

# On Adequate Performance Measures for Paging

Konstantinos Panagiotou\*  
panagiok@inf.ethz.ch

Alexander Souza  
asouza@inf.ethz.ch

Institute of Theoretical Computer Science, ETH Zürich  
Universitätstr. 6, CH - 8092 Zürich

## Abstract

Memory management is a fundamental problem in computer architecture and operating systems. We consider a two-level memory system with fast, but small cache and slow, but large main memory. The underlying theoretical problem is known as the paging problem. A sequence of requests to pages has to be served by making each requested page available in the cache. A paging strategy replaces pages in the cache with requested ones. The aim is to minimize the number of page faults that occur whenever a requested page is not in the cache.

Experience shows that the LEAST-RECENTLY-USED (LRU) paging strategy usually achieves a factor around 2 to 3 compared to the optimum number of faults. This contrasts the theoretical worst case, in which this factor can be as large as the cache size  $k$ .

One difficulty in analyzing the paging problem was the lack of an appropriate lower bound for the minimum number of page faults. We address this issue and propose a general lower bound which provides insight into the global structure of a given request sequence. In addition, we derive a characterization for the number of faults incurred by LRU.

We give a theoretical explanation why LRU performs well in practice. We classify the set of all request sequences according to certain parameters and prove a bound on the competitive ratio of LRU, which depends on them. This bound varies between 2 and  $k$ , i.e., it includes the worst-case, but explains for which sequences LRU achieves constant competitive ratio. The classification is motivated from the structure of request sequences of practical applications: locality of reference and characteristic data access patterns. We argue that this structure yields values around 2 for our bound. Indeed, it is between 2 and 5 in extensive practical experiments.

Moreover, we show that the alternative approach of variable cache size, which was also studied previously, is not appropriate to explain this phenomenon. Further, our analysis provides evidence that the approach of the expected competitive ratio  $\frac{\mathbb{E}[\text{ALG}]}{\mathbb{E}[\text{OPT}]}$  in the diffuse adversary model can be misleading. We propose the use of the expected performance ratio  $\mathbb{E}\left[\frac{\text{ALG}}{\text{OPT}}\right]$  instead, and prove tight bounds for both measures.

## 1 Introduction

Memory management is a fundamental problem in computer architecture and operating systems. In its simplest form, the memory system consists of two levels: a fast but small cache and a slow but large main memory. The cache is a temporary memory for data needed by the programs in execution. Copying from main memory to the cache takes time, but has the advantage that the

---

\*Partially supported by the SNF, grant number: 20021-107880/1

data in the cache can be accessed fast. Hence, the strategy of data replacement in the cache is crucial for the overall system performance.

The underlying theoretical problem is known as the *paging* problem: a sequence of requests to memory pages has to be served, where each requested page must be made available in the cache. The cache has *size*  $k$ , i.e., it can store up to  $k$  pages. If the page is already there, the request is served without additional cost. Otherwise, a *page fault* occurs and the requested page must be brought to fast memory. If the cache stores already  $k$  pages, one of these must be evicted, in order to clear space for the requested one. A (reasonable) paging strategy aims at minimizing the number of page faults.

In the *offline* version of the problem, the whole request sequence is known in advance. The LONGEST-FORWARD-DISTANCE algorithm (denoted OPT), which was introduced by Belady [2], is an optimal offline algorithm, which always evicts the page with the most distant next request.

In *online* paging, requests arrive one-by-one and eviction decisions must be made at every arriving request, without any knowledge of the future. Examples for online strategies are LEAST-RECENTLY-USED (LRU), which evicts the page in the cache whose last access was earliest, and FIRST-IN-FIRST-OUT (FIFO), which evicts the page that has been in fast memory longest.

Sleator and Tarjan [9] proposed to evaluate the performance of online algorithms by using competitive analysis. In that model, an online algorithm ALG is compared with an offline optimum algorithm OPT on the same input sequence. The request sequence is chosen by an offline *adversary*  $A$  out of a set  $S_A$  of *admissible* sequences.

Denote the respective number of faults of ALG and OPT on the request sequence  $R$  by  $\text{ALG}(R)$  and  $\text{OPT}(R)$ . The *competitive ratio* of ALG against the adversary  $A$  is defined by

$$\text{CR}_A(\text{ALG}) = \max \left\{ \frac{\text{ALG}(R)}{\text{OPT}(R)} : R \in S_A \right\}.$$

If there is a function  $c = c(k)$ , which may depend on the cache size  $k$ , such that  $\text{CR}_A(\text{ALG}) \leq c$ , then ALG is called *c-competitive*; otherwise, it is said to be not competitive.

Sleator and Tarjan [9] proved that LRU and FIFO are both  $k$ -competitive against the unrestricted adversary, and that this bound is tight. However, the theoretical  $k$ -competitiveness of LRU seems to be overly pessimistic compared to practical observations. In the experimental study of Sites and Agarwal [8], LRU achieves a factor of approximately two to three compared to the optimum.

This motivates research in order to explain this phenomenon. Apparently, the unrestricted adversary is too powerful in the choice of the sequence. Two ways to limit the power are *randomized* adversaries, where request sequences obey an underlying probability distribution and *deterministic* adversaries, for which the set of admissible sequences is restricted explicitly.

In one of our main results we propose a deterministic adversary, which is only mildly restricted by two parameters  $\alpha$  and  $\beta$ . The adversary especially covers two important principles, which are widely observed in practice: *locality of reference*, which states that a page which was requested recently is likely to be requested soon again, and *typical memory access patterns* that occur due to requesting data.

We prove a bound for the competitive ratio of LRU which depends on  $\alpha$  and  $\beta$ . We argue that realistic values for  $\alpha$  and  $\beta$  indeed are such that our bound gives values between two and five, coming close to practical experience – see Section 2.3.

## 1.1 Previous Work

In the randomized *diffuse adversary* framework of Koutsoupias and Papadimitriou [5], an adversary is allowed to choose the input distribution  $D$  out of a given class  $\Delta$  of distributions. The *expected* competitive ratio of an algorithm ALG is defined by

$$\text{ECR}_\Delta(\text{ALG}) = \max \left\{ \frac{\mathbb{E}[\text{ALG}]}{\mathbb{E}[\text{OPT}]} : D \in \Delta \right\},$$

where the expectations are taken over all request sequences weighted with the respective probability according to the distribution  $D$ . Instead of choosing a worst possible input, the adversary is only free to choose a worst input distribution.

The diffuse  $\Delta_\varepsilon$ -*adversary* of Koutsoupias and Papadimitriou [5] chooses each request randomly, such that no page is requested with probability more than some  $\varepsilon > 0$ . They proved that LRU is optimal against this adversary, but did not determine the actual ratio. Young [13, 14] proved that the function  $\text{ECR}_\Delta(\text{LRU})$  varies from constant to linear in  $k$  as  $\varepsilon$  varies from zero to one. Moreover, it is logarithmic in  $k$  if  $\varepsilon \approx \frac{1}{k}$ .

Recently, Becchetti [1] analyzed LRU in a probabilistic setting and, for the first time, proved a constant expected competitive ratio. A diffuse adversary chooses a probability distribution for each page request, such that the expected number of distinct pages until the page is requested again is at most  $\frac{k}{2}$ . In this setting, the expected competitive ratio of LRU is at most 22.

The *access graph* model of Borodin et. al. [3] is an example for a deterministic adversary, which captures locality of reference. The pages stored in slow memory are identified with the vertices of a graph. The adversary is free to choose a sequence if it maps to a walk along the edges of that graph. Fiat and Mendel [4] gave an  $\mathcal{O}(1)$ -competitive algorithm.

Another approach, carried out by Young [12], is to vary the cache size and to consider a relaxation of the competitive ratio. Roughly speaking, an algorithm is called *loosely  $c(k)$ -competitive* if its fault rate on a given sequence is “irrelevant”, or its competitive ratio is at most  $c(k)$  for “most” values of  $k \in \{1, 2, \dots, p\}$ , where  $p$  is the number of distinct pages in the sequence. Young [12] proved that LRU and FIFO are loosely log  $k$ -competitive.

## 1.2 Our Contribution

One difficulty in analyzing the paging problem was the lack of an strong lower bound on the optimum number of faults. So far, this number was lower bounded by partitioning the sequence into phases, giving insight only into *local* structure of the paging problem. We propose a novel analysis, which is based on a lower bound which captures the structure of the *entire* sequence.

Two requests to the same page with exactly  $\ell$  distinct pages inbetween are called a *pair* with *distance*  $\ell$ . Define the *characteristic vector*  $\mathbf{c}(R) = (c_0, c_1, \dots, c_p)$  of a sequence  $R$ , where every entry  $c_\ell$  counts the number of pairs with distance  $\ell$ .

This definition allows us to characterize the number of faults of LRU. This algorithm will have evicted a certain page from the cache if and only if at least  $k$  distinct pages are requested in the meantime. Therefore, we have  $\text{LRU}(R) = \sum_{\ell \geq k} c_\ell + p$ , where  $p$  denotes the number of distinct pages within the sequence  $R$ .

The crucial result for our analysis is the lower bound  $\text{OPT}(R) \geq \frac{1}{2} \sum_{\ell \geq k} \frac{\ell - k + 1}{\ell} c_\ell$ . Because both bounds depend on the characteristic vector, we are able to compare *directly* the number of faults of LRU and OPT on *any* given sequence  $R$ .

We are able to give a theoretical explanation why LRU performs well in practice. We argue that request sequences generated by running programs feature two characteristics: *locality of reference* due to executing code and *typical memory access patterns* due to requesting data.

We formalize this intuition with the  $(\alpha, \beta)$ -*adversary*, which is free to choose any sequence such that its characteristic vector satisfies  $\sum_{\ell=k}^{\alpha k-1} c_\ell \leq \beta \sum_{\ell=\alpha k}^{\alpha k-1} c_\ell$ . The intuition behind this definition is that there are not “too many” request pairs with “critical distance”, see Section 2.3. Our analysis yields that the competitive ratio of LRU is bounded by

$$\text{CR}_{(\alpha, \beta)}(\text{LRU}) \leq 2(1 + \beta) \left( 1 + \frac{1}{\alpha - 1} \right) + \varepsilon,$$

where  $\varepsilon$  depends on a worst-case sequence and is between zero and one. Depending on the values of  $\alpha$  and  $\beta$ , our bound can actually be as small as two. We argue that the sequences of running programs in practice feature “large”  $\alpha$  and “small”  $\beta$ . We performed extensive experiments, see Figure 1 for two representative results, and calculated the corresponding values for  $\alpha$  and  $\beta$ . Our bound for LRU ranged there between three and seven, coming close to practical experience.

We emphasize that our result holds for a deterministic adversary and (depending on  $\alpha$  and  $\beta$ ) improves upon a result of Becchetti [1], which states  $\text{ECR}_\Delta(\text{LRU}) \leq 22$  in a probabilistic setting.

We also consider variable cache size, which was studied earlier, see e.g., Young [12]. We introduce the  $K$ -*adversary*, which is unrestricted in the choice of the sequence, but the cache size is a random variable ranging between two integers  $a$  and  $b$ . We analyze LRU in the two measures, the expected *competitive* ratio  $\text{ECR}_K(\text{ALG}) = \frac{\mathbb{E}[\text{ALG}]}{\mathbb{E}[\text{OPT}]}$  and the expected *performance* ratio  $\text{EPR}_K(\text{ALG}) = \mathbb{E} \left[ \frac{\text{ALG}}{\text{OPT}} \right]$ .

We conclude that variable cache size is *not* appropriate to explain the performance of LRU in practice, because we can construct a sequence  $R$  such that  $\frac{\text{LRU}(R)}{\text{OPT}(R)} \simeq k$  for *every* cache size  $k \in \{a, \dots, b\}$ . This yields the second conclusion, which is more important. The expected competitive ratio can give a misleading answer, in the following sense. We can prove  $\text{ECR}_K(\text{LRU}) = \Theta(a)$  and  $\text{EPR}_K(\text{LRU}) = \Theta(b)$ . Hence, if we fix  $a$  and let  $b$  grow,  $\text{ECR}_K(\text{LRU}) = \Theta(a)$  suggests that LRU performs well, although this is not the case. It turns out that this phenomenon is because the expected competitive ratio loses an important property of competitive analysis: the *instance-wise* comparison of algorithms.

## 2 On Adequate Performance Measures for Paging

### 2.1 The Characteristic Vector

Let  $M = \{1, 2, \dots, m\}$  be the set of *pages* stored in the large memory. Let  $R = (r_1, r_2, \dots, r_n)$  denote a sequence of *requests* of *length*  $n$ , where we refer to the set  $T = \{1, 2, \dots, n\}$  as *time*. We require  $r_t \in M$  for every  $t \in T$  and call  $P = \{r_1, \dots, r_n\} \subseteq M$  the *requested pages*. Throughout,  $p = p(R) = \|P\|$  counts the number of requested pages in the sequence.

A *request pair*  $(i, j)$  is defined by two time-indices  $i$  and  $j$  within  $R$  such that  $1 \leq i < j \leq n$ . We call a request-pair  $(i, j)$  *consecutive* if  $r_i = r_j$  and all pages  $r_\ell$  in the range  $i < \ell < j$  are distinct from  $r_i$ . Define the set of distinct pages *between*  $(i, j)$  by  $D(i, j) = \{r_{i+1}, \dots, r_{j-1}\}$  and define the *distance* of a consecutive request pair  $(i, j)$  by  $\text{dist}(i, j) = \|D(i, j)\|$ . Let

$$C_\ell(R) = \{(s, t) : \text{consecutive request pair } (s, t) \text{ in } R \text{ with } \text{dist}(s, t) = \ell\}$$

denote the set of consecutive pairs with distance  $\ell$ .

The *characteristic vector* of a sequence  $R$  is defined by

$$\mathbf{c}(R) = (c_0, c_1, \dots, c_{p-1}), \text{ where } c_\ell = c_\ell(R) = \|C_\ell(R)\|.$$

Each  $c_\ell$  counts the number of consecutive pairs  $(i, j)$  that have exactly  $\ell$  distinct pages between  $i$  and  $j$ . Each  $c_\ell = c_\ell(R)$  actually depends on  $R$  but we mostly omit the argument  $R$  for simplicity.

## 2.2 Bounds for Paging Algorithms

The characteristic vector enables us to describe algebraically the number of faults of LRU, see Theorem 1, and to bound OPT from below, see Theorem 2. Especially the latter result is crucial for our analysis. We believe that our concept of the characteristic vector will also be useful for understanding further paging heuristics, e.g., FIFO.

**Theorem 1.** *Let  $k$  denote the cache size and let  $R$  be a request sequence with characteristic vector  $\mathbf{c}(R) = (c_0, \dots, c_{p-1})$ , then*

$$\text{LRU}(R) = \sum_{\ell \geq k} c_\ell(R) + p(R).$$

*Proof.* If a certain page is in the cache, it will be evicted by LRU before it is requested the next time if and only if there are requests to at least  $k$  distinct pages before that. The additional  $p$  in the bound is due to the initial faults, i.e., the faults incurred when the page is brought to the cache for the first time. ■

**Theorem 2.** *Let  $k$  denote the cache size and let  $R$  be a request sequence with characteristic vector  $\mathbf{c}(R) = (c_0, \dots, c_{p-1})$  and  $p(R) \geq k + 1$ , then*

$$\text{OPT}(R) \geq \frac{1}{1 + \frac{k-1}{k} - \frac{k-1}{p-1}} \sum_{\ell \geq k} \frac{\ell - k + 1}{\ell} c_\ell(R).$$

*The bound is tight for  $(k + 1)$ -multicycles as  $c_k$  tends to infinity.*

**Corollary 3.** *Let  $k$  denote the cache size and let  $R$  be a request sequence with characteristic vector  $\mathbf{c}(R) = (c_0, \dots, c_{p-1})$  and  $p(R) \geq k + 1$ , then*

$$\text{OPT}(R) \geq \frac{1}{2} \sum_{\ell \geq k} \frac{\ell - k + 1}{\ell} c_\ell(R).$$

For an intuitive understanding of this bound, consider an  $\ell + 1$ -multicycle, i.e., a sequence  $(1, 2, \dots, \ell + 1, 1, 2, \dots, \ell + 1, \dots)$ . It is easy to see that the optimum algorithm faults  $\ell - k + 1$  times per repetition of the cycle  $(1, 2, \dots, \ell + 1)$ , and that every consecutive request pair has distance  $\ell$ . Hence, averaging over the whole sequence yields that each pair contributes  $\frac{\ell - k + 1}{\ell}$  to the total number of faults.

The essential statement of the lower bound is the abstraction that an *arbitrary* sequence can be seen as if it were a *collection of multicycles* with the same characteristic vector.

Before we proceed to the proof of Theorem 2, we want to give some intuition about the obstacles we have to overcome. Let  $R$  be a request sequence which requests  $p$  distinct pages; then

$$\text{OPT}(R) \geq \max_{\ell \geq k} \left\{ \frac{\ell - k + 1}{\ell} c_\ell(R) \right\},$$

where  $k$  denotes the cache size. To see this, observe that OPT faults at least  $\ell - k + 1$  times between two consecutive requests to a page, with  $\ell$  distinct pages in between, as at the point of time of the first request it has cached at most  $k - 1$  of the pages inbetween. Therefore, by considering only consecutive request pairs of distance  $\ell$ , we overcount by at most a factor  $\ell$  because every fault is counted for at most  $\ell$  consecutive request pairs. Observe that the bound is tight for  $(\ell + 1)$ -multicycles.

Theorem 2 states that the max in the above lower bound can be replaced by a *sum*, losing a factor which is at most two. The main difficulty in proving it is that consecutive request pairs of different distances may *overlap*, which has to be handled appropriately.

*Proof of Theorem 2.* Recall that  $C$  denotes the set of all consecutive request pairs and also recall the set  $T$  which is referred to as time. Consider an execution of an arbitrary optimal offline algorithm OPT on a given sequence  $R$ , e.g., LONGEST-FORWARD-DISTANCE. Define

$$F = F(R) = \{t \in T : \text{OPT faults at time } t\} \quad (1)$$

and

$$f_t = f_t(R) = \begin{cases} 1 & \text{if } t \in F(R), \\ 0 & \text{otherwise,} \end{cases}$$

which yields  $\text{OPT}(R) = \sum_{t \in T} f_t(R)$ .

Below, we prove that for every  $t \in T$  and every pair  $(i, j) \in C$ , there exists a function  $\delta_t(i, j)$ , which has the two properties expressed in Claim 4 and Claim 5.

**Claim 4.** *Let  $(i, j) \in C$ . If  $\text{dist}(i, j) \geq k$ , then it holds that*

$$\sum_{t \in T} \delta_t(i, j) = \frac{\text{dist}(i, j) - k + 1}{\text{dist}(i, j)}.$$

*Otherwise, i.e., if  $\text{dist}(i, j) < k$ , then  $\sum_{t \in T} \delta_t(i, j) = 0$ .*

**Claim 5.** *For every  $t \in T$  we have that*

$$\sum_{(i, j) \in C} \delta_t(i, j) \leq \left(1 + \frac{k-1}{k} - \frac{k-1}{p-1}\right) f_t.$$

If Claim 4 and Claim 5 hold, then the following argument, which is based on changing the order of summation, completes the proof of the lower bound.

$$\begin{aligned} \text{OPT}(R) &= \sum_{t \in T} f_t(R) \stackrel{\text{Claim 5}}{\geq} \frac{1}{1 + \frac{k-1}{k} - \frac{k-1}{p-1}} \sum_{t \in T} \sum_{(i, j) \in C} \delta_t(i, j) \\ &= \frac{1}{1 + \frac{k-1}{k} - \frac{k-1}{p-1}} \sum_{(i, j) \in C} \sum_{t \in T} \delta_t(i, j) \\ &\stackrel{\text{Claim 4}}{=} \frac{1}{1 + \frac{k-1}{k} - \frac{k-1}{p-1}} \sum_{\substack{(i, j) \in C, \\ \text{dist}(i, j) \geq k}} \frac{\text{dist}(i, j) - k + 1}{\text{dist}(i, j)} \\ &= \frac{1}{1 + \frac{k-1}{k} - \frac{k-1}{p-1}} \sum_{\ell \geq k} \frac{\ell - k + 1}{\ell} c_\ell. \end{aligned}$$

Before we actually prove Claim 4 and Claim 5 we need additional notation. For every pair  $(i, j) \in C$  define the set

$$F(i, j) = \{t \in F : i < t < j\},$$

i.e., the times when OPT faults between  $i$  and  $j$ . For every  $t \in F(i, j)$ , define the *rank* of the fault at time  $t$  within  $(i, j)$  by

$$\text{rank}_t(i, j) = \|\{s \in F(i, j) : s \geq t\}\|.$$

The intuition is that the fault with rank  $r$  is the  $r$ -th *last* fault within  $(i, j)$ . Let

$$L(i, j) = \{t \in F(i, j) : \text{rank}_t(i, j) \leq \text{dist}(i, j) - k + 1\}$$

denote the set of the (at most)  $\text{dist}(i, j) - k + 1$  *last* faults of  $(i, j)$ .

Define the *value* of  $(i, j)$  at time  $t \in T$  by

$$\text{val}_1(i, j) = \text{dist}(i, j), \tag{2}$$

$$\text{val}_{t+1}(i, j) = \begin{cases} \text{val}_t(i, j) - 1 & \text{if } t \in L(i, j), \\ \text{val}_t(i, j) & \text{otherwise.} \end{cases} \tag{3}$$

Observe that the value of a pair  $(i, j)$  decreases by exactly one at a time whenever one of the  $\text{dist}(i, j) - k + 1$  last faults in  $L(i, j)$  occurs. At all other times, the value remains constant. It is not hard to see that the following equality holds. It relates the rank of a fault at time  $t \in L(i, j)$  with the value at that time:

$$\text{val}_t(i, j) = \text{rank}_t(i, j) + k - 1. \tag{4}$$

Hence we have  $\text{val}_t(i, j) \in \{k, \dots, \text{dist}(i, j)\}$  for all  $t \in L(i, j)$ .

We are now in position to define the function  $\delta_t(i, j)$  based on  $\text{val}_t(i, j)$ :

$$\delta_t(i, j) = \begin{cases} \frac{k-1}{\text{val}_{t+1}(i, j)} - \frac{k-1}{\text{val}_t(i, j)} & \text{if } \text{dist}(i, j) \geq k, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Observe that  $\delta_t(i, j) = 0$  for all times except  $t \in L(i, j)$ . The two crucial properties of this definition are the following. First, for each pair  $(i, j)$  with  $\text{dist}(i, j) \geq k$  we have  $\sum_{t \in T} \delta_t(i, j) = \frac{\text{dist}(i, j) - k + 1}{\text{dist}(i, j)}$ , which will be shown below. Second, observe that (5) is defined in such a way that the smaller the rank of a fault is, the more it contributes to  $\sum_{t \in L(i, j)} \delta_t(i, j)$ . This second property is crucially needed for the proof of Claim 5 below.

*Proof of Claim 4.* Let  $(i, j) \in C$  be a consecutive request pair.

First, let  $\text{dist}(i, j) < k$  and observe that  $L(i, j) = \emptyset$  in this case. Hence (3), and (5) imply  $\sum_{t \in T} \delta_t(i, j) = 0$ .

Second, let  $\text{dist}(i, j) \geq k$ . Observe that the number of faults of OPT within  $(i, j)$  is  $\|F(i, j)\| \geq \text{dist}(i, j) - k + 1$  for all pairs with distance at least  $k$ . This is because there are  $\text{dist}(i, j)$  distinct pages between  $i$  and  $j$  and there are at most  $k - 1$  of these in the cache of OPT at time  $i$ . Hence, for pairs  $(i, j)$  with  $\text{dist}(i, j) \geq k$  the property  $\|F(i, j)\| \geq \text{dist}(i, j) - k + 1$  implies that  $\|L(i, j)\| = \text{dist}(i, j) - k + 1$ .

Then the definitions (1), (2), (3), and (5) imply

$$\begin{aligned}
\sum_{t \in T} \delta_t(i, j) &= \sum_{t \in F} \delta_t(i, j) = \sum_{t \in L(i, j)} \delta_t(i, j) \\
&= \sum_{t \in L(i, j)} \frac{k-1}{\text{val}_{t+1}(i, j)} - \frac{k-1}{\text{val}_t(i, j)} = (k-1) \sum_{v=k}^{\text{dist}(i, j)} \frac{1}{v-1} - \frac{1}{v} \\
&= (k-1) \left( \frac{1}{k-1} - \frac{1}{\text{dist}(i, j)} \right) = (k-1) \frac{\text{dist}(i, j) - k + 1}{(k-1)\text{dist}(i, j)} \\
&= \frac{\text{dist}(i, j) - k + 1}{\text{dist}(i, j)}
\end{aligned}$$

and Claim 4 is established.  $\blacksquare$

Before we proceed to the proof of Claim 5, we give some of the intuition behind it and introduce additional notation.

Let  $t \in F$  be a point in time when an OPT fault occurs. We have to prove that the sum  $\sum_{(i, j) \in C} \delta_t(i, j)$  of all the pairs  $(i, j)$  for which  $\delta_t(i, j) > 0$  is not “too large”. As mentioned above, for every pair  $(i, j)$  with  $\text{dist}(i, j) \geq k$  a fault at time  $t$  contributes the more to  $\sum_{t \in L(i, j)} \delta_t(i, j)$  the smaller the value of that pair is at time  $t$ . Hence it is crucial to prove that there are not “too many” pairs with small value at time  $t$ . This central property is expressed in Lemma 6.

For every  $t \in T$  define the set  $X(t) = \{(i, j) \in C : t \in L(i, j)\}$ . Observe that if no fault occurs at time  $t$ , then  $X(t)$  is empty. Otherwise, i.e., for  $t \in F$ , the set  $X(t)$  comprises all the pairs for which the fault at time  $t$  is one of the  $\text{dist}(i, j) - k + 1$  last faults. Hence, for these pairs the value  $\text{val}_t(i, j)$  decreases by definition (3) at time  $t + 1$ . Furthermore, observe that

$$\sum_{(i, j) \in C} \delta_t(i, j) = \sum_{(i, j) \in X(t)} \delta_t(i, j),$$

i.e., it actually suffices to consider the pairs  $(i, j) \in X(t)$ , because  $\delta_t(v, w) = 0$  for all the other pairs  $(v, w) \notin X(t)$ .

For every  $\ell = 1, \dots, p-1$  define the sets and quantities

$$X_\ell = X_\ell(t) = \{(i, j) \in X(t) : \text{rank}_t(i, j) = \ell\} \quad \text{and} \quad x_\ell = x_\ell(t) = |X_\ell(t)|$$

i.e., the (number of) pairs for which the fault at time  $t$  has rank  $\ell$ .

**Lemma 6.** *For all  $t \in F$  it holds that*

$$\sum_{\ell=1}^r x_\ell(t) \leq r + k - 1$$

for all  $r = 1, \dots, p-k$  and  $x_{p-k+1}(t) = \dots = x_{p-1}(t) = 0$ .

*Proof.* Consider a fixed time  $t \in F$ . For simplicity of notation, we omit the argument  $t$ , e.g., we write  $X_\ell$  and  $X$  instead of  $X_\ell(t)$  and  $X(t)$ .

First observe that (4) implies  $\text{rank}_t(i, j) \leq \text{dist}(i, j) - k + 1 \leq p - k$ . Hence  $X_{p-k+1} = \dots = X_{p-1} = \emptyset$  and  $x_{p-k+1} = \dots = x_{p-1} = 0$ .

The sets  $X_\ell$  partition the pairs  $(i, j) \in X$  according to their rank. Hence the sets  $X_\ell$  induce a disjoint partition of  $X$  and we have  $|\cup_{\ell=1}^r X_\ell| = \sum_{\ell=1}^r |X_\ell| = \sum_{\ell=1}^r x_\ell$ .

For sake of contradiction, fix  $r$  such that  $|\cup_{\ell=1}^r X_\ell$  comprises  $m > r + k - 1$  pairs. Let these be  $(i_1, j_1), \dots, (i_m, j_m)$ . Without loss of generality  $t < j_1 < \dots < j_{m-1} < j_m$ . Observe that all the pages  $r_t, r_{j_1}, \dots, r_{j_m}$  are distinct from each other. Hence, at time  $t$ , the set  $\{r_t, r_{j_1}, \dots, r_{j_{m-1}}, r_{j_m}\}$  comprises exactly  $m + 1$  distinct pages. Therefore *any* algorithm with cache size  $k$  faults at least  $m + 1 - k > r + k - 1 + 1 - k = r$  times. Hence there are *more than*  $r$  faults of OPT within the time range  $t, \dots, j_m$ . Consider the pair  $(i_m, j_m)$  and observe that the fault at time  $t$  within that pair has rank more than  $r$ , i.e.,  $\text{rank}_t(i_m, j_m) > r$ . Hence  $(i_m, j_m) \notin \cup_{\ell=1}^r X_\ell$  yields the desired contradiction.  $\blacksquare$

*Proof of Claim 5.* First observe that for every  $t \in T \setminus F$  we have

$$\sum_{(i,j) \in C} \delta_t(i, j) = 0 \leq \left(1 + \frac{k-1}{k} - \frac{k-1}{p-1}\right) f_t$$

by definition of  $\delta_t(i, j)$ .

Second, let  $t \in F$  and consider the set  $X(t)$ . Recall the disjoint partition  $X(t) = X_1(t) \cup \dots \cup X_{p-k}(t)$ . By (3), (4), and (5) each pair  $(i, j) \in X_\ell(t)$  contributes

$$\delta_t(i, j) = \frac{k-1}{\ell+k-2} - \frac{k-1}{\ell+k-1}$$

to  $\sum_{(i,j) \in X_t} \delta_t(i, j)$ .

Now interpret the  $x_\ell$  as variables of the following linear program, where the constraint (7) is implied from Lemma 6.

$$\text{maximize} \quad \sum_{\ell=1}^{p-k} x_\ell \left( \frac{k-1}{\ell+k-2} - \frac{k-1}{\ell+k-1} \right) \quad (6)$$

$$\text{subject to} \quad \sum_{\ell=1}^r x_\ell \leq r + k - 1 \quad \text{for } r = 1, \dots, p - k \quad (7)$$

Clearly, the optimum value of the linear program (6) immediately yields an upper bound on  $\sum_{(i,j) \in X_t} \delta_t(i, j)$ .

Now we prove that the unique optimal solution to (6) is  $\mathbf{x}^* = (k, 1, \dots, 1)$ . Consider an arbitrary optimal solution  $\mathbf{x} = (x_1, \dots, x_{p-k})$  and let  $v$  be the smallest index such that  $\sum_{\ell=1}^v x_\ell = v + k - 1 - \varepsilon$  for some  $\varepsilon > 0$ . If  $v$  does not exist, there is nothing to prove, because then  $\mathbf{x} = (k, 1, \dots, 1)$ .

If  $v = p - k$  holds, increasing  $d_v$  by  $\varepsilon$  yields the feasible solution  $(k, 1, \dots, 1)$  with larger objective value and hence a contradiction. Otherwise, i.e., for  $v < p - k$ , let  $w$  be an index with  $v < w$  and consider the solution  $\mathbf{x}_\varepsilon = (x_1, \dots, x_v + \varepsilon, \dots, x_w - \varepsilon, \dots, x_{p-k})$ , which is indeed feasible. We have that the objective value of  $\mathbf{x}_\varepsilon - \mathbf{x}$  is

$$\varepsilon(k-1) \left( \frac{1}{(v+k-1)(v+k-2)} - \frac{1}{(w+k-1)(w+k-2)} \right) > 0 \quad \text{by } v < w \text{ and } \varepsilon > 0$$

which is a contradiction. Hence  $\mathbf{x}^* = (k, 1, \dots, 1)$  is the unique optimum solution to (6).

This solution and a simple calculation yields

$$\sum_{(i,j) \in C} \delta_t(i, j) \leq \left(1 + \frac{k-1}{k} - \frac{k-1}{p-1}\right) f_t \leq 2f_t \quad (8)$$

and the proof of Claim 5 is complete.  $\blacksquare$

It is easy to see from the discussion preceding this proof, that the bound is tight for  $(k + 1)$ -multicycles  $(1, 2, \dots, k + 1, 1, 2, \dots, k + 1, \dots)$  because  $p = k + 1$ . As discussed above, the proofs of Claim 4 and Claim 5 imply Theorem 2. Corollary 3 is implied by (8).  $\blacksquare$

### 2.3 The $(\alpha, \beta)$ -Adversary

How does the request sequence of a typical program look like? To answer this question, recall that programs are organized in two parts: *code* and *data*.

The algorithms executed by the program are implemented in the code segment, which is usually small compared to the cache size. By *locality of reference*, most consecutive requests to code pages will have short distance. For example, a lot of time of the control flow is spent in executing loops resulting in an enormous number of requests to few pages.

However, the size of the datastructures needed by an algorithm can be assumed to be large compared to the cache size, e.g., databases, matrices, or graphs. This data is mostly accessed in a structured way, e.g., linearly reading, or in an irregular way, e.g., the queries to a database. In both cases, we expect a *typical memory access pattern*: a consecutive request to a page either has small or large distance compared to the cache size.

We argue that request distances can be divided into three classes: short, long, and uncertain. Intuitively, for short distance, “no” (reasonable) algorithm faults, for long distance, “any” algorithm faults, but it is not clear for the distances inbetween. As discussed below, the treatment of the latter turns out to be crucial for the performance of a strategy.

This observation motivates the definition of the  $(\alpha, \beta)$ -adversary. Let  $k$  denote the cache size, let  $\alpha > 1$  be such that  $\alpha k$  is an integer, and let  $\beta \geq 0$ . Then, the  $(\alpha, \beta)$ -adversary is free to choose any sequence  $R$  in the set  $S_{(\alpha, \beta)}$  defined by

$$S_{(\alpha, \beta)} = \left\{ R \text{ request sequence with } \mathbf{c}(R) \text{ such that } \sum_{\ell=k}^{\alpha k-1} c_{\ell}(R) \leq \beta \sum_{\ell=\alpha k}^{p-1} c_{\ell}(R) \right\}. \quad (9)$$

The set of all sequences is classified according to the parameters  $\alpha$  and  $\beta$ , which is motivated from the above reasoning and has the following intuition. The parameter  $\beta$  controls that there are not “too many” consecutive requests with distance in the *crucial interval*  $[k, \alpha k - 1]$ . To see why this interval is critical, consider an  $\ell + 1$ -multicycle and observe that each request pair has distance  $\ell$ . For these sequences, the competitive ratio of LRU is  $\frac{\ell}{\ell - k + 1}$ , decreasing from  $k$  to  $\frac{\alpha}{\alpha - 1}$  as  $\ell$  grows from  $k$  to  $\alpha k - 1$ .

Notice that we do not restrict the number of requests with very short distance, i.e., in the interval  $[0, k - 1]$ . Hence, our model *implicitly* captures the notion of locality of reference. Observe that the interval  $[\alpha k, p - 1]$  is also not constrained, allowing sequences that have a typical memory access pattern.

Summarizing, the  $(\alpha, \beta)$ -adversary restricts the crucial interval  $[k, \alpha k - 1]$  of request distances; but only mildly. We just require that the total number of crucial requests  $\sum_{\ell=k}^{\alpha k-1} c_{\ell}$  can be balanced with the long-distance requests  $\sum_{\ell=\alpha k}^{p-1} c_{\ell}$ .

For technical reasons, we define the function  $\varepsilon(R) = \frac{p(R)}{\text{OPT}(R)}$  which relates the number of initial faults to the optimum number of faults of the sequence  $R$ . Observe that  $0 \leq \varepsilon(R) \leq 1$  always holds. Define  $\varepsilon = \varepsilon(R)$ , where  $R \in S_{(\alpha, \beta)}$  is a sequence that maximizes  $\frac{\text{LRU}(S)}{\text{OPT}(S)}$  over all  $S \in S_{(\alpha, \beta)}$ .

Legend	
$x$ -axis	$\ell$
$y$ -axis	$c_\ell$ (log-scale)
Left Image	
SPEC	Hydro
purpose	fluid dynamics
pages	$\sim 7 \cdot 10^6$
$k$	1024 (4 MB)
$(\alpha, \beta)$	(16.002, 0.012)
$f(\alpha, \beta)$	2.16
Right Image	
SPEC	Nasa7
purpose	weather sim.
pages	$\sim 3 \cdot 10^7$
$k$	1024 (4 MB)
$(\alpha, \beta)$	(3.97, 0.6)
$f(\alpha, \beta)$	4.28

Figure 1: Characteristic vectors of two SPEC benchmarks

**Theorem 7.** For the  $(\alpha, \beta)$ -adversary and cache size  $k$ , it holds that

$$\text{CR}_{(\alpha, \beta)}(\text{LRU}) \leq f(\alpha, \beta) := 2(1 + \beta) \left( 1 + \frac{1}{\alpha - 1} \right) + \varepsilon.$$

*Proof.* Let  $R \in S_{(\alpha, \beta)}$  be a sequence that maximizes  $\frac{\text{LRU}(S)}{\text{OPT}(S)}$  over all  $S \in S_{(\alpha, \beta)}$  with  $\varepsilon = \varepsilon(R)$  relative contribution of initial faults. Further, Theorem 1, Theorem 2, and (9) yield

$$\frac{\text{LRU}(R)}{\text{OPT}(R)} \leq 2 \frac{\sum_{\ell=k}^{\alpha k-1} c_\ell + \sum_{\ell=\alpha k}^p c_\ell}{\sum_{\ell \geq k} \frac{\ell-k+1}{\ell} c_\ell} + \varepsilon \leq 2 \frac{(1 + \beta) \sum_{\ell=\alpha k}^p c_\ell}{\frac{\alpha-1}{\alpha} \sum_{\ell=\alpha k}^p c_\ell} + \varepsilon = 2(1 + \beta) \left( 1 + \frac{1}{\alpha - 1} \right) + \varepsilon$$

and the proof is complete.  $\blacksquare$

This result is a theoretical underpinning of the good practical behaviour of LRU, since it gives a criterion for the sequences on which the algorithm achieves constant competitive ratio. The bound  $f(\alpha, \beta)$  is constant if  $\alpha$  is “large” and  $\beta$  is “small”. Indeed, the discussion above suggests that practical sequences are in a set  $S_{(\alpha, \beta)}$  with this property.

To substantiate this claim, we performed extensive experiments with memory traces generated by our implementations of standard algorithms and by the execution of SPEC benchmark programs [11]. SPEC benchmarks are standard for performance evaluation in computer industry. They provide programs for a broad variety of applications, e.g., numerical simulations or user programs. We calculated  $\alpha$  and  $\beta$  and obtained that our bound  $f(\alpha, \beta)$  gives values between two and five for these request sequences, see Figure 1.

Further experimental study is needed to determine consistent values of  $\alpha$  and  $\beta$ . However, according to the above reasoning, the size of data needed by a program is large compared to the cache size, i.e., at least a constant factor. We hence expect that  $\alpha$  is “large” in practice. In our experiments, we also monitored the structure of the characteristic vector of these sequences. Figure 1 depicts two representative examples of our results. The  $x$ -axis shows the length  $\ell$  of the consecutive request and the  $y$ -axis shows the value of the respective  $c_\ell$  in log-scale. There is a peak for smaller values of  $\ell$  and there are many large entries  $c_\ell$  for larger values of  $\ell$ , corresponding to short and long distance. This structure suggests that  $\beta$  should be “small” in practice.

## 2.4 The $K$ -Adversary

An alternative approach is to consider variable cache size, which was already studied by Young [12]. He measured the performance of an algorithm with a relaxation of the competitive ratio. We propose an alternative measure, the *expected performance ratio*, see Souza [10] for further reading, and compare it with the expected competitive ratio in a diffuse adversary model.

We argue that the expected competitive ratio loses an important property of competitive analysis: the *instance-wise* comparison of a certain algorithm against an optimum algorithm. Theorem 8 is evidence that the expected competitive ratio can be misleading due to this fact.

The  $K$ -adversary is completely free to choose a request sequence, i.e.,  $S_K$  is not restricted, but the cache size  $K$  is uniform distributed  $K \sim \text{Uni}\{a, \dots, b\}$  and beyond the power of the adversary.

The *expected competitive ratio* ECR and the *expected performance ratio* EPR of an algorithm ALG against the  $K$ -adversary are defined by

$$\text{ECR}_K(\text{ALG}) = \max \left\{ \frac{\mathbb{E}[\text{ALG}(R)]}{\mathbb{E}[\text{OPT}(R)]} : R \in S_K \right\} \quad \text{and} \quad \text{EPR}_K(\text{ALG}) = \max \left\{ \mathbb{E} \left[ \frac{\text{ALG}(R)}{\text{OPT}(R)} \right] : R \in S_K \right\},$$

where each expectation is taken over the random cache size  $K$ .

**Theorem 8.** *For the  $K$ -adversary with cache size  $K \sim \text{Uni}\{a, \dots, b\}$ , and  $a \geq 2$  it holds that*

$$a \leq \text{ECR}_K(\text{LRU}) \leq 2a + 1 \quad \text{and} \quad \text{EPR}_K(\text{LRU}) = \frac{a + b}{2}.$$

The proof of Theorem 8 is deferred to Appendix A.1. The value of the expected *competitive* ratio varies from  $\Theta(1)$  to  $\Theta(b)$  as  $a$  varies from one to  $b$ . The expected *performance* ratio is  $\Theta(b)$ , independent of  $a$ . Both bounds are tight for sequences  $R$  with the characteristic vector  $\mathbf{c}(R)$

$$c_\ell(R) = x^{b-\ell+1} \text{ for } \ell = a, \dots, b, \quad \text{and} \quad c_\ell(R) = 0 \text{ for all other } \ell,$$

where  $x$  is a free variable. Then LRU indeed exhibits its worst-case behaviour because we have  $\lim_{x \rightarrow \infty} \frac{\text{LRU}_k(R)}{\text{OPT}_k(R)} = k$  for every cache size  $k \in \{a, \dots, b\}$ . The bound  $\text{EPR}_K(\text{LRU}) = \Theta(b)$  reflects this result, but  $\text{ECR}_K(\text{LRU}) = \Theta(a)$  does *not*, if we fix  $a$  and let  $b$  grow.

We draw two conclusions from the  $K$ -adversary model. First, variable cache size and an unrestricted adversary are *not* appropriate to explain the good performance of LRU in practice because  $\text{EPR}_K(\text{LRU}) = \Theta(b)$ . Hence we argue that assumptions on the sequence are necessary and justify restrictions, e.g., the  $(\alpha, \beta)$ -adversary. Second, the expected *competitive* ratio can be *misleading* in the following sense. Although there exist sequences such that LRU yields competitive ratio  $k$  on every cache size  $k \in \{a, \dots, b\}$ , we have that  $\text{ECR}_K(\text{LRU})$  is constant if  $a$  is chosen to be constant. LRU appears to perform quite well in this measure against an unrestricted adversary with variable cache size, although the opposite is the case. The reason for this inconsistency is that algorithms are not compared instance-wise, which can be seen in the proof of Theorem 8.

## 2.5 Outlook

An interesting outlook of this paper is that the proof of the lower bound, Theorem 2, could also lead to new insights concerning the long-standing  $k$ -server conjecture, which was raised by Manasse McGeoch and Sleator [7]. The  $k$ -server problem is a natural generalization of the paging problem, where the cost of moving a page into the cache may differ from one. The conjecture states that there exists a  $k$  competitive algorithm for this problem. See, e.g., Koutsoupias and Papadimitriou [6] for further reading.

**Acknowledgement.** We thank Angelika Steger and especially Riko Jacob for carefully reading earlier versions of the manuscript and Justus Schwartz for discussions and help with our experiments.

## References

- [1] BECCHETTI, L. Modeling locality: A probabilistic analysis of LRU and FWF. In *12th Annual European Symposium on Algorithms (ESA '04)* (2004), pp. 98 – 109.
- [2] BELADY, L. A. A study of replacement algorithms for virtual storage computers. *IBM Systems Journal* 5 (1966), 78 – 101.
- [3] BORODIN, A., RAGHAVAN, P., IRANI, S., AND SCHIEBER, B. Competitive paging with locality of reference. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC '91)* (1991), pp. 249 – 259.
- [4] FIAT, A., AND MENDEL, M. Truly online paging with locality of reference. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS '97)* (1997), pp. 326 – 335.
- [5] KOUTSOUPIAS, E., AND PAPADIMITRIOU, C. Beyond competitive analysis. *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS '94)* (1994), 394 – 400.
- [6] KOUTSOUPIAS, E., AND PAPADIMITRIOU, C. On the  $k$ -server conjecture. *Journal of the ACM* 42, 5 (1995), 971 – 983.
- [7] MANASSE, M. S., MCGEOCH, L. A., AND SLEATOR, D. D. Competitive algorithms for on-line problems. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC '88)* (1988), pp. 322 – 333.
- [8] SITES, R. L., AND AGARWAL, A. Multiprocessor cache analysis using ATUM. In *15th IEEE International Symposium on Computer Architecture* (1988), pp. 186 – 195.
- [9] SLEATOR, D. D., AND TARJAN, R. E. Amortized efficiency of list update and paging rules. *Communications of the ACM* 28, 2 (1985), 202 – 208.
- [10] SOUZA, A. *Average Performance Analysis*. PhD thesis, ETH Zürich, to appear.
- [11] SPEC – STANDARD PERFORMANCE EVALUATION CORPORATION. [www.spec.org](http://www.spec.org).
- [12] YOUNG, N. E. On-line caching as cache size varies. In *Proceedings of the 2nd ACM-SIAM Symposium on Discrete Algorithms (SODA '91)* (1991), pp. 241 – 250.
- [13] YOUNG, N. E. Bounding the diffuse adversary. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms (SODA '98)* (1998), pp. 420 – 425.
- [14] YOUNG, N. E. On-line paging against adversarially biased random inputs. *Journal of Algorithms* 37, 1 (2000), 218 – 235.

## A Appendix – Omitted Proofs

### A.1 Proofs of Section 2

Before we actually prove Theorem 8, we state Lemma 9 and its variant Corollary 10 which are needed therein.

**Lemma 9 (Omitting Lemma).** *Let  $\mathbf{x} = (x_1, \dots, x_n) \geq \mathbf{0}$  and  $\mathbf{y} = (y_1, \dots, y_n) > \mathbf{0}$  be such that*

$$\frac{x_1}{y_1} \geq \dots \geq \frac{x_n}{y_n} \geq 0.$$

*Then we have*

$$\frac{x_1 + x_2 + \dots + x_n}{y_1 + y_2 + \dots + y_n} \leq \frac{x_1 + x_2 + \dots + x_{n-1}}{y_1 + y_2 + \dots + y_{n-1}} \leq \dots \leq \frac{x_1 + x_2}{y_1 + y_2} \leq \frac{x_1}{y_1}.$$

*Proof.* For every  $i \in \{1, \dots, n\}$ , define the values  $n_i = \sum_{j=1}^i x_j$ ,  $d_i = \sum_{j=1}^i y_j$ , and  $r_i = \frac{n_i}{d_i}$ . Observe that the claimed assertion  $r_i \geq r_{i+1}$  is equivalent to  $r_i \geq \frac{x_{i+1}}{y_{i+1}}$  because

$$r_i - r_{i+1} = \frac{n_i}{d_i} - \frac{n_i + x_{i+1}}{d_i + y_{i+1}} = \frac{n_i y_{i+1} - d_i x_{i+1}}{d_i d_{i+1}} \geq 0 \quad \text{if and only if} \quad \frac{n_i}{d_i} = r_i \geq \frac{x_{i+1}}{y_{i+1}}.$$

We proceed by induction on the assertion  $r_i \geq \frac{x_{i+1}}{y_{i+1}}$ . The base case  $i = 1$  is trivial:

$$\frac{x_1}{y_1} - \frac{x_1 + x_2}{y_1 + y_2} = \frac{x_1 y_2 - x_2 y_1}{y_1 (y_1 + y_2)} \geq 0 \quad \text{if and only if} \quad \frac{x_1}{y_1} \geq \frac{x_2}{y_2}.$$

For the inductive case, we prove that  $r_{i+1} \geq \frac{x_{i+2}}{y_{i+2}}$  follows from  $r_i \geq \frac{x_{i+1}}{y_{i+1}}$  and  $\frac{x_i}{y_i} \geq \frac{x_{i+1}}{y_{i+1}} \geq \frac{x_{i+2}}{y_{i+2}}$ . First,  $r_{i+1} \geq \frac{x_{i+2}}{y_{i+2}}$  holds if and only if

$$n_i y_{i+2} + x_{i+1} y_{i+2} - x_{i+2} y_{i+1} - d_i x_{i+2} \geq 0. \tag{10}$$

Second, the assumption  $\frac{x_{i+1}}{y_{i+1}} \geq \frac{x_{i+2}}{y_{i+2}}$ , i.e.,  $x_{i+1} y_{i+2} - x_{i+2} y_{i+1} \geq 0$  implies that (10) is true if  $n_i y_{i+2} - d_i x_{i+2} \geq 0$  or equivalently  $\frac{x_{i+2}}{y_{i+2}} \leq \frac{n_i}{d_i} = r_i$ . The induction hypothesis implies

$$r_i = \frac{n_i}{d_i} \geq \frac{x_{i+1}}{y_{i+1}} \geq \frac{x_{i+2}}{y_{i+2}}$$

and the proof is complete. ■

**Corollary 10.** *Let  $\mathbf{x} = (x_1, \dots, x_n) \geq \mathbf{0}$ ,  $\mathbf{y} = (y_1, \dots, y_n) > \mathbf{0}$ , and let  $\pi$  denote a permutation of  $(1, \dots, n)$ . Then we have*

$$\frac{\sum_{i=1}^n x_i}{\sum_{i=1}^n y_i} \leq \min_{\pi} \max \left\{ \frac{x_i}{y_{\pi(i)}} : i = 1, \dots, n \right\} \leq \max \left\{ \frac{x_i}{y_{\pi(i)}} : i = 1, \dots, n, \text{ and fixed } \pi \right\}.$$

*Proof of Theorem 8.* First note that we may assume characteristic vectors  $\mathbf{c} = (c_a, \dots, c_b)$  without loss of generality, i.e.,  $\ell$  is in the range  $a, \dots, b$ . If there is a  $c_\ell > 0$  with  $\ell < a$ , it does not contribute, neither to LRU nor to OPT. If we want to prove an upper bound of  $\frac{\text{LRU}}{\text{OPT}}$  and there is a  $c_\ell > 0$  with  $\ell > b$ , then we can simply modify the characteristic vector  $\mathbf{c}$  to  $\mathbf{c}'$  by setting  $c'_b = c_b + c_\ell$  and  $c'_\ell = 0$ .

This change does not affect LRU and only decreases the lower bound for OPT. If we want to prove a lower bound of  $\frac{\text{LRU}}{\text{OPT}}$  we can choose a characteristic vector anyway.

The upper bound on  $\text{EPR}_K(\text{LRU})$  is easy to prove, because  $\frac{\text{LRU}}{\text{OPT}} \leq k$  holds. We find

$$\mathbb{E} \left[ \frac{\text{LRU}}{\text{OPT}} \right] \leq \frac{1}{b-a+1} \sum_{k=a}^b k = \frac{a+b}{2}.$$

The lower bound on  $\text{EPR}_K(\text{LRU})$  is obtained for sequences  $R$  with the following characteristic vector with the free variable  $x$

$$c_\ell = x^{b-\ell+1} \text{ for } \ell = a, \dots, b. \quad (11)$$

Consider for example a collection of multicycles with this characteristic vector. For this sequence  $R$ , we find that  $\text{LRU}(R) \geq \sum_{l \geq k} c_\ell$  and  $\text{OPT}(R) \leq \sum_{\ell \geq k} \frac{\ell-k+1}{\ell} c_\ell$ . Therefore, as  $x$  tends to infinity, we obtain

$$\begin{aligned} \mathbb{E} \left[ \frac{\text{LRU}}{\text{OPT}} \right] &\geq \frac{1}{b-a+1} \sum_{k=a}^b \frac{\sum_{l \geq k} c_\ell}{\sum_{l \geq k} \frac{\ell-k+1}{\ell} c_\ell} \\ &= \frac{1}{b-a+1} \sum_{k=a}^b \frac{\sum_{l \geq k} x^{b-\ell+1}}{\sum_{l \geq k} \frac{\ell-k+1}{\ell} x^{b-\ell+1}} = \frac{1}{b-a+1} \sum_{k=a}^b k \\ &= \frac{a+b}{2}. \end{aligned}$$

For the upper bound on  $\text{ECR}_K(\text{LRU})$ , we first derive the following facts by changing the index of summation:

$$\begin{aligned} \mathbb{E}[\text{LRU}] &= \frac{1}{b-a+1} \sum_{k=a}^b \sum_{\ell \geq k} c_\ell + p = \sum_{\ell=a}^b (\ell-a+1) c_\ell + p \\ \mathbb{E}[\text{OPT}] &\geq 2 \frac{1}{b-a+1} \sum_{k=a}^b \sum_{\ell \geq k} \frac{\ell-k+1}{\ell} c_\ell = 4 \sum_{\ell=a}^b \frac{(\ell-a+1)(\ell-a+2)}{\ell} c_\ell \end{aligned}$$

With Corollary 10, and the trivial lower bound  $\text{OPT} \geq p$ , we obtain

$$\begin{aligned} \frac{\mathbb{E}[\text{LRU}]}{\mathbb{E}[\text{OPT}]} &\leq 1 + 4 \frac{\sum_{\ell=a}^b (\ell-a+1) c_\ell}{\sum_{\ell=a}^b \frac{(\ell-a+1)(\ell-a+2)}{\ell} c_\ell} \\ &\leq 1 + 4 \max \left\{ \frac{\ell}{\ell-a+2} : \ell = a, \dots, b \right\} \\ &= 1 + 2a, \end{aligned}$$

as  $a \geq 2$ .

The lower bound on  $\text{ECR}_K(\text{LRU})$  is obtained for sequences with the same characteristic vector (11) as for the lower bound  $\text{EPR}_K(\text{LRU}) = \Omega(b)$ , for example. As  $x$  tends to infinity, we obtain  $\frac{\mathbb{E}[\text{LRU}]}{\mathbb{E}[\text{OPT}]} \geq a$  and the proof is complete.  $\blacksquare$