Reordering Buffer Management for Non-Uniform Cost Models*

Matthias Englert and Matthias Westermann

Department of Computer Science
RWTH Aachen, D-52056 Aachen, Germany
{englert,marsu}@cs.rwth-aachen.de

Abstract. A sequence of objects which are characterized by their color has to be processed. Their processing order influences how efficiently they can be processed: Each color change between two consecutive objects produces non-uniform cost. A reordering buffer which is a random access buffer with storage capacity for k objects can be used to rearrange this sequence in such a way that the total cost are minimized. This concept is useful for many applications in computer science and economics.
We show that a reordering buffer reduces the cost of each sequence by a factor of at most $2k - 1$. This result even holds for cost functions modeled by arbitrary metric spaces. In addition, a matching lower bound is presented. From this bound follows that each strategy that does not increase the cost of a sequence is at least $(2k - 1)$-competitive.
As main result, we present the deterministic Maximum Adjusted Penalty (MAP) strategy which is $O(\log k)$-competitive. Previous strategies only achieve a competitive ratio of k in the non-uniform model. For the upper bound on MAP, we introduce a basic proof technique. We believe that this technique can be interesting for other problems.

1 Introduction

Frequently, a number of tasks has to be processed and their processing order influences how efficiently they can be processed. Hence, a reordering buffer can be expedient to influence the processing order. This concept is useful for many applications in computer science and economics. In the following and in Sect. 1.2 we give some examples.

In computer graphics, a rendering system displays a 3D scene which is composed of primitives. In current rendering systems a significant factor for the performance are the state changes performed by the graphics hardware. A state change occurs when two consecutively rendered primitives differ in their attribute values, e.g., in their texture or shader program. These state changes slow down a rendering system. Note that the duration of a state change is non-uniform and heavily depends on the attribute values of the primitive causing this state change, e.g., textures and shader programs vary significantly in size which has

* Supported by the DFG grant WE 2842/1.

a great impact on the state change. To reduce the cost of the state changes, a reordering buffer can be included between application and graphics hardware. This reordering buffer, which is a random access buffer with limited memory capacity, can be used to rearrange the incoming sequence of primitives online in such a way that the cost of the state changes are minimized.

Hard disks consist of one or more rotating platters. A read/write head is positioned above the rotating surface of each platter. The position of a head determines which cylinder can be accessed. The latency of an access is mainly induced by the movement of the head to the respective cylinder. The latencies are the dominating factor for the performance of a hard disk. This can be modeled by a non-uniform metric, e.g., the line metric: Accesses are categorized according to their destination cylinder, and the cost are defined as the distance between start and destination cylinder. A reordering buffer can be used to rearrange the incoming sequence of accesses online in such a way that latencies are minimized. This problem is known as disk scheduling (see, e.g., [1]).

File servers are high-capacity storage devices which each computer in a network can access to retrieve files. A file on a server is denoted as open, if it is ready to be accessed. Otherwise, it is denoted as closed. By technical reasons, the number of open files on a server is limited. The overhead induced by the opening and closing processes is a significant factor for the performance of a file server. Note that the cost of an opening or closing process is non-uniform and depends on the characteristics of the involved file. This overhead can be minimized by preceding a file server with a reordering buffer. If several open files are allowed, this scenario is a generalization of the rendering scenario. In addition to the possibility to choose a file that has to be opened next, there is the possibility to choose a file that has to be closed in the case that the maximum number of files is open. This scenario is equivalent to the classical paging problem (see, e.g., [2]) if the reordering buffer has storage capacity for only one file.

1.1 The Model

An *input sequence* $\sigma = \sigma_1\sigma_2\cdots$ of objects which are only characterized by a specific attribute has to be processed. To simplify matters, we suppose that the objects are characterized by their color, and, for each object σ_i, let $c(\sigma_i)$ denote the color of σ_i. A *reordering buffer* which is a random access buffer with storage capacity for k objects can be used to rearrange the input sequence in the following way.

The *current input object* σ_i, i.e., the first object of σ that is not handled yet, can be stored in the reordering buffer, or objects currently stored in the reordering buffer can be removed. These removed objects result in an *output sequence* $\sigma_{\pi^{-1}} = \sigma_{\pi^{-1}(1)}\sigma_{\pi^{-1}(2)}\cdots$ which is a partial permutation of σ. Let the *current output object* denote the object that was last assigned to the output sequence. We suppose that the reordering buffer is initially empty and, after processing the whole input sequence, the buffer is empty again.

For each color c, we are given *weight* b_c. Cost b_c are produced for each color change to color c, i.e., for two consecutive objects $\sigma_{\pi^{-1}(i)}$ and $\sigma_{\pi^{-1}(i+1)}$ of the

output sequence, we define the cost function $d(\sigma_{\pi^{-1}(i)}, \sigma_{\pi^{-1}(i+1)}) = b_{c(\sigma_{\pi^{-1}(i+1)})}$, if $c(\sigma_{\pi^{-1}(i)}) \neq c(\sigma_{\pi^{-1}(i+1)})$, and $d(\sigma_{\pi^{-1}(i)}, \sigma_{\pi^{-1}(i+1)}) = 0$, otherwise. Then, the goal is to minimize the *cost* $C^A(\sigma) = \sum_i d(\sigma_{\pi^{-1}(i)}, \sigma_{\pi^{-1}(i+1)})$ *of a management strategy* A. Note that this models the presented application examples well except disk scheduling (see also Sect. 1.3).

The notion of an online strategy is intended to formalize the realistic scenario, where the strategy does not have knowledge about the whole input sequence in advance. The online strategy has to serve the input sequence σ one after the other, i.e., a new object is not issued before there is a free slot in the reordering buffer. Online strategies are typically evaluated in a competitive analysis. In this kind of analysis the cost of the online strategy are compared with the cost of an optimal offline strategy. For a given sequence σ, let $C^{\text{op}}(\sigma)$ denote the minimum cost produced by an optimal offline strategy. An online strategy is denoted as α-*competitive* if it produces cost at most $\alpha \cdot C^{\text{op}}(\sigma) + \kappa$, for each sequence σ, where κ is a term that does not depend on σ. The value α is also called the *competitive ratio* of the online strategy.

W.l.o.g., we only consider *lazy* strategies, i.e., strategies that fulfill the following two properties:

- If an object with the same color as the current output object is stored in the reordering buffer, a lazy strategy does not make a color change.
- If the current input object can be stored in the reordering buffer, a lazy strategy does not remove an object from the reordering buffer.

Note that every (in particular every optimal offline) strategy can be transformed into a lazy strategy without increasing the cost.

1.2 Previous Work

Web caching with request reordering extends the classic paging model by allowing reordering of requests under the constraint that a request is delayed by no longer than a predetermined number of time steps (see, e.g., [3, 4]). Albers [3] presents a deterministic strategy that achieves an optimal competitive ratio of $k+1$, where k denotes the storage capacity of the cache. Feder et al. [4] introduce a randomized strategy that achieves an asymptotically optimal competitive ratio of $\Theta(\log k)$.

The uniform case of our problem is studied in, e.g., [5, 6]. In the uniform model, for each color c, weight $b_c = 1$, i.e., just the number of color changes is considered. Räcke, Sohler and Westermann [5] show that several standard strategies are unsuitable for a reordering buffer, i.e., the competitive ratio of the First In First Out and Least Recently Used strategy is $\Omega(\sqrt{k})$ and the competitive ratio of the Most Common First strategy is $\Omega(k)$, where k denotes the buffer size. Further, the deterministic Bounded Waste strategy is presented and it is proven that this strategy achieves a competitive ratio of $O(\log^2 k)$ in the uniform model. Kohrt and Pruhs [6] present a polynomial-time offline algorithm that achieves a constant approximation ratio. However, their goal is to maximize

the number of saved color changes. Note that a constant approximation of the minimal number of color changes in the output sequence is preferable, if it is possible to save a large number of color changes.

Krokowski et al. [7] examine the previously mentioned rendering application in an uniform version, i.e., just the number of state changes is considered. They use a small reordering buffer to rearrange the incoming sequence of primitives online in such a way that the number of state changes is minimized. Due to its simple structure and its low memory requirements this method can easily be implemented in software or even hardware. In their experimental evaluation this method typically reduces the number of state changes by an order of magnitude and the rendering time by roughly 30%. Note that the studied strategies do not consider the individual cost of a state change. A conclusion is that there is a lack of efficient strategies that consider these individual cost.

In the painting shop of a car plant, a sequence of cars bodies traverses the final layer painting where each car body is painted with its own top coat. If two consecutive cars have to be painted in different colors then a color change is required which causes set-up cost. In addition to the color change cost, further important non-uniform cost arise, e.g., the individual accessing times of the parking slots for the car bodies. These costs can be minimized by preceding the final layer painting with a reordering buffer. In several practical work, heuristic strategies for reordering buffers are evaluated by simulation (see, e.g., [8]). Efficient strategies for reordering buffer are considered to be a major problem of operating a paint shop.

1.3 Results and Further Work

In Sect. 2, the possible gain of a reordering buffer is investigated. We show that a reordering buffer of size k reduces the cost of each sequence by a factor of at most $2k - 1$. This result holds for online and offline strategies and even for cost functions modeled by arbitrary metric spaces, i.e., the cost function $d(\sigma_{\pi^{-1}(i)}, \sigma_{\pi^{-1}(i+1)})$ can be any positive and symmetric function obeying the triangle inequality. In addition, a matching lower bound is presented. From this basic upper bound follows immediately that each strategy that does not increase the cost of an input sequence is at least $(2k - 1)$-competitive. In particular, the simple online strategy that does no reordering at all is already $(2k - 1)$-competitive. This shows the poor performance of some strategies. For example, Yeh et al. [9] give a lower bound of $2k - 1$ on the competitive ratio of the disk scheduling strategies Shortest Seek Time First and Look in the line metric model.

In Sect. 3, we show a lower bound of k on the competitive ratio of the Bounded Waste (BW) strategy in our model, where k denotes the size of the reordering buffer. The BW strategy is introduced by Räcke, Sohler and Westermann [5] for the uniform case of our problem: For each color c, weight $b_c = 1$, i.e., just the number of color changes is considered. Note that this lower bound even holds for the case that BW takes the non-uniform cost into account, i.e., BW is aware of the individual cost b_c, for each color c. In Sect. 4, the deterministic Maximum Adjusted Penalty (MAP) strategy is presented. We show that the MAP strategy

is $O(\log k)$-competitive, where k denotes the size of the reordering buffer. Note that, although MAP is equivalent to BW in the uniform case of our model, our analysis provides a better result. Currently, we only know a trivial lower bound of 5/3 on the competitive ratio of any deterministic strategy.

For the upper bound on MAP we introduce the following basic proof technique: First, it is shown that MAP with buffer size k is 4-competitive against an optimal offline strategy with buffer size $k/4$. Finally, it is proven that an optimal offline strategy with buffer size $k/4$ is $O(\log k)$-competitive against an optimal offline strategy with buffer size k. We believe that this technique can be interesting for other problems.

Our non-uniform scenario can be modeled by the following star-like metric space: $d(x, y) = (b_x + b_y)/2$, if $x \neq y$, and $d(x, y) = 0$, otherwise. Above, we conclude that there is a lack of efficient strategies for the disk scheduling problem, i.e., the line metric space. We consider to transfer this technique from star-like to line metric spaces.

2 Basic Upper Bound

In this section, we show that a reordering buffer of size k reduces the cost of each sequence by a factor of at most $2k - 1$. This result holds for online and offline strategies and even for cost functions modeled by arbitrary metric spaces. Note that this result is tight. Fix the two colors c_1 and c_2 with weights $b_{c_1} = b_{c_2} = 1$. The input sequence $\sigma = (c_1 c_2)^k$ of length $2k$ can obviously be reordered to $\sigma_{\pi^{-1}} = c_1^k c_2^k$ with a reordering buffer of size k. The cost of σ is $C(\sigma) = 2k - 1$, and the cost of $\sigma_{\pi^{-1}}$ is $C(\sigma_{\pi^{-1}}) = 1$. Hence, $C(\sigma) = (2k - 1) \cdot C(\sigma_{\pi^{-1}}) \geq (2k - 1) \cdot C^{\mathrm{op}}(\sigma)$, where $C^{\mathrm{op}}(\sigma)$ denotes the cost of an optimal offline strategy using a reordering buffer of size k.

Theorem 1. *For every metric space (M, d), and every input sequence $\sigma = \sigma_1 \cdots \sigma_l$, with $\sigma_i \in M$,*

$$C(\sigma) \leq (2k - 1) \cdot C^{\mathrm{op}}(\sigma) \ ,$$

where $C(\sigma)$ denotes the cost of σ, and $C^{\mathrm{op}}(\sigma)$ denotes the cost of an optimal offline strategy using a reordering buffer of size k.

Proof. Fix an input sequence $\sigma = \sigma_1 \cdots \sigma_l$. Let $\sigma_{\pi^{-1}} = \sigma_{\pi^{-1}(1)} \cdots \sigma_{\pi^{-1}(l)}$ denote the output sequence of an optimal offline strategy using a reordering buffer of size k. We define a *subsequence* $I_r^s = \sigma_{\pi^{-1}(r)} \cdots \sigma_{\pi^{-1}(s)}$, if $r \leq s$, and $I_r^s = \sigma_{\pi^{-1}(s)} \cdots \sigma_{\pi^{-1}(r)}$, otherwise. Let $C(I_r^s)$ denote the cost of this subsequence, i.e., $C(I_r^s) = \sum_{j=\min\{r,s\}}^{\max\{r,s\}-1} d(\sigma_{\pi^{-1}(j)}, \sigma_{\pi^{-1}(j+1)})$.

Due to the triangle inequality, $d(\sigma_i, \sigma_{i+1}) \leq C(I_{\pi(i)}^{\pi(i+1)})$. Thus,

$$C(\sigma) = \sum_{i=1}^{l-1} d(\sigma_i, \sigma_{i+1}) \leq \sum_{i=1}^{l-1} C(I_{\pi(i)}^{\pi(i+1)}) \ .$$

Fix two consecutive objects $\sigma_{\pi^{-1}(j)}$ and $\sigma_{\pi^{-1}(j+1)}$ of $\sigma_{\pi^{-1}}$. We show that these objects do only occur in at most $2k - 1$ of the subsequences above. If

$\sigma_{\pi^{-1}(j)}$ and $\sigma_{\pi^{-1}(j+1)}$ are part of a subsequence $I_{\pi(i)}^{\pi(i+1)}$, one of the following two cases is true:

1. $\pi(i) \leq j$ and $\pi(i+1) \geq j+1$
2. $\pi(i+1) \leq j$ and $\pi(i) \geq j+1$

In case (1), σ_i is one of the first j objects and σ_{i+1} is not under the first j objects of $\sigma_{\pi^{-1}}$. In case (2), it is the other way around.

Obviously, the following observation can be made.

Observation 2. *For each input sequence $\sigma = \sigma_1 \cdots \sigma_l$, the output sequence of a reordering buffer of size k is a permutation $\sigma_{\pi^{-1}} = \sigma_{\pi^{-1}(1)} \cdots \sigma_{\pi^{-1}(l)}$ of σ, with $\pi^{-1}(i) < i + k$, for each i. In addition, each such permutation can be generated using a reordering buffer of size k.*

The observation shows that only one of the first $i + k - 1$ objects of σ can be placed at the i-th position of $\sigma_{\pi^{-1}}$. Thus, we conclude for case (1) that $i \leq j + k - 1$. In the same way, we conclude for case (2) the even stronger inequality $i + 1 \leq j + k - 1$.

In the following, we consider case (1). The above conclusions provide that σ_{i+1} must be one of the first $j + k$ objects of σ. But, σ_{i+1} is not one of the first j objects of $\sigma_{\pi^{-1}}$. Recall that the observation shows that the first j objects of $\sigma_{\pi^{-1}}$ have to be under the first $j+k-1$ objects of σ. Hence, at most k objects of the first $j + k$ objects of σ cannot be under the first j objects of $\sigma_{\pi^{-1}}$. It follows that case (1) is true for at most k different subsequences. Obviously, case (2) can be addressed analogously. It follows that case (2) is true for at most $k - 1$ different subsequences.

Hence,

$$C(\sigma) \leq \sum_{i=1}^{l-1} C(I_{\pi(i)}^{\pi(i+1)}) \leq (2k-1) \sum_{i=1}^{l-1} d(\sigma_{\pi^{-1}(i)}, \sigma_{\pi^{-1}(i+1)}) = (2k-1) \cdot C^{\mathrm{op}}(\sigma) \ ,$$

since at most $2k - 1$ subsequences are containing the two objects $\sigma_{\pi^{-1}(i)}$ and $\sigma_{\pi^{-1}(i+1)}$. □

3 Lower Bound for the BW Strategy

The Bounded Waste (BW) strategy is introduced in [5] for the uniform case of our model: For each color c, weight $b_c = 1$, i.e., just the number of color changes is considered. BW chooses one color as the active color, and continues to remove objects of this active color from the reordering buffer until all objects in the buffer have a color different from the active color. Then a new active color has to be chosen. For this purpose, a counter P_c, which is initially set to zero, is assigned to each color c. At each color change, the counter of each color c is increased by the number of objects of color c currently stored in the buffer. Then a color c' with maximal counter $P_{c'}$ is chosen as the new active color and $P_{c'}$ is reset to zero.

The following theorem shows a lower bound of k, where k denotes the size of the reordering buffer, on the competitive ratio of BW in our non-uniform model. Note that this lower bound even holds for the case that BW takes the non-uniform weights of the colors into account.

Theorem 3. *The competitive ratio of BW is at least k, where k denotes the size of the reordering buffer.*

Proof. Fix one expensive color x with weight $b_x = 1$, and several inexpensive colors $c_1, \dots c_l$ with weights $b_{c_1} = \cdots = b_{c_l} = \varepsilon$. The input sequence $\sigma = \sigma_1 \cdots \sigma_l$ with $l = k \cdot (k+1)$ is defined as follows: σ_i is of color x, if i is divisible by $(k+1)$, and of color c_i, otherwise. Obviously, it is possible to produce an output sequence $\sigma^{\mathrm{op}}_{\pi^{-1}}$ with cost $C(\sigma^{\mathrm{op}}_{\pi^{-1}}) = 1 + (k^2 - 1) \cdot \varepsilon$ by aggregating objects of the expensive color.

Each pair of objects in σ has different colors, except the pairs where both objects have the color x. However, BW will never hold two objects of color x in the reordering buffer at the same time. Assume this statement holds until some step i in which an object of color x arrives. If each pair of objects in the buffer of BW has different colors, the values of the counters are exclusively based on the number of steps the corresponding objects are stored in the buffer. The object of color x is after $k - 1$ steps the oldest object in the buffer. Hence, this object is removed from the buffer in the next step, i.e., one step before the next object of color x arrives.

The cost of the produced output sequence $\sigma^{\mathrm{on}}_{\pi^{-1}}$ is $C(\sigma^{\mathrm{on}}_{\pi^{-1}}) = C(\sigma) = k + k^2 \cdot \varepsilon$. Since the input sequence σ can be iterated and ε can be chosen arbitrarily small, this yields a lower bound of k on the competitive ratio. Of course, BW could take the non-uniform weights of the colors into account, i.e., the counter P_c is increased by the number of objects of color c currently stored in the buffer times the weight of color c. However, this proof does not dependent on this decision. □

4 The MAP Strategy

In this section, we present the Maximal Adjusted Penalty (MAP) strategy. We show that the MAP strategy is $O(\log k)$-competitive, where k denotes the size of the reordering buffer.

MAP chooses one color as the active color, and removes at each time step one object of this active color from the reordering buffer until all objects in the buffer have a color different from the active color. Then a new active color has to be chosen. For this purpose, a penalty counter P_c, which is initially set to zero, is assigned to each color c. MAP chooses a color c as the new active color with $P_c - k \cdot b_c \geq P_{c'} - k \cdot b_{c'}$, for each color c'. The counters are updated after a new active color is chosen. Suppose a step in which a color change from color x to color y occurs. Let n_c denote the number of objects of color c stored in the buffer at the beginning of this step. Then each counter P_c is increased by $n_c \cdot b_y$ and counter P_x is reset to zero.

The MAP strategy does not need to know the weights of all colors in advance. It is sufficient to provide the weight b_c when the first object of color c arrives. In addition, a counter P_c can be deleted if no objects of color c are stored in the buffer. Hence, each step can be performed in time $O(k)$, since at most k counters are active at the same time.

Theorem 4. *The MAP strategy is $O(\log k)$-competitive, where k denotes the size of the reordering buffer.*

Proof. The proof consists of two parts.

1. First, we prove that MAP with buffer size k is 4-competitive against an optimal offline strategy with buffer size $h = k/4$.
2. Finally, we show that an optimal offline strategy with buffer size h is $O(\log k)$-competitive against an optimal offline strategy with buffer size k.

Together, this yields the theorem.

Part 1. Fix an input sequence σ and a lazy optimal offline strategy OPT. MAP has a reordering buffer of size k and OPT has a reordering buffer of size h. We exclude the last k color changes of MAP. Hence, it can be assumed that there are k objects in the buffer of MAP at any time. Under this assumption we show that MAP is 4-competitive against OPT. This yields part 1, since the last k color changes of MAP produce at most cost $k \cdot \max_{\text{color } c}\{b_c\}$.

Color changes of MAP and OPT are denoted as online and offline color changes, respectively. An *online (offline) c-interval* starts with an online (offline) color change from color c to a different color and ends right before the next online (offline) color change from c to a different color (the first online (offline) c-interval starts with the first step). Each object of color c falls into exactly one online and one offline c-interval, and it enters and leaves the buffer of the respective strategy in the same c-interval. Also each step i falls into exactly one online and one offline c-interval, and these intervals are denoted as *active* at step i.

Now, we introduce counters to which k-times the cost of each online color change is assigned. For each color c, and each online c-interval I, the two counters $w_c^{\text{on},I}$ and $\hat{w}_c^{\text{on},I}$ are introduced. $w_c^{\text{on},I}(i)$ and $\hat{w}_c^{\text{on},I}(i)$ denote the value of $w_c^{\text{on},I}$ and $\hat{w}_c^{\text{on},I}$ at the beginning of step i, respectively. The counters are initially set to zero, and they are monotonously increasing. A counter is denoted as active at step i, if the according online c-interval I is active at step i. Otherwise, the counter is denoted as inactive. For simplicity, we just write $w_c^{\text{on}}(i)$ and $\hat{w}_c^{\text{on}}(i)$ to denote the active counters for color c at the beginning of step i. Inactive counters do not change their value.

Fix a step i. In the following, we describe how k-times the cost of an online color change is distributed among the counters. Let $n_c^{\text{on}}(i)$ denote the number of objects with color c in the buffer of MAP at the beginning of step i. For simplicity, we just write n_c^{on}, if the step is fixed. Note that $\sum_{\text{color } c} n_c^{\text{on}} = k$. Suppose there is an online color change to color c' in step i. Then $k \cdot b_{c'}$ has to

be assigned to the counters. For each color c, w_c^{on} is increased by $n_c^{\text{on}} \cdot b_{c'}$. In total, we assign $\sum_{\text{color } c} n_c^{\text{on}} \cdot b_{c'} = k \cdot b_{c'}$ to active counters.

But we prevent a counter w_c^{on} from becoming larger than $k \cdot b_c$. This restriction might cause that nothing or only a part of the value $n_c^{\text{on}} \cdot b_{c'}$ is really assigned to w_c^{on}. The remaining part, i.e, the part that would lead to a counter w_c^{on} larger than $k \cdot b_c$, is assigned to $\hat{w}_{c'}^{\text{on}}$ instead. Note that w_c^{on} equals the P_c counter in MAP, as long as $P_c \leq k \cdot b_c$. Otherwise, w_c^{on} remains on the value $k \cdot b_c$, but P_c is further increased. Note in addition that, for each color c' and each online c'-interval I, the counter $\hat{w}_{c'}^{\text{on},I}$ is increased in at most one step.

Since we have assigned k-times the produced cost to counters, we can express the cost of MAP $C^{\text{on}} = (1/k) \cdot \sum_{\text{color } c} \sum_{\text{on. } c\text{-int. } I} (W_c^{\text{on},I} + \hat{W}_c^{\text{on},I})$, where $W_c^{\text{on},I}$ and $\hat{W}_c^{\text{on},I}$ denote the final, i.e., maximum, value of the counters $w_c^{\text{on},I}$ and $\hat{w}_c^{\text{on},I}$, respectively.

In addition, for each color c and each online c-interval I, the counter $w_c^{\text{op},I}$ is introduced. $w_c^{\text{op},I}(i)$ denotes the value of $w_c^{\text{op},I}$ at the beginning of step i. The counters are initially set to zero, and they are monotonously increasing. A counter is denoted as active at step i, if the according online c-interval I is active at step i. Otherwise, the counter is denoted as inactive. For simplicity, we just write $w_c^{\text{op}}(i)$ to denote the active counter for color c at the beginning of step i. Inactive counters do not change their value.

Fix a step i. Let $n_c^{\text{op}}(i)$ denote the number of objects with color c in the buffer of OPT at the beginning of step i. For simplicity, we just write n_c^{op}, if the step is fixed. Note that $\sum_{\text{color } c} n_c^{\text{op}} = h$. Suppose there is an online color change to color c' in step i. For each color c, w_c^{op} is increased by $n_c^{\text{op}} \cdot b_{c'}$. In total, we assign $\sum_{\text{color } c} n_c^{\text{op}} \cdot b_{c'} = h \cdot b_{c'}$ to active counters.

Hence, we yield a new possibility to express the cost of MAP $C^{\text{on}} = (1/h) \cdot \sum_{\text{color } c} \sum_{\text{on. } c\text{-int. } I} W_c^{\text{op},I}$, where $W_c^{\text{op},I}$ denotes the final, i.e., maximum, value of the counter $w_c^{\text{op},I}$.

For each color c, we show the following *main inequality*

$$4k \cdot C_c^{\text{op}} + \sum_{\text{on. } c\text{-int. } I} (4W_c^{\text{op},I} - (W_c^{\text{on},I} + \hat{W}_c^{\text{on},I})) \geq \sum_{\text{on. } c\text{-int. } I} (W_c^{\text{on},I} + \hat{W}_c^{\text{on},I}) \ ,$$

where C_c^{op} denotes the total cost produced by offline color changes to color c. Summing up over all colors, we yield $4k \cdot C^{\text{op}} + 4h \cdot C^{\text{on}} - k \cdot C^{\text{on}} \geq k \cdot C^{\text{on}}$. Hence, $4C^{\text{op}} \geq C^{\text{on}}$. This yields part 1.

We distinguish between two kinds of online c-intervals. An online c-interval I is denoted as *problematic*, if $4W_c^{\text{op},I} < 2(W_c^{\text{on},I} + \hat{W}_c^{\text{on},I})$. Otherwise, I is denoted as *non-problematic*. Now, we show the following inequality

$$4k \cdot C_c^{\text{op}} \geq 2 \sum_{\text{prob. } c\text{-int. } I} (W_c^{\text{on},I} + \hat{W}_c^{\text{on},I}) \ .$$

Obviously, the main inequality can be concluded with the help of the above inequality.

The following lemma provides an upper bound on $W_c^{\text{on},I}$ and $\hat{W}_c^{\text{on},I}$. Then, an upper bound on the number of problematic c-intervals is shown. These two results together complete part 1 of the proof.

Lemma 5. *For each color c and each online c-interval I,*

$$\hat{W}_c^{\mathrm{on},I} \leq W_c^{\mathrm{on},I} \leq k \cdot b_c \ .$$

Proof. Due to the cost assignment for $w_c^{\mathrm{on},I}$, $W_c^{\mathrm{on},I} \leq k \cdot b_c$. Suppose that $\hat{W}_c^{\mathrm{on},I} > 0$. Recall that $\hat{w}_c^{\mathrm{on},I}$ is increased in at most one step. We consider the step i in which $\hat{w}_c^{\mathrm{on},I}$ is increased due to the online color change to color c in the online c-interval I. In this step, for each color c', $P_c - k \cdot b_c \geq P_{c'} - k \cdot b_{c'}$.

Now, we distinguish the following two cases.

- Suppose that $W_c^{\mathrm{on},I} = k \cdot b_c$.
  The color change in step i produces cost b_c. Hence, $k \cdot b_c$ is assigned to the counters in this step. Even if the whole value $k \cdot b_c$ is assigned to $\hat{w}_c^{\mathrm{on},I}$, $\hat{W}_c^{\mathrm{on},I} \leq k \cdot b_c = W_c^{\mathrm{on},I}$, since $\hat{w}_c^{\mathrm{on},I}$ is increased in at most one step.
- Suppose that $W_c^{\mathrm{on},I} < k \cdot b_c$.
  In this case, $P_c = w_c^{\mathrm{on}}(i)$ in step i. In fact, for each color c', $P_{c'} = w_{c'}^{\mathrm{on}}(i)$ at the beginning of step i, since $w_{c'}^{\mathrm{on}}(i) < k \cdot b_{c'}$. If, for some color $c' \neq c$, $w_{c'}^{\mathrm{on}}(i) = k \cdot b_{c'}$, MAP would have chosen color c' as new active color in step i.
  Of course, for a color $c' \neq c$, the active counter $w_{c'}^{\mathrm{on}}$ can reach its limit $k \cdot b_{c'}$ in this step. $w_{c'}^{\mathrm{on}}$ exceeds its limit by $x_{c'} = \max\{0, w_{c'}^{\mathrm{on}}(i) + n_{c'}^{\mathrm{on}}(i) \cdot b_c - k \cdot b_{c'}\}$. Due to MAP, $w_{c'}^{\mathrm{on}}(i) - k \cdot b_{c'} \leq w_c^{\mathrm{on}}(i) - k \cdot b_c$. Hence, $x_{c'} \leq \max\{0, w_c^{\mathrm{on}}(i) + n_{c'}^{\mathrm{on}}(i) \cdot b_c - k \cdot b_c\}$.
  Let V denote the set of all colors c' with $x_{c'} > 0$. If $V = \emptyset$, $\hat{W}_c^{\mathrm{on},I} = 0$. Otherwise,

$$\begin{aligned} \hat{W}_c^{\mathrm{on},I} &= \sum_{c' \in V} x_{c'} \leq |V| \cdot (w_c^{\mathrm{on}}(i) - k \cdot b_c) + \sum_{c' \in V} n_{c'}^{\mathrm{on}}(i) \cdot b_c \\ &\leq w_c^{\mathrm{on}}(i) - k \cdot b_c + \sum_{c' \in V} n_{c'}^{\mathrm{on}}(i) \cdot b_c \leq w_c^{\mathrm{on}}(i) \leq W_c^{\mathrm{on},I} \ , \end{aligned}$$

  since $w_c^{\mathrm{on}}(i) - k \cdot b_c < 0$.

  This finishes the proof of the lemma. □

The beginning of an offline c-interval I, before the offline color change to color c occurs, is denoted as increasing phase, since the number of objects of color c in the offline buffer is monotonously increasing. The remaining part of I, after the offline color change to color c, is denoted as decreasing phase, since the number of objects of color c in the offline buffer is monotonously decreasing.

Lemma 6. *At most one problematic online c-interval starts in an offline c-interval.*

Proof. Fix a problematic online c-interval I. From Lem. 5 follows $4W_c^{\mathrm{op},I} < 4W_c^{\mathrm{on},I}$. Hence, there exists at least one step i in I with $n_c^{\mathrm{op}}(i) < n_c^{\mathrm{on}}(i)$. Let I' denote the offline c-interval in which I starts.

Suppose I starts in the increasing phase of I'. Let start(I) denote the first step of interval I. Then $n_c^{\text{op}}(\text{start}(I)) \geq 0 = n_c^{\text{on}}(\text{start}(I))$. Hence, no step i with $n_c^{\text{op}}(i) < n_c^{\text{on}}(i)$ can exist in this increasing phase, since every arriving object of color c is stored in the offline buffer and no objects of color c are removed from the offline buffer.

Consider the decreasing phase of I'. If n_c^{on} is decreased, then n_c^{op} is decreased by the same amount. Hence, if there exists a step i with $n_c^{\text{op}}(i) < n_c^{\text{on}}(i)$ in the decreasing phase of I', the offline c-interval I' ends before the end of the problematic online c-interval I. □

The total number of offline c-intervals is C_c^{op}/b_c. We can exclude the first offline c-interval, if c is the color of the first object in the output sequence of OPT, since the total produced cost by MAP in the only problematic interval starting in this offline c-interval can be bounded by a term independent of σ. Note that we exclude, for only one color c, an offline c-interval. From the lemma above it follows that the total number of problematic online c-intervals is at most C_c^{op}/b_c. Then

$$\begin{aligned} 2 \sum_{\text{prob. } c\text{-int. } I} (W_c^{\text{on},I} + \hat{W}_c^{\text{on},I}) &\leq 2 \sum_{\text{prob. } c\text{-int. } I} k \cdot b_c + k \cdot b_c \\ &\leq 4k \cdot b_c \cdot C_c^{\text{op}}/b_c = 4k \cdot C_c^{\text{op}} \ . \end{aligned}$$

This completes part 1 of the proof.

Part 2. It remains to show, that an optimal offline strategy with buffer size $h = k/4$ is $O(\log k)$-competitive against an optimal offline strategy with buffer size k. Fix an input sequence σ. For each step i, let $n_c^h(i)$ denote the number of objects of color c in the buffer of size h and let $n_c^k(i)$ denote the number of objects of color c in the buffer of size k.

Fix a lazy optimal offline strategy LARGE for the reordering buffer of size k. The offline strategy SMALL for the reordering buffer of size h chooses a new active color c, with $n_c^h(i) \geq n_c^k(i)/4$. Note that there exists always such a color, since $\sum_{\text{color } c} n_c^h(i) = \sum_{\text{color } c} n_c^k(i)/4$.

Large (small) c-intervals are defined for LARGE (SMALL) according to the definition of online and offline c-intervals. The definitions of increasing and decreasing phases apply to large c-intervals, too. Note that the total number of large (small) c-intervals is order of the total number of color changes to color c of LARGE (SMALL).

For every color c, we show that there are at most $O(\log k)$ small c-intervals in one large c-interval. Then, $C_c^h \leq O(\log k) \cdot C_c^k$, where C_c^h (C_c^k) denotes the cost of LARGE (SMALL) for color c. This yields part 2 of the proof.

Fix a color c. In the following, we only consider small c-intervals that are completely contained in a large c-interval. In addition, we exclude a small c-interval, if LARGE performs a color change to color c during this interval. Hence, in total at most two small c-intervals are excluded for every large c-interval. The

remaining small c-intervals are completely contained either in an increasing or in a decreasing phase of a large c-interval.

The following lemma shows that there are at most $O(\log k)$ small c-intervals in a large c-interval, since the buffer size of LARGE is k.

Lemma 7. *Let* $\text{start}(I)$ *and* $\text{end}(I)$ *denote the first and last step of a small* c*-interval* I*, respectively.*

- *In an increasing phase:* $n_c^k(\text{end}(I)) \geq (5/4)n_c^k(\text{start}(I))$.
- *In a decreasing phase:* $n_c^k(\text{start}(I)) \geq (5/4)n_c^k(\text{end}(I))$.

Proof. We only prove the inequality for the increasing phase. The inequality for the decreasing phase can be addressed analogously. Fix a small c-interval I. Let i be the step in I at which SMALL performs a color change to color c. Due to SMALL, $n_c^h(i) \geq n_c^k(i)/4$.

At least $n_c^h(i)$ objects of color c arrive in I. Since I lies in an increasing phase, these objects are not removed from the buffer of LARGE during I. At the end of I, the $n_c^k(\text{start}(I))$ objects of color c stored in the buffer of LARGE before I, are still there, and at least $n_c^h(i) \geq n_c^k(i)/4 \geq n_c^k(\text{start}(I))/4$ new objects of color c are added. Hence, at least $(5/4)n_c^k(\text{start}(I))$ objects of color c are stored in the buffer of LARGE at the end of I. □

This completes part 2 of the proof. □

References

1. Teorey, T., Pinkerton, T.: A comparative analysis of disk scheduling policies. Communications of the ACM **15** (1972) 177–184
2. Fiat, A., Karp., R.M., Luby, M., McGeoch, L.A., Sleator, D.D., Young, N.E.: Competitive paging algorithms. Journal of Algorithms **12** (1991) 685–699
3. Albers, S.: New results on web caching with request reordering. In: Proceedings of the 16th ACM Symposium on Parallel Algorithms and Architectures (SPAA). (2004) 84–92
4. Feder, T., Motwani, R., Panigrahy, R., Seiden, S., van Stee, R., Zhu, A.: Combining request scheduling with web caching. Theoretical Compuer Science **324** (2004) 201–218
5. Räcke, H., Sohler, C., Westermann, M.: Online scheduling for sorting buffers. In: Proceedings of the 10th European Symposium on Algorithms (ESA). (2002) 820–832
6. Kohrt, J., Pruhs, K.: A constant approximation algorithm for sorting buffers. In: Proceedings of the 6th Latin American Symposium on Theoretical Informatics (LATIN). (2004) 193–202
7. Krokowski, J., Räcke, H., Sohler, C., Westermann, M.: Reducing state changes with a pipeline buffer. In: Proceedings of the 9th International Fall Workshop Vision, Modeling, and Visualization (VMV). (2004) 217–224
8. Gutenschwager, K., Spieckermann, S., Voss, S.: A sequential ordering problem in automotive paint shops. International Journal of Production Research **42** (2004) 1865–1878
9. Yeh, T., Kuo, C., Lei, C., Yen, H.: Competitive analysis of on-line disk scheduling. Theory of Computing Systems **31** (1998) 491–506