was developed for a rather simple scheduling problem. Consider a single resource and a discrete time model. The resource is available for one unit in every time step. In the following this resource is called *server*. Every time step a task can occur that has a demand of one unit of the server. This task is called *request*. Such a request includes a set of time steps which specifies acceptable times for serving. These times must not be situated in the past and neither need to be consecutive nor include the time step of the arrival of the request. It is obvious that we can model this problem by a bipartite graph $G = (R \dot{\cup} S, E)$. The two disjoint partitions represent the server resource at every time step (S) and the set of requests (R). Whenever a request $r \in R$ can be served at time j, there is an edge $\{r, s\}$ between request vertex r and the vertex $s \in S$ which represents the j-th time step. Then the problem to decide when a request should be served is nothing else than to construct a matching in G.
Our scheduling problem is *online*, i. e. the requests appear through time while the current usage of the server has to be determined. These decisions have to be made without knowledge of further requests and cannot be taken back. The definition of our problem implies that in general it is impossible to serve all requests[1]. Additionally, the lack of information about the future prevents an optimal solution to be found which maximizes the number of successful requests.
This online matching problem in a bipartite graph is called *online request server matching* (ORSM).
We make use of *competitive analysis* to investigate the described problem. Here the worst case factor between the quality of an optimal solution and a solution calculated by an online algorithm on the same input is determined. This factor is called *competitive ratio*. For our matching problem the size of a maximum matching in G is the value of the optimal solution. It is well-known how to calculate it in $O(\sqrt{|R \cup S|} \cdot |E|)$ time (see [ET75] or any comprehensive textbook on algorithms).
In this report we present an optimal online algorithm which constructs a matching of at least $\frac{2}{3}$ of the maximal size. We also investigate a weighted variant of this online matching problem. Then all edges have a weight and the objective is the construction of a maximal weighted matching. For this problem our analysis shows a lower bound of $\frac{\sqrt{5}+1}{2} \approx 1.618034$ and an upper bound of 2 for the competitive ratio.

The material of this report is organized as follows. The next subsection presents a short introduction to competitive analysis. Thereafter, an overview of related work is given. Section 2 defines our model formally. It will be compared precisely with two models that have been studied in literature. Additionally, a few definitions and notations are given in the end. The simple, unweighted variant of our model is analysed in Sect. 3. It includes a general lower bound for the competitive ratio, a deterministic online algorithm, and a matching upper bound. Section 4 investigates the weighted version of this problem. The proof of a general lower bound is followed by the presentation of an algorithm. Its tight

[1] Imagine two requests to exactly the same, single server resource.

analysis establishes a gap to the previous lower bound. The section ends with some remarks and a suggestion for a more sophisticated online algorithm. This report is completed by a description of a few open problems.

1.1 Competitive Analysis

Algorithms are often based on the principle of the following sequence: input, computation, and output. After reading the complete input, which is a description of a finite problem, it computes a solution and presents it at the end. Such a behaviour is called *offline*.
In real world applications like e. g. embedded systems we find different requirements. Here, parts of the input appear by and by for an indefinite period of time and decisions of the solution have to be made immediately without knowing the future. These types of problems, including their solving algorithms, are called *online*. Different methods were suggested to analyse such online algorithms.
The most popular methods of the past, which are still in use, assume a known input distribution (a stochastic model for the input). Of course, the expressiveness of results of these studies are highly dependent on the correct choice of the distribution.
In contrast 1984 Daniel D. Sleator and Robert E. Tarjan introduced a new method for analysing online algorithms (journal version is [ST85]). It is a method of worst case analysis which avoids these difficulties and has been becoming more and more popular in the last decade.
The basic idea is a comparison of the quality of the solution computed by the online algorithm and the quality of an optimal, offline solution for the same input.
Let $\mathcal{A}$ be an online algorithm and let OPT be an optimal offline algorithm for a payoff maximization problem[2]. Then, for an input sequence σ we denote with $perf_{\mathcal{A}}(\sigma)$ the performance of the algorithm $\mathcal{A}$ and $perf_{\mathsf{OPT}}(\sigma)$ is the performance of OPT respectively.

Definition 1 (c-competitive)

$\mathcal{A}$ is called c-competitive *if there is a constant α such that*

$$\forall \sigma : \quad perf_{\mathsf{OPT}}(\sigma) \leq c \cdot perf_{\mathcal{A}}(\sigma) + \alpha \ .$$

The constant α has to be independent of the input and can compensate irregularities right at the beginning. Then the infimum over all values c such that $\mathcal{A}$ is c-competitive is called the *competitive ratio* of $\mathcal{A}$. Whenever the constant $\alpha = 0$ we can alternatively define the competitive ratio as

$$\sup_{\sigma} \frac{perf_{\mathsf{OPT}}(\sigma)}{perf_{\mathcal{A}}(\sigma)} \ .$$

[2]In this report we do not consider problems which have to minimize costs. The definitions would slightly differ to ensure that a competitive ratio is always at least 1.

We immediately realize that this analysis is independent of a stochastic model for the inputs and gives performance *guarantees*. On the other hand this kind of analysis is sometimes unrealistically pessimistic.
Indeed, an online algorithm approximates an optimal solution while working under the restriction of incomplete knowledge. So the distinguished name competitive ratio for the approximation factor seems to be adequate.
An excellent, introductory article on competitive analysis is [Kar92]. A comprehensive introduction is the textbook by Allan Borodin and Ran El-Yaniv [BEY98] which includes an extensive bibliography.
Two essential techniques to prove competitive ratios have been established. A description of these methods illustrates the character of this type of analysis.
Lower bounds for the competitive ratio are typically shown by an adversary strategy. We can interpret competitive analysis as a game between an online algorithm $\mathcal{A}$ and an omniscient adversary. The adversary creates the input sequence σ with knowledge of $\mathcal{A}$. So it can in advance calculate every decision of $\mathcal{A}$ on σ, $perf_{\mathcal{A}}(\sigma)$, and $perf_{\mathsf{OPT}}(\sigma)$. Due to the unlimited computational power the malicious adversary is able to construct a worst case input sequence.
When we derive general lower bounds for the competitive ratio of a problem we slightly change our view. Then a strategy is given which tells the adversary how to construct the next part of the input sequence dependent on the decisions an online algorithm was able to make. The performance analysis of these sequences gives a lower bound for the competitive ratio when the strategy is able to generate infinite inputs. Otherwise the loss of performance of an online algorithm could be compensated by the constant α in the above definition.
To prove upper bounds for an online problem the performance of a designed algorithm $\mathcal{A}$ is investigated. Sometimes, ad hoc arguments can be applied to show how $\mathcal{A}$ can bound the competitive ratio. However, it is more common to apply a potential function argument for an analysis of amortized performance. Hence, every pair of a state of $\mathcal{A}$ and a state of OPT is mapped to a real value by a potential function Φ. Whenever it can be shown that for every step i with input σ_i holds

$$perf_{\mathsf{OPT}}(\sigma_i) \;\leq\; c \cdot perf_{\mathcal{A}}(\sigma_i) + \Phi_i - \Phi_{i-1}$$

and Φ is bounded above, then $\mathcal{A}$ is c-competitive. It was shown that there exists such a potential function for every online algorithm (see the description in [IK96], page 534). Nevertheless the variant usage of potential functions and combinations with additional arguments are nowadays commonly used in proofs of the performance of online algorithms.

Several extensions of the competitive analysis were suggested. A major influence had the seminal work [BDBK$^+$94] which introduces randomized online algorithms and adapted methods for their analysis. However, in this report we limit our study to deterministic online algorithms.

1.2 Previous Work

We introduced our online problem in terms of a scheduling problem. However, there is a vast literature on the subject of online scheduling and related problems like load balancing and routing. We will not review these works here. The reader may consult the survey by Jiří Sgall [Sga98] instead. The publications of online matching problems are more relevant to the studies of this report. In the following, we discuss these papers.

The first article about an online version of a bipartite matching problem is by Richard M. Karp, Umesh V. Vazirani and Vijay V. Vazirani [KVV90]. The partition U of the graph $G = (U \cup V, E)$ is known in advance and the vertices of V including their edges arrive through time. Whenever a vertex $v \in V$ is revealed, an online algorithm has to decide which edge incident to v is added to the online matching M. The objective is the maximization of the size of M. For deterministic online algorithms the competitive ratio is exactly 2. An adversary can present an input with a vertex $v_1 \in V$ being adjacent to two vertices $u, u' \in U$. After the decision of the online algorithm, the adversary presents vertex $v_2 \in V$ which is adjacent to the previous matched vertex u or u', only. The online algorithm is not able to match v_1 *and* v_2. However, there is such an offline solution. The infinite repetition of this strategy results in the lower bound of 2. The greedy algorithm which adds the first possible edge of an incoming vertex $v \in V$ to the matching M achieves this competitive ratio. From graph theory we know that this greedy algorithm constructs a maximal matching (every edge of the graph is either itself a matching edge or is adjacent to one) and every maximal matching has at least half the size of an optimal, maximum matching.
The key contribution of the discussed paper is the analysis of a randomized online algorithm for the described problem.

Ming-Yang Kao and Stephen R. Tate investigated the above model with blocked inputs [KT91]. When an input occurs k vertices were revealed 'in a block' instead of one at a time. Let $n = |V|$ be the number of vertices in the partition V of the bipartite graph. Of course with $k = 1$ the problem is the same as above and for $k = n$ we have the offline version of the matching problem. For deterministic online algorithms, no improvements are possible as long as $k \leq \frac{n}{2}$. For randomized algorithms the result of [KVV90] cannot be improved apart from low order terms as long as $k = o(n)$.

The online b-matching was analysed by Bala Kalyanasundaram and Kirk Pruhs in [KP95]. In this problem the vertices of the partition V can be incident to at most b edges out of the matching. Again, the size of the matching has to be maximized. The authors give an optimal deterministic algorithm with competitive ratio

$$\frac{(1+\frac{1}{b})^b}{(1+\frac{1}{b})^b - 1} \ .$$

A weighted variant of the online bipartite matching was studied by the same authors in [KP93]. When allowing an arbitrary weight function for the edges, no competitive algorithm can exist. To see this consider that an online algorithm has chosen a matching

edge of weight one. As a next step an adjacent edge of arbitrary weight is revealed. Then no online algorithm can bound the competitive ratio.
Hence, the model was restricted to a weighted, complete bipartite graph of partitions of equal size $n = |U| = |V|$ which is a metric space (especially the triangle inequation is fulfilled). When asked for a maximal weighted matching, the greedy algorithm is optimal and 3-competitive.
On the other hand in the minimum weighted matching problem, a *perfect* matching of minimal weight has to be determined in a bipartite graph defined as above. Then an optimal and $(2n-1)$-competitive algorithm is shown. The same result was independently discovered by Samir Khuller, Stephen G. Mitchell and Vijay V. Vazirani in [KMV94][3]. The last article also contains a study of an online version of the stable marriage problem.

In [ANR95] Yossi Azar, Joseph Naor and Raphael Rom studied a different model based on bipartite graphs. They called it online assignment. The partition U has a fixed size and vertices out of V are adjacent to a subset of U. For each $v \in V$ one of its weighted edge must be selected immediately after its arrival with the objective to minimize the maximal weight of all selected edges incident to a vertex in U. Indeed, this is a load balancing problem. For deterministic online algorithms a lower bound of $\lceil \log_2(n+1) \rceil$ and an upper bound of $\lceil \log_2 n \rceil + 1$ is shown for the competitive ratio.

The technical report by Ethan Bernstein and Sridhar Rajagopalan [BR93] is of major importance for our following studies. An online matching problem in general graphs called *roommates problem* has been introduced. The graph $G = (V, E, w)$ is undirected, simple and weighted. An unweighted version of this model has also been investigated.
The input sequence consists of the vertices of V. Whenever a vertex is revealed, all of its adjacent vertices and the weighted edges in between become known. We want to emphasize that this process includes adjacent vertices never seen before in the input sequence. Then, an edge of the current vertex to a non-matched vertex (that has been revealed previously) can be added to the online matching. An edge to a non-revealed and incompletely known vertex can be selected later when this adjacent vertex is the current one in the input sequence.
For the roommates problem the unweighted model is interpreted as follows: People arrive one at a time to a hotel where a conference takes place. The hotel consists of double rooms, only. Every person gives a list of possible roommates independently of whether they have arrived yet. The model assumes that these lists are *symmetric*, i. e., every potential roommate will accept to share this room. The hotel manager has to decide in which room the person will stay. The objective is to minimize the allocated rooms, i. e. to maximize the matching in the implicitly defined graph.
In the weighted version the aim is an online construction of a weighted matching of maximal size in G.
In the paper a tight analysis of the unweighted model is given. Therefore the competitive ratio is 1.5 . For the weighted model a lower bound of 3 is proven. A suggested online algorithm is shown to be 4-competitive.

[3]Both articles ([KP93] and [KMV94]) were previously published on conferences in 1991.

At the end of the next section we will revisit the roommates problem and we will compare it to our model. Indeed, our model is a special case of the roommates problem. For our investigations we were able to adapt proofs taken from the discussed paper.

2 The Model

In the beginning of the introduction we described an online matching problem. Now we will present a formal definition for the *online request server matching* problem (ORSM).
The underlying structure of the problem is a bipartite graph $G := (R \dot\cup S, E)$. 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 indicate the position within the order. We interpret this order as a discrete time model. The vertices of partition S represent a simple resource called *server*, which 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. We call them *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 the request can be served in time step j. The set of edges $E \subset R \times S$ is constructed with a restriction:

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

This means that 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 would be possible.
Now we have to specify how this model works online: When the system starts the partition R is completely unknown[4]. In the beginning of a time step i the request r_i is revealed as input, i. e. the vertex r_i and all of its edges are inserted into the previous known part of G. If no request appears, vertex r_i is isolated.
After this input, an algorithm has to decide how to use the server in the current time step i. Indeed, it can add an edge incident to s_i to the online matching M. It is worth noting that due to the restriction (1) *all* edges incident to s_i are known when this decision has to be made. The online algorithm has the objective to maximize the cardinality of the matching M, i. e. to serve as much requests as it can.
Up to now the graph G was unweighted. We also study the weighted variant. Then the graph is defined as $G := (R \dot\cup S, E, w)$. The weight function $w : E \to \mathbb{R}_+$ defines a positive real value for every edge. The objective is the construction of a weighted matching of maximal overall weight. Otherwise, the problem and its online fashion is completely the same as in the ORSM problem. This version is named *online request server weighted matching* problem or in short wORSM problem.

[4]When we take a close look this is not the truth. We know that every time step i a new vertex r_i is inserted but its set of incident edges is unknown before. For our convenience we will interpret the input process in the introduced way.

2.1 Our Model in Comparison with Models out of Literature

At first we want to make a clear distinction between the ORSM model and the online bipartite matching problem in [KVV90]. In the later one the vertices from the unknown partition and their edges appear in the same way as in the ORSM problem. However, in the online bipartite matching problem no order on the vertices of the already known partition is given and no restrictions on the set of edges. After a new vertex v is inserted the online algorithm has to decide which edge incident to v should be added to the matching. In contrast, in the ORSM model, one asks for an edge incident to the current server vertex to add into the matching. This server vertex is situated in the *known* partition of the bipartite graph. The restriction (1) of the set of edges guarantees that all edges of this server vertex are known at that time. Due to the specified input process (revealing a request vertex and all of its edges), all adjacent edges are known as well. There is therefore the advantage to have some extra knowledge of the graph structure whenever a decision has to be taken. That is also the reason why it is possible to achieve a better competitive ratio for the ORSM problem. Additionally, in the weighted version, a restriction on a metric space, or the other restrictions described in Sect. 1.2 are not necessary.
Again, we want to emphasize the difference: In the online bipartite matching problem decisions about adding an edge to the online matching are made with respect to a set of edges incident to a *just revealed* vertex. In the ORSM problem such a decision is made with respect to a set of edges incident to a vertex of *the other* (known) partition.

In the previous section it was claimed that the ORSM problem is a special case of the roommates problem. Now we are able to give a transformation of an instance of the wORSM problem with its bipartite graph $G = (R \dot\cup S, E, w)$ to an instance of the roommates problem with the underlying graph $G_{\mathrm{R}} = (V_{\mathrm{R}}, E_{\mathrm{R}}, w_{\mathrm{R}})$ and its total order $\prec$ on the set V_{R}. This transformation defines:

$$\begin{aligned} V_{\mathrm{R}} &:= R \cup S \ , \\ E_{\mathrm{R}} &:= E \ , \\ w_{\mathrm{R}} &:= w \ , \end{aligned}$$

and the order $\prec$ on V_{R} is defined by the use of the orders on R and S such that:

$$r_1 \prec s_1 \prec r_2 \prec s_2 \prec r_3 \prec s_3 \prec r_4 \prec s_4 \prec \dots$$

Whenever a vertex r_i is inserted, the roommates problem on G_{R} is not able to add any edge to M because no adjacent s-vertices are revealed (remember restriction (1) on E). Then vertex s_i is inserted. All of its incident edges and adjacent vertices respectively, are known at that time and so every edge of s_i can be selected as a matching edge in the roommates problem. No edge of an unrevealed vertex was given simultaneously. That means s_i will never be a candidate for the matching again. We conclude that the roommates problem can simulate the wORSM problem using the given transformation.

Nevertheless, the two models are not able to simulate each other. Hence, we have to prove the lower bound of the competitive ratio for our more restrictive model. Additional we present an online algorithm and an analysis applying simplified arguments which are tailor-made to the ORSM problem. In the weighted model, the increase in knowledge of the graph structure, compared to the roommates problem, results in a lower value for the competitive ratio.

Before starting our investigations a few definitions and notations are presented.

2.2 Definitions and Notations

Although we assume that the reader is familiar with basic concepts in graph theory, and although we already used some graph theoretical notations, a few standard definitions are included in the following list.

Definition 2 (weighted graph)

$G = (V, E, w)$ *is called a* weighted graph *iff*

V *is a finite set of vertices,*
$E \subset \{\{u, v\} \mid u, v \in V \wedge u \neq v\}$ *is a set of edges, and*
$w : E \to \mathbb{R}_+$ *is a weight function.*

When using the unit weight function $w : E \to 1$, an unweighted graph $G = (V, E)$ is derived. As you can see, the weight function is omitted in the notation. This conversation from a weighted to an unweighted graph applies to the next definitions of graphs and matchings.
Bipartite graphs consist of two disjoint sets of vertices without edges inside the sets:

Definition 3 (bipartite weighted graph)

$G = (U \dot\cup V, E, w)$ *is called a* bipartite weighted graph *iff*

U *and* V *are finite sets of vertices,*
$E \subset \{\{u, v\} \mid u \in U \wedge v \in V\}$ *is a set of edges, and*
$w : E \to \mathbb{R}_+$ *is a weight function.*

Definition 4 (vertex induced subgraph)

Let $G = (V, E, w)$ *be a weighted graph and* $V_S \subset V$. *Then*

$$G|_{V_S} := (V_S, \{\{u, v\} \mid u, v \in V_S, \{u, v\} \in E\}, w)$$

is the subgraph of G induced by vertex set V_S .

Definition 5 (matching)

Let $G = (V, E, w)$ be a weighted graph. M is called a matching *in G iff*

$$M \subset E \text{ and}$$
$$\{u, v\} \in M \;\Rightarrow\; \nexists\, x \in V : \{u, x\} \in M \wedge x \neq v \;.$$

Definition 6 (weight of a matching, $|M|$)

Let $G = (V, E, w)$ be a weighted graph and let M be a matching in G.

$$|M| \;:=\; \sum_{\{u,v\} \in M} w(\{u, v\})$$

is called the weight of matching M .

It is obvious that $|M|$ counts the number of edges in M when it is applied to an unweighted graph. In that case $|M|$ is called the *cardinality* or *size* of the matching M .

Definition 7 (notation: $\overset{\circ}{\in}$)

Let $G = (V, E)$ be a graph and let $M \subset E$ be a matching. For $v \in V$

$$v \overset{\circ}{\in} M \;\;:\Leftrightarrow\;\; \exists\, u \in V : \{v, u\} \in M \text{ and}$$
$$v \overset{\circ}{\notin} M \;\;:\Leftrightarrow\;\; \nexists\, u \in V : \{v, u\} \in M \;.$$

Definition 8 (maximum weighted matching, $\mathcal{M}(G)$)

Let $G = (V, E, w)$ be a weighted graph. Then $\mathcal{M}(G)$ denotes a maximum weighted matching:

$$\mathcal{M}(G) \in \{M \mid \nexists\, M' : |M| < |M'| \wedge M, M' \text{ matchings in } G\} \;.$$

We will frequently denote a specially defined or calculated maximum matching by $\mathcal{M}(G)$.

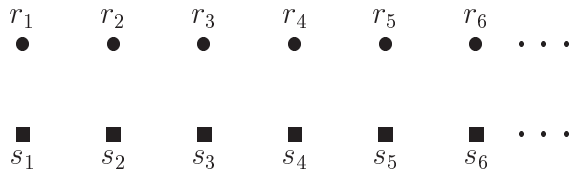
Definition 9 (symmetric difference, $\oplus$)

Let A and B be two sets. Then $A \oplus B$ denotes the symmetric difference:

$$A \oplus B \;:=\; (A \cup B) \setminus (A \cap B) \;=\; (A \setminus B) \cup (B \setminus A) \;.$$

To illustrate structures of graphs and matchings, we use a graphical notation. It is fully interchangeable with the set theoretical representation. Vertices of the two sets in bipartite graphs were depicted by small, filled circles ● and squares ■ respectively. The circles

represent vertices from the request partition R, and squares represent vertices out of the server partition S. Additionally, the label of such a vertex is written right next to its symbol. When we sketch an instance of the wORSM problem, the vertices of the request partition are drawn in their order along a horizontal line from left to right. The server vertices are drawn in the same way in a distance below. Indeed, a request vertex is situated precisely on top of the server vertex of the same time step:



Edges are depicted by a line between two vertices. Of course in bipartite graphs such a line has a circle and a square on its ends:
To mark the edges of a matching they are symbolized by a double line:
Whenever edges cannot be selected for the matching anymore, due to previous decisions made by the online algorithm, they were depicted in grey:

3 The Online Request Server Matching Problem

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

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 $\mathcal{A}$ for the ORSM problem has a competitive ratio of at least 1.5.

Proof. The adversary strategy starts with the following input structure:
$\{\{r_1, s_2\}, \{r_1, s_3\}, \{r_2, s_2\}, \{r_2, s_4\}\}$.

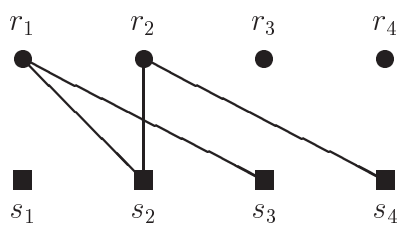


Figure 1: Situation at time $t = 2$.

$\mathcal{A}$ can react to this input at time $t = 2$ in three different ways:

Case 1: $\mathcal{A}$ puts the edge $\{r_1, s_2\}$ to the online matching $M_{\mathcal{A}}$. In the next step the adversary presents edge $\{r_3, s_4\}$.

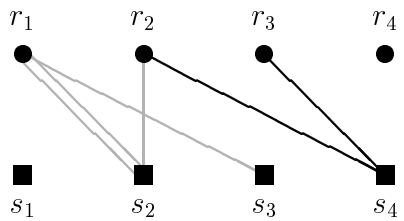


Figure 2: $\{r_1, s_2\} \in M_{\mathcal{A}}$, situation at time $t = 3$.

$\mathcal{A}$ is not able to use the serving interval s_3. Therefore, $|M_{\mathcal{A}}| \leq 2$ whereas the optimal solution gives $|M_{\mathsf{OPT}}| = 3$.

Case 2: $\mathcal{A}$ puts the edge $\{r_2, s_2\}$ to the online matching $M_{\mathcal{A}}$. In the next step the adversary presents edge $\{r_3, s_3\}$.

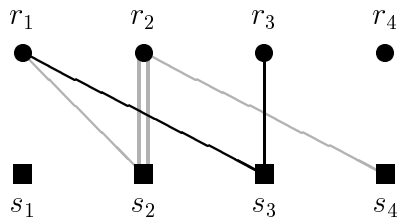


Figure 3: $\{r_2, s_2\} \in M_{\mathcal{A}}$, situation at time $t = 3$.

$\mathcal{A}$ cannot use s_4. Again $|M_{\mathcal{A}}| \leq 2$ and the optimal matching results in $|M_{\mathsf{OPT}}| = 3$.

Case 3: $\mathcal{A}$ decides not to match s_2.

The adversary will present the input of Case 1 (the input of Case 2 would work as well) and $|M_{\mathcal{A}}| \leq 2$, $|M_{\mathsf{OPT}}| = 3$ holds.

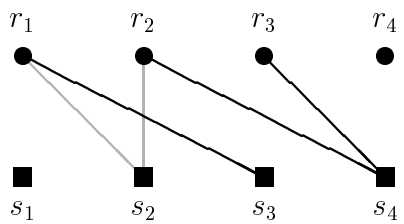


Figure 4: $M_{\mathcal{A}} = \varnothing$, situation at time $t = 3$.

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

$$\frac{|M_{\mathsf{OPT}}|}{|M_{\mathcal{A}}|} \geq \frac{3}{2} .$$

■ Theorem 1

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, we know the subgraph of G induced by the set $V_i = \{r_k |\, r_k \in R, 1 \leq k \leq i\} \cup S$. Due to the irreversible decisions of the former time steps all previous server vertices $s_1, \ldots, s_{i-1}$ and all hitherto matched request vertices $r_k \overset{\circ}{\in} M_i$ cannot be rearranged anymore (M_i is the online matching up to time i; $M_i \subset M$). So we have a vertex induced bipartite subgraph of G:

$$\begin{aligned} \tilde{V}_i &:= \{r_k |\, r_k \in R, r_k \overset{\circ}{\notin} M_i, 1 \leq k \leq i\} \cup \{s_k |\, s_k \in S, i \leq k\} \\ B_i &:= G|_{\tilde{V}_i} . \end{aligned}$$

Our online algorithm is called 'Local Maximum Matching' (LMM) because it constructs a maximum matching on every local subgraph B_i (denoted by $\mathcal{M}(B_i)$). The exact function of LMM follows:

```
1:  i ⇐ 0
2:  loop
3:     i ⇐ i + 1
4:     read input of time i and build up B_i
5:     construct a maximum matching M(B_i) on B_i :
           start with all matching edges of M(B_{i-1}) which are edges in B_i;
6:         look for an augmenting path which starts at vertex r_i and do the
           augmentation when found[5]
7:     if s_i ∈̊ M(B_i) then
8:        add the matching edge of s_i to the online matching M_LMM
9:     else if s_i is not isolated in B_i then {all neighbours of s_i are matched in M(B_i)}
10:       add an arbitrary edge {s_i, r} of B_i to the matching M_LMM and delete the
          matching edge of r in M(B_i)
11:    end if
12: end loop
```

Line 10 of this algorithm is essential and prefers a matching which includes the current vertex s_i. Now we will analyse the performance of LMM.

[5] due to the maximum cardinality of $\mathcal{M}(B_{i-1})$ any augmenting path must have r_i on one end

3.3 The Upper Bound

At first, two invariants of LMM will be formalized in lemmata.

Lemma 2

After a request vertex r_i has been matched in B_i for the first time, it is in all following maximum matchings $r_i \mathring{\in} \mathcal{M}(B_j)$, $j > i$ up to the time step where its current matching edge is added to M_{LMM} and so $r_i \mathring{\in} M_{\mathsf{LMM}}$ holds in the end.

Proof. In line 5 of LMM at time j the matching edge of a previous matched request vertex $r \in R, r \mathring{\in} \mathcal{M}(B_{j-1})$ is copied to $\mathcal{M}(B_j)$ and an augmentation in line 6 can change r's matching edge but $r \mathring{\in} \mathcal{M}(B_j)$ holds. In line 10 the matching edge of a vertex $r \mathring{\in} \mathcal{M}(B_j)$ is changed but after this step $r \mathring{\in} M_{\mathsf{LMM}}$ holds. ■ Lemma 2

Lemma 3

If s_i is not isolated in B_i, then $s_i \mathring{\in} M_{\mathsf{LMM}}$ holds.

The lemma follows directly from line 7 to 11 of LMM. Applying these two invariants we can prove the upper bound now.

Theorem 4

LMM is 1.5-competitive.

Proof. We will show that no online matching M_{LMM} can be increased by augmenting paths of length one or three. Therefore, shortest augmenting paths for M_{LMM} must have a length of five.
This fact immediately results in the theorem because by performing the augmentation, two matching edges out of M_{LMM} become three edges in M_{OPT}. Longer augmenting paths have a lower ratio and matching edges outside such paths decrease the overall ratio as well.
Let us fix an arbitrary input graph G_{I} and a maximum matching $M_{\mathsf{OPT}} = \mathcal{M}(G_{\mathrm{I}})$. Then we can compare M_{OPT} with the matching M_{LMM} constructed by LMM on G_{I}. The symmetric difference $M_{\mathsf{OPT}} \oplus M_{\mathsf{LMM}}$ defines a set of *disjoint* augmenting paths for M_{LMM} (for more details see [Edm65]). The augmentation of all of these paths transforms M_{LMM} to M_{OPT}. By contradiction, we will prove the non-existence of paths of length one and three in this set.

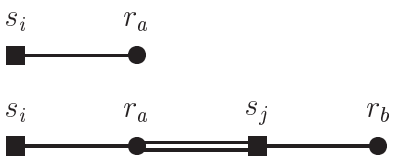


Figure 5: Structure of augmenting paths of length one and three.

Case 1: augmenting path of length one $\{s_i, r_a\} \notin M_{\mathsf{LMM}}$:

Vertex r_a was never matched because $r_a \overset{\circ}{\notin} M_{\mathsf{LMM}}$ (reverse application of Lemma 2). So $\{s_i, r_a\}$ is in B_i and this fact contradicts Lemma 3.

Case 2: augmenting path of length three and $i < j$:

r_a was matched at time j only, which implies the edge $\{s_i, r_a\}$ was in B_i. Then $s_i \overset{\circ}{\notin} M_{\mathsf{LMM}}$ contradicts Lemma 3.

Case 3: augmenting path of length three and $i > j$:

At time j, request vertices $r_a, r_b \overset{\circ}{\notin} M_{\mathsf{LMM}}$ and so the whole path $\{\{s_i, r_a\}, \{r_a, s_j\}, \{s_j, r_b\}\}$ is in B_j.

The case $s_i \overset{\circ}{\notin} \mathcal{M}(B_j)$ contradicts the optimality of $\mathcal{M}(B_j)$ because the path is an augmenting one. Therefore, at time j $s_i \overset{\circ}{\in} \mathcal{M}(B_j)$ must hold, 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 10 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 $\{s_k, r_c\}$ 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}.

■ Theorem 4

4 The Online Request Server Weighted Matching Problem

Like in section 3, we present a general lower bound for the wORSM problem first. Then the algorithm wLMM is given and analysed. Unfortunately this algorithm cannot achieve the lower bound and so we suggest the algorithm PHI at the end of this section.

4.1 The Lower Bound

Let $\phi := \frac{\sqrt{5}+1}{2} \approx 1.618034$ be the golden ratio.

Theorem 5

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

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.

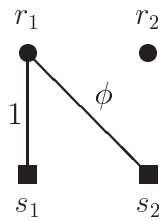


Figure 6: Situation at time $t = 1$.

$\mathcal{A}$ can react to this input in two ways:

Case 1: $\mathcal{A}$ adds edge $\{r_1, s_1\}$ to the weighted online matching $M_{\mathcal{A}}$. Then the adversary does not presents any new edge incident to s_2. So $|M_{\mathcal{A}}| = 1$ and $|M_{\mathsf{OPT}}| = \phi$ holds.

Case 2: $\mathcal{A}$ does not change the online matching $M_{\mathcal{A}}$. Then the adversary presents edge $\{r_2, s_2\}$ with weight $w(\{r_2, s_2\}) = \phi$.

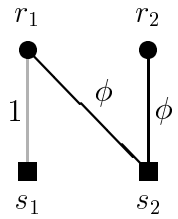


Figure 7: Situation at time $t = 2$.

Now $\mathcal{A}$ is able to construct a matching with weight $|M_{\mathcal{A}}| \leq \phi$ only, whereas it holds $|M_{\mathsf{OPT}}| = 1 + \phi$. The ratio of these two values is:

$$\frac{|M_{\mathsf{OPT}}|}{|M_{\mathcal{A}}|} = \frac{1+\phi}{\phi} = \frac{1+\frac{\sqrt{5}+1}{2}}{\frac{\sqrt{5}+1}{2}} = \frac{2\sqrt{5}+6}{2(\sqrt{5}+1)} = \frac{\sqrt{5}+1}{2} \cdot \frac{\sqrt{5}+1}{\sqrt{5}+1} = \phi \ .$$

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 $\phi = \frac{\sqrt{5}+1}{2}$. ■ Theorem 5

4.2 The Algorithm wLMM

The algorithm wLMM works very similarly to LMM. Of course, wLMM determines a maximum weighted matching on the local bipartite graph B_i. Furthermore, the algorithm works without the special preference of the vertex s_i. Later, on page 30, we will explain why a

special treatment of s_i, in the way LMM does, cannot increase the performance of wLMM. Our investigation will indicate the problems arising from this fact. A formal description of wLMM follows:

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** $s_i \mathring{\in} \mathcal{M}(B_i)$ **then**
7: $\quad\quad$ add the matching edge of s_i to the online weighted matching M_{LMM}
8: $\quad$ **end if**
9: **end loop**

4.3 The Performance of wLMM

In this section we present a tight analysis of wLMM being 2-competitive. At first we exhibit a lower bound of wLMM which establishes the fact that wLMM is not able to achieve the lower bound of theorem 5.

The Lower Bound

Theorem 6

The online algorithm wLMM has a competitive ratio of at least 2.

Proof. An adversary presents the input $w(\{r_1, s_1\}) = 1$, $w(\{r_1, s_2\}) = 1 + \varepsilon$ ($\varepsilon \in \mathbb{R}_+$) and in the next time step $w(\{r_2, s_2\}) = 1$.

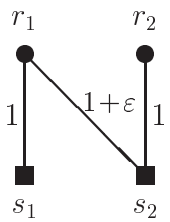


Figure 8: Input structure that is used by the adversary

The algorithm wLMM determines the online matching $M_{\mathsf{wLMM}} = \{\{r_1, s_2\}\}$ which has a weight of $|M_{\mathsf{wLMM}}| = 1 + \varepsilon$. The optimal solution is $M_{\mathsf{OPT}} = \{\{r_1, s_1\}, \{r_2, s_2\}\}$ with weight $|M_{\mathsf{OPT}}| = 2$. Consequently,

$$\frac{|M_{\mathsf{OPT}}|}{|M_{\mathsf{wLMM}}|} = \frac{2}{1+\varepsilon}$$

and the limit for $\varepsilon \to 0$ is:

$$\lim_{\varepsilon \to 0} \frac{2}{1+\varepsilon} \;=\; 2 \;.$$

■ Theorem 6

The Upper Bound

Firstly, we repeat a few definitions and give new ones which will be used extensively in the following proofs. Most of the notations are similar and comparable to [BR93], whereas one different notation is introduced. By the use of this notation, we aim to make the formulae more accessible.

Definitions and Notations

$G = (R \,\dot\cup\, S, E_G, w)$ is the bipartite, weighted graph of the wORSM problem with the weight function $w : E_G \to \mathbb{R}_+$.

M_i is the online matching which is calculated up to step i by wLMM.

$\mathcal{M}(B)$ is a maximum weighted matching of a bipartite graph B.

$m(B) := |\mathcal{M}(B)| := \sum_{\{r,s\} \in \mathcal{M}(B)} w(\{r,s\})$ is the overall weight of a maximum weighted matching of B.

$B_i := G|_{R_i \dot\cup S_i}$ is the local bipartite graph of step i which consists of all known and not matched request vertices $R_i \subset R$ with $r_j \in R_i \Rightarrow (j \le i$ and $r_j \overset{\circ}{\notin} M_i)$, all non-passed server vertices $S_i \subset S$ with $s_j \in S_i \Rightarrow i \le j$, and all edges between this two sets $(E_i = (R_i \times S_i) \cap E_G)$.

$B_{i \leftarrow v} := G|_{R_i \dot\cup S_i \dot\cup \{v\}}$, $v \in (R \,\dot\cup\, S) \setminus (R_i \,\dot\cup\, S_i)$,

$B_i{}^{\rightarrow v} := G|_{R_i \dot\cup S_i \setminus \{v\}}$, $v \in (R_i \,\dot\cup\, S_i)$.

Both notations above can be combined. We use indexed r's and s's to denote vertices of R and S, so the partition where these vertices come from is unambiguous. Additionally, we employ a list of inserted or deleted vertices when necessary. A typical usage of this notation is $B_i \overset{\rightarrow r_k, s_i}{\leftarrow r_{i+1}}$ which describes the vertex induced subgraph $G|_{R_i \dot\cup S_i \dot\cup \{r_{i+1}\} \setminus \{r_k\} \setminus \{s_i\}}$, $r_{i+1}, r_k \in R$, $s_i \in S$.

Let $\mathcal{M}(B_i)$ be a maximum weighted matching of B_i and let $\mathcal{M}(B_i{}^{\rightarrow s})$ be such a matching of B_i after removing the vertex s. The symmetric difference of these two sets results in a path $P := \mathcal{M}(B_i) \oplus \mathcal{M}(B_i{}^{\rightarrow s})$. It is an augmenting path[6] in B_i with respect to the matching $\mathcal{M}(B_i{}^{\rightarrow s})$. We will use notation P for the set of edges as defined above and

[6] this path can be empty

for a graph $P = (V_P, E_P, w_P)$ synonymously. Here E_P is the defined set of edges, and set V_P gathers all the incident vertices. With this definition, we define the *value* of the corresponding server vertex as:

$\beta_i(s) \;:=\; m(B_i) - m(B_i{}^{\rightarrow s}) \quad , \; s \in S_i \;\; .$

It is obvious that $\beta_i(s) = m(B_i) - m(B_i{}^{\rightarrow s}) = m(P) - m(P{}^{\rightarrow s})$ when we fix $\mathcal{M}(B_i)$ and $\mathcal{M}(B_i{}^{\rightarrow s})$ as above. Sometimes we call $\beta_i(s)$ the *weight* of path P because this value is added to the overall matching when augmenting P.

Next we define the global potential function:

$\Phi_i \;:=\; 2\,|M_i| + m(B_i) \;\; .$

At last the following abbreviated notations will be used for an expression f :

$f_i^+ \;:= f_{i+1} \;\; ,$

$\Delta f_i \;:= f_i^+ - f_i \;\; ,$

$f_{|v}$ is the value of f in step i with $v \in \{r_i, s_i\}$,

$\Delta f_{|v}$ is the difference of the value of expression f before and after processing vertex v.

Index i is omitted whenever it is clear from the context.

Basic Properties of Weighted Augmenting Paths Corresponding to $\beta(v)$

Now some important properties of weighted augmenting paths which corresponds to $\beta(v)$ are listed. In this context we can relax the restriction to bipartite graphs. All above mentioned definitions applies to simple, undirected, weighted graphs.

Lemma 7

Let $G = (V, E, w)$ be a simple, undirected, weighted graph. Then it holds:

$$\forall\, v \in V : \quad \beta(v) \geq 0 \;\; .$$

Proof. $G^{\rightarrow v}$ is a subgraph of G and both $\mathcal{M}(G)$ and $\mathcal{M}(G^{\rightarrow v})$ are optimal. So it holds $m(G) \geq m(G^{\rightarrow v})$ and $\beta(v) = m(G) - m(G^{\rightarrow v}) \geq 0$. ■ Lemma 7

Lemma 8

(I) *Every subpath P_b which arises from P by deleting an* even *number of edges from the start vertex v and which starts itself by vertex b is optimally matched by the matching of P, i. e. $\mathcal{M}(P_b) = \mathcal{M}(P) \cap P_b$.*

(II) *Every subpath P_c which arises from P by deleting an* odd *number of edges from the start vertex v and which starts itself by vertex c is optimally matched by the matching of $P^{\rightarrow v}$, i. e. $\mathcal{M}(P_c) = \mathcal{M}(P^{\rightarrow v}) \cap P_c$.*

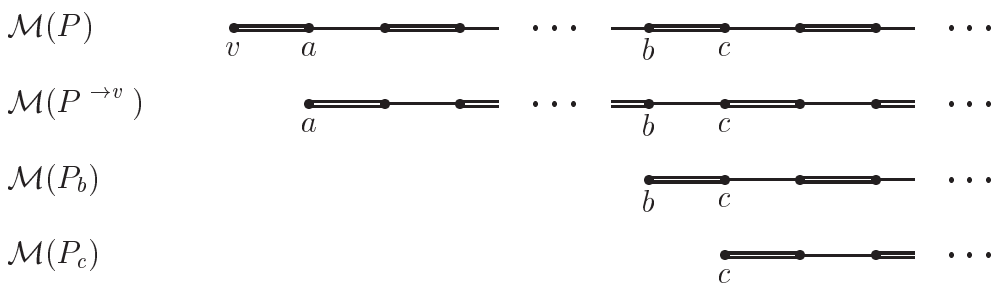


Figure 9: Sketch of the paths described in Lemma 8 including their optimal matchings.

Proof. We prove statement (I) by contradiction. Assume $\mathcal{M}(P) \cap P_b$ is not optimal:

$$|\mathcal{M}(P) \cap P_b| \; < \; |\mathcal{M}(P_b)| = m(P_b) \; .$$

The matching edge of b is in P_b, $\{b, c\} \in \mathcal{M}(P)$ and $\{b, c\} \in P_b$ because $\{v, a\} \in \mathcal{M}(P)$ and b is in a distance of *even* edges away from v. Path P can be divided into two subpaths at vertex b and it holds:

$$\mathcal{M}(P) \; = \; (\mathcal{M}(P) \setminus P_b) \,\dot\cup\, (\mathcal{M}(P) \cap P_b) \; .$$

Then the above assumption implies:

$$m(P) \; = \; |\mathcal{M}(P)| \; < \; |\mathcal{M}(P) \setminus P_b \,\dot\cup\, \mathcal{M}(P_b)| \; = \; |\mathcal{M}(P) \setminus P_b| + |\mathcal{M}(P_b)|$$

which contradicts the optimality of $\mathcal{M}(P)$.

In the same way statement (II) is proven. The line of argumentation is about $P^{\rightarrow v}$ and P_c instead. Notice that in $\mathcal{M}(P^{\rightarrow v})$ the matching and non-matching edges are exchanged in comparison to $\mathcal{M}(P)$ by definition of P. ■ Lemma 8

Lemma 9

Let path P and subpaths P_b and P_c be defined as in Lemma 8. The value $\beta(v)$ can be expressed from vertex b or c onwards by $\beta(b)$ or $\beta(c)$ in the following way:

(I) $$\beta(v) \; = \; m(P \setminus P_b) - m((P \setminus P_b)^{\rightarrow v}_{\leftarrow b}) \; + \beta(b)$$

(II) $$\beta(v) \; = \; m((P \setminus P_c)_{\leftarrow c}) - m((P \setminus P_c)^{\rightarrow v}) \; - \beta(c)$$

Proof.

(I) Matching $\mathcal{M}(P)$ can be divided into $\mathcal{M}(P \setminus P_b)$ and $\mathcal{M}(P_b)$ by Lemma 8. In the same way $\mathcal{M}(P^{\rightarrow v})$ can be divided into $\mathcal{M}((P \setminus P_b)^{\rightarrow v}_{\leftarrow b})$ and $\mathcal{M}(P_b{}^{\rightarrow b})$. Then it holds:

$$\begin{aligned}\beta(v) &= m(P) - m(P^{\rightarrow v}) \\ &= m(P \setminus P_b) + m(P_b) - [m((P \setminus P_b)^{\rightarrow v}_{\leftarrow b}) + m(P_b{}^{\rightarrow b})] \\ &= m(P \setminus P_b) - m((P \setminus P_b)^{\rightarrow v}_{\leftarrow b}) + \underbrace{m(P_b) - m(P_b{}^{\rightarrow b})}_{\beta(b)}\end{aligned}$$

(II) Matching $\mathcal{M}(P)$ can be divided into $\mathcal{M}((P \setminus P_c)_{\leftarrow c})$ and $\mathcal{M}(P_c{}^{\rightarrow c})$ by Lemma 8. In the same way $\mathcal{M}(P^{\rightarrow v})$ can be divided into $\mathcal{M}((P \setminus P_c)^{\rightarrow v})$ and $\mathcal{M}(P_c)$. Then it holds:

$$\begin{aligned}\beta(v) &= m(P) - m(P^{\rightarrow v}) \\ &= m((P \setminus P_c)_{\leftarrow c}) + m(P_c{}^{\rightarrow c}) - [m((P \setminus P_c)^{\rightarrow v}) + m(P_c)] \\ &= m((P \setminus P_c)_{\leftarrow c}) + m((P \setminus P_c)^{\rightarrow v}) - \underbrace{[m(P_c) - m(P_c{}^{\rightarrow c})]}_{\beta(c)}\end{aligned}$$

■ Lemma 9

Lemma 10

$$\forall\ \{v,u\} \in \mathcal{M}(G):\ w(\{v,u\}) \geq \beta(v)$$

Proof. The Lemma follows from Lemma 9 part (II) with the subpath P_c starts with vertex u. Then $m((P \setminus P_u)_{\leftarrow u}) = w(\{v,u\})$, and $(P \setminus P_u)^{\rightarrow v}$ is the empty graph with $m((P \setminus P_u)^{\rightarrow v}) = 0$. Then:

$$\beta(v) \ = \ w(\{v,u\}) - \beta(u)\ ,$$

and the application of Lemma 7 ($\beta(u) \geq 0$) completes the proof. ■ Lemma 10

Lemma 11

Let $G = (V, E, w)$ be a simple, undirected, weighted graph, $v \in V$, and let $\beta(v)$ correspond to a path P of even *length. When a manipulation of G increases the length of P, the value of $\beta(v)$ will never decrease, i. e.*

$$\Delta\beta(v) \geq 0\ .$$

Proof. Suppose b to be the last vertex of path P and P_b be the extension of P. We can use Lemma 9 part (I) because P has an even length. We then achieve:

$$\begin{aligned}\beta^+(v) &= m(P^+ \setminus P_b) - m(P^+ \setminus P_b \overset{\rightarrow v}{\leftarrow b}) + \beta^+(b) \\ &= m(P) - m(P^{\rightarrow v}) + \beta^+(b) \\ &= \beta(v) + \beta^+(b) \ .\end{aligned}$$

Then

$$\begin{aligned}\Delta\beta(v) &= \beta^+(v) - \beta(v) \\ &= \beta(v) + \beta^+(b) - \beta(v) \\ &= \beta^+(b)\end{aligned}$$

and by applying of Lemma 7:

$$\Delta\beta(v) \ = \ \beta^+(b) \ \geq 0 \ .$$

■ Lemma 11

Lemma 12

Let $G = (V, E, w)$ be a simple, undirected, weighted graph, $v \in V$, and let $\beta(v)$ correspond to a path P. When a manipulation on G decreases the length of P such that $\beta^+(v)$ corresponds to a path of odd *length, then the value of $\beta(v)$ will never decrease, i. e.*

$$\Delta\beta(v) \geq 0 \ .$$

Proof. Suppose c to be the last vertex of the reduced path P^+ (corresponding to $\beta^+(v)$) and $P_c{}^{\rightarrow c}$ be the shortening itself. We can use Lemma 9 part (II) because P^+ has an odd length, and we achieve:

$$\begin{aligned}\beta(v) &= m((P \setminus P_c)_{\leftarrow c}) - m((P \setminus P_c)^{\rightarrow v}) - \beta(c) \\ &= m(P^+) - m(P^{+\rightarrow v}) - \beta(c) \\ &= \beta^+(v) - \beta(c) \ .\end{aligned}$$

Then

$$\begin{aligned}\Delta\beta(v) &= \beta^+(v) - \beta(v) \\ &= \beta(v) + \beta(c) - \beta(v) \\ &= \beta(c)\end{aligned}$$

and by applying of Lemma 7:

$$\Delta\beta(v) \;=\; \beta(c) \;\geq 0 \;\;.$$

■ Lemma 12

Now we are able to give two key lemmata, which are needed to prove the upper bound. The statements of these lemmata are the same as in [BR93] and there are called identically. The line of argumentation in our proof follows E. Bernstein's and S. Rajagopalan's proof, too.

The Key Lemmata and the Proof

Lemma 13 (Stability Lemma)

The value of a server vertex never decreases, i. e.

$$\Delta\beta(s) \;\geq\; 0 \qquad \forall s \in S_i \;\;.$$

Proof. When wLMM is running the following changes on the local graph B can happen:

Case 1 'matching': A matching edge $\{r, s_i\} \in \mathcal{M}(B_i)$ is added to the online matching, $\{r, s_i\} \in M_{\mathsf{wLMM}}$. This edge including its both vertices is removed from B_i :

$$B_i^+ = B_i \,^{\rightarrow r, s_i} \;\;.$$

Case 2 'non-matching': The current server vertex s_i is not matched ($s_i \mathring{\notin} \mathcal{M}(B_i)$ and $s_i \mathring{\notin} M_{\mathsf{wLMM}}$) and is removed from B_i :

$$B_i^+ = B_i \,^{\rightarrow s_i} \;\;.$$

This can happen when the weights of edges incident to s_i are too small.

Case 3 'input': A new request vertex r_i is added to B_i :

$$B_i^+ = B_i \,_{\leftarrow r_i} \;\;.$$

We will show that in all three cases the value $\beta(s)$ of an arbitrary server vertex $s \in S_i$ will not decrease. Let Q be the path corresponding to $\beta(s)$. Whenever Q is not affected by one of the above cases, it holds: $\beta^+(s) = \beta(s)$ and $\Delta\beta(s) \geq 0$. Henceforth, it is sufficient to assume a modification in the structure of Q.

Case 1 ($\{r, s_i\} \in \mathcal{M}(B_i)$)**:** Path Q is shortened by removing one of its matching edges. This removal includes all adjacent non-matching edges. Hence, the shortened path Q^+ has a matching edge on both of its ends and therefore it is of *odd* length. An application of Lemma 12 gives $\Delta\beta(s) \geq 0$.

Case 2 ($s_i \overset{\circ}{\notin} \mathcal{M}(B_i)$)**:** The path Q is shortened by removing its last vertex s_i (all other vertices of Q are matched in $\mathcal{M}(B_i)$). Again the removing of s_i removes all incident edges and Q^+ has two matching edges on its ends. So it is of *odd* length, and by Lemma 12 it holds $\Delta\beta(s) \geq 0$.

Case 3: At first we focus on the possibility that the new request vertex is not matched, $r_i \overset{\circ}{\notin} \mathcal{M}(B_i)$. However, such a situation does not change any path Q and therefore it holds $\forall s \in S_i : \Delta\beta(s) = 0$. The reason is that r_i could be the last, non-matched vertex of Q^+ only. Then Q^+ has an even length. By definition, Q starts with a server vertex $s \in S_i$ which implies that all request vertices are situated in an odd distance to s. This fact implies a contradiction.
Henceforth, we assume that r_i is matched to a server vertex s_j in the enlarged graph $B^+ = B_{\leftarrow r_i}$ ($\{r_i, s_j\} \in \mathcal{M}(B^+)$). Then a path P was augmented which is described exactly by $\mathcal{M}(B_{\leftarrow r_i}) \oplus \mathcal{M}(B)$. We define the weight of path P just as the value of a server vertex[7]:

$$\beta'(r_i) := m(B_{\leftarrow r_i}) - m(B) > 0 .$$

Now we have to distinguish whether s_j was matched in $\mathcal{M}(B)$ before.

Case 3.a: $s_j \overset{\circ}{\notin} \mathcal{M}(B)$. No path Q of *odd* length could become extended because these paths have two matched end vertices. This is a contradiction to the assumption of this case. If Q is of *even* length and has been extended, then Lemma 11 can be applied and this results in $\Delta\beta(s) \geq 0$.

Case 3.b: $s_j \overset{\circ}{\in} \mathcal{M}(B)$. The following statements are implicated by the preconditions of this case:

- P is an augmenting path of length > 1 because s_j was matched before.
- There exists a vertex v which is the first common vertex of path P and Q. At that point, the paths meet first time with respect to their start vertices.

From this facts we can conclude:

- $v \in S_i$; otherwise v would be matched to a vertex of Q which is not in P *and* to another vertex of P. This is a contradiction.

[7]Note: the definition of $\beta(s)$ was made for server vertices $s \in S$ only

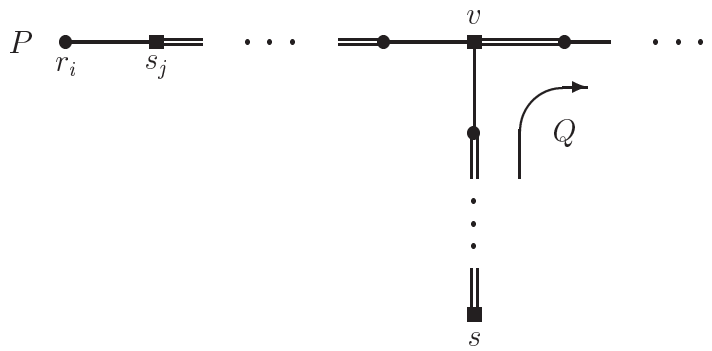


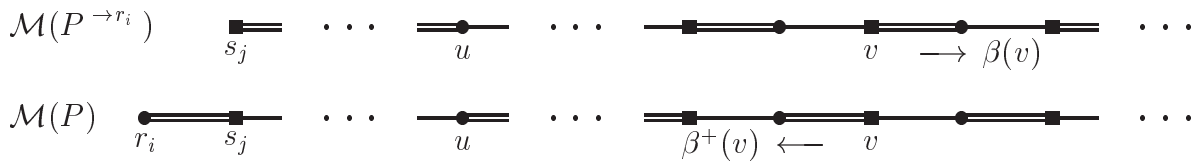
Figure 10: Situation just before P is augmented.

- From vertex v onwards, paths P and Q are identical. Lemma 9 part (I) gives the reason for this statement[8], since $\beta(v)$ is the same for both paths.

Lemma 9 part (I) allows us to express $\beta(s)$ by $\beta(v)$. Due to the fact that no matching edge is changed between s and v on path Q (see also Fig. 10) it holds:

$$\Delta\beta(s) = \Delta\beta(v) \ .$$

Hence, it is sufficient to investigate $\Delta\beta(v)$. We can observe that v is matched to the farther request vertex with respect to r_i, before path P is augmented. This augmentation exchanges all matching and non-matching edges on P. Then, v is matched to the request vertex that is closer to r_i and the path corresponding to $\beta^+(v)$ has the 'opposite direction', i. e. it starts at vertex v and ends at vertex u where u is situated between v and r_i.



This vertex u has to be a request vertex, $u \in R_i$. Otherwise it would be matched in $\mathcal{M}(P)$ outside the path described by $\beta^+(v)$ which is a contradiction to the definition of $\beta^+(v)$. Vertex u and r_i can be identical.

[8]Note: A very careful inspection shows the possibility that this statement does not necessarily hold. On the one hand $\beta(v)$ is an optimized value and it is equal in P and Q. On the other hand, there *can exist* two different paths P_{tail} and Q_{tail} which start at vertex v and have the same value $\beta(v)$. Without loss of generality, select the maximum matchings which define P such that Q_{tail} is the part of P form v onward. It is obvious that these matchings exist and that we get proper definitions for P, $\beta(s_j)$, and $\beta'(r_i)$ too.

The definition of $\Delta\beta(v)$ gives:

$$\begin{aligned}\Delta\beta(v) &= \beta^+(v) - \beta(v) \\ &= m(P) - m(P^{\to v}) - [m(P^{\to r_i}) - m(P^{\to r_i, v})] \\ &= \underbrace{m(P) - m(P^{\to r_i})}_{=\beta'(r_i)} - [m(P^{\to v}) - m(P^{\to r_i, v})] \end{aligned} \tag{2}$$

Claim:

$$m(P^{\to v}) - m(P^{\to r_i, v}) \;=\; \beta'(r_i) - \beta'(u) \;\; .$$

Substituting this claim in equation (2):

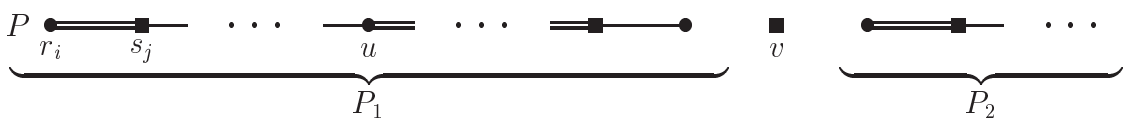
$$\Delta\beta(v) = \beta'(r_i) - [\beta'(r_i) - \beta'(u)] = \beta'(u) \;\; ,$$

and by Lemma 7 $\beta'(u) \geq 0$, which completes the proof. Lemma 7 can be applied here because it holds for the value $\beta(\cdot)$ of a vertex in a general, simple, undirected graph.

Proof of the claim:

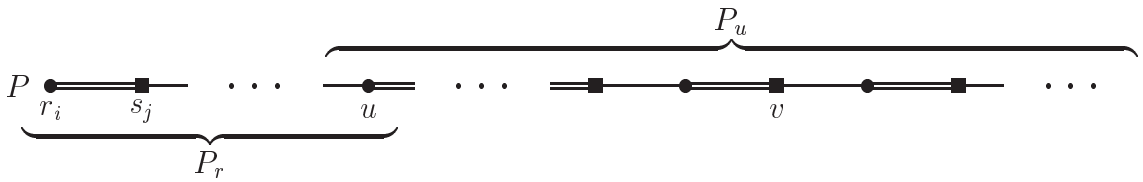
$$m(P^{\to v}) - m(P^{\to r_i, v}) \;=\; \beta'(r_i) - \beta'(u)$$

On the left hand side, the vertex v is removed out of path P in both terms. Hence, P is divided into two subpaths $P_1 \,\dot\cup\, \{v\} \,\dot\cup\, P_2$ whereas r_i is situated in P_1.



The maximum weighted matching of P_2 is the same in $\mathcal{M}(P^{\to v})$ and $\mathcal{M}(P^{\to r_i, v})$, and it has no influence on the difference $m(P^{\to v}) - m(P^{\to r_i, v})$.

Let P_r be a subpath between r_i and u, and let P_u be the subpath of P from u onward.



By definition of $\beta'(r_i)$ and Lemma 8:

$$\begin{aligned}\beta'(r_i) &= m(P) - m(P^{\to r_i}) \\ &= m(P_r) + m(P_u) - [m(P_r^{\to r_i}) + m(P_u^{\to u})] \\ &= \underbrace{m(P_r) - m(P_r^{\to r_i})}_{m(P^{\to v}) - m(P^{\to r_i, v})} + \underbrace{m(P_u) - m(P_u^{\to u})}_{\beta'(u)} \end{aligned}$$

To establish the statement $m(P_r) - m(P_r{}^{\rightarrow r_i}) = m(P^{\rightarrow v}) - m(P^{\rightarrow r_i,v})$, it is sufficient to have a look at P_1. When P was augmented (from $\mathcal{M}(P^{\rightarrow r_i})$ to $\mathcal{M}(P)$) all matching and non-matching edges were exchanged. After the removal of vertex v, path P_1 is augmented, and again all matching and non-matching edges are exchanged in the subpath between v and u which is described by $\beta^+(v)$. In this subpath we have the same situation as before when $r_i$had been inserted. Then the difference between $\mathcal{M}(P_1)$ and $\mathcal{M}(P_1{}^{\rightarrow r_i})$ can only be found in P_r (to be more precise: $\mathcal{M}(P_1) \oplus \mathcal{M}(P_1{}^{\rightarrow r_i}) = P_r$).
From the above equation, we get:

$$\begin{aligned} & & \beta'(r_i) &= m(P^{\rightarrow v}) - m(P^{\rightarrow r_i,v}) + \beta'(u) \\ \Leftrightarrow & & m(P^{\rightarrow v}) - m(P^{\rightarrow r_i,v}) &= \beta'(r_i) - \beta'(u) \end{aligned}$$

■ Lemma 13

Lemma 14 (Pay-Off - Lemma)

During a run of wLMM *holds:*

(I) $\Delta\Phi_{|s} \geq \beta(s)_{|s}$ $\qquad , s \in S$

(II) $\Delta\Phi_{|r} \geq w(\{r,s\}) - \beta(s)_{|r} \qquad \forall s : \{r,s\} \in E \qquad , r \in R$

Before proving the lemma we would like to present an interpretation of the formulae. Statement (I) establishes the fact that the potential function Φ increases by at least the value of $\beta(s)$ when the server vertex s is processed. Statement (II) ensures the choice of the heaviest augmenting path when a request vertex r is inserted and matched. The left hand side of this inequation is the weight of the selected augmenting path whereas the right hand side describes the weight of all possible augmenting paths that start in vertex r (see Lemma 9 part (II)).

Proof. For simplicity of notation the index i is omitted.

(I) Case 1: 'matching', $s \mathring{\in} M$. Then

$$\begin{aligned} M^+ &= M \cup \{r,s\} \\ \Rightarrow |M^+| &= |M| + w(\{r,s\}) \ , \end{aligned}$$

and

$$\begin{aligned} B^+ &= B^{\rightarrow r,s} \\ \Rightarrow m(B^+) &= m(B) - w(\{r,s\}) \ . \end{aligned}$$

Substituting the above equations into the definition of $\Delta\Phi_{|s}$:

$$\begin{aligned}\Delta\Phi_{|s} &= 2\,|M^+| + m(B^+) - [2\,|M| + m(B)] \\ &= 2\,|M| + 2w(\{r,s\}) + m(B) - w(\{r,s\}) - 2\,|M| - m(B) \\ &= w(\{r,s\})\end{aligned}$$

and by Lemma 10:

$$\Delta\Phi_{|s} = w(\{r,s\}) \;\geq\; \beta(s)_{|s} \;\; .$$

Case 2: 'non-matching', $s \mathring{\notin} M$. Then

$$\begin{aligned} M^+ &= M \\ \Rightarrow\; |M^+| &= |M| \;\; , \end{aligned}$$

and

$$\begin{aligned} B^+ &= B^{\rightarrow s} \\ \Rightarrow\; m(B^+) &= m(B) \;\; . \end{aligned}$$

Substituting the above equations into the definition of $\Delta\Phi_{|s}$:

$$\begin{aligned}\Delta\Phi_{|s} &= 2\,|M^+| + m(B^+) - [2\,|M| + m(B)] \\ &= 2\,|M| + m(B) - 2\,|M| - m(B) \\ &= 0\end{aligned}$$

Since, $s \mathring{\notin} \mathcal{M}(B)$:

$$\beta(s)_{|s} = m(B) - m(B^{\rightarrow s}) = 0 \;\; ,$$

and together we have:

$$\Delta\Phi_{|s} = 0 = \beta(s)_{|s} \;\; .$$

(II) (proof by contradiction)
Let $r \in R$ and $m(B^{\leftarrow r})$ be the overall weight of an optimal matching of B including vertex r.
Assumption: $\exists\, s \in S_i$ such that

$$\Delta\Phi_{|r} < w(\{r,s\}) - \beta(s)_{|r} \;\; .$$

The matching M is not changed by the insertion of r and we have:

$$\Delta\Phi_{|r} = m(B^{\leftarrow r}) - m(B) \;\; .$$

Substituting the assumption by this equation:

$$\begin{aligned} \Delta\Phi_{|r} = m(B^{\leftarrow r}) - m(B) &< w(\{r,s\}) - \beta(s)_{|r} = w(\{r,s\}) - [m(B) - m(B^{\rightarrow s})] \\ \Leftrightarrow \qquad m(B^{\leftarrow r}) &< w(\{r,s\}) + m(B^{\rightarrow s}) \end{aligned} \tag{3}$$

The term $w(\{r,s\}) + m(B^{\rightarrow s})$ is the weight of a matching in graph $B_{\leftarrow r}$ because both vertices r and s are not in $B^{\rightarrow s}$. This contradicts the assumed optimality of $m(B_{\leftarrow r})$ and the lemma is proven. ■ Lemma 14

Theorem 15

The deterministic online algorithm wLMM *for the* wORSM *problem is 2-competitive.*

Proof. For all request vertices $r \in R$ we get from Lemma 14 part (II):

$$\Delta\Phi_{|r} \;\geq\; w(\{r,s\}) - \beta(s)_{|r} \qquad \forall s \in S,\ \{r,s\} \in E_G \;\;, \tag{4}$$

while the definition of the wORSM problem ensures that r is processed before s .
We get from Lemma 14 part (I) for all server vertices $s \in S$:

$$\Delta\Phi_{|s} \;\geq\; \beta(s)_{|s} \;\;, \tag{5}$$

and by the use of Lemma 13, since s is processed after r :

$$\beta(s)_{|s} \;\geq\; \beta(s)_{|r} \;\;. \tag{6}$$

The sum of equations (4), (5) and (6) is

$$\Delta\Phi_{|r} + \Delta\Phi_{|s} \;\geq\; w(\{r,s\}) \;\;.$$

Then, for an *arbitrary* matching M_G of G holds:

$$\Phi_{\text{final}} = \sum_{r\in R} \Delta\Phi_{|r} + \sum_{s\in S} \Delta\Phi_{|s} \geq \sum_{\{r,s\}\in M_G} \Delta\Phi_{|r} + \Delta\Phi_{|s} \geq \sum_{\{r,s\}\in M_G} w(\{r,s\}) = |M_G| \;\;.$$

Hence, we get for M_{OPT} :

$$\Phi_{\text{final}} \;\geq\; |M_{\mathsf{OPT}}|$$

and from the definition of Φ, and the fact that $B_{\text{final}} = \varnothing$, it follows:

$$\Phi_{\text{final}} = 2\,|M_{\mathsf{wLMM}}| \;\geq\; |M_{\mathsf{OPT}}|$$

which shows the 2-competitiveness of wLMM. ■ Theorem 15

A Comment on the wLMM-Algorithm

Why does wLMM not prefer the current server vertex s_i in the online matching, like LMM does? The answer to that question is a very short one: It does not help anyway.
Suppose an algorithm wLMM* which prefers the current server vertex in the way like LMM does. Furthermore, assume an input graph $G = (V, E, w)$ such that $M_{\mathsf{wLMM}^*} \neq M_{\mathsf{wLMM}}$. Let s_i be the first vertex where wLMM* makes a different decision than wLMM ($s_i \mathring{\in} M_{\mathsf{wLMM}^*}$

and $s_i \mathring{\notin} M_{\mathsf{wLMM}}$). When decreasing all weights of edges incident to s_i by ε, wLMM^* will behave like wLMM on s_i. With the help of this trick, we can construct an input graph $G^* = (V, E, w^*)$ such that wLMM^* calculates the same matching as wLMM does on input G. The difference in the weights of the resulting online matchings $|M_{\mathsf{wLMM}^*}|$ and $|M_{\mathsf{wLMM}}|$ is small and disappears for $\varepsilon \to 0$.

Possibly, the next section shows a way out.

4.4 The Algorithm PHI

The algorithm wLMM implements a special greedy strategy. From its current point of view, it takes the maximal, additional weight for the online matching. Theorem 6 shows a situation where this strategy is not very clever. With regard to theorem 5 (construction of the general lower bound) and theorem 6 we suppose that there is an algorithmic improvement. Using the current vertex s_i in the online matching M, and therefore removing a vertex r out of B_i simultaneously, may be more valuable than the originated loss in $\mathcal{M}(B_i)$. This observation leads us to suggest another online algorithm for the wORSM problem. It works similarly to wLMM but after computing the local maximum weighted matching $\mathcal{M}(B_i)$, it checks the vertex s_i. Whenever $s_i \mathring{\notin} \mathcal{M}(B_i)$ holds, the weights of all edges of B_i incident to s_i will be increased by the factor $\phi = \frac{\sqrt{5}+1}{2}$. This new local bipartite graph is called B_i^ϕ. Now the algorithm determines $\mathcal{M}(B_i^\phi)$. Whenever $s_i \mathring{\in} \mathcal{M}(B_i)$ or $s_i \mathring{\in} \mathcal{M}(B_i^\phi)$ holds[9], its matching edge is added to the online matching M. The new online algorithm is called PHI and a formal description follows:

```
1:  i ⇐ 0
2:  loop
3:     i ⇐ i + 1
4:     read input of time i and build up B_i
5:     construct a maximum weighted matching M(B_i) on B_i
6:     if s_i ∉̊ M(B_i) then
7:        construct B_i^φ {increase the weight of every edge of B_i incident to s_i by factor (√5+1)/2}
8:        calculate the maximum weighted matching M(B_i^φ)
9:     end if
10:    if s_i ∈̊ M(B_i) or s_i ∈̊ M(B_i^φ) then
11:       add the matching edge of s_i to the online weighted matching M_PHI
12:    end if
13: end loop
```

A problem still unsolved is the performance analysis of PHI. It seems as if a modification of the technique, that is used in the proof of theorem 15 does not work. It is likely that

[9] $s_i \mathring{\in} \mathcal{M}(B_i)$ implies $s_i \mathring{\in} \mathcal{M}(B_i^\phi)$, then the algorithm does not need to calculate B_i^ϕ, $\mathcal{M}(B_i^\phi)$ and to test $s_i \mathring{\in} \mathcal{M}(B_i^\phi)$

these facts (the necessity of a modified algorithm like PHI and a new technique to analyse it) also explain the gap in the analysis of the roommates problem in [BR93].
It is worth noting that the algorithm LMM is a special implementation of PHI for unit edge weights. The increasing of unit weights of s_i-edges by a factor of ϕ is equivalent to the simple preference of an edge incident to s_i at time step i.

5 Open Problems

In the end of the last subsection we presented the most urgent research task: To close the gap between the lower and upper bound of the wORSM problem (Theorem 5 and Theorem 15). If we achieve this aim, the roommates problem (weighted version) of [BR93] should be revisited.

A completely different research task deals with the ORSM problem (unweighted version). Here, the performance on inputs of strong, pre-determined structures is of interest. Finally, we would like to model a set of parallel, homogeneous resources and requests with deadlines. First results have been already proven. It turned out that dependent on the concrete model:

- lower competitive ratios are achievable,
- the algorithm LMM is not optimal anymore,
- different arguments to prove upper bounds are necessary.

These investigations are not completed by now but they will be presented in a later publication.

References

[ANR95] Yossi Azar, Joseph Naor, and Raphael Rom. The competitiveness of on-line assignments. *Journal of Algorithms*, 18:221–237, 1995.

[BDBK+94] Shai Ben-David, Allan Borodin, Richard M. Karp, Gábor Tardos, and Avi Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1):2–14, 1994.

[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.

[ET75] Shimon Even and Robert Endre Tarjan. Network flow and testing graph connectivity. *SIAM Journal on Computing*, 4(4):507–518, December 1975.

[IK96] Sandy Irani and Anna R. Karlin. Online computation. In Dorit S. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, chapter 13, pages 521–564. PWS Publishing Company, Boston, Mass., 1996.

[Kar92] Richard M. Karp. On-line algorithms versus off-line algorithms: How much is it worth to know the future? In Jan van Leeuwen, editor, *Algorithms, Software, Architecture; Proceedings of the 12th IFIP World Computer Congress*, pages 416–429. Elsevier Science Publishers B.V. (North Holland), 1992.

[KMV94] Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, 127(2):255–267, May 1994.

[KP93] Bala Kalyanasundaram and Kirk Pruhs. Online weighted matching. *Journal of Algorithms*, 14(3):478–488, 1993.

[KP95] Bala Kalyanasundaram and Kirk Pruhs. An optimal deterministic algorithm for online b-matching. In Pazhamaneri S. Thiagarajan, editor, *Proceedings of the 15th Conference of Foundations of Software Technology and Theoretical Computer Science, (Bangalore, India, December 18–20, 1995)*, LNCS 1026, Berlin-Heidelberg-New York, 1995. Springer-Verlag.

[KT91] Ming-Yang Kao and Stephen R. Tate. Online matching with blocked input. *Information Processing Letters*, 38(3):113–116, May 1991.

[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 —STOC '90 (Baltimore, Maryland, May 14–16, 1990)*, pages 352–358, New York, 1990. ACM SIGACT, ACM Press.

[Sga98] Jiří Sgall. On-line scheduling — a survey. In Amos Fiat and Gerhard J. Woeginger, editors, *Dagstuhl Seminar on On-Line Algorithms (Schloß Dagstuhl (Wadern), Germany, June 24–28, 1996)*, to appear in LNCS. Springer-Verlag, Berlin-Heidelberg-New York, 1998.

[ST85] Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, February 1985.