# Deferred on-line bipartite matching

Jakub Kozik* Grzegorz Matecki†

Theoretical Computer Science
Faculty of Mathematics and Computer Science
Jagiellonian University in Kraków, Poland

Jakub.Kozik@tcs.uj.edu.pl Grzegorz.Matecki@tcs.uj.edu.pl

Submitted: Dec 3, 2015; Accepted: Apr 24, 2018; Published: May 11, 2018


**Abstract**

We present a new model for the problem of on-line matching on bipartite graphs. Suppose that one part of a graph is given, but the vertices of the other part are presented in an on-line fashion. In the classical version, each incoming vertex is either irrevocably matched to a vertex from the other part or stays unmatched forever. In our version, an algorithm is allowed to match the new vertex to a group of elements (possibly empty). Later on, the algorithm can decide to remove some vertices from the group and assign them to another (just presented) vertex, with the restriction that each element belongs to at most one group. We present an optimal (deterministic) algorithm for this problem and prove that its competitive ratio equals $1 - \pi/\cosh(\frac{\sqrt{3}}{2}\pi) \approx 0.588$.

**Mathematics Subject Classifications:** 68W27, 05C70, 05C85

## 1 Introduction

A number of task-server assignment problems can be modelled as finding a match in a bipartite graph $G = (U, D, E)$. The vertices of one part (set $D$) correspond to servers, and the vertices of the other part (set $U$) correspond to tasks. An edge between a task and a server indicates that the server is capable of performing the task. In a simple setting where a single server can perform at most one task, the problem of maximising the number of performed tasks is reduced to finding a maximum matching. However, in real-life applications, it is very common that not all tasks are known a priori, and some decisions about the assignments have to be made with no knowledge about the future tasks. A simple model for this situation is on-line bipartite matching. In this setting,

*Partially supported by the Polish National Science Centre UMO-2011/03/D/ST6/01370.
†Partially supported by the Polish National Science Centre 2011/03/B/ST6/01367.

the servers are known from the beginning, and the tasks are revealed one by one. The decision about the assignment of each task has to be made just after its arrival and cannot be changed in the future. Suppose that there are $k$ servers, and that $k$ tasks are going to be revealed. It is easy to show that it is possible to present the tasks in such a way that the constructed assignment matches at most $\lceil k/2 \rceil$ tasks, whereas it would be possible to match all the tasks if they were revealed all at once. On the other hand, any greedy strategy guarantees that at least half of the tasks will be assigned. These observations imply that the competitive ratio of the on-line bipartite matching problem equals $1/2$.

In their classical contribution, Karp, Vazirani and Vazirani [18] took an approach in which the graph to be presented is fixed before the first task is revealed. In particular, the graph does not depend on the decisions of the assigning algorithm. While the approach does not make any difference for the worst-case analysis of the algorithm, it provides a framework for analysing randomised algorithms. The authors presented a randomised algorithm, which constructs a matching with the expected competitive ratio of at least $1-1/e$ (the original paper [18] contained a mistake, which was corrected in [14]; see also a simplified exposition [5]). The result is the best possible among all randomised algorithms. The approach of [18] has been applied to many variants of the original problem, with numerous practical applications (the switch routing problem [1, 3], on-line auctions [19], the Adwords problem [9, 14, 21], etc.) Recently, the problem of on-line stochastic matching [4, 12, 17, 19, 20], where the competitive ratio can be greater than $1-\frac{1}{e}$, has attracted a deep interest. A different approach (called $b$-matching) is presented in [16], where the authors allowed a server to perform up to $b$ tasks at the same time. They showed an optimal deterministic algorithm with a competitive ratio of $1-\frac{1}{(1+\frac{1}{b})^b}$ (which tends to $1-\frac{1}{e}$ with $b \to \infty$).

It is not uncommon for the cost of a running server to be roughly the same as the cost of an idle one. Consequently, it may be profitable to perform some task on many servers simultaneously. In order to capture this situation, we allow an algorithm to assign more than one server to an incoming task, and later in the future, to forgot some calculations and switch the freed servers over to new tasks. One server is assumed to be enough to complete a task. In this model, called *deferred* (on-line) matching, an algorithm can improve performance by delaying some decisions concerning which server should accomplish which task. We present an algorithm that always matches at least $(1-\pi/\cosh(\frac{\sqrt{3}}{2}\pi))n + o(n) \approx 0.588n + o(n)$ vertices, when $n$ is the size of the maximum matching in the presented graph.

This type of approach was first introduced by Felsner in [13] as an *adaptive* generalisation of the on-line chain partitioning problem. In this problem, an algorithm has to partition into chains a partial order whose vertices are presented in an on-line manner. Felsner proposed a variant in which each incoming vertex can be initially assigned to several chains and later withdrawn from some of these chains, unless only one chain remains. This modification gives more freedom to a partitioning algorithm but, the classical behaviour whereby just one chain is used for each new vertex is still allowed. Interestingly, as reported in [6], no adaptive algorithm has been proved to essentially outperform the best non-adaptive algorithm and, in both cases, the best lower bounds for the number

of used chains are roughly twice the width of the presented partial order. It seems challenging to verify whenever the adaptive (deferred) approach to chain partitioning enables more efficient on-line algorithms.

### 1.1 Further related work

The problem of $b$-matching [16] seems similar to deferred matching in which the roles of servers and tasks are switched. However, in the deferred matching, an algorithm always ends up with each server performing at most one task, while $b$-matching allows a server to perform many tasks.

Another similar approach is proposed by Feldman et al. [11] as *free disposal*. They consider the weighted matching problem, in which each incoming vertex (server) $u \in U$ can be assigned to one of its neighbours (in $D$) or left alone. All vertices of $D$ (tasks) are given in advance (the roles of servers and tasks are switched). Tasks are allowed to be assigned to servers multiple times. In the end, each task $d \in D$ chooses at most $n(d)$ servers from all the vertices of $U$ assigned to it – the ones with the highest-weighted edge. The numbers $n(d)$ (for $d \in D$) are given in advance and can be interpreted as the maximum number of times the realization of the task brings a profit. The main difference from the deferred matching is that once a connection between a server and a task is established, it cannot be changed until the very end. In deferred matching, a server may drop its task and take on a new one during the on-line process.

The idea of dropping an edge from already-constructed matching is investigated in the *pre-emptive model*. Here, edges with weights are incoming on-line, and an algorithm is allowed to remove previously accepted edges in order to add a new one. A collection of results from pre-emptive matching can be found in [8, 10].

An approach where a task (with a given weight) can be assigned to multiple servers and, therefore, each server can run many tasks was already studied in [2]. The goal here is to minimise the maximum load (the sum of the weights of all tasks assigned to a server) of all servers. Furthermore, assignment decisions may not be changed in the future. The authors prove that the best competitive ratio is $O(\log n)$, where $n$ is the number of servers. In deferred matching, we are not interested in the load of each server that may change (it can only decrease) during the process (assuming that weights are equal to 1). However, we explicitly forbid the load of a server to be greater than $\alpha$, and our results depend on $\alpha$.

In most papers about on-line bipartite matching, decisions made by an algorithm are irrevocable. Some authors allow a limited number of reassignments (see [7, 15]), and analyse the strategies that minimise the reassignments.

### 1.2 Problem definition

For a positive integer $\alpha$, the *deferred $\alpha$-matching game* is played in rounds between the Scheduler and the Builder. They play on a bipartite graph, say $G = (U, D, E)$. The set of vertices $D$ is known in advance, and set $U$ (with its neighbouring edges) is revealed step-by-step by the Builder within a finite number of rounds. Each round consists of three steps:

(R1) Builder presents a vertex $u \in U$ and reveals all its neighbours $N(u) = \{d \in D : (u, d) \in E\} \subseteq D$.

(R2) Scheduler assigns to $u$ a set $m(u) \subseteq N(u)$ with a maximum size of $\alpha$.

(R3) Scheduler updates $m(x) := m(x) \setminus m(u)$ for every vertex $x$ presented before.

The *size* of the game, denoted by $n$, is the maximum size of a matching in $G$. The final function $m$ is called *the multi-match* constructed by the Scheduler. We allow $\alpha$ to amount to infinity, in which case the game is called *deferred $\infty$-matching* or simply *deferred matching*.

The goal of the Scheduler is to maximise the number $k$ of non-empty sets $m(u)$ over all vertices $u \in U$. Intuitively, every such vertex $u$ can be successfully matched with an arbitrarily chosen $d \in m(u)$. The number $k$ denotes *the size of the multi-match constructed by the Scheduler*. The goal of the Builder is to make $k$ as small as possible.

The interpretation of the deferred $\alpha$-matching game in terms of servers-tasks assignment is clear: $D$ is the set of servers, $U$ is the set of incoming tasks. An algorithm assigns the servers $m(u)$ to an incoming $u$ (possibly cancelling the previous computations of the servers in $m(u)$). The performance of the Scheduler is measured through the number (or the fraction) of the tasks are performed at the end of the game. Note that for $\alpha = 1$, the game is reduced to the classical on-line bipartite matching.

Let $\mathcal{A}$ be the assigning algorithm. We denote by $\mathrm{val}_{\mathcal{A}}(n)$ the worst-case value of the size of the multi-match constructed by $\mathcal{A}$ in all possible games of size $n$. The value of the deferred $\alpha$-matching problem $\mathrm{val}_{\alpha}(n)$ is the maximum value of $\mathrm{val}_{\mathcal{A}}(n)$ among all $\alpha$-assigning algorithms $\mathcal{A}$. Since no algorithm produces a multi-match larger than $n$, we additionally use the *competitive ratio* defined as $\liminf_{n\to\infty} \mathrm{val}_{\alpha}(n)/n$. Similarly, *the competitive ratio of the algorithm* $\mathcal{A}$ is $\liminf_{n\to\infty} \mathrm{val}_{\mathcal{A}}(n)/n$.

### 1.3 Main results

To solve the problem of on-line deferred $\alpha$-matching we consider a deterministic algorithm called $\alpha$-BALANCED. In one round, for the presented vertex $u$, the algorithm successively enlarges the new matching set $m(u)$, possibly diminishing previously mapped sets, as long as $|m(u)|$ does not exceed the size of any diminished sets. It is greedy in the sense that no task is rejected as long as it is possible to perform. The main idea of the algorithm is to locally balance the sizes of the assigned sets. The details of the algorithm are described in Section 3. The main result of our work is the proof that $\alpha$-BALANCED is the best possible algorithm.

**Theorem 1.** *$\alpha$-BALANCED is an optimal strategy for the Scheduler in the deferred $\alpha$-matching game.*

The proof is split into two parts, described in the two subsequent sections. The

following schema of a system of inequalities is crucial for both arguments:

$$\begin{cases} (1+\alpha)x_0 \leqslant n, \\ (x_0+\cdots+x_i)(1+x_i) \leqslant n-i, \quad i=1,\ldots,k, \\ x_1 \geqslant x_2 \geqslant \cdots \geqslant x_k \geqslant 0, \\ x_0+\cdots+x_k \geqslant 0. \end{cases} \tag{1}$$

We will operate on fixed $n$ and $\alpha$. Thus, the schema is parametrised by a positive integer $k$. We say that a pair $(k,x)$ satisfies system (1) if $x=(x_0,x_1,\ldots,x_k)$ is an integer vector satisfying the instance of the schema for this particular $k$.

In Section 2, we prove (Proposition 4) that for every solution $(k,x)$ of (1), every deferred $\alpha$-matching algorithm $\mathcal{A}$ can be limited by the Builder's strategy in a game of the size $n$, so that $\mathcal{A}$ matches at most $n-(x_0+\cdots+x_k)$ vertices. On the other hand, in Section 3, we show (Proposition 5) that when $\alpha$-BALANCED constructs a multi-match of size $k$ in a game of size $n$, then $k=n-(x_0+\cdots+x_k)$ for some $(k,x)$ satisfying (1). These two facts ensure that $\alpha$-BALANCED is an optimal strategy for Scheduler.

In order to determine the competitive ratio of $\alpha$-BALANCED, we need to maximise the sum $x_0+\cdots+x_k$ over all feasible solutions $(k,x)$ of (1). In Section 4 we present and solve a linear programming formulation of (1). Finally, we prove

**Theorem 2.** *The competitive ratio of the deferred on-line $\alpha$-matching problem on bipartite graphs equals $1-\frac{\alpha}{1+\alpha}\prod_{i=1}^{\alpha-1}\frac{i+i^2}{1+i+i^2}$. For $\alpha\to\infty$, it converges to $1-\pi/\cosh\frac{\sqrt{3}\pi}{2}\approx 0.588$.*

The ratio converges fast, we obtain $5/9\approx 0.556$ and $4/7\approx 0.571$ for $\alpha=2$ and $\alpha=3$, respectively.

## 2 Worst case scenario

Inequalities (1) allow $x_0$ to be negative. We start with an observation that in order to maximise the sum $x_0+\cdots+x_k$, it suffices to consider the solutions of (1) with $x_0=\left\lfloor\frac{n}{1+\alpha}\right\rfloor$.

**Proposition 3.** *For any pair $(k,x)$ satisfying (1), there exists a pair $(k,x')$ satisfying (1), such that $x'_0=\left\lfloor\frac{n}{1+\alpha}\right\rfloor$ and $x'_0+\cdots+x'_k\geqslant x_0+\cdots+x_k$.*

*Proof.* For the case in which $x_1=0$, take $x'_0=\left\lfloor\frac{n}{1+\alpha}\right\rfloor$, $x'_1=\cdots=x'_k=0$ to observe that $x_0+\cdots+x_k=x_0\leqslant\left\lfloor\frac{n}{1+\alpha}\right\rfloor=x'_0+\cdots+x'_k$.

From now on, we shall assume that $x_1>0$. The proposition is proved by induction on the number $m=\left\lfloor\frac{n}{1+\alpha}\right\rfloor-x_0$. The base case of $m=0$ (where $x_0=\left\lfloor\frac{n}{1+\alpha}\right\rfloor$) is immediate, as we can simply take $x'=x$. For the induction step, let $m>0$. It is equivalent to $x_0<\left\lfloor\frac{n}{1+\alpha}\right\rfloor$. Let $j$ be the highest index for which $x_j=x_1$. Then, consider the following sequence $(\bar{x}_0,\ldots,\bar{x}_k)$: $\bar{x}_0=x_0+1$, $\bar{x}_j=x_j-1$ and $\bar{x}_i=x_i$ for all $i\notin\{0,j\}$. First, we need to test whether the pair $(k,\bar{x})$ satisfies (1). The condition $m>0$ implies that

$$\bar{x}_0=x_0+1\leqslant\left\lfloor\frac{n}{1+\alpha}\right\rfloor. \tag{2}$$

The definition of $j$ guarantees that $x_j > x_{j+1}$ as long as $j < k$. Therefore, $\bar{x}_j \geqslant \bar{x}_{j+1}$, and indeed,

$$\bar{x}_1 \geqslant \ldots \geqslant \bar{x}_k. \tag{3}$$

For all $i \geqslant j$, we have $\bar{x}_0 + \cdots + \bar{x}_i = x_0 + \cdots + x_i$, and thus

$$(\bar{x}_0 + \cdots + \bar{x}_i)(1 + \bar{x}_i) \leqslant (x_0 + \cdots + x_i)(1 + x_i) \leqslant n - i, \tag{4}$$

and

$$\bar{x}_0 + \cdots + \bar{x}_k = x_0 + \cdots + x_k \geqslant 0. \tag{5}$$

For $0 < i < j$, we arrive at

$$\begin{aligned}(\bar{x}_0 + \cdots + \bar{x}_i)(1 + \bar{x}_i) = (x_0 + \cdots + x_i + 1)(1 + x_j) \leqslant \\ \leqslant (x_0 + \cdots + x_{j-1} + x_j)(1 + x_j) \leqslant \\ \leqslant n - j < n - i.\end{aligned} \tag{6}$$

Inequalities (2)-(6) imply that $(k, \bar{x})$ does, indeed, satisfy (1). Next, observe that for $(k, \bar{x})$ there is a pair $(k, x')$ satisfying (1) such that $x'_0 = \left\lfloor \frac{n}{1+\alpha} \right\rfloor$ and $x'_0 + \cdots + x'_k \geqslant \bar{x}_0 + \cdots + \bar{x}_k$. It has already been proved for $\bar{x}_1 = 0$, and the case $\bar{x}_1 > 0$ is given through the induction hypothesis, since $0 \leqslant \left\lfloor \frac{n}{1+\alpha} \right\rfloor - \bar{x}_0 < m$. Finally, with (5), we obtain $x'_0 + \cdots + x'_k \geqslant x_0 + \cdots + x_k$, and through the induction hypothesis, we end the proof. □

**Proposition 4.** *For any pair $(k, x)$ satisfying (1), there exists a strategy for the Builder in the deferred $\alpha$-matching game of size $n$ such that any Scheduler constructs a multi-match with a size of at most $n - (x_0 + \cdots + x_k)$.*

*Proof.* Assume for a moment that the proposition is true whenever $x_0 \geqslant 0$. For the case $x_0 < 0$, by Proposition 3, there is a pair $(k, x')$, such that $x'_0 \geqslant 0$ and

$$n - (x'_0 + \cdots + x'_k) \leqslant n - (x_0 + \cdots + x_k). \tag{7}$$

Thus, for $(k, x')$, we obtain a Builder's strategy $S$ that force the Scheduler to use at most $n - (x'_0 + \cdots + x'_k)$ vertices for its match mapping. By (7), we can use the strategy $S$ for the pair $(x, k)$ with $x_0 < 0$.

The above argument shows that it is enough to consider pairs with $x_0 \geqslant 0$. Without a loss of generality, we assume that $x_k > 0$ and describe a strategy for the Builder that does not allow the Scheduler to construct a multi-match larger than $n - (x_0 + \cdots + x_k)$. During the game, the Builder presents a bipartite graph $G = (U, D, E)$ with $|U| = |D| = n$ and maintains an auxiliary structure: a partition of $U = U_0 \cup U_1 \cup \ldots \cup U_k \cup R$ and a partition of $D = D_0 \cup D_1 \cup \ldots \cup D_k \cup S$, such that

$$\begin{aligned}|U_0| = |D_0| &= x_0, \\ |U_i| = |D_i| &= 1 + x_i, \qquad \text{for } i = 1, \ldots, k, \\ N(u_i) &= D - (D_0 \cup \ldots \cup D_{i-1}), \qquad \text{for each } u_i \in U_i, \qquad (8) \\ N(r) &= S, \qquad \text{for each } r \in R. \qquad (9)\end{aligned}$$

Note that (1) guarantees that $x_0 + \cdots + x_k \leqslant n - k$ and thus $\sum_{i=0}^{k} |U_i| = \sum_{i=0}^{k} |D_i| \leqslant x_0 + \sum_{i=1}^{k}(1 + x_i) \leqslant n$. Therefore, $|R| = |S| \geqslant 0$. It is clear that any bipartite graph that can be partitioned in such manner contains a perfect matching.

The strategy of the Builder is divided into $k + 2$ phases enumerated from 0 to $k + 1$. Figure 1 depicts the evolution of the strategy described below for $\alpha = 2$, $n = 18$ and $k = 2$. In the beginning of the $i$-th phase ($0 \leqslant i \leqslant k$), the sets $U_j$ and $D_j$, for $j < i$ are already fixed. Next, during the $i$-th phase, the Builder presents $1 + x_i$ vertices (or $x_0$ vertices when $i = 0$), which form the set $U_i$ with their neighbourhoods defined by (8). Then, the Builder chooses in a special manner the set $D_i \subseteq D - (D_0 \cup \ldots D_{i-1})$ with a size of $1 + x_i$ (or a size of $x_0$ if $i = 0$). This concludes phase $i$. In the last phase of the game (i.e. phase $k+1$), the Builder presents a set $R$ with a size of $n - k - (x_0 + \cdots + x_k)$ with vertices adjacent to all vertices in $S = D - \bigcup_{i=0}^{k} D_i$.

We now only need to define the Builder's choice of $D_i$. For that, after each phase $i$ ($0 \leqslant i \leqslant k$), the Builder maintains the following:

($\star$) there are $i$ distinct vertices $y_1, \ldots, y_i \in U_0 \cup \ldots \cup U_i$ such that $m(u) \cap \bigcup_{j=0}^{i} D_j = \emptyset$ for all $u \in (U_0 \cup \ldots \cup U_i) \setminus \{y_1, \ldots, y_i\}$.

Observe that, for a fixed $X \subseteq D$ and a fixed $u \in U$, once the condition $X \cap m(u) = \emptyset$ is satisfied, it will stay so to the end of the game, as $m(u)$ may only shrink later on in the game. We prove by induction that choosing such $D_i$ is possible in every phase by successively finding the correct $y_i$'s.

For $i = 0$, the Builder chooses[1] any $D_0 \subseteq D - m(U_0)$ with a size of $x_0$, which is possible since $|D - m(U_0)| \geqslant n - \alpha x_0 \geqslant x_0$ (because $|m(u)| \leqslant \alpha$ for all $u \in U_0$, and through (1)). Thus, $D_0 \cap m(u) = \emptyset$ for all $u \in U_0$, as required.

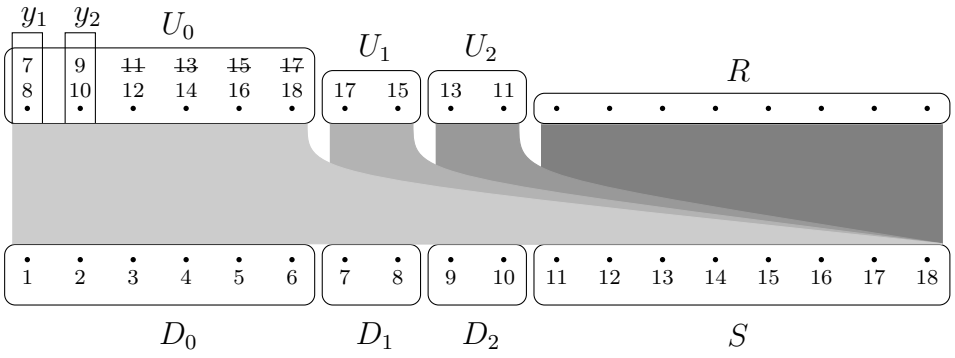


Figure 1: Sample construction of the solution $x = (6, 1, 1)$ of (1) with $n = 18$ and $\alpha = 2$. Note that $N(U_0) = D$, $N(U_1) = D - D_0$, $N(U_2) = D_2 \cup S$ and $N(R) = S$. The numbers under the vertices of the bottom part represent their names. The numbers in the upper part correspond to sets $m(u)$ (e.g. $m(y_1) = \{7, 8\}$).

For $1 \leqslant i \leqslant k$, we assume that ($\star$) holds for the vertices $Y = \{y_1, \ldots, y_{i-1}\}$ after the $(i-1)$-th phase. Let $U' = U_0 \cup \ldots \cup U_i$ and $D' = D - (D_0 \cup \ldots \cup D_{i-1})$. First, note

[1] Let $m(X) = \bigcup_{x \in X} m(x)$ for a set $X$.

that $m(U'-Y) \subseteq D'$ through the $(\star)$-property after the $(i-1)$-th phase. We split the vertices in $D'$ into two (disjointed) blocks: $D' = m(U'-Y) \cup X$, where $X$ is simply the set of all unmatched vertices in $D'$. Next, with $|U'| = x_0 + \cdots + x_i + i$, $|Y| = i-1$ and $|D'| = n - (x_0 + \cdots + x_{i-1} + i - 1)$, we set a lower bound on the average size of $m(u)$ for $u \in U' - Y$:

$$\begin{aligned}\frac{|m(U'-Y)|}{|U'-Y|} = \frac{|D'|-|X|}{|U'|-|Y|} &= \frac{n-i-|X|-(x_0+\cdots+x_{i-1}-1)}{x_0+\cdots+x_i+1} \\ &\overset{(1)}{\geqslant} \frac{(x_0+\cdots+x_i)(1+x_i)-|X|-(x_0+\cdots+x_{i-1}-1)}{x_0+\cdots+x_i+1} \\ &= x_i + \frac{1-|X|}{x_0+\cdots+x_i+1} > x_i - |X|\,.\end{aligned}$$

Clearly, there must exist a vertex $y_i \in U' - Y$ with $|m(y_i)| \geqslant \frac{|m(U'-Y)|}{|U'-Y|} > x_i - |X|$. Since all of the values are integers, we have $|X| + |m(y_i)| \geqslant 1 + x_i$. The Builder picks for $D_i$ any subset of $X \cup m(y_i)$ with a size of $1 + x_i$, and hence keeps the property $(\star)$ satisfied after the $i$-th phase.

Based on the condition $(\star)$, after the $k$-th phase, there is $Y = \{y_1, \ldots, y_k\}$ such that $D - S$ and $m(u)$ are disjoint for all $u \in U - Y$. Therefore, whenever $m(u) \neq \emptyset$, for $u \in U$, then either $u \in Y$ or $m(u) \subseteq S$. This means that the number of such $u$'s is at most $|Y| + |S| = k + n - (|D_0| + \cdots + |D_k|) = n - (x_0 + \cdots + x_k)$. Consequently, the size of the multi-match produced by the Scheduler is at most $n - (x_0 + \cdots + x_k)$. □

## 3 The best matching algorithm

To describe the main algorithm, we shall introduce certain useful definitions. Consider a deferred $\alpha$-matching algorithm. Its (partial) multi-match produced at the end of round $t$ is denoted by $m^t : U \to \mathcal{P}(D)$. In particular, for the $u \in U$ that is presented in round $t_0$, we have $m^t(u) = \emptyset$ for $t < t_0$, and subsequently $(m^{t_0}(u), m^{t_0+1}(u), \ldots, m^T(u))$ is a weakly decreasing (see rule ((R3))) sequence of sets, i.e. $m^{t_0}(u) \supseteq m^{t_0+1}(u) \supseteq \ldots \supseteq m^T(u)$. After the last round (i.e. round $T$) the function $m^T$ is equal to the function $m$.

At a round $t$ in which a vertex $u \in U$ is presented, we say that $d \in D$ is *available* for $u$ if $d \in N(u)$ and $m^{t-1}(x) \neq \{d\}$ for any $x$ presented earlier. Furthermore, $d$ is *strongly available* for $u$ if it is available for $u$ and $d$ does not belong to any $m^{t-1}(x)$. Vertex $e \in U$ is *ready* for $u$ if $m^{t-1}(e)$ contains an element that is available for $u$. See Figure 2 for an example.

We present an algorithm for the Scheduler called $\alpha$-BALANCED. Suppose that a vertex $u$ is presented in round $t$, and let $U_t$ be the set of vertices presented so far (including $u$). The function $m^{t-1}$ defined on $U_{t-1}$ is already known, and the algorithm has to construct the function $m^t$ defined on $U_t$. The algorithm initially assigns $m^t(x) := m^{t-1}(x)$ for all $x \in U_{t-1}$ and sets $m^t(u) := \emptyset$. Then, the set $m^t(u)$ is increased, one element at a time, until a certain condition is satisfied. During the process some other sets $m^t(x)$ may be decreased. The precise description is listed in Algorithm 1.

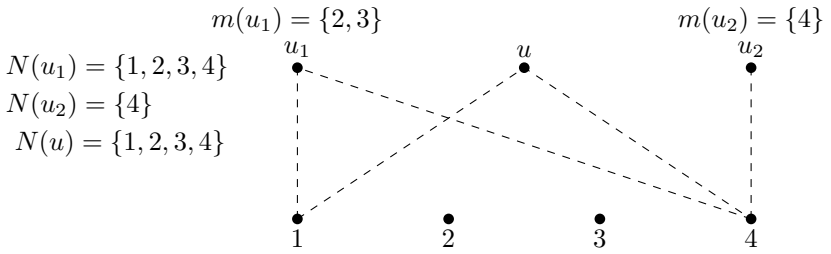


Figure 2: Sample graph for when a vertex $u$ is presented. Element 1 is strongly available, and both 2 and 3 are (just) available for $u$. Element 4 is not available for $u$. Vertex $u_1$ is ready for $u$.

**Algorithm 1:** $\alpha$ – BALANCED($u$) - round $t$

1 **let** $m^t := m^{t-1}$
2 **let** $m^t(u) := \emptyset$
3 pick up at most $\alpha$ strongly available elements for $u$ and put it into $m^t(u)$
4 **while** *there exists a vertex* $e \in U$ *that is ready for* $u$ *and satisfies*

$$\left|m^t(u)\right| + 2 \leqslant \left|m^t(e)\right|$$

**do**
5 from the set of all such vertices pick $e$ with maximal size of $m^t(e)$
6 move one vertex available for $u$ from $m^t(e)$ to $m^t(u)$

The condition in line 3 guarantees that the size of $m^t(u)$ is never greater than $\alpha$. Note also that $\alpha$-BALANCED never leaves $m^t(u)$ empty if there exists an element available for $u$; thus, in this respect, $\alpha$-BALANCED can be considered as greedy. For $\alpha = 1$, the above algorithm is just a simple greedy construction of a bipartite matching.

The following proposition describes the performance of $\alpha$-BALANCED.

**Proposition 5.** *The size $k$ of the multi-match produced by $\alpha$-BALANCED in a deferred $\alpha$-matching game with a size of $n$ equals $n - (x_0 + x_1 + \cdots + x_k)$ for a certain pair $(k, (x_0, \ldots, x_k))$ satisfying* (1).

*Proof.* Consider an instance of the matching game with a size of $n$ in which the Builder produced the graph $G = (U, D, E)$, and $\alpha$-BALANCED algorithm constructed the multi-match $m : U \to \mathcal{P}(D)$. Suppose that $T$ rounds have been played in the game. *The presentation time* of an element $u \in U$ is the index of the round in which $u$ has been revealed.

Let $X$ be the set of all vertices in $D$ such that $m(U) \cap X = \emptyset$. The size of the multi-match produced by $\alpha$-BALANCED is equal to the size of $\mathbf{Y} = \{u \in U : m(u) \neq \emptyset\}$. The proof of the proposition requires the following claims.

**Claim 6.** *Suppose that $x \neq y$, $m^{t_1}(x) \cap m^{t_2}(y) \neq \emptyset$ and $t_1 < t_2$, then $|m^{t_1+1}(x)| \geqslant |m^{t_2}(y)|$.*

*Proof.* We will prove the claim by the induction on $t_2 - t_1$. For the induction base suppose that $t_2 = t_1 + 1$. This means that during round $t_2$, the $\alpha$-BALANCED algorithm removed (at least) one element from $m^{t_2}(x)$ and inserted it into $m^{t_2}(y)$. Let $d \in m^{t_1}(x) \cap m^{t_2}(y)$ be the last such element (see Figure 3). This only happens when the condition from line 4 of the algorithm is satisfied, and $x$ is a vertex with the maximum size of the assigned set among the vertices ready for $y$. Let $s$ be the size of the set assigned to $x$ at that moment (in terms of Algorithm 1, this is $|m^{t_2}(x)|$). Clearly, $|m^{t_1}(x)| \geqslant s$, since $|m^{t_1}(x)|$ is the size of the set assigned to $x$ in the beginning of round $t_2$, and since that set can only become smaller during the round. Furthermore, $s - 1 = |m^{t_1+1}(x)|$, since element $d$ was the last one removed from $m^{t_2}(x)$.

The condition from line 4 of $\alpha$-BALANCED guarantees that the set that has just been increased has no more elements than the set that has been decreased. That property, combined with the fact that for any vertex that was ready for $y$, the assigned set was not greater than $s$, gives $s > |m^{t_2}(y)|$. Thus, indeed, $|m^{t_1+1}(x)| \geqslant |m^{t_2}(y)|$.

For the general case take any $d \in m^{t_1}(x) \cap m^{t_2}(y)$ and consider the last round $t \geqslant t_1$, where $d \in m^t(x)$. Note that $t < t_2$, since through the assumption, we know that $d \in m^{t_2}(y)$ for $y \neq x$ and $t_2 > t_1$. Therefore, there is $z \in U$ different from $x$ such that $d \in m^{t+1}(z)$. Based on the induction base and the fact that the sizes of the matching sets of fixed $u \in U$ never increase, we obtain $|m^{t_1}(x)| \geqslant |m^{t_1+1}(x)| \geqslant |m^{t+1}(x)| \geqslant |m^{t+1}(z)|$. If $z = y$, then simply $|m^{t+1}(z)| \geqslant |m^{t_2}(y)|$, and the claim follows. The case $z \neq y$ implies that $t+1 < t_2$; however we also have $t_2 - (t+1) < t_2 - t_1$. Therefore, based on the induction, we obtain $|m^{t+1}(z)| \geqslant |m^{t+2}(z)| \geqslant |m^{t_2}(y)|$, which concludes the proof. □

This claim has the following interesting consequence: if we focus on a single element $d \in D$ and track the sizes of the matching sets that contain this element during the game, then these sizes form a non-increasing sequence.

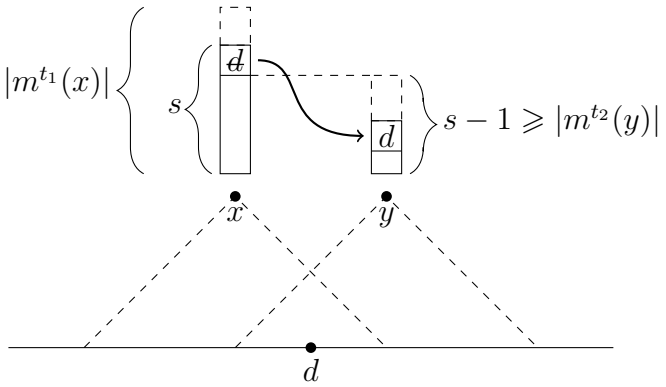


Figure 3: Moving element $d$ from $m(x)$ into $m(y)$.

**Claim 7.** *If $t$ is the presenting time of $u$ and $N(u) \cap X \neq \emptyset$, then $|m^t(u)| = \alpha$.*

*Proof.* Let $d \in N(u) \cap X$. By the definition of $X$, element $d$ is strongly available for $u$ when $u$ is presented. But $d$ is not chosen in the line 3 of $\alpha$-BALANCED. It means

that there were at least $\alpha$ other strongly available elements for $u$ that had been added to $m^t(u)$. Thus, indeed $|m^t(u)| = \alpha$. □

The next claim will allow us to prove that inequalities (1) are satisfied for carefully chosen $(k, (x_0, \ldots, x_k))$. We will apply it for quite specific values of $Y \subseteq \mathbf{Y}$ (we sort $\mathbf{Y}$ so that the sizes of $m(y)$ do not increase along the list, and the consider prefixes of the resulting list). However, as this does not seem to simplify the proof, the claim is formulated for arbitrary $Y \subseteq \mathbf{Y}$. Once it is fixed (see Figure 4), we denote by $\mu$ the minimum size of $m(y)$ for $y \in Y$. Set $M \subset D$ consists of elements assigned to the elements of $Y$ after the game (i.e. $M = m(Y)$). Finally, set $Q \subset U$ consists of all elements that contain at least one element of $M$ or $X$ in their neighbourhood (in particular, $Y \subset Q$). Intuitively these are the elements that directly influence the final shape of $M$.

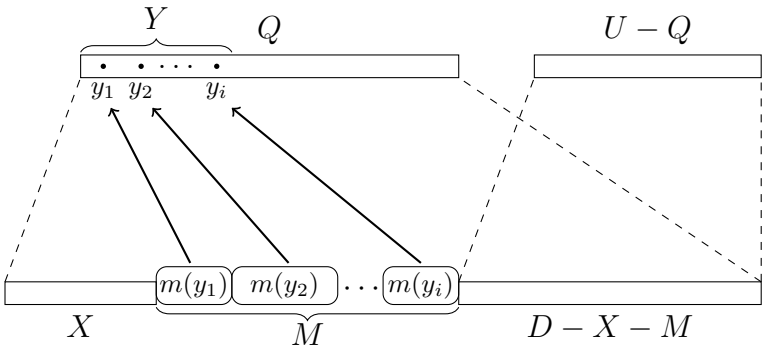


Figure 4: A partition of $U$ and $D$ with a chosen subset $Y = \{y_1, y_2, \ldots, y_i\} \subseteq U$. Set $M$ is equal to $\bigcup_{y \in Y} m(y)$. Each element $q \in Q$ has a neighbour in $M$ or in $X$, i.e. $N(q) \cap (M \cup X) \neq \emptyset$. For element $u \in U - Q$, we have $N(u) \subseteq D - M - X$.
Recall that $x \in X$ iff $x \notin m(u)$ for all $u \in U$.

**Claim 8.** *For any subset $Y \subseteq \mathbf{Y}$ we have*

$$(|Q| - |Y|)(\mu - 1) + |M| \leqslant |D - X|\,,$$

*where $\mu = \min\{|m(y)| : y \in Y\}, M = \bigcup_{y \in Y} m(y)$ and $Q$ is the set of all vertices $q \in U$ for which $N(q) \cap (M \cup X) \neq \emptyset$.*

*Proof.* For every element $q \in Q$, we will assign a witnessing set $Z_q \subseteq D - X$, such that

(a) all sets $Z_q$ are mutually disjoint,

(b) $|Z_q| \geqslant |m(q)|$ for $q \in Y$,

(c) $|Z_q| \geqslant \mu - 1$ for $q \in Q - Y$.

Consequently, the left hand side of the claimed inequality will be the lower bound for the total count of the witnessing elements, which is $\sum_{q\in Q}|Z_q|$. This number must be lower than the total number of the assigned elements, which is the value of the right hand side.

The claim is obvious for $\mu = 1$, since $M \cap X = \emptyset$. We assume that $\mu > 1$. Let $i = |Y|, s = |Q|$, and let $(q_1, \ldots, q_s)$ be the enumeration of $Q$ for which the sequence of the corresponding presenting times $(t_1, \ldots, t_s)$ is strictly decreasing. This means that $q_1, \ldots, q_s$ are in the reverse of the arrival order. For each $q_j$, we recursively define its witnessing set as

$$Z_j := m^{t_j}(q_j) - (Z_1 \cup \ldots \cup Z_{j-1}) \subseteq D - X.$$

Clearly, these sets are disjoint and consist of elements of $D-X$; therefore, $|Z_1|+\cdots+|Z_s| \leqslant |D - X|$. All we need to complete the proof now is to show that $|Z_j| \geqslant \mu - 1$ for all $j$ $(1 \leqslant j \leqslant s)$, and $Z_j \supseteq m(q_j)$ for $q_j \in Y$. See (b) and (c).

Suppose first that $q_j \in Y$. The set $m^t(q_j)$ of the elements assigned to $q_j$ can only become smaller after vertex $q_j$ has been presented; therefore, we have $m(q_j) \subseteq m^t(q_j)$ for all $t \geqslant t_j$. In particular, for every $j' \leqslant j-1$, we have $t_{j'} \geqslant t_j$, hence $m(q_j) \cap Z_{j'} = \emptyset$. This gives $Z_j \supseteq m(q_j)$.

Suppose now that $q_j \notin Y$. Let $R_j := (Z_1 \cup \ldots \cup Z_{j-1}) \cap m^{t_j}(q_j)$ (so that $m^{t_j}(q_j) = Z_j \cup R_j$). Assume also that $|Z_j| < \alpha$, since otherwise $|Z_j| = \alpha > \mu - 1$. Consider two cases:

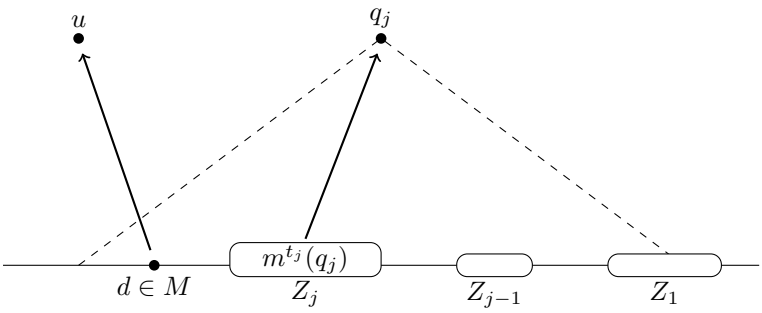


Figure 5: Case $R_j = \emptyset$ at the round $t_j$. The element $d \in M$ is available for $q_j$, but $d \in m_{t_j}(u)$ for some $u \in U$.

**Case 1: $R_j = \emptyset$.** Based on Claim 7 and the definition of $q_j \in Q$, inequality $|Z_j| < \alpha$ implies that $N(q_j) \cap X = \emptyset$ and thus $N(q_j) \cap M \neq \emptyset$. Let $d \in N(q_j) \cap M$. Note that element $d$ is not strongly available for $q_j$ (at the time $t_j$). Otherwise, it would be inserted into $m^{t_j}(q_j)$ (knowing that $|Z_j| < \alpha$). However, since after the game $d \in m(y)$ for some $y \in Y$, this would also mean that $d \in R_j$, which contradicts our assumption that $R_j = \emptyset$. Therefore, at the time $t_j$, element $d$ must already belong to $m^{t_j}(u)$ for some $u \in U$ presented earlier (see Figure 5).

We show that $d$ is available for $q_j$ (at the time $t_j$). At the end of the game $m(y)$ has the size of at least $\mu$; therefore, based on Claim 6, the sets containing $d$ during the game were at least of that size. In particular, we have $|m^{t_j}(u)| \geqslant \mu \geqslant 2$ (since $\mu > 1$). It means that $d$ is indeed available for $q_j$ at the time $t_j$.

The algorithm in round $t_j$ did not select the element $d$ to be assigned to $q_j$; consequently, at the end of the round, the inequality in line 4 of the algorithm was not satisfied. This means that $|Z_j| = |m^{t_j}(q_j)| \geqslant |m^{t_j}(u)| - 1 \geqslant \mu - 1$.

**Case 2: $R_j \neq \emptyset$.** Let $t > t_j$ be the lowest number (the first moment) for which $R_j \cap m^t(q_j) = \emptyset$ (this is a straightforward consequence of the definition that such $t$ exists). This means that $t$ is the last round in which an element of $R_j$ was removed from the set assigned to $q_j$, in particular, $|Z_j| \geqslant |m^t(q_j)|$. Consider any element $d \in R_j \cap m^{t-1}(q_j)$. Based on the definition of $Z_j$, there exists a number $l < j$ such that $d \in Z_l \subseteq m^{t_l}(q_l)$ with $t - 1 < t_l$. Based on Claim 6, this means that $|m^t(q_j)| \geqslant |m^{t_l}(q_l)|$, thus $|Z_j| \geqslant |Z_l|$. Straightforward induction (with Case 1 as basis) gives $|Z_j| \geqslant \mu - 1$. □

We are ready to prove the proposition. Fix any optimal (maximum) matching in graph $G$ and let $F \subseteq D$ be the set of all elements in $D$ outside the matching. Consider an enumeration $(y_1, \ldots, y_k)$ of $\mathbf{Y}$ such that, for $x_i = |m(y_i)| - 1$, we have $x_1 \geqslant x_2 \geqslant \ldots \geqslant x_k \geqslant 0$. Let $x' = |X - F|$, $f' = |F - X|$ and $x_0 = x' - f'$. Observe that $|F| = f' + |X| - x'$. This implies that

$$|D - X| = |D| - |X| = n + |F| - |X| = n - x_0. \tag{10}$$

To show that $(k, (x_0, x_1, \ldots, x_k))$ satisfies (1), fix $1 \leqslant i \leqslant k$ and apply Claim 8 for $Y = \{y_1, \ldots, y_i\}$. Thus, $|M| = x_1 + \cdots + x_i + i$ and $\mu = x_i + 1$. Recall that in the selected optimal matching each vertex in $D - F$ has a unique match in $U$. Therefore, $|Q| \geqslant |M - F| + |X - F| = |M - (F - X)| + |X - F| \geqslant |M| - f' + x'$ since $M \cap X = \emptyset$, and thus,

$$(|M| - i + x_0)(\mu - 1) + |M| \leqslant |D - X| \overset{(10)}{=} n - x_0,$$

which can be rewritten as

$$(x_0 + x_1 + \cdots + x_i) \cdot (1 + x_i) \leqslant n - i.$$

To show $(1 + \alpha)x_0 \leqslant n$, we will consider the set $Q'$ of all vertices $q \in U$ for which $N(q) \cap X \neq \emptyset$. For each $q \in Q'$ we define $s(q)$ as the set of all strongly available elements inserted into $m^t(q)$, at the time $t$ when $q$ was presented, in line 3 of the algorithm. Firstly, note that $s(q_1)$ and $s(q_2)$ are disjoint for distinct $q_1, q_2 \in Q'$. This is because $s(q)$ gathers only strong available elements, which will never be strong available elements again. Secondly, based on Claim 7, we know that $|s(q)| = \alpha$ for each $q \in Q'$.

The above two facts imply that $\alpha\,|Q'| \leqslant n - x_0$, since $\bigcup_{q \in Q'} s(q) \subseteq D - X$ and through (10). Furthermore, since each element in $D - F$ has a unique match in $U$, we have $|Q'| \geqslant |X - F| = x' \geqslant x_0$. Therefore, $(1 + \alpha)x_0 \leqslant n$.

To complete the proof, recall that $\bigcup_{y \in \mathbf{Y}} m(y) = D - X$. Thus, $x_1 + \cdots + x_k + k = n - x_0$, and consequently, the size of the multi-match constructed by the algorithm equals $k = n - (x_0 + x_1 + \cdots + x_k)$. Furthermore, since $k$ cannot be larger than $n$, we have $x_0 + \cdots + x_k \geqslant 0$. □

Combining Proposition 4 and Proposition 5 we finally arrive at Theorem 1.

## 4 Competitiveness of the algorithm

Recall that the worst (minimum) size of the multi-match constructed by $\alpha$-BALANCED in any deferred $\alpha$-matching game with a size of $n$ is denoted by $\mathrm{val}_{\alpha-\mathrm{BALANCED}}(n)$. To simplify this notation we use $\mathrm{bal}(\alpha, n)$ instead of $\mathrm{val}_{\alpha-\mathrm{BALANCED}}(n)$. Similarly, for the competitive ratio of $\alpha$-BALANCED, i.e. $\liminf_{n\to\infty} \mathrm{bal}(\alpha, n)/n$, we use the notation $\mathrm{bal}(\alpha)$.

Propositions 4 and 5 imply that in order to determine $\mathrm{bal}(\alpha, n)$, it is sufficient to find a pair $(k, (x_0, \ldots, x_k))$ that satisfies (1) and maximises $\sum_{i=0}^{k} x_i$. Moreover, based on Proposition 3, we can assume that in the maximising solution, we have $x_0 = \lfloor \frac{n}{1+\alpha} \rfloor$. From now on, we will consider $x_0$ in system (1) as fixed together with $n$ and $\alpha$.

Consider a pair $(k, x)$ such that $x_i = x_{i+1}$ for some $i \geqslant 1$. If $(k, x)$ satisfies the $(i+1)$-th inequality from (1), i.e. $(x_0 + \cdots + x_{i+1})(1 + x_{i+1}) \leqslant n - (i+1)$, then it also satisfies the $i$-th inequality from (1):

$$(x_0 + \cdots + x_i)(1 + x_i) \leqslant (x_0 + \cdots + x_i + x_{i+1})(1 + x_{i+1}) \leqslant n - (i+1) < n - i.$$

This suggests another representation of the solutions. Let $(k, x)$ be a pair satisfying (1) with $x = (x_1, \ldots, x_k)$. Note that $x$ is a non-increasing sequence of non-negative integers. We will assume, without a loss of generality, that $x_k > 0$. Let $Y(x) = (y_1, \ldots, y_m)$, such that $y_j = |\{i > 0 : 1 + x_i = j\}|$, for $j = 1, \ldots, m$, and $m$ be such that $y_m \neq 0$. Thus, $y_j$ is simply the number of the (consecutive) entries of $x$ with the value $j - 1$. Clearly, $m - 1$ equals the maximum value of the sequence $x$. This value is realised in $x_1$, which gives $m = x_1 + 1$. Consequently, for every $i$ for which $x_i \neq x_{i+1}$, the inequality

$$(x_0 + x_1 + \cdots + x_i)(1 + x_i) \leqslant n - i$$

can be rewritten as

$$(x_0 + (m-1) \cdot y_m + \cdots + (t-1) \cdot y_t) \cdot t \leqslant n - (y_m + \cdots + y_t),$$

where $t = x_i + 1$. Thus, the sequence $(y_1, \ldots, y_m)$ belongs to the image of $Y$ whenever it satisfies the following system of inequalities

$$\Psi_{n,m}(x_0): \qquad t \cdot x_0 + \sum_{i=t}^{m} (1 + (i-1)t) y_i \leqslant n \quad \text{for } t = 1, \ldots, m. \tag{11}$$

Moreover, since $y_i \geqslant 0$, the $m$-th inequality in (11) implies that

$$n - m x_0 \geqslant 0. \tag{12}$$

On the other hand, having any $m > 0$ satisfying (12) and any solution $(y_1, \ldots, y_m)$ of (11), we can easily find (using the definition of the $y_j$'s) a solution $(k, x)$ of (1) such that $x_0 + x_1 + \cdots + x_k = x_0 + \sum_{i=1}^{m} (i-1) y_i$.

The above considerations are summarised in the following proposition.

**Proposition 9.** *The minimal size of the multi-match constructed by $\alpha$-BALANCED in all deferred $\alpha$-matching games with a size of $n$ is equal to*

$$\mathrm{bal}(\alpha, n) = n - x_0 - \sup\left\{\sum_{i=1}^{m}(i-1)y_i\right\},$$

*where $x_0 = \lfloor n/(1+\alpha)\rfloor$ and the supremum is taken over all integers $m > 0$ such that $n - mx_0 \geqslant 0$ and over all vectors $(y_1, \ldots, y_m)$ of the non-negative integers that satisfy $\Psi_{n,m}(x_0)$.*

### 4.1 LP formulation

In order to maximise $\sum_{i=1}^{m}(i-1)y_i$, we will consider the following linear program. We present the primal and the dual formulation.

**Primal**

$$\begin{aligned} \text{maximise:} \quad & \textstyle\sum_{j=1}^{m}(j-1)y_j \\ \text{subject to:} \quad & \textstyle\sum_{j=i}^{m}(1+(j-1)i)y_j \leqslant n - ix_0, \text{ and } y_i \geqslant 0 \quad \text{for } i = 1, \ldots, m \end{aligned}$$

**Dual**

$$\begin{aligned} \text{minimise:} \quad & \textstyle\sum_{i=1}^{m}(n-ix_0)z_i \\ \text{subject to:} \quad & \textstyle\sum_{i=1}^{j}(1+(j-1)i)z_i \geqslant j-1, \text{ and } z_j \geqslant 0 \text{ for } j = 1, \ldots, m \end{aligned}$$

Let $P_i(y) = \sum_{j=i}^{m}(1+(j-1)i)y_j$ and $D_j(z) = \sum_{i=1}^{j}(1+(j-1)i)z_i$. Note that both systems $(P_i(y) = n - ix_0)_{i=1,\ldots,m}$ and $(D_j(z) = j-1)_{j=1,\ldots,m}$ are quadratic and have unique solutions. Based on the Complementary Slackness Theorem, these solutions will be optimal as long as both are non-negative vectors.

To solve the first system, note that $P_{i+2}(y)+P_i(y)-2P_{i+1}(y) = (1-i+i^2)y_i-(1+i)^2y_{i+1}$. This, together with the initial values for $y_m$ and $y_{m-1}$, lead to

$$y_i = y_{i+1}\frac{(i+1)^2}{1-i+i^2} \quad \text{for } i = 1, \ldots, m-2, \tag{13}$$

$$y_m = \frac{n - m\cdot x_0}{1+m(m-1)}, \qquad y_{m-1} = \frac{x_0 + (m-1)y_m}{1+(m-1)(m-2)}. \tag{14}$$

Through (12), $y_m \geqslant 0$, and thus all $y_i$ are non-negative.

For the second system observe that $D_{j+1}(z) + D_{j-1}(z) - 2D_j(z) = (1+j+j^2)z_{j+1} - (1-j)^2z_j$. Again, with the initial values for $z_1$ and $z_2$, we arrive at

$$\begin{aligned} & z_{j+1} = z_j\frac{(j-1)^2}{1+j+j^2} \quad \text{for } j = 2, \ldots, m-1, \\ & z_1 = 0, \qquad z_2 = 1/3. \end{aligned}$$

Therefore, $z_i \geqslant 0$, and, indeed, the above vectors $y$ and $z$ are the optimal solutions for the primal and the dual LP.

To calculate the target function $\sum_{j=1}^{m}(j-1)y_j$, consider a new variable $y_0$ defined by the additional condition: $P_0(y) = n$. Equation (13) still works for $y_0$. Furthermore, after combining $P_0(y)$ with $P_1(y)$, we obtain $\sum_{j=1}^{m}(j-1)y_j = y_0 - x_0$. Finally, using (13) and (14) and after rearrangement, we arrive at

$$x_0 + \sum_{j=1}^{m}(j-1)y_j = y_0 = \frac{(m-1)n + x_0}{m}\prod_{i=1}^{m-1}\frac{i+i^2}{1+i+i^2}. \tag{15}$$

### 4.2 Lower bound

The solution for the LP relaxation of (11) may not be an integer; therefore, with the solution (15), we only have the following bound:

$$\mathrm{bal}(\alpha, n) \geqslant n - \sup\{y_0 : n - mx_0 \geqslant 0\}.$$

Let $F(z,m) = \frac{(m-1)+z}{m}\prod_{i=1}^{m-1}\frac{i+i^2}{1+i+i^2}$. Thus, for fixed $m$, we obtain $y_0 = n \cdot F(x_0/n, m)$. Note that the function $F(z,m)$ increases with $m$. For $m \leqslant 1/z$ and $F(0,m)$, it increases indefinitely with $m$. Therefore for $x_0 = \lfloor n/(1+\alpha)\rfloor > 0$, we have

$$\begin{aligned}\mathrm{bal}(\alpha, n)/n &\geqslant 1 - \max_{m \leqslant n/x_0} F(x_0/n, m)\\ &= 1 - F(x_0/n, \lfloor n/x_0\rfloor) \overset{n\to\infty}{\longrightarrow} 1 - F(1/(1+\alpha), 1+\alpha),\end{aligned} \tag{16}$$

and for $x_0 = 0$ (this occurs when $\alpha \geqslant n$) the bound is

$$\mathrm{bal}(\alpha, n)/n \geqslant 1 - \lim_{m\to\infty} F(0,m) = 1 - \prod_{i=1}^{\infty}\frac{i+i^2}{1+i+i^2} = 1 - \frac{\pi}{\cosh\frac{\sqrt{3}}{2}\pi}. \tag{17}$$

### 4.3 Upper bound

Consider $m = \min(\lfloor n/x_0\rfloor, \ln n)$, with the assumption that $n/x_0$ is greater than $\ln n$ when $x_0 = 0$, and let $(y_1, \ldots, y_m)$ be the optimal rational solution of $\Psi_{n,m}(x_0)$. Let $v = (v_1, \ldots, v_m)$ be such that $v_i = \lfloor y_i\rfloor$. The vector $v$ contains only non-negative, integer entries and $v \leqslant y$. The shape of the system $\Psi_{n,m}(x_0)$ guarantees that $v$ also satisfies $\Psi_{n,m}(x_0)$. Finally, we have

$$x_0 + \sum_{i=1}^{m}(i-1)v_i > x_0 + \sum_{i=1}^{m}(i-1)y_i - \sum_{i=1}^{m}(i-1).$$

The definition of $m$ indicates that $m = o(n)$. Therefore, through Proposition 4, we arrive at

$$\mathrm{bal}(\alpha, n) < n \cdot (1 - F(x_0/n, m)) + m(m-1)/2 = n \cdot (1 - F(x_0/n, m)) + o(n)$$

Hence, for $x_0 = \lfloor n/(1+\alpha) \rfloor > 0$ the bound is

$$\mathrm{bal}(\alpha, n)/n < 1 - F(x_0/n, \lfloor n/x_0 \rfloor) + o(1) \stackrel{n\to\infty}{\longrightarrow} 1 - F(1/(1+\alpha), 1+\alpha), \tag{18}$$

and for $x_0 = 0$ we have

$$\mathrm{bal}(\alpha, n)/n < 1 - F(0, \ln n) + o(1) \stackrel{n\to\infty}{\longrightarrow} 1 - \prod_{i=1}^{\infty} \frac{i+i^2}{1+i+i^2} = 1 - \frac{\pi}{\cosh \frac{\sqrt{3}}{2}\pi}. \tag{19}$$

### 4.4 Competitive ratio

If $\alpha$ is finite, then based on (16) and (18), the ratio of $\alpha$-BALANCED is equal to $\mathrm{bal}(\alpha) = 1 - F(\frac{1}{\alpha+1}, \alpha+1)$. Note that for the case $\alpha \to \infty$, $F(1/(1+\alpha), 1+\alpha) \to \pi/\cosh \frac{\sqrt{3}}{2}\pi$. Thus, based on (16-19), we arrive at $\mathrm{bal}(\alpha) = 1 - \pi/\cosh \frac{\sqrt{3}}{2}\pi$. This finally proves Theorem 2.

## 5 Conclusions and remarks

In the classical on-line matching problem, the randomised approach has a considerable advantage over the deterministic approach. This paper shows that in the model with the deferred decisions, the deterministic bounds can be pushed further. It is interesting to know what can be achieved with randomised strategies.

**Problem 10.** What is the competitive ratio of the randomised version of the deferred matching problem?

The authors of [16] consider a variant (called $b$-matching), in which each server can perform up to $b$ tasks. The competitive ratio of their optimal algorithm approaches $1 - \frac{1}{e}$ with $b \to \infty$, which is a barrier for any randomised matching algorithm (see [18]). We expect that in our model, it will be possible to break the $1 - \frac{1}{e}$ limit in case of $b$-matching.

**Problem 11.** What is the competitive ratio of the deferred $(\alpha, b)$-matching problem?

Note that the underlying graph $G$ of a partial order $P$ with a height of most 2 is bipartite. The following Claim 12 implies that the problems of finding the maximal matching of $G$ and finding the minimal chain partition of $P$ are dual.

**Claim 12.** *Let $G$ be a bipartite graph with $N$ vertices. If $n$ is the size of the maximal matching in $G$, and $w$ is the size of the maximal independent set in $G$, then $n + w = N$.*

This allows the results of this paper to be adopted for the deferred (a.k.a. adaptive) approach of the on-line chain partition problem (which was mentioned in the introduction). We only provide the result without the prove, which is only a technical modification of the proofs in this paper.

**Theorem 13.** *There is an algorithm for the $\alpha$-adaptive chain partitioning of up-growing orders with a height of 2, with competitive ratio equal to $1 + \frac{\alpha}{1+\alpha} \prod_{i=1}^{\alpha-1} \frac{i+i^2}{1+i+i^2}$.*
*For unlimited $\alpha$, i.e. $\alpha \to \infty$, the competitive ratio equals $1 + \pi/\cosh \frac{\sqrt{3}}{2}\pi \approx 1.412$.*

## References


[1] Yossi Azar and Yoel Chaiutin. Optimal node routing. In Bruno Durand and Wolfgang Thomas, editors, *STACS 2006*, volume 3884 of *Lecture Notes in Computer Science*, pages 596–607. Springer Berlin Heidelberg, 2006.

[2] Yossi Azar, Joseph Naor, and Raphael Rom. The competitiveness of on-line assignments. *Journal of Algorithms*, 18(2):221 – 237, 1995.

[3] Yossi Azar and Yossi Richter. Management of multi-queue switches in qos networks. *Algorithmica*, 43(1):81–96, Sep 2005.

[4] Bahman Bahmani and Michael Kapralov. Improved bounds for online stochastic matching. In *Proceedings of the 18th annual European conference on Algorithms: Part I*, ESA'10, pages 170–181, Berlin, Heidelberg, 2010. Springer-Verlag.

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

[6] Bartłomiej Bosek, Stefan Felsner, Kamil Kloch, Tomasz Krawczyk, Grzegorz Matecki, and Piotr Micek. On-line chain partitions of orders: A survey. *Order*, 29(1):49–73, 2012.

[7] Kamalika Chaudhuri, Constantinos Daskalakis, Robert D. Kleinberg, and Henry Lin. Online bipartite perfect matching with augmentations. In *INFOCOM 2009, IEEE*, pages 1044–1052, April 2009.

[8] Ashish Chiplunkar, Sumedh Tirodkar, and Sundar Vishwanathan. On randomized algorithms for matching in the online preemptive model. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms – ESA 2015*, volume 9294 of *Lecture Notes in Computer Science*, pages 325–336. Springer Berlin Heidelberg, 2015.

[9] Nikhil R. Devenur and Thomas P. Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *Proceedings of the 10th ACM conference on Electronic commerce*, EC '09, pages 71–78, New York, NY, USA, 2009. ACM.

[10] Leah Epstein, Asaf Levin, Danny Segev, and Oren Weimann. Improved Bounds for Online Preemptive Matching. In Natacha Portier and Thomas Wilke, editors, *30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013)*, volume 20 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 389–399, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[11] Jon Feldman, Nitish Korula, Vahab Mirrokni, S. Muthukrishnan, and Martin Pál. Online ad assignment with free disposal. In Stefano Leonardi, editor, *Internet and Network Economics*, volume 5929 of *Lecture Notes in Computer Science*, pages 374–385. Springer Berlin Heidelberg, 2009.

[12] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '09, pages 117–126, Washington, DC, USA, 2009. IEEE Computer Society.

[13] Stefan Felsner. On-line chain partitions of orders. *Theoret. Comput. Sci.*, 175(2):283–292, 1997.

[14] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '08, pages 982–991, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[15] Anupam Gupta, Amit Kumar, and Cliff Stein. Maintaining assignments online: Matching, scheduling, and flows. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 468–479. SIAM, 2014.

[16] Bala Kalyanasundaram and Kirk R. Pruhs. An optimal deterministic algorithm for online b-matching. *Theoretical Computer Science*, 233(1-2):319 – 325, 2000.

[17] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, STOC '11, pages 587–596, New York, NY, USA, 2011. ACM.

[18] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, STOC '90, pages 352–358, New York, NY, USA, 1990. ACM.

[19] Vahideh H. Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research*, 37(4):559–573, 2012.

[20] Aranyak Mehta and Debmalya Panigrahi. Online matching with stochastic rewards. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12, pages 728–737, Washington, DC, USA, 2012. IEEE Computer Society.

[21] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5):no. 5, October 2007.