

An Improved Competitive Algorithm for Reordering Buffer Management[§]

Noa Avigdor-Elgrabli[¶]

Yuval Rabani^{||}

Abstract

We design and analyze an on-line reordering buffer management algorithm with improved $O\left(\frac{\log k}{\log \log k}\right)$ competitive ratio for non-uniform costs, where k is the buffer size. This improves on the best previous result (even for uniform costs) of Englert and Westermann (ICALP 2005) giving $O(\log k)$ competitive ratio, which was also the best (off-line) polynomial time approximation guarantee for this problem. Our analysis is based on an intricate dual fitting argument using a linear programming relaxation for the problem that we introduce in this paper.

1 Introduction

Problem statement and motivation. In the reordering buffer management problem [23], a stream of colored items arrives at a service station equipped with a buffer that has a limited storage capacity of k items. The buffer is used to permute the input stream (in a limited way) in order to minimize the context switching cost, which is incurred whenever there is a color change between consecutive items served. More specifically, when the buffer fills up (after receiving the first k items), an item from the buffer is chosen to be evicted and served, and then the next input item enters the buffer. At this point, the next output item is chosen and evicted from the buffer, and another input item enters the buffer. The eviction process goes on until the buffer is empty (this happens k steps after the input stream is exhausted). The total cost depends on the color changes in the output sequence. We denote the length of the input stream by N and the set of colors present in the input stream by C .

There are quite a number of compelling applications

that motivate this problem, in areas such as production engineering, shipping, network optimization, file servers, computer graphics, storage systems, and information retrieval [23, 18, 22, 24, 7]. For example, a node in a computer network may delay temporarily some outgoing communication streams in order to merge streams with the same destination and thus reduce the overhead required to initiate a connection. To suit the generic model, different applications may require different service cost functions. The simplest model is uniform costs. Each color change costs 1. A more complicated model involves non-uniform costs. The cost of changing the color to c is $w(c)$, depending on c . Both models may apply to the above communication stream merging example, depending on the appropriate cost model for establishing a connection. To illustrate further the versatility of this model, consider two additional settings discussed in the literature. The first setting is an automotive paint shop [18], where switching paint colors between consecutive cars costs fixed cleanup and setup time and materials. In this case uniform costs apply. The second setting is that of a 3D graphic rendering engine [22], where a change in attributes between consecutive rendered polygons slows down the graphic processor as the rendering program needs to be replaced. Here non-uniform costs are appropriate, because the size of the rendering program that needs to be loaded depends on the attributes.

In most applications, it is more reasonable to assume that the input stream has to be handled on-line, so the decision on which item to evict from the buffer has to be taken before the rest of the input sequence is known. Moreover, in the on-line setting the problem is fundamentally appealing due to its simplicity and elegance, the difficulty encountered in attempting to solve it, and its interpretation as a natural and well-motivated model generalizing lookahead. (Notice that if $k = 1$ then the output stream is identical to the input stream, and if $k = N$, then the entire input stream can be stored and then permuted optimally, even by an online algorithm.) We follow previous work on the problem and adopt the pervasive notion of *competitive analysis* to evaluate the performance of on-line algorithms for reordering buffer

[§]Research supported by Israel Science Foundation grant number 1109/07.

[¶]Computer Science Department, Technion—Israel Institute of Technology, Haifa 32000, Israel. Email: noaelg@cs.technion.ac.il.

^{||}The Rachel and Selim Benin School of Computer Science and Engineering, The Hebrew University of Jerusalem, Jerusalem 91904, Israel. Email: yrabani@cs.huji.ac.il.

management. In other words, we compare the cost of an algorithm's output to that of an optimal off-line solution, and bound the worst case approximation guarantee of the on-line algorithm.

Our contribution. We give an $O\left(\frac{\log k}{\log \log k}\right)$ -competitive deterministic on-line algorithm, named *threshold or lowest cost* (TLC), for reordering buffer management with non-uniform costs. This improves the previous best on-line as well as off-line approximation guarantees known for the problem with either uniform or non-uniform costs (see below). The analysis of TLC is an intricate dual fitting argument. We introduce a new linear programming relaxation for reordering buffer management, and we use it to lower bound the cost of an optimal off-line algorithm. Our on-line algorithm does not use the relaxation. It is only used in the analysis. We use the algorithm to construct a partial solution to the dual program. We then use a second duality argument to show that the partial solution can be completed to form a feasible dual solution whose value is close to the cost incurred by the algorithm.

Previous work. Racke, Sohler, and Westermann [23] introduced the reordering buffer management problem. They demonstrated that the problem with uniform costs is substantially distinct from seemingly similar on-line problems, such as paging, by showing that standard algorithms (FIFO, LRU) perform poorly in this case ($\Omega(\sqrt{k})$ competitive ratio).¹ Racke et al. proposed a new algorithm named *bounded waste* (BW) and proved that with uniform costs it is $O(\log^2 k)$ -competitive. Englert and Westermann [13] later presented an algorithm named *maximum adjusted penalty* (MAP) that generalizes BW to handle the non-uniform case (BW performs poorly in this case). They also improved the competitive analysis to $O(\log k)$. The analyses of [23, 13] differ appreciably from each other and also from our analysis. For example, the analysis of [13] relates the performance of MAP to that of an optimal solution using a smaller buffer of size $\frac{k}{4}$, and then shows that an optimal solution for a buffer of size k can gain at most a factor of $O(\log k)$ over the solution for a buffer of size $\frac{k}{4}$. Aboud [1] proved that the analysis of the latter step in this argument is asymptotically tight, and therefore it seems that a different approach is indeed needed in order to improve those results.² Our algorithm is similar to but not identical to MAP. We do not know if our analysis can be modified to apply to MAP.

¹*Largest color first* is $\Omega(k)$ -competitive [23], whereas doing nothing (leaving the input stream as is) is exactly $(2k - 1)$ -competitive [13].

²In fact, the argument remains tight even if the optimal cost for a size $\frac{k}{4}$ buffer is replaced by the value of our relaxation for such a buffer.

The above algorithms and analysis also provide the best off-line approximation guarantees for reordering buffer management with uniform or non-uniform costs prior to our work. Kohrt and Pruhs [21] gave constant factor (off-line) approximation algorithms for the complement objective of maximizing the number of color changes in the input stream that are avoided in the output stream. Their approximation guarantees were later improved by Bar-Yehuda and Laserson [5]. No non-trivial lower bounds are known for any of the above problems. It is not known if the problems are polynomial-time solvable or NP-hard, and there are no lower bounds on their competitive ratio. This is also true of the specific algorithms analyzed, including our own. These algorithms might actually have a much better competitive ratio than what was proven about them, possibly even a constant competitive ratio.

Some applications require other cost functions, and those were addressed in several papers. The upper bounds on the competitive ratio that these papers prove depend on the number of colors or on the length of the input stream, and not only on the buffer size. The case where the colors are points in a line metric, and a color change costs the distance between the two colors corresponds to a problem of disk arm scheduling. This was first addressed by Khandekar and Pandit [19, 20], who gave a quasi-polynomial time constant factor off-line approximation algorithm, and an $O(\log^2 |C|)$ -competitive randomized on-line algorithm for the problem.³ Gamzu and Segev [17] gave an improved $O(\log |C|)$ -competitive deterministic on-line algorithm for evenly spaced points on a line, and an $O(\log N \log \log N)$ -competitive deterministic on-line algorithm for the continuous line. They also gave a 2.154 lower bound on the competitive ratio of the line problem. This is the only non-trivial lower bound on the competitive ratio of reordering buffer management under *any* cost function. Finally, Englert, Racke, and Westermann [12] considered the problem where the colors are points in an arbitrary metric space. This general case is known to be NP-hard, as the special case of a buffer with infinite capacity is simply the metric traveling salesman problem. Englert et al. gave an $O(\log^2 k \log |C|)$ -competitive randomized on-line algorithm for this case. Their algorithm is based on an $O(\log^2 k)$ -competitive deterministic on-line algorithm for hierarchically separated tree (HST) metrics (see [6]) and both the randomness and the $O(\log |C|)$ factor stem from approximating arbitrary metrics by distributions on HSTs (see [14]). When adapted to the non-uniform costs model, their algorithm is determin-

³The on-line algorithms proposed for uniform and non-uniform costs, including our own, are all deterministic.

istic and gives an alternative $O(\log k)$ guarantee. The proof involves a rather intricate potential function argument, a different approach from either the analysis of MAP or the analysis of our algorithm.

Several other problems with a similar flavor were discussed in the literature. The k -client problem [3] is a multi-threaded variant of the reordering buffer problem. Each position in the buffer is fed by a separate input stream (but there's a single output stream). The choice of item to evict affects the next input item, so intuitively it seems that the adversary generating the input streams has greater power in this case. Our results give the first rigorous demonstration of this intuition: there is an $\Omega(\log k)$ lower bound on the competitive ratio of any deterministic algorithm for the k -client problem,⁴ even in the uniform costs case [3], whereas our reordering buffer algorithm beats this bound. Another rather closely related problem was proposed in [11]. The paper analyzes the classical on-line multiprocessor makespan minimization problem with a reordering buffer prepended to the scheduling process. Thus, computing the cost of the output stream is itself a hard optimization problem. Finally, alternative reordering models also make sense in the context of some applications. For example, the web caching with reordering problem [15, 2], motivated by the application of caching web pages to speed up access, offers an alternative reordering model, where an input request can be delayed up to k steps. Clearly, a buffer of size k can generate all such permutations, but it can also generate additional permutations that delay some requests more than k steps.

Linear programming relaxations have been used in the past in the competitive analysis of on-line algorithms, most notably in the on-line primal-dual schema introduced in [4, 9, 16] (see also [8] for additional references). The approach applies to covering/packing linear programs (possibly with some ad-hoc adjustment). The algorithms use the linear programs and their duals explicitly to form a good fractional solution on-line. The fractional solution is then rounded (often using randomness) on-the-fly to solve the original problem. Our relaxation does not seem to fit into their framework. Moreover, we derive a deterministic algorithm. We note that dual fitting, the tool that we use in our analysis, is a common technique in analyzing off-line approximation algorithms; this tool, too, was previously used primarily to analyze greedy algorithms for problems that can be relaxed to covering/packing linear programs, such as set cover (see [25, Chapter 13] and see [10] for an example in on-line computing).

⁴Also, the same paper gives $(2k - 1)$ -competitive algorithms.

2 Preliminaries

We denote the input sequence by $1, 2, \dots, N$. The color of input element i is $c(i)$. For an input element i let $p(i)$ (respectively, $n(i)$) denote the previous (respectively, next) input element with color $c(i)$. Let $\text{last}(i)$ (respectively, $\text{first}(i)$) denote the last (respectively, first) input element with color $c(i)$. We use the convention $p(\text{first}(i)) = 0$ for all i . For the first k steps input elements are stored in the buffer and no element is output, so the elements are output in steps $k+1, k+2, \dots, k+N$. Given a feasible permutation π (a permutation of the input sequence that can be realized using a buffer of size k), we denote by $e_\pi(j)$ the input element output at step j , and by $t_\pi(i)$ the output step in which the input element i is output. In other words $e_\pi(j) = \pi^{-1}(j - k)$ and $t_\pi(i) = \pi(i) + k$. Whenever π is clear from the context we omit the subscript π from the above notation, and use $e(j), t(i)$.

Our algorithm maintains and updates a counter for every input element. We will denote the value of i 's counter in step j by φ_j^i . For every i , φ_j^i is monotonically non-decreasing in j , and $\varphi_j^i = 0$ for all $j < i$. We denote by B_j the content of the buffer that the algorithm holds in step j .

3 The TLC Algorithm

The algorithm has an active color at each step. As long as the buffer contains an element of the active color, the algorithm outputs the first such element. When the buffer no longer contains such an element the algorithm chooses a new active color. The new active color is chosen using the counters φ_j^i , where j is the current step, as follows. (Recall that $w(c)$ is the cost of changing the active color to c .) If there is a color c such that $\sum_{i \in B_j \wedge c(i)=c} \varphi_j^i \geq w(c)$ then choose any such color c to be the new active color (free change). Otherwise, choose a color c with the smallest $w(c)$ such that there is an element of color c in the buffer (paid change). In the latter case, update the counters as follows: add $\frac{w(c)}{k}$ to the counter of every element in the buffer. To simplify notation we use φ^i to denote $\varphi_{t(i)}^i$, which is the maximum counter value of element i . Let C_{TLC} denote the total cost of the algorithm's solution (on the instance being discussed).

FACT 3.1. *For every set I of elements of the same color c that are in the buffer at the same time,*

$$\sum_{i \in I} \varphi^i < 2 \cdot w(c).$$

Proof. Assume for contradiction that there exists such a set I , and $\sum_{i \in I} \varphi^i \geq 2 \cdot w(c)$. Let j be the step in which

there is a color change to color c' , $\sum_{i \in I} \varphi_{j-1}^i < w(c)$ and $\sum_{i \in I} \varphi_j^i \geq w(c)$. Because this is a paid change and the algorithm chooses the color in the buffer with the minimum cost, $w(c') \leq w(c)$. Moreover, it holds that

$$\sum_{i \in I} \varphi_j^i \leq \sum_{i \in I} \left(\varphi_{j-1}^i + \frac{w(c')}{k} \right).$$

As $|I| < k$,

$$\sum_{i \in I} \varphi_j^i \leq \sum_{i \in I} \varphi_{j-1}^i + w(c') < w(c) + w(c') \leq 2 \cdot w(c).$$

Notice that after step j the counters of the elements in I do not increase, since every color change between step j and the step that the algorithm outputs I is a free color change. Therefore, $\sum_{i \in I} \varphi^i = \sum_{i \in I} \varphi_j^i < 2 \cdot w(c)$, in contradiction to our assumption. ■

FACT 3.2.

$$C_{\text{TLC}} \leq 2 \cdot \sum_{i=1}^N \varphi^i.$$

Proof. Let C_1, C_2 denote the total cost of the free and paid color changes, respectively. Let J_1, J_2 be the steps in which those color changes occur. Let $j \in J_1$ be a color change to color c_j . It holds that $\sum_{i \in B_j \wedge c(i)=c_j} \varphi_j^i \geq w(c_j)$. Moreover $C_1 = \sum_{j \in J_1} w(c_j)$. Therefore,

$$C_1 = \sum_{j \in J_1} w(c_j) \leq \sum_{j \in J_1} \sum_{i \in B_j \wedge c(i)=c_j} \varphi_j^i \leq \sum_{i=1}^N \varphi^i,$$

where the last inequality follows because all elements i for which φ_j^i contributes to the summation are removed from the buffer in sequence starting at step j . Therefore each element appears in the sum at most once. Let $j \in J_2$ be a color change to color c . Then for each element $i \in B_j$, φ_j^i increases by $\frac{w(c)}{k}$. At each step there are k elements in the buffer. Therefore, the total counters value increases by $w(c)$, and thus $C_2 \leq \sum_{i=1}^N \varphi^i$. Finally, we sum the two parts to get

$$C_{\text{TLC}} = C_1 + C_2 \leq 2 \cdot \sum_{i=1}^N \varphi^i,$$

which completes the proof. ■

4 Competitive Analysis

In order to analyze TLC we characterize the optimal offline solution using the following integer linear program, which we denote by IP. For every input element i and

for every output time slot $j \geq i$ we have a 0-1 variable $x_{i,j}$ that indicates if i is output in step j .

$$\begin{aligned} z_{\text{IP}} = \min & \sum_{c \in C} w(c) + \sum_{i \neq \text{last}(i)} \sum_{j=\max\{i, k+1\}}^{n(i)-2} w(c(i)) \cdot x_{i,j} \\ \text{s.t.} & \sum_{j=\max\{i, k+1\}}^{k+N} x_{i,j} \geq 1 \quad \forall i \\ & \sum_{i=1}^j x_{i,j} \leq 1 \quad \forall j \\ & x_{n(i),j} - x_{i,j-1} \geq 0 \quad \forall i \neq \text{last}(i); \forall j \geq n(i) \\ & x_{i,j} \in \{0, 1\} \quad \forall i; \forall j \geq i, \end{aligned}$$

Informally, the first and second sets of constraints imply that every element i is output exactly once and that in every time step exactly one element is output. The third set of constraints implies that elements of the same color are output in the same order they arrived, and that there will be no color change as long as the buffer contains an element of the active color. The second term in the objective function implies a cost of $w(c)$ whenever an element of color c is output before the next element arrives. (Note that this requires a color change in the output sequence.) The additive (first) term pays for the last time each color is output (so there is no next element of this color in the input sequence). Formally, we prove the following proposition.

PROPOSITION 4.1. *For every input sequence, $z_{\text{IP}} = C_{\text{OPT}}$.*

Proof. We first show that given an optimal solution OPT of cost C_{OPT} , we can construct a solution x to IP with the same cost, proving that $C_{\text{OPT}} \geq z_{\text{IP}}$. We set $x_{i,j}$ to indicate if i is output at step j . Thus, for every element

$$i, \sum_{j=\max\{i, k+1\}}^{k+N} x_{i,j} = 1 \geq 1. \text{ For every step } j \geq k+1,$$

$$\sum_{i=1}^j x_{i,j} = 1 \leq 1, \text{ as OPT outputs a single element in step } j.$$

For any optimal solution we may assume that for any color, the order of the elements of this color in the input sequence is preserved in the output sequence ([21, Lemma 2]). Moreover, at any step of the solution if there is an element of the color last served in the buffer, the solution will output this element ([20]). Therefore, if i is output at step $j-1$ and $n(i)$ entered the buffer before step j , then $n(i)$ is still in the buffer at step j as it wasn't served before i , and it must be served at step j as it is the next element of the same color. Thus, if $x_{i,j-1} = 1$ and $j \geq n(i)$ then $x_{n(i),j} = 1$, so the third set of constraints of IP is also satisfied. OPT pays $w(c)$

at step j when there is a color change from color c at that step. This is equivalent to the model of paying $w(c)$ when changing to color c . A color change (from color $c(i)$) occurs when i is output at step j ($x_{i,j} = 1$) and $n(i)$ is still not in the buffer in the following step ($n(i) > j + 1$), or if $i = \text{last}(i)$. Therefore

$$C_{\text{OPT}} = \sum_{c \in C} w(c) + \sum_{i \neq \text{last}(i)} \sum_{j=\max\{i, k+1\}}^{n(i)-2} w(c(i)) \cdot x_{i,j}.$$

Next, we show that given a solution x to IP we can construct a reordering buffer solution to our instance. At any step j we will remove the element i for which $x_{i,j} = 1$. Notice that from the first and second sets of constraints it follows that in every step $j, j \geq k+1$ there is exactly one element i for which $x_{i,j} = 1$, and for every element i there is exactly one step j for which $x_{i,j} = 1$. From these two observations it follows that the number of elements in the buffer (k) is maintained as long as the input sequence is not finished. Moreover every element i enters the buffer at step i . As $x_{i,j}$ is defined only for $j \geq i$,

it holds that if $x_{i,j} = 1$ then the element i is in the buffer at step j . Therefore the solution we constructed is well defined and feasible. The constructed solution pays $w(c)$ at step j when there is a color change from color c at that step. Such a color change occurs iff there exists an element i of color c , such that $x_{i,j} = 1$ and either $i = \text{last}(i)$, or $i \neq \text{last}(i)$ and $x_{n(i),j+1} = 0$. The first case only happens once for each color c and it is taken into account by the additive term $\sum_{c \in C} w(c)$. The second case can only happen if $n(i) > j + 1$ (otherwise $x_{n(i),j+1} - x_{i,j} = -1$ for $j + 1 \geq n(i)$ in contradiction to the IP third set of constraints). Therefore the IP solution also pays $w(c(i)) \cdot x_{i,j} = w(c)$. Thus, we conclude that the cost of the IP solution x is equal to the cost of the solution we constructed. ■

We lower bound z_{IP} using the obvious linear programming relaxation LP of IP: replace the constraints $x_{i,j} \in \{0, 1\}$ by $x_{i,j} \geq 0, \forall i = 1, 2, \dots, N, \forall j = \max\{i, k+1\}, \dots, k+N$. We denote by z_{LP} the value

of LP. The dual of LP which we denote by DP is :

$$\begin{aligned} z_{\text{DP}} = \max \quad & \sum_{c \in C} w(c) + \sum_{i=1}^N y_i - \sum_{j=k+1}^{k+N} z_j \\ \text{s.t.} \quad & y_i - z_j + u_{p(i),j} \leq w(c(i)) & \forall i \neq \text{last}(i), \\ & & \forall j \in [i, n(i) - 2]; \\ & y_i - z_j + u_{p(i),j} - u_{i,j+1} \leq 0 & \forall i \neq \text{last}(i), \\ & & \forall j \geq n(i) - 1; \\ & y_i - z_j + u_{p(i),j} \leq 0 & \forall i = \text{last}(i), \\ & & \forall j \geq i; \\ & u_{0,j} = 0 & \forall j; \\ & y, z, u \geq 0 \end{aligned}$$

Note that $z_{\text{DP}} = z_{\text{LP}} \leq z_{\text{IP}} = C_{\text{OPT}}$.

The following lemma characterizes the the set of feasible solutions to DP. We require the following definition. A subsequence $I = i_1, i_2, \dots, i_m$ of input elements is called a monochromatic sequence iff for every $s = 1, 2, \dots, m-1$ it holds that $c(i_s) = c(i_{s+1})$ and $n(i_s) = i_{s+1}$. A pair I, j where I is a monochromatic sequence and j is a time slot is called a monochromatic matching sequence iff the following two conditions are satisfied: (1) $j + s \geq i_s$ for every $s = 1, 2, \dots, m$; (2) $j + m < n(i_m) - 1$. We show in the following lemma that if y, z satisfy a certain inequality on each monochromatic matching sequence, then they can be extended (by fixing some u) to a feasible solution. We later use the algorithm to generate y, z that on the one hand satisfy the conditions in the lemma, and on the other hand give a DP value proportional the cost of the algorithm.

LEMMA 4.1. *Given two vectors $y, z \geq 0$ there exists $u \geq 0$ such that y, z, u is a feasible solution for DP iff for every monochromatic matching sequence $I, j, I = i_1, i_2, \dots, i_m$, we have that*

$$\sum_{s=1}^m (y_{i_s} - z_{j+s}) \leq \begin{cases} w(c(i_1)) & i_m \neq \text{last}(i_m), \\ 0 & \text{otherwise.} \end{cases}$$

Proof. Assign y, z in the inequalities of DP. We obtain a system of the following types of inequalities for u : (1) $u_{p(i),j} \leq 1 - (y_i - z_j)$ ($i \neq \text{last}(i)$); (2) $u_{p(i),j} \leq -(y_i - z_j)$ ($i = \text{last}(i)$); (3) $u_{p(i),j} - u_{i,j+1} \leq -(y_i - z_j)$. Denote by A the matrix of coefficients of u in this system. The row i, j in A represents the coefficients of u in the dual constraint indexed by i, j . Let $Au \leq b$ denote the system of inequalities. By the Farkas Lemma, there is $u \geq 0$ such that $Au \leq b$ iff for every $v \geq 0$ for which $v^T A \geq 0$ it holds that $v^T b \geq 0$. Notice that the entries of v correspond to pairs i, j where i is an input element and

$j \geq \max\{i, k + 1\}$ is an output time slot. A vector v is called the indicator vector of a monochromatic matching sequence I, j , $I = i_1, i_2, \dots, i_m$, iff $v_{i_s, j+s} = 1$ for all $s = 1, \dots, m$, and $v_{i, j} = 0$ otherwise.

LEMMA 4.2. *A vector $v \geq 0$ satisfies $v^T A \geq 0$ iff v is a linear combination of indicator vectors of monochromatic matching sequences with positive coefficients.*

Proof. We define a maximal sequence in v to be a sequence of entries $v_{i_1, j+1}, v_{i_2, j+2}, \dots, v_{i_m, j+m}$ that satisfies the following two conditions: (i) for $I = i_1, i_2, \dots, i_m$, I, j is a monochromatic matching sequence; (ii) $v_{i_1, j+1} = 0$. Let $v \geq 0$ satisfy $v^T A \geq 0$. If $v = \vec{0}$ then there is nothing to prove, so pick $v_{i_1, j+1} > 0$ such that $v_{p(i_1), j} = 0$. Assuming that $i_1 \neq \text{last}(i_1)$, if $j + 1 \geq n(i_1) - 1$ then in $v^T A$, $v_{i_1, j+1}$ multiplies a coefficient of -1 for $u_{i_1, j+2}$. In order to cancel this negative value, we must have that $v_{n(i_1), j+2} \geq v_{i_1, j+1}$. We can continue this argument to get a monotonically non-decreasing maximal sequence $0 = v_{p(i_1), j} < v_{i_1, j+1} \leq v_{i_2, j+2} \leq \dots \leq v_{i_m, j+m}$ (if $i_1 = \text{last}(i_1)$ then $m = 1$ and it is also trivially true). So, if $v^T A \geq 0$ then every maximal sequence in v is monotonically non-decreasing. Notice that the converse is also true. If every maximal sequence in v is monotonically non-decreasing, then $v^T A \geq 0$. Consider the vector v' defined by $v'_{i_s, j+s} = v_{i_1, j+1}$ for all $s = 1, \dots, m$ and $v'_{i, j} = 0$ otherwise. Notice that v' is a scaled indicator vector of a monochromatic matching sequence, such that $v - v' \geq 0$. Further notice that every maximal sequence in $v - v'$ remains the same as in v except for the maximal sequence indicated by v' . For the maximal sequence indicated by v' , we have $0 = (v - v')_{p(i_1), j} = (v - v')_{i_1, j+1} \leq (v - v')_{i_2, j+2} \leq \dots \leq (v - v')_{i_m, j+m}$. Therefore every maximal sequence in $v - v'$ is monotonically non-decreasing, and therefore $(v - v')^T A \geq 0$. We can apply this argument repeatedly, until we are left with the zero vector. The vector v is equal to the sum of all v' -s generated by this process. Hence, v is a linear combination of indicator vectors of monochromatic matching sequences with positive coefficients. The proof of the other direction of the lemma is simple and it is not needed for the rest of the analysis. ■

We now return to the proof of the lemma. By the above claim, it is sufficient to show that for every indicator vector v , $v^T b \geq 0$. For such v , if $i_m \neq \text{last}(i_m)$ then $v^T b = \sum_{s=1}^m (z_{j+s} - y_{i_s}) + w(c(i_1))$ as all the $u_{i, j}$ are canceled. We assume that $\sum_{s=1}^m (y_{i_s} - z_{j+s}) \leq w(c(i_1))$ so $v^T b \geq 0$. A similar argument holds for the case of $i_m = \text{last}(i_m)$. The other direction of the lemma is easy and is not needed for the rest of our analysis. ■

Using our algorithm we define vectors y, z that satisfy the conditions of Lemma 4.1. Informally, z_j indicates the total increase in the counter value (per element), up to step j , and y_i is determined when i is output in step $t(i)$ so that $y_i - z_{t(i)}$ is proportional to the final value of the counter of element i , which is i 's share in the cost of the algorithm. We first define z_j by induction on j . The base case is $z_k = 0$. For $j \geq k + 1$, we set

$$z_j = z_{j-1} + \frac{\epsilon_j}{4}$$

where ϵ_j is the increase in the counter of each element in the buffer in time j . (Notice that in every time slot j either none of the counters increase or they all increase by the same amount.) To define y recall that $t(i)$ denotes the time slot in which the algorithm outputs i . For every i set

$$y_i = \begin{cases} z_{t(i)} + \frac{1}{\alpha} \varphi^i & i \neq \text{last}(i), \\ z_{t(i)} + \frac{1}{\alpha} \varphi^i - w(c(i)) & \text{otherwise.} \end{cases}$$

where α is defined below. (If $y_i < 0$ for some i , we can shift all entries in y, z by the same amount M to get $y \geq 0$. This does not change the feasibility of the other conditions or the value of the solution.)

We now show that the above y, z can be extended to a feasible solution to DP. This is the main technical difficulty in the argument. Recall that by Lemma 4.1, we need to examine monochromatic matching sequences. A monochromatic matching sequence may glue together many blocks of the same color output by the algorithm, the first ones matched after they were output by the algorithm and the latter ones matched before they were output by the algorithm. If z_j increases too slowly, then the first blocks might contribute too much to the cost of the monochromatic matching sequence, because the differences $y_{i_s} - z_{j+s}$ will be too close to $y_{i_s} - z_{t(i_s)}$. If z_j increases too quickly, then the latter blocks might contribute too much to the cost of the monochromatic matching sequence, because $y_{i_s} - z_{j+s}$ will be much larger than $y_{i_s} - z_{t(i_s)}$. Thus, the main idea of the analysis is to balance the two opposing requirements on the rate of increment of z . The asymptotically optimal balancing yields the competitive ratio of $O\left(\frac{\log k}{\log \log k}\right)$. This is shown in the following lemma.

LEMMA 4.3. *For $\alpha = O\left(\frac{\log k}{\log \log k}\right)$ there exists u such that y, z, u is a feasible solution for DP.*

Proof. By Lemma 4.1, it is sufficient to prove that for every monochromatic matching sequence I, j , $I = i_1, i_2, \dots, i_m$, we have that $\sum_{s=1}^m (y_{i_s} - z_{j+s}) \leq w(c(i_1))$ if $i_m \neq \text{last}(i_m)$ and $\sum_{s=1}^m (y_{i_s} - z_{j+s}) \leq 0$ otherwise.

Consider a monochromatic sequence i_1, i_2, \dots, i_m of color c . The algorithm partitions this sequence into blocks, where each block is output in consecutive time slots and between blocks other colors are output. Denote the blocks by I_1, I_2, \dots, I_s . Notice that all blocks, except perhaps I_1, I_s , are maximal monochromatic sequences that the algorithm outputs consecutively. Denote by σ_q the index of the first element in block I_q . Denote by ℓ_q the number of elements that enter the buffer between the time that the algorithm outputs the last element of I_q and the time that the algorithm outputs the first element of I_{q+1} . In other words, $\ell_q = t(i_{\sigma_{q+1}}) - \max_{i \in I_q} t(i) - 1$. Notice that the number of elements $i \in I_{q+1}$ with $\varphi^i > 0$ is strictly less than ℓ_q , because all these elements must have entered the buffer between the time I_q is output and the time I_{q+1} is output, and the first element in this time interval must be of a different color (otherwise I_q would have continued).

We partition $\sum_{\rho=1}^m (y_{i_\rho} - z_{j+\rho})$ into three parts. The first part deals with the last block, which might be incomplete. The second part deals with the blocks that are matched before the time they were output by the algorithm (excluding the last block). The third part deals with the blocks that are matched after the time they were output by the algorithm.

Part 1: the last block I_s :

$$\begin{aligned} \sum_{i_\rho \in I_s} (y_{i_\rho} - z_{j+\rho}) &\leq \sum_{i_\rho \in I_s} (y_{i_\rho} - z_{i_\rho}) \\ &= \sum_{i_\rho \in I_s} (y_{i_\rho} - z_{t(i_\rho)}) + \sum_{i_\rho \in I_s} (z_{t(i_\rho)} - z_{i_\rho}) \\ &= \sum_{i_\rho \in I_s} \frac{1}{\alpha} \varphi^{i_\rho} + \sum_{i_\rho \in I_s} \frac{1}{4} \varphi^{i_\rho} \\ &< \left(\frac{2}{\alpha} + \frac{1}{2} \right) \cdot w(c), \end{aligned}$$

where the first inequality follows from the fact that z is monotonically non-decreasing, and the second inequality follows from Observation 3.1. The reader can verify that if $i_m = \text{last}(i_m)$ then

$$\sum_{i_\rho \in I_s} (y_{i_\rho} - z_{j+\rho}) < \left(\frac{2}{\alpha} - \frac{1}{2} \right) \cdot w(c).$$

Part 2: Let r be the largest index for which $j+1 > t(i_{\sigma_r})$. We now upper bound

$$\begin{aligned} \sum_{q=r+1}^{s-1} \sum_{i_\rho \in I_q} (y_{i_\rho} - z_{j+\rho}) &= \sum_{q=r+1}^{s-1} \sum_{i_\rho \in I_q} (y_{i_\rho} - z_{t(i_\rho)}) \\ &\quad - \sum_{q=r+1}^{s-1} \sum_{i_\rho \in I_q} (z_{j+\rho} - z_{t(i_\rho)}). \end{aligned}$$

Notice that in every block I_q , the first element i_{σ_q} accumulates the largest counter and less than ℓ_{q-1} elements accumulate a positive counter. Therefore,

$$\begin{aligned} \sum_{q=r+1}^{s-1} \sum_{i_\rho \in I_q} (y_{i_\rho} - z_{t(i_\rho)}) &< \sum_{i_\rho \in I_{r+1}} \frac{1}{\alpha} \varphi^{i_\rho} \\ &\quad + \sum_{q=r+2}^{s-1} \frac{1}{\alpha} \varphi^{i_{\sigma_q}} \cdot \ell_{q-1}. \end{aligned}$$

Notice that for every ρ , $j+\rho > t(i_\rho)$, otherwise there would be a gap between two consecutive blocks, and therefore $z_{j+\rho} - z_{t(i_\rho)} \geq 0$. Further notice that for every $i_\rho \in I_{s-2}$, $j+\rho > t(i_\rho) + \ell_{s-2}$ and therefore the last ℓ_{s-2} elements i_ρ in the blocks before I_{s-2} have $j+\rho \geq t(i_{\sigma_{s-2}})$ and therefore $z_{j+\rho} \geq z_{t(i_\rho)} + \frac{1}{4} \varphi^{i_{\sigma_{s-2}}}$. More generally, block I_q “moves” at least $\ell_{q+1} + \ell_{q+2} + \dots + \ell_{s-2}$ steps ahead and therefore at least the $\ell_{q+1} + \ell_{q+2} + \dots + \ell_{s-2}$ last elements i_ρ from previous blocks have $j+\rho \geq t(i_{\sigma_q})$ so $z_{j+\rho} \geq z_{t(i_\rho)} + \frac{1}{4} \varphi^{i_{\sigma_q}}$. Thus we have that

$$\begin{aligned} 4 \sum_{q=r+1}^{s-1} \sum_{i_\rho \in I_q} (z_{j+\rho} - z_{t(i_\rho)}) &\geq \ell_{s-2} \cdot \varphi^{i_{\sigma_{s-2}}} + (\ell_{s-2} + \ell_{s-3}) \varphi^{i_{\sigma_{s-3}}} \\ &\quad + \dots + (\ell_{s-2} + \dots + \ell_{r+1}) \varphi^{i_{\sigma_{r+1}}} \\ &= \ell_{s-2} (\varphi^{i_{\sigma_{r+1}}} + \dots + \varphi^{i_{\sigma_{s-2}}}) \\ &\quad + \ell_{s-3} (\varphi^{i_{\sigma_{r+1}}} + \dots + \varphi^{i_{\sigma_{s-3}}}) + \dots + \ell_{r+1} \cdot \varphi^{i_{\sigma_{r+1}}}. \end{aligned}$$

Combining the two bounds, we obtain that

$$\begin{aligned} \sum_{q=r+1}^{s-1} \sum_{i_\rho \in I_q} (y_{i_\rho} - z_{j+\rho}) &= \sum_{q=r+1}^{s-1} \sum_{i_\rho \in I_q} (y_{i_\rho} - z_{t(i_\rho)}) \\ &\quad - \sum_{q=r+1}^{s-1} \sum_{i_\rho \in I_q} (z_{j+\rho} - z_{t(i_\rho)}) \\ &< \sum_{q=r+2}^{s-1} \ell_{q-1} \left(\frac{1}{\alpha} \varphi^{i_{\sigma_q}} - \frac{1}{4} (\varphi^{i_{\sigma_{r+1}}} + \dots + \varphi^{i_{\sigma_{q-1}}}) \right) \\ &\quad + \sum_{i_\rho \in I_{r+1}} \frac{1}{\alpha} \varphi^{i_\rho}. \end{aligned}$$

We now show that the number of indexes q for which $\frac{1}{\alpha} \varphi^{i_{\sigma_q}} - \frac{1}{4} (\varphi^{i_{\sigma_{r+1}}} + \dots + \varphi^{i_{\sigma_{q-1}}}) > 0$ is at most $1 + \log_{\alpha/4} 2k$. Let $q_1 < q_2 < \dots < q_a$ be the indexes for

which the above condition holds. For every $b = 1, \dots, a$,

$$\begin{aligned}\varphi^{i\sigma_{qb}} &> \frac{\alpha}{4} \left(\varphi^{i\sigma_{r+1}} + \varphi^{i\sigma_{r+2}} + \dots + \varphi^{i\sigma_{qb-1}} \right) \\ &\geq \frac{\alpha}{4} \left(\varphi^{i\sigma_{q1}} + \varphi^{i\sigma_{q2}} \dots + \varphi^{i\sigma_{qb-1}} \right).\end{aligned}$$

Therefore,

$$2w(c) > \varphi^{i\sigma_{qa}} > \left(\frac{\alpha}{4}\right)^{a-1} \varphi^{i\sigma_{q1}} \geq \left(\frac{\alpha}{4}\right)^{a-1} \frac{w(c)}{k},$$

where the last inequality follows from the fact that for every block I_q one of the following two situations occurs. If the block is output due to a free color change, then $\sum_{i \in I_q} \varphi^i \geq w(c)$, so $\varphi^{i\sigma_q} \geq \frac{w(c)}{k}$ (as $|I_q| \leq k$). Otherwise, the block is output due to a paid color change, then the counter of every element in the buffer, including $i\sigma_q$, is increased by $\frac{w(c)}{k}$ at step $t(i\sigma_q)$. Therefore $a < 1 + \log_{\alpha/4} 2k$. Thus we conclude that

$$\begin{aligned}\sum_{q=r+1}^{s-1} \sum_{i_\rho \in I_q} (y_{i_\rho} - z_{j+\rho}) &\leq \sum_{b=1}^a \sum_{i_\rho \in I_{q_b}} (y_{i_\rho} - z_{j+\rho}) \\ &\leq \sum_{b=1}^a \sum_{i_\rho \in I_{q_b}} (y_{i_\rho} - z_{t(i_\rho)}) \\ &< \frac{2a}{\alpha} w(c) \\ &< \frac{2 + 2 \log_{\alpha/4} 2k}{\alpha} w(c).\end{aligned}$$

Part 3: Finally we evaluate

$$\begin{aligned}\sum_{q=1}^r \sum_{i_\rho \in I_q} (y_{i_\rho} - z_{j+\rho}) &= \sum_{q=1}^r \sum_{i_\rho \in I_q} (y_{i_\rho} - z_{t(i_\rho)}) \\ &\quad - \sum_{q=1}^r \sum_{i_\rho \in I_q} (z_{j+\rho} - z_{t(i_\rho)}).\end{aligned}$$

For $q = 1, 2, \dots, r$ consider

$$\sum_{i_\rho \in I_q} (y_{i_\rho} - z_{t(i_\rho)}) - \sum_{i_\rho \in I_q} (z_{j+\rho} - z_{t(i_\rho)}).$$

Let $q_1 < q_2 < \dots < q_a$ be the indexes for which the above difference is positive. For every $i_\rho \in I_q$, the following two inequalities hold:

- (i) $y_{i_\rho} - z_{t(i_\rho)} = \varphi^{i_\rho} \leq \varphi^{i\sigma_q} = y_{i\sigma_q} - z_{t(i\sigma_q)}$,
- (ii) $z_{j+\rho} - z_{t(i_\rho)} \geq z_{j+\sigma_q} - z_{t(i_\rho)} = z_{j+\sigma_q} - z_{t(i\sigma_q)}$.

The inequality in (ii) follows because z_j is monotonically non-decreasing in j , and the equality in (i) follows because z_j is the same for all $j \in [t(i\sigma_q), t(i_\rho)]$, as there is no color change in this interval (steps in

which I_q is output). Hence, for every $b = 1, \dots, a$, $(y_{i\sigma_{qb}} - z_{t(i\sigma_{qb})}) - (z_{j+\sigma_{qb}} - z_{t(i\sigma_{qb})}) > 0$. So,

$$\begin{aligned}\frac{1}{\alpha} \varphi^{i\sigma_{qb}} &= y_{i\sigma_{qb}} - z_{t(i\sigma_{qb})} > z_{j+\sigma_{qb}} - z_{t(i\sigma_{qb})} \\ &\geq \frac{1}{4} \sum_{q=q_b+1}^r \varphi^{i\sigma_q} \geq \frac{1}{4} \sum_{b'=b+1}^a \varphi^{i\sigma_{q_{b'}}}.\end{aligned}$$

The penultimate inequality follows from the fact that :

$$\begin{aligned}t(i\sigma_{q_b}) &< i\sigma_{q_{b+1}} \leq t(i\sigma_{q_{b+1}}) < i\sigma_{q_{b+2}} \leq t(i\sigma_{q_{b+2}}) \\ &< \dots < i\sigma_{q_a} \leq t(i\sigma_{q_a}) \leq t(i\sigma_r) < j+1 \leq j+\sigma_{q_b},\end{aligned}$$

as for all q , the first element of I_{q+1} entered the buffer after the last element of I_q was output. Hence

$$z_{j+\sigma_{q_b}} - z_{t(i\sigma_{q_b})} \geq \sum_{q=q_b+1}^r (z_{t(i\sigma_q)} - z_{i\sigma_q}) \geq \frac{1}{4} \sum_{q=q_b+1}^r \varphi^{i\sigma_q}.$$

Therefore, as in Part 2,

$$2w(c) > \varphi^{i\sigma_{q1}} > \left(\frac{\alpha}{4}\right)^{a-1} \varphi^{i\sigma_{qa}} \geq \left(\frac{\alpha}{4}\right)^{a-1} \frac{w(c)}{k},$$

so, $a < 1 + \log_{\alpha/4} 2k$, and thus

$$\sum_{q=1}^r \sum_{i_\rho \in I_q} (y_{i_\rho} - z_{j+\rho}) < \frac{2 + 2 \log_{\alpha/4} 2k}{\alpha} w(c).$$

Summing the three parts, we conclude that

$$\sum_{s=1}^m (y_{i_s} - z_{j+s}) \leq \left(\frac{1}{2} + \frac{2}{\alpha} + \frac{2}{\alpha} (1 + \log_{\alpha/4} 2k) \right) \cdot w(c).$$

If we set $\alpha = \gamma \cdot \frac{\log k}{\log \log k}$ for a sufficiently large constant γ then $\log_{\alpha/4} 2k \ll \alpha$ so the above bound is at most $w(c)$. Again, the reader can verify that if $i_m = \text{last}(i_m)$ then

$$\begin{aligned}\sum_{s=1}^m (y_{i_s} - z_{j+s}) \\ \leq \left(-\frac{1}{2} + \frac{2}{\alpha} + \frac{2}{\alpha} (1 + \log_{\alpha/4} 2k) \right) \cdot w(c) \leq 0,\end{aligned}$$

for γ sufficiently large. ■

THEOREM 4.1. *The algorithm is 2α -competitive.*

Proof.

$$\begin{aligned}C_{\text{OPT}} \geq z_{\text{DP}} &\geq \sum_{c \in C} w(c) + \sum_{i=1}^N (y_i - z_{t(i)}) \\ &= \sum_{c \in C} w(c) + \frac{1}{\alpha} \cdot \sum_{i=1}^N \varphi^i - \sum_{i=\text{last}(i)} w(c(i)) \\ &= \frac{1}{\alpha} \cdot \sum_{i=1}^N \varphi^i \geq \frac{1}{2\alpha} C_{\text{TLC}}.\end{aligned}$$
■

References

- [1] A. Aboud. Correlation clustering with penalties and approximating the reordering buffer management problem. Master's thesis, Computer Science Department, The Technion - Israel Institute of Technology, January 2008.
- [2] S. Albers. New results on web caching with request reordering. In *Proc. of the 16th ACM Symp. on Parallel Algorithms and Architectures*, pages 84–92, 2004.
- [3] H. Alborzi, E. Torng, P. Uthaisombut, and S. Wagner. The k -client problem. *J. Algorithms*, 41(2):115–173, 2001.
- [4] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. The online set cover problem. In *Proc. of the 35th Ann. ACM Symp. on Theory of Computing*, pages 100–105, 2003.
- [5] R. Bar-Yehuda and J. Laserson. Exploiting locality: approximating sorting buffers. *J. of Discrete Algorithms*, 5(4):729–738, 2007.
- [6] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Proc. of the 30th Ann. ACM Symp. on Theory of Computing*, pages 161–168, 1998.
- [7] D. Blandford and G. Blelloch. Index compression through document reordering. In *Data Compression Conference*, pages 342–351, 2002.
- [8] N. Buchbinder. *Designing competitive online algorithms via a primal-dual approach*. PhD thesis, Computer Science Department, The Technion - Israel Institute of Technology, July 2008.
- [9] N. Buchbinder and J. Naor. Online primal-dual algorithms for covering and packing problems. In *Proc. of the 13th Ann. European Symposium on Algorithms*, pages 689–701, 2005.
- [10] M. Charikar, H.J. Karloff, C. Mathieu, J. Naor, and M.E. Saks. Online multicast with egalitarian cost sharing. In *Proc. of the 20th ACM Symp. on Parallel Algorithms and Architectures*, pages 70–76, 2008.
- [11] M. Englert, D. Özmen, and M. Westermann. The power of reordering for online minimum makespan scheduling. In *Proc. of the 49th Ann. IEEE Symp. on Foundations of Computer Science*, 2008.
- [12] M. Englert, H. Räcke, and M. Westermann. Reordering buffers for general metric spaces. In *Proc. of the 39th Ann. ACM Symp. on Theory of Computing*, pages 556–564, 2007.
- [13] M. Englert and M. Westermann. Reordering buffer management for non-uniform cost models. In *Proc. of the 32nd Ann. International Colloquium on Algorithms, Languages, and Programming*, pages 627–638, 2005.
- [14] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004.
- [15] T. Feder, R. Motwani, R. Panigrahy, and A. Zhu. Web caching with request reordering. In *Proc. of the 13th Ann. ACM-SIAM Symposium on Discrete Algorithms*, pages 104–105, 2002.
- [16] D. Fotakis. A primal-dual algorithm for online non-uniform facility location. In *Panhellenic Conference on Informatics*, pages 47–56, 2005.
- [17] I. Gamzu and D. Segev. Improved online algorithms for the sorting buffer problem. In *Proc. of the 24th Ann. International Symposium on Theoretical Aspects of Computer Science*, pages 658–669, 2007.
- [18] K. Gutenschwager, S. Spiekermann, and S. Vos. A sequential ordering problem in automotive paint shops. *International Journal of Production Research*, 42(9):1865–1878, 2004.
- [19] R. Khandekar and V. Pandit. Offline sorting buffers on line. In *ISAAC*, pages 81–89, 2006.
- [20] R. Khandekar and V. Pandit. Online sorting buffers on line. In *Proc. of the 23rd Ann. International Symposium on Theoretical Aspects of Computer Science*, pages 584–595, 2006.
- [21] J. Kohrt and K. Pruhs. Constant approximation algorithm for sorting buffers. In *Proc. of the 6th Latin American Symp. on Theoretical Informatics*, pages 193–202, Buenos Aires, Argentina, 2004.
- [22] J. Krokowski, H. Räcke, C. Sohler, and M. Westermann. Reducing state changes with a pipeline buffer. In *VMV*, page 217, 2004.
- [23] H. Räcke, C. Sohler, and M. Westermann. Online scheduling for sorting buffers. In *Proc. of the 10th Ann. European Symposium on Algorithms*, pages 820–832, 2002.
- [24] A. Silberschatz, P. Galvin, and G. Gagne. *Applied Operating System Concepts*. J. Wiley, 2000.
- [25] V.V. Vazirani. *Approximation Algorithms*. Springer, 2001.