

A 3.3904-Competitive Online Algorithm for List Update with Uniform Costs


Mateusz Basiak 
University of Wrocław, Poland


Martin Böhm 
University of Wrocław, Poland

Łukasz Jeż 
University of Wrocław, Poland

Agnieszka Tatarczuk 
University of Wrocław, Poland

Marcin Bienkowski 
University of Wrocław, Poland

Marek Chrobak 
University of California, Riverside, CA, USA

Jiří Sgall 
Charles University, Prague, Czech Republic

Abstract

We consider the List Update problem where the cost of each swap is assumed to be 1. This is in contrast to the “standard” model, in which an algorithm is allowed to swap the requested item with previous items for free. We construct an online algorithm FULL-OR-PARTIAL-MOVE (*FPM*), whose competitive ratio is at most 3.3904, improving over the previous best known bound of 4.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases List update, work functions, amortized analysis, online algorithms, competitive analysis

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.76

Related Version *Full Version*: <https://arxiv.org/abs/2503.17264> [14]

Funding Supported by Polish National Science Centre grants 2022/45/B/ST6/00559, 2020/39/B/ST6/01679 and 2022/47/D/ST6/02864, National Science Foundation grant CCF-2153723, and by grant 24-10306S of GA ČR.

1 Introduction

The List Update problem. In the online *List Update problem* [3, 20, 21], the objective is to maintain a set of items stored in a linear list in response to a sequence of access requests. The cost of accessing a requested item is equal to its distance from the front of the list. After each request, an algorithm is allowed to rearrange the list by performing an arbitrary number of swaps of adjacent items. In the model introduced by Sleator and Tarjan in their seminal 1985 paper on competitive analysis [25], an algorithm can repeatedly swap the requested item with its preceding item at no cost. These swaps are called *free*. All other swaps are called *paid* and have cost 1 each. As in other problems involving self-organizing data structures [7], the goal is to construct an *online* algorithm, i.e., operating without the knowledge of future requests. The cost of such an algorithm is compared to the cost of the optimal *offline* algorithm; the ratio of the two costs is called the *competitive ratio* and is subject to minimization.

Sleator and Tarjan proved that the algorithm *MOVE-TO-FRONT* (*MTF*), which after each request moves the requested item to the front of the list, is 2-competitive [25]. This ratio is known to be optimal if the number of items is unbounded. Their work was the culmination of previous extensive studies of list updating, including experimental results, probabilistic approaches, and earlier attempts at amortized analysis (see [15] and the references therein).



© Mateusz Basiak, Marcin Bienkowski, Martin Böhm, Marek Chrobak, Łukasz Jeż, Jiří Sgall, and Agnieszka Tatarczuk;
licensed under Creative Commons License CC-BY 4.0

33rd Annual European Symposium on Algorithms (ESA 2025).

Editors: Anne Benoit, Haim Kaplan, Sebastian Wild, and Grzegorz Herman; Article No. 76; pp. 76:1–76:15



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

As shown in subsequent work, *MTF* is not unique – there are other strategies that achieve ratio 2, such as *TIMESTAMP* [2] or algorithms based on work functions [9]. In fact, there are infinitely many algorithms that achieve ratio 2 [13, 21].

The uniform cost model. Following [24], we will refer to the cost model of [25] as *standard*, and we will denote it here by LUP_S . This model has been questioned in the literature for not accurately reflecting true costs in some implementations [24, 23, 22, 18], with the concept of free swaps being one of the main concerns.

A natural approach to address this concern, considered in some later studies (see, e.g., [24, 7, 21, 4, 12]), is simply to charge cost 1 for *any* swap. We will call it here the *uniform cost model* and denote it LUP_1 . A general lower bound of 3 on the competitive ratio of deterministic algorithms for LUP_1 was given by Reingold *et al.* [24]. Changing the cost of free swaps from 0 to 1 at most doubles the cost of any algorithm, so *MTF* is no worse than 4-competitive for LUP_1 . Surprisingly, no algorithm is known to beat *MTF*, i.e., achieve ratio lower than 4.¹

Our main result. To address this open problem, we develop an online algorithm *FULL-OR-PARTIAL-MOVE* (*FPM*) for LUP_1 with competitive ratio $\frac{1}{8} \cdot (23 + \sqrt{17}) \approx 3.3904$, significantly improving the previous upper bound of 4.

Our algorithm *FPM* remains 3.3904-competitive even for the *partial cost* function, where the cost of accessing location ℓ is $\ell - 1$, instead of the *full cost* of ℓ used in the original definition of List Update [25]. Both functions have been used in the literature, depending on context and convenience. For any online algorithm, its partial-cost competitive ratio is at least as large as its full-cost ratio, although the difference typically vanishes when the list size is unbounded. We present our analysis in terms of partial costs.

FPM remains 3.3904-competitive also in the dynamic scenario of List Update that allows operations of insertions and deletions, as in the original definition in [25] (see the full version of the paper [14]).

Technical challenges and new ideas. The question whether ratio 3 can be achieved remains open. We also do not know if there is a *simple* algorithm with ratio below 4. We have considered some natural adaptations of *MTF* and other algorithms that are 2-competitive for LUP_S , but for all we were able to show lower bounds higher than 3 for LUP_1 .

As earlier mentioned, *MTF* is 4-competitive for LUP_1 . It is also easy to show that its ratio is not better than 4: repeatedly request the last item in the list. Ignoring additive constants, the algorithm pays twice the length of the list at each step, while any algorithm that just keeps the list in a fixed order, pays only half the length on the average.

The intuition why *MTF* performs poorly is that it moves the requested items to front “too quickly”. For the aforementioned adversarial strategy against *MTF*, ratio 3 can be obtained by moving the items to front only *every other time* they are requested. This algorithm, called *DBIT*, is a deterministic variant of algorithm *BIT* from [24] and a special case of algorithm *COUNTER* in [24], and it has been also considered in [21]. In the full version of the paper [14], we show that *DBIT* is not better than 4-competitive in the partial cost model, and not better than 3.302-competitive in the full cost model.

¹ The authors of [21] claimed an upper bound of 3 for LUP_1 , but later discovered that their proof was not correct (personal communication with S. Kamali).

One can generalize these approaches by considering a more general class of algorithms that either leave the requested item at its current location or move it to the front. We show that such a strategy cannot achieve ratio better than 3.25, even for just three items. A naive fix would be, for example, to always move the requested item half-way towards front. This algorithm is even worse: its competitive ratio is at least 6 (both those proofs are in the full version of the paper [14]).

We also show (see Section 5) via a computer-aided argument that, for LUP_1 , the work function algorithm's competitive ratio is larger than 3, even for lists of length 5. This is in contrast to its performance for the standard model, where it achieves optimal ratio 2 [9].

Our algorithm *FPM* overcomes the difficulties mentioned above by combining a few new ideas. The first idea is a more sophisticated choice of the target location for the requested item. That is, aside from *full moves* that move the requested items to the list front, *FPM* sometimes performs a *partial move* to a suitably chosen target location in the list. This location roughly corresponds to the front of the list when this item was requested earlier.

The second idea is to keep track, for each pair of items, of the work function for the two-item subsequence consisting of these items. These work functions are used in two ways. First, they roughly indicate which relative order between the items in each pair is “more likely” in an optimal solution. The algorithm uses this information to decide whether to perform a full move or a partial move. Second, the simple sum of all these pair-based work functions is a lower bound on the optimal cost, which is useful in analyzing the competitive ratio of *FPM*.

Related work. Better bounds are known for randomized algorithms both in the standard model (LUP_S) and the uniform cost model (LUP_1). For LUP_S , a long line of research culminated in a 1.6-competitive algorithm [19, 24, 2, 6], and a 1.5-lower bound [26]. The upper bound of 1.6 is tight in the class of so-called projective algorithms, whose computation is uniquely determined by their behavior on two-item instances [8]. For LUP_1 , the ratio is known to be between 1.5 [4] and 2.64 [24].

It is possible to generalize the uniform cost function by distinguishing between the cost of 1 for following a link during search and the cost of $d \geq 1$ for a swap [24, 4]. This model is sometimes called the P^d model; in this terminology, our LUP_1 corresponds to P^1 . While *MTF* is 4-competitive for P^1 , it does not generalize in an obvious way to $d > 1$. Other known algorithms for the P^d model (randomized and deterministic *COUNTER*, *RANDOMRESET* and *TIMESTAMP*) have bounds on competitive ratios that monotonically decrease with growing d [24, 4]. In particular, deterministic *COUNTER* achieves ratio 4.56 when d tends to infinity [7] and for the same setting ($d \rightarrow \infty$) a recent result by Albers and Janke [4] shows a randomized algorithm *TIMESTAMP* that is 2.24-competitive.

A variety of List Update variants have been investigated in the literature over the last forty years, including models with lookahead [1], locality of reference [10, 5], parameterized approach [17], algorithms with advice [16], prediction [11], or alternative cost models [18, 22, 23]. A particularly interesting model was proposed recently by Azar *et al.* [12], where an online algorithm is allowed to postpone serving some requests, but is either required to serve them by a specified deadline or pay a delay penalty.

In summary, List Update is one of canonical problems in the area of competitive analysis, used to experiment with refined models of competitive analysis or to study the effects of additional features. This underscores the need to fully resolve the remaining open questions regarding its basic variants, including the question whether ratio 3 is attainable for LUP_1 .

2 Preliminaries

Model. An algorithm has to maintain a list of items, while a sequence σ of access requests is presented in an online manner. In each step $t \geq 1$, the algorithm is presented an access request σ^t to an item in the list. If this item is in a location ℓ , the algorithm incurs cost $\ell - 1$ to access it. (The locations in the list are indexed $1, 2, \dots$.) Afterwards, the algorithm may change the list configuration by performing an arbitrary number of swaps of neighboring items, each of cost 1.

For an algorithm A , we denote its cost for processing a sequence σ by $A(\sigma)$. The optimal algorithm is denoted by OPT .

Notation. Let \mathcal{P} be the set of all unordered pairs of items. For a pair $\{x, y\} \in \mathcal{P}$, we use the notation $x \prec y$ ($x \succ y$) to denote that x is before (after) y in the list of an online algorithm. (The relative order of x, y may change over time, but it will be always clear from context what step of the computation we are referring to.) We use $x \preceq y$ ($x \succeq y$) to denote that $x \prec y$ ($x \succ y$) or $x = y$.

For an input σ and a pair $\{x, y\} \in \mathcal{P}$, σ_{xy} is the subsequence of σ restricted to requests to items x and y only. Whenever we say that an algorithm serves input σ_{xy} , we mean that it has to maintain a list of two items, x and y .

2.1 Work Functions

Work functions on item pairs. For each prefix σ of the input sequence, an online algorithm may compute a so-called *work function* W^{xy} , where $W^{xy}(xy)$ (or $W^{xy}(yx)$) is the optimal cost of the solution that serves σ_{xy} and ends with the list in configuration xy (or yx). (Function W^{xy} also has prefix σ as an argument. Its value will be always uniquely determined from context.) The values of W for each step can be computed iteratively using straightforward dynamic programming. Note that the values of W are non-negative integers and $|W^{xy}(xy) - W^{xy}(yx)| \leq 1$.

Modes. For a pair $\{x, y\} \in \mathcal{P}$, we define its *mode* depending on the value of the work function W^{xy} in the current step and the mutual relation of x and y in the list of an online algorithm. In the following definition we assume that $y \prec x$.

- Pair $\{x, y\}$ is in mode α if $W^{xy}(yx) + 1 = W^{xy}(xy)$.
- Pair $\{x, y\}$ is in mode β if $W^{xy}(yx) = W^{xy}(xy)$.
- Pair $\{x, y\}$ is in mode γ if $W^{xy}(yx) - 1 = W^{xy}(xy)$.

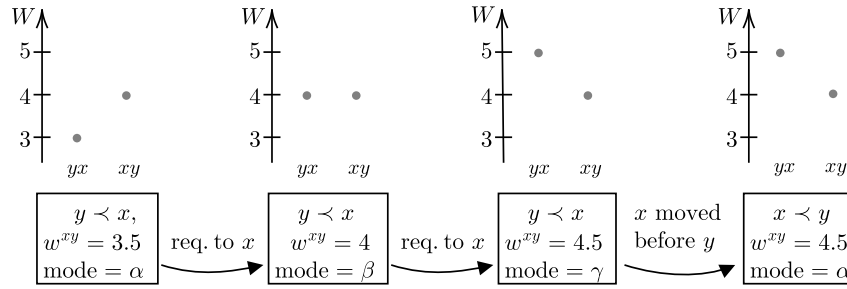
For an illustration of work function evolution and associated modes, see Figure 1.

If a pair $\{x, y\}$ is in mode α , then the minimum of the work function W^{xy} is at configuration yx , i.e., the one that has y before x . That is, an online algorithm keeps these items in a way that “agrees” with the work function. Note that α is the initial mode of all pairs. Conversely, if a pair $\{x, y\}$ is in state γ , then the minimum of the work function W^{xy} is at configuration xy . In this case, an online algorithm keeps these items in a way that “disagrees” with the work function.

2.2 Lower Bound on OPT

Now we show how to use the changes in the work functions of item pairs to provide a useful lower bound on the cost of an optimal algorithm. The following lemma is a standard and straightforward result of the list partitioning technique [9].²

² There are known input sequences on which the relation of Lemma 1 is not tight [9].



■ **Figure 1** The evolution of the work function W^{xy} . Initially, W^{xy} has its minimum in the state yx and $y \prec x$ (in the list of an online algorithm). Thus, the mode of the pair $\{x, y\}$ is α . Next, because of the two requests to x , the value of $W^{xy}(yx)$ is incremented, while the value at xy remains intact. The mode is thus changed from α to β and then to γ . Finally, when an algorithm moves item x before y , the mode of the pair $\{x, y\}$ changes to α . The value of w^{xy} (the average of $W^{xy}(xy)$ and $W^{xy}(yx)$) increases by $\frac{1}{2}$ whenever the mode changes due to a request.

► **Lemma 1.** For every input sequence σ , it holds that $\sum_{\{x,y\} \in \mathcal{P}} OPT(\sigma_{xy}) \leq OPT(\sigma)$.

Averaging work functions. We define the function w^{xy} as the average value of the work function W^{xy} , i.e.,

$$w^{xy} \triangleq \frac{1}{2} \cdot (W^{xy}(xy) + W^{xy}(yx)).$$

We use w_t^{xy} to denote the value of w^{xy} after serving the first t requests of σ , and define $\Delta_t w^{xy} \triangleq w_t^{xy} - w_{t-1}^{xy}$. The growth of w^{xy} can be related to $OPT(\sigma)$ in the following way.

► **Lemma 2.** For a sequence σ consisting of T requests, it holds that $\sum_{t=1}^T \sum_{\{x,y\} \in \mathcal{P}} \Delta_t w^{xy} \leq OPT(\sigma)$.

Proof. We first fix a pair $\{x, y\} \in \mathcal{P}$. We have $w_0^{xy} = \frac{1}{2}$ and $w_T^{xy} \leq OPT(\sigma_{xy}) + \frac{1}{2}$. Hence, $\sum_{t=1}^T \Delta_t w^{xy} = w_T^{xy} - w_0^{xy} \leq OPT(\sigma_{xy})$. The proof follows by summing over all pairs $\{x, y\} \in \mathcal{P}$ and invoking Lemma 1. ◀

Pair-based OPT. Lemma 2 gives us a convenient tool to lower bound $OPT(\sigma)$. We define the cost of *pair-based OPT* in step t as $\sum_{\{x,y\} \in \mathcal{P}} \Delta_t w^{xy}$. For a given request sequence σ , the sum of these costs over all steps is a lower bound on the actual value of $OPT(\sigma)$.

On the other hand, we can express $\Delta_t w^{xy}$ (and thus also the pair-based OPT) in terms of the changes of the modes of item pairs. See Figure 1 for an illustration.

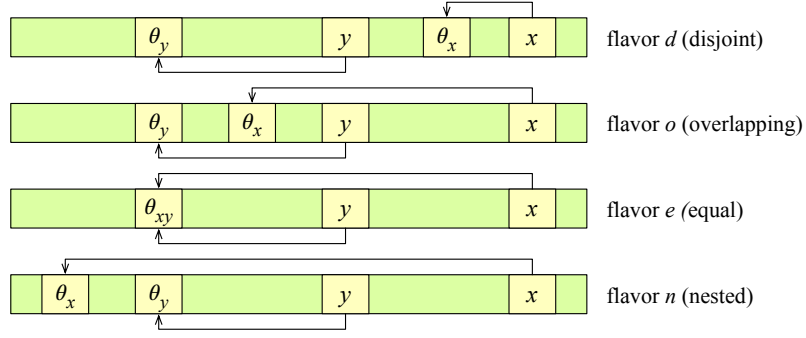
► **Observation 3.** If a pair $\{x, y\}$ changes its mode due to the request in step t then $\Delta_t w^{xy} = \frac{1}{2}$, otherwise $\Delta_t w^{xy} = 0$.

3 Algorithm Full-Or-Partial-Move

For each item x , FPM keeps track of an item denoted θ_x and called the *target* of x . At the beginning, FPM sets $\theta_x = x$ for all x . Furthermore, at each time, FPM ensures that $\theta_x \preceq x$.

The rest of this section describes the overall strategy of algorithm FPM . Our description is top-down, and proceeds in three steps:

- First we describe, in broad terms, what actions are involved in serving a request, including the choice of a move and the principle behind updating target items.



■ **Figure 2** The flavor of a pair $\{x, y\}$ (where $y \prec x$) depends on the position of target θ_x with respect to items $\theta_y \preceq y \prec x$. In the figure for flavor e , we write θ_{xy} for the item $\theta_x = \theta_y$.

- Next, we define the concept of states associated with item pairs and their potentials.
- Finally, we explain how algorithm FPM uses these potential values to decide how to adjust the list after serving the request.

This description will fully specify how FPM works, providing that the potential function on the states is given. Thus, for any choice of the potential function the competitive ratio of FPM is well-defined. What remains is to choose these potential values to optimize the competitive ratio. This is accomplished by the analysis in Section 4 that follows.

Serving a request. Whenever an item z^* is requested, FPM performs the following three operations, in this order:

1. *Target cleanup.* If z^* was a target of another item y (i.e., $\theta_y = z^*$ for $y \neq z^*$), then θ_y is updated to the successor of z^* . This happens for all items y with this property.
2. *Movement of z^* .* FPM executes one of the two actions: a *partial move* or a *full move*. We will explain how to choose between them later.
 - In the partial move, item z^* is inserted right before θ_{z^*} . (If $\theta_{z^*} = z^*$, this means that z^* does not change its position.)
 - In the full move, item z^* is moved to the front of the list.
3. *Target reset.* θ_{z^*} is set to the front item of the list.

It is illustrative to note a few properties and corner cases of the algorithm.

- Target cleanup is executed only for items following z^* , and thus the successor of z^* exists then (i.e., FPM is well defined).
- If $\theta_{z^*} = z^*$ and a partial move is executed, then z^* is not moved, but the items that targeted z^* now target the successor of z^* .
- For an item x , the items that precede θ_x in the list were requested (each at least once) after the last time x had been requested.

Modes, flavors and states. Fix a pair $\{x, y\}$ such that $y \prec x$, and thus also $\theta_y \preceq y \prec x$. This pair is assigned one of four possible *flavors*, depending on the position of θ_x (cf. Figure 2):

- flavor d (disjoint): if $\theta_y \preceq y \prec \theta_x \preceq x$,
- flavor o (overlapping): if $\theta_y \prec \theta_x \preceq y \prec x$,
- flavor e (equal): if $\theta_y = \theta_x \preceq y \prec x$,
- flavor n (nested): if $\theta_x \prec \theta_y \preceq y \prec x$

For a pair $\{x, y\}$ that is in a mode $\xi \in \{\alpha, \beta, \gamma\}$ and has a flavor $\omega \in \{d, o, e, n\}$, we say that the *state* of $\{x, y\}$ is ξ^ω . Recall that all pairs are initially in mode α , and note that their initial flavor is d . That is, the initial state of all pairs is α^d .

We will sometimes combine the flavors o with e , stating that the pair is in state ξ^{oe} if its mode is ξ and its flavor is o or e . Similarly, we will also combine flavors n and e .

Pair potential. To each pair $\{x, y\}$ of items, we assign a non-negative *pair potential* Φ_{xy} . We abuse the notation and use ξ^ω not only to denote the pair state, but also the corresponding values of the pair potential. That is, we assign the potential $\Phi_{xy} = \xi^\omega$ if pair $\{x, y\}$ is in state ξ^ω . We pick the actual values of these potentials only later in Subsection 4.5.

We emphasize that the states of each item pair depend on work functions for this pair that are easily computable in online manner. Thus, *FPM* may compute the current values of Φ_{xy} , and also compute how their values would change for particular choice of a move.

Choosing the cheaper move. For a step t , let $\Delta_t FPM$ be the cost of *FPM* in this step, and for a pair $\{x, y\} \in \mathcal{P}$, let $\Delta_t \Phi_{xy}$ be the change of the potential of pair $\{x, y\}$ in step t . Let z^* denote the requested item. *FPM* chooses the move (full or partial) with the smaller value of

$$\Delta_t FPM + \sum_{y \prec z^*} \Delta_t \Phi_{z^* y},$$

breaking ties arbitrarily. Importantly, note that the value that *FPM* minimizes involves only pairs including the requested item z^* and items currently preceding it.

4 Analysis of Full-Or-Partial-Move

In this section, we show that for a suitable choice of pair potentials, *FPM* is 3.3904-competitive.

To this end, we first make a few observations concerning how the modes, flavors (and thus states) of item pairs change due to the particular actions of *FPM*. These are summarized in Table 1, Table 2, and Table 3, and proved in Subsection 4.2 and Subsection 4.3.

Next, in Subsection 4.4 and Subsection 4.5, we show how these changes influence the amortized cost of *FPM* pertaining to particular pairs. We show that for a suitable choice of pair potentials, we can directly compare the amortized cost of *FPM* with the cost of *OPT*.

4.1 Structural Properties

Note that requests to items other than x and y do not affect the mutual relation between x and y . Moreover, to some extent, they preserve the relation between targets θ_x and θ_y , as stated in the following lemma.

► **Lemma 4.** *Fix a pair $\{x, y\} \in \mathcal{P}$ and assume that $\theta_y \preceq \theta_x$ (resp. $\theta_y = \theta_x$). If *FPM* serves a request to an item z^* different than x and y , then these relations are preserved. The relation $\theta_y \prec \theta_x$ changes into $\theta_y = \theta_x$ only when θ_y and θ_x are adjacent and $z^* = \theta_y$.*

Proof. By the definition of *FPM*, the targets of non-requested items are updated (during the target cleanup) only if they are equal to z^* . In such a case, they are updated to the successor of z^* . We consider several cases.

- If $z^* \notin \{\theta_x, \theta_y\}$, the targets θ_x and θ_y are not updated.
- If $z^* = \theta_y = \theta_x$, both targets are updated to the successor of z^* , and thus they remain equal.

- If $z^* = \theta_y \prec \theta_x$, target θ_y is updated. If θ_y and θ_x were adjacent (θ_y was an immediate predecessor of θ_x), they become equal. In either case, $\theta_y \preceq \theta_x$.
- If $\theta_y \prec \theta_x = z^*$, target θ_x is updated, in which case the relation $\theta_y \prec \theta_x$ is preserved. ◀

4.2 Mode Transitions

Now, we focus on the changes of modes. It is convenient to look first at how they are affected by the request itself (which induces an update of the work function), and subsequently due to the actions of *FPM* (when some items are swapped). The changes are summarized in the following observation.

► **Observation 5.** *Let z^* be the requested item.*

- *Fix an item $y \succ z^*$. The mode transitions of the pair $\{z^*, y\}$ due to request are $\alpha \rightarrow \alpha$, $\beta \rightarrow \alpha$, and $\gamma \rightarrow \beta$. Subsequent movement of z^* does not further change the mode.*
- *Fix an item $y \prec z^*$. Due to the request, pair $\{z^*, y\}$ changes first its mode due to the request in the following way: $\alpha \rightarrow \beta$, $\beta \rightarrow \gamma$, and $\gamma \rightarrow \gamma$. Afterwards, if *FPM* moves z^* before y , the subsequent mode transitions for pair $\{z^*, y\}$ are $\beta \rightarrow \beta$ and $\gamma \rightarrow \alpha$.*

4.3 State Transitions

Throughout this section, we fix a step and let z^* be the requested item.

We analyze the potential changes for both types of movements. We split our considerations into three cases corresponding to three types of item pairs. The first two types involve z^* as one pair item, where the second item either initially precedes z^* (cf. Lemma 7) or follows z^* (cf. Lemma 8). The third type involves pairs that do not contain z^* at all (cf. Lemma 9).

While we defined 12 possible states (3 modes \times 4 flavors), we will show that α^n , γ^d , and γ^o never occur. This clearly holds at the very beginning as all pairs are then in state α^d .

For succinctness, we also combine some of the remaining states, reducing the number of states to the following six: α^d , β^d , α^{oe} , β^o , β^{ne} and γ^{ne} . (For example, a pair is in state α^{oe} if it is in state α^o or α^e .) In the following we analyze the transitions between them. We start with a simple observation.

► **Lemma 6.** *Fix an item $y \neq z^*$. If *FPM* performs a full move, then the resulting flavor of the pair $\{z^*, y\}$ is d .*

Proof. Item z^* is moved to the front of the list, and its target θ_{z^*} is reset to this item, i.e., $z^* = \theta_{z^*}$. After the movement, we have $z^* \prec \theta_y$. This relation follows trivially if z^* is indeed moved. However, it holds also if z^* was already on the first position: even if $\theta_y = z^*$ before the move, θ_y would be updated to the successor of z^* during the target cleanup. The resulting ordering is thus $\theta_{z^*} = z^* \prec \theta_y \preceq y$, i.e., the flavor of the pair becomes d . ◀

► **Lemma 7.** *Fix $y \prec z^*$. The state transitions for pair $\{z^*, y\}$ are given in Table 1.*

Proof. First, suppose *FPM* performs a full move. The pair $\{z^*, y\}$ is swapped, which changes its mode according to Observation 5 ($\alpha \rightarrow \beta$, $\beta \rightarrow \alpha$, $\gamma \rightarrow \alpha$). By Lemma 6, the flavor of the pair becomes d . This proves the correctness of the transitions in row three of Table 1.

In the rest of the proof, we analyze the case when *FPM* performs a partial move. We consider three sub-cases depending on the initial flavor of the pair $\{z^*, y\}$. For the analysis of mode changes we will apply Observation 5.

■ **Table 1** State transitions for pairs $\{z^*, y\}$ where $y \prec z^*$ right before the request to z^* .

State before move	α^d	β^d	α^{oe}	β^o	β^{ne}	γ^{ne}
State after partial move	β^{ne}	γ^{ne}	β^d or β^o or β^{ne}	α^{oe}	α^d	α^d
State after full move	β^d	α^d	β^d	α^d	α^d	α^d

■ **Table 2** State transitions for pairs $\{z^*, y\}$ where $z^* \prec y$ right before the request to z^* .

State before move	α^d	β^d	α^{oe}	β^o	β^{ne}	γ^{ne}
State after move	α^d	α^d	α^d	α^d	α^d or α^{oe}	β^d or β^o or β^{ne}

- Before the movement, the flavor of the pair was d , i.e., $\theta_y \preceq y \prec \theta_{z^*} \preceq z^*$.
The movement of z^* does not swap the pair, i.e., its mode transitions are $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$. As θ_{z^*} is set to the list front, after the movement $\theta_{z^*} \preceq \theta_y \preceq y \prec z^*$, i.e., the flavor of the pair becomes either e or n . This explains the state transitions $\alpha^d \rightarrow \beta^{ne}$ and $\beta^d \rightarrow \gamma^{ne}$.
- Before the movement, the flavor of the pair was o , i.e., $\theta_y \prec \theta_{z^*} \preceq y \prec z^*$.
Due to the movement, the pair is swapped, and its mode transitions are $\alpha \rightarrow \beta$ and $\beta \rightarrow \alpha$. As θ_{z^*} is set to the list front, we have $\theta_{z^*} \preceq \theta_y$. After the movement, $\theta_y \preceq z^* \prec y$, i.e., the flavor of the pair becomes either o or e . This explains the state transitions $\alpha^{oe} \rightarrow (\beta^o \text{ or } \beta^{ne})$ and $\beta^o \rightarrow \alpha^{oe}$.
- Before the movement, the flavor of the pair was e or n , i.e., $\theta_{z^*} \preceq \theta_y \preceq y \prec z^*$.
The movement swaps the pair, and its mode transitions are $\alpha \rightarrow \beta$, $\beta \rightarrow \alpha$ and $\gamma \rightarrow \alpha$. After the movement $z^* \prec \theta_y$, and target θ_{z^*} is set to the list front, which results in $\theta_{z^*} \preceq z^* \prec \theta_y \preceq y$, i.e., the pair flavor becomes d . This explains the state transitions $\alpha^{oe} \rightarrow \beta^d$, $\beta^{ne} \rightarrow \alpha^d$, and $\gamma^{ne} \rightarrow \alpha^d$ ◀

► **Lemma 8.** Fix $y \succ z^*$. The state transitions for pair $\{z^*, y\}$ are given in Table 2.

Proof. The pair $\{z^*, y\}$ is not swapped due to the request, and thus, by Observation 5, its mode transition is $\alpha \rightarrow \alpha$, $\beta \rightarrow \alpha$, $\gamma \rightarrow \beta$.

If *FPM* performs a full move, the flavor of the pair becomes d by Lemma 6. This explains the state transitions $\alpha^d \rightarrow \alpha^d$, $\beta^d \rightarrow \alpha^d$, $\alpha^{oe} \rightarrow \alpha^d$, $\beta^o \rightarrow \alpha^d$, $\beta^{ne} \rightarrow \alpha^d$, and $\gamma^{ne} \rightarrow \beta^d$.

In the following, we assume that *FPM* performs a partial move, and we will identify cases where the resulting pair flavor is different than d . We consider two cases.

- The initial flavor is d , o or e . That is $\theta_{z^*} \preceq z^* \prec y$ and $\theta_{z^*} \preceq \theta_y \preceq y$. During the target cleanup, θ_y may be updated to its successor, but it does not affect these relations, and in particular we still have $\theta_{z^*} \preceq \theta_y$. Thus, when z^* is moved, it gets placed before θ_y . This results in the ordering $\theta_{z^*} \preceq z^* \prec \theta_y \preceq y$, i.e., the resulting flavor is d .
- The initial flavor is n , i.e., $\theta_y \prec \theta_{z^*} \preceq z^* \prec y$. As $\theta_y \neq z^*$, the target θ_y is not updated during the target cleanup. As z^* is moved right before original position of θ_{z^*} , it is placed after θ_y , and the resulting ordering is $\theta_{z^*} \preceq \theta_y \prec z^* \prec y$. That is, the flavor becomes o or e , which explains the state transitions $\beta^{ne} \rightarrow \alpha^{oe}$ and $\gamma^{ne} \rightarrow (\beta^o \text{ or } \beta^{ne})$. ◀

► **Lemma 9.** Fix $y \prec x$, such that $x \neq z^*$ and $y \neq z^*$. State transitions for pair $\{x, y\}$ are given in Table 3.

■ **Table 3** State transitions for pairs $\{x, y\}$ where $x \neq z^*$ and $y \neq z^*$ right before the request to z^* .

State before move	α^d	β^d	α^{oe}	β^o	β^{ne}	γ^{ne}
State after move	α^d	β^d	α^{oe}	β^o or β^{ne}	β^{ne}	γ^{ne}

Proof. The mode of the pair $\{x, y\}$ is not affected by the request to z^* .

The flavor of the pair $\{x, y\}$ depends on mutual relations between x , y , θ_x and θ_y . By Lemma 4, the only possible change is that θ_x and θ_y were different but may become equal: this happens when they were adjacent and the earlier of them is equal to z^* . We consider four cases depending on the initial flavor of the pair.

- The initial flavor was e ($\theta_y = \theta_x \preceq y \prec x$). As θ_y and θ_x are not adjacent, the flavor remains e .
- The initial flavor was d ($\theta_y \preceq y \prec \theta_x \preceq x$). Suppose $\theta_y = z^*$. As $y \neq z^*$, we have $\theta_y = z^* \prec y \prec \theta_x$. That is, θ_x and θ_y are not adjacent, and thus the flavor remains d .
- The initial flavor was o ($\theta_y \prec \theta_x \preceq y \prec x$). In this case it is possible that θ_y and θ_x are adjacent and $z^* = \theta_y$. The flavor may thus change to e or remain o .
- The initial flavor was n ($\theta_x \prec \theta_y \preceq y \prec x$). Similarly to the previous case, it is possible that θ_x and θ_y are adjacent and $z^* = \theta_x$. The flavor may thus change to e or remain n . ◀

4.4 Amortized Analysis

We set $R = \frac{1}{8}(23 + \sqrt{17}) \leq 3.3904$ as our desired competitive ratio.

In the following, we fix a step t in which item z^* is requested. We partition \mathcal{P} into three sets corresponding to the three types of pairs:

- $\mathcal{P}_1^t \triangleq \{\{y, z^*\} : y \prec z^*\}$ (pairs where z^* is the second item, analyzed in Table 1),
- $\mathcal{P}_2^t \triangleq \{\{y, z^*\} : z^* \prec y\}$ (pairs where z^* is the first item, analyzed in Table 2),
- $\mathcal{P}_3^t \triangleq \{\{x, y\} : x \neq z^* \wedge y \neq z^*\}$ (pairs where z^* is not involved, analyzed in Table 3).

For succinctness, wherever it does not lead to ambiguity, we omit subscripts t , i.e., write $\Delta\Phi_{xy}$ and Δw^{xy} instead of $\Delta_t\Phi_{xy}$ and $\Delta_t w^{xy}$. We also omit superscripts t in \mathcal{P}_1^t , \mathcal{P}_2^t , and \mathcal{P}_3^t .

Our goal is to show the following three bounds:

- $\Delta FPM + \sum_{\{x, y\} \in \mathcal{P}_1} \Delta\Phi_{xy} \leq R \cdot \sum_{\{x, y\} \in \mathcal{P}_1} \Delta w^{xy},$
- $\sum_{\{x, y\} \in \mathcal{P}_2} \Delta\Phi_{xy} \leq R \cdot \sum_{\{x, y\} \in \mathcal{P}_2} \Delta w^{xy},$
- $\sum_{\{x, y\} \in \mathcal{P}_3} \Delta\Phi_{xy} \leq R \cdot \sum_{\{x, y\} \in \mathcal{P}_3} \Delta w^{xy}.$

Note that the left hand sides of these inequalities correspond to the portions of amortized cost of FPM corresponding to sets $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$, while the right hand sides are equal to R times the corresponding portion of the cost of pair-based OPT . Hence, if we can show the above inequalities for every step t , the competitive ratio of R will follow by simply adding them up.

As we show in the sections that follow, these bounds reduce to some constraints involving state potentials $\alpha^d, \beta^d, \alpha^{oe}, \beta^o, \beta^{ne}$ and γ^{ne} . The bounds for \mathcal{P}_2 and \mathcal{P}_3 , while they involve summations over pairs, can be justified by considering individual pairs and the needed constraints are simple inequalities between state potentials, summarized in the following assumption:

► **Assumption 10.** We assume that

- $\alpha^d = 0,$
- all constants $\beta^d, \alpha^{oe}, \beta^o, \beta^{ne}$ and γ^{ne} are non-negative,
- $\alpha^{oe} \leq \beta^{ne} + \frac{1}{2}R \leq \beta^o + \frac{1}{2}R,$
- $\max\{\beta^d, \beta^o\} \leq \gamma^{ne} + \frac{1}{2}R.$

The bound for \mathcal{P}_1 is most critical (not surprisingly, as it corresponds to requesting the second item of a pair). To analyze this bound, the needed constraints, besides the state potentials also need to involve the numbers of pairs that are in these states. This gives rise to a non-linear optimization problem that we need to solve.

4.4.1 Analyzing pairs from set \mathcal{P}_1

The proof of the following lemma is deferred to Subsection 4.5.

► **Lemma 11.** *There exist parameters $\alpha^d, \beta^d, \alpha^{oe}, \beta^o, \beta^{ne}$ and γ^{ne} , satisfying Assumption 10, such that for any step t , it holds that $\Delta F_{PM} + \sum_{\{x,y\} \in \mathcal{P}_1} \Delta \Phi_{xy} \leq R \cdot \sum_{\{x,y\} \in \mathcal{P}_1} \Delta w^{xy}$.*

4.4.2 Analyzing pairs from set \mathcal{P}_2

► **Lemma 12.** *For any step t , it holds that $\sum_{\{x,y\} \in \mathcal{P}_2} \Delta \Phi_{xy} \leq R \cdot \sum_{\{x,y\} \in \mathcal{P}_2} \Delta w^{xy}$.*

Proof. Recall that \mathcal{P}_2 contains all pairs $\{z^*, y\}$, such that $z^* \prec y$. Thus, it is sufficient to show that $\Delta \Phi_{z^*y} \leq R \cdot \Delta w^{z^*y}$ holds for any such pair $\{z^*, y\}$. The lemma will then follow by summing over all pairs from \mathcal{P}_2 .

By Assumption 10, we have

$$\alpha^d - \alpha^{oe} \leq 0, \quad (1)$$

$$\max\{\alpha^d, \alpha^{oe}\} - \beta^{ne} = \alpha^{oe} - \beta^{ne} \leq \frac{1}{2}R, \quad (2)$$

$$\max\{\beta^d, \beta^o, \beta^{ne}\} - \gamma^{ne} = \max\{\beta^d, \beta^o\} - \gamma^{ne} \leq \frac{1}{2}R. \quad (3)$$

We consider two cases depending on the initial mode of the pair $\{z^*, y\}$. In each case, we upper-bound the potential change on the basis of possible state changes of this pair (cf. Table 2).

- The initial mode of $\{z^*, y\}$ is α . By Table 2, this mode remains α , and thus by Observation 3, $\Delta w^{z^*y} = 0$. Then,

$$\begin{aligned} \Delta \Phi_{z^*y} &\leq \max\{\alpha^d - \alpha^d, \alpha^d - \alpha^{oe}\} && \text{(by Table 2)} \\ &= 0 = R \cdot \Delta w^{z^*y}. && \text{(by (1))} \end{aligned}$$

- The initial mode of $\{z^*, y\}$ is β or γ . By Table 2, its mode changes due to the request to z^* , and hence, by Observation 3, $\Delta w^{z^*y} = \frac{1}{2}$. Then,

$$\begin{aligned} \Delta \Phi_{z^*y} &\leq \max\{\alpha^d - \alpha^d, \alpha^d - \beta^d, \alpha^d - \alpha^{oe}, \alpha^d - \beta^o, \\ &\quad \max\{\alpha^d, \alpha^{oe}\} - \beta^{ne}, \\ &\quad \max\{\beta^d, \beta^o, \beta^{ne}\} - \gamma^{ne}\} && \text{(by Table 2)} \\ &\leq \max\{-\beta^d, -\beta^o, \frac{1}{2}R, \frac{1}{2}R\} && \text{(by } \alpha^d = 0, (1), (2) \text{ and (3))} \\ &= \frac{1}{2}R = R \cdot \Delta w^{z^*y}. \end{aligned} \quad \blacktriangleleft$$

4.4.3 Analyzing pairs from set \mathcal{P}_3

► **Lemma 13.** *For any step t , it holds that $\sum_{\{x,y\} \in \mathcal{P}_3} \Delta \Phi_{xy} \leq R \cdot \sum_{\{x,y\} \in \mathcal{P}_3} \Delta w^{xy}$.*

Proof. As in the previous lemma, we show that the inequality $\Delta \Phi_{xy} \leq R \cdot \Delta w^{xy}$ holds for any pair $\{x, y\} \in \mathcal{P}_3$, i.e., for a pair $\{x, y\}$, such that $x \neq z^*$ and $y \neq z^*$. The lemma will then follow by summing over all pairs $\{x, y\} \in \mathcal{P}_3$.

Possible state transitions of such a pair $\{x, y\}$ are given in Table 3. Hence, such a pair either does not change its state (and then $\Delta \Phi_{xy} = 0$) or it changes it from β^o to β^{ne} (and then $\Delta \Phi_{xy} = \beta^{ne} - \beta^o \leq 0$ by Assumption 10). In either case, $\Delta \Phi_{xy} \leq 0 \leq R \cdot \Delta w^{xy}$. ◀

4.4.4 Proof of R -competitiveness

We now show that the three lemmas above imply that FPM is R -competitive.

► **Theorem 14.** *For an appropriate choice of parameters, the competitive ratio of FPM is at most $R = \frac{1}{8}(23 + \sqrt{17}) \leq 3.3904$.*

Proof. Fix any sequence σ consisting of T requests. By summing the guarantees of Lemma 11, Lemma 12, and Lemma 13, we obtain that for any step t , it holds that

$$\Delta_t FPM + \sum_{\{x,y\} \in \mathcal{P}} \Delta_t \Phi_{xy} \leq R \cdot \sum_{\{x,y\} \in \mathcal{P}} \Delta_t w^{xy}.$$

By summing over all steps, observing that the potentials are non-negative and the initial potential is zero (cf. Assumption 10), we immediately obtain that $FPM(\sigma) \leq R \cdot \sum_{t=1}^T \sum_{\{x,y\} \in \mathcal{P}} \Delta w^{xy} \leq R \cdot OPT(\sigma)$. The second inequality follows by Lemma 2. ◀

4.5 Proof of Lemma 11

Again, we focus on a single step t , in which the requested item is denoted z^* . We let $A^d, B^d, A^{oe}, B^o, B^{ne}, C^{ne}$ be the number of items y preceding z^* such that pairs $\{z^*, y\}$ have states $\alpha^d, \beta^d, \alpha^{oe}, \beta^o, \beta^{ne}$ and γ^{ne} , respectively. Let

$$V \triangleq [A^d, B^d, A^{oe}, B^o, B^{ne}, C^{ne}].$$

Note that $\|V\|_1$ is the number of items preceding z^* , and thus also the access cost FPM pays for the request. Moreover, $A^d + B^d$ is the number of items that precede θ_{z^*} . We use \odot to denote scalar product (point-wise multiplication) of two vectors.

We define three row vectors G_{OPT} , G_{PM} , and G_{FM} , such that

$$\begin{pmatrix} G_{PM} \\ G_{FM} \\ G_{OPT} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix} + \begin{pmatrix} \beta^{ne} & (\gamma^{ne} - \beta^d) & (\max\{\beta^d, \beta^o\} - \alpha^{oe}) & (\alpha^{oe} - \beta^o) & -\beta^{ne} & -\gamma^{ne} \\ \beta^d & -\beta^d & (\beta^d - \alpha^{oe}) & -\beta^o & -\beta^{ne} & -\gamma^{ne} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Expressing costs as vector products. Recall that to prove Lemma 11, we need to relate the \mathcal{P}_1 portion of the amortized cost of FPM , i.e., $\Delta FPM + \sum_{\{x,y\} \in \mathcal{P}_1} \Delta \Phi_{xy}$ and the corresponding portion of the cost of pair-based OPT , i.e., $\sum_{\{x,y\} \in \mathcal{P}_1} \Delta w^{xy}$. In the following two lemmas, we show how to express both terms as vector products.

► **Lemma 15.** *It holds that $\sum_{y \prec z^*} \Delta w^{z^*y} = G_{OPT} \odot V$.*

Proof. The right-hand side of the lemma relation is equal to $\frac{1}{2}(A^d + B^d + A^{oe} + B^o + B^{ne})$. By Observation 5, due to request to z^* :

- $A^d + A^{oe}$ pairs change mode from α to β , and
- $B^d + B^o + B^{ne}$ pairs change mode from β to γ .

By Observation 3, each such mode change contributes $\frac{1}{2}$ to the left hand side of the lemma equation. Note that C^{ne} pairs of mode γ do not change their mode due to the request. ◀

► **Lemma 16.** *It holds that $\Delta FPM + \sum_{y \prec z^*} \Delta \Phi_{z^*y} = G \odot V$, where*

- $G = G_{PM}$ if FPM performs a partial move, and
- $G = G_{FM}$ if FPM performs a full move.

Proof. First, assume that FPM performs a partial move. By the definitions of A^d , B^d , A^{oe} , B^o , B^{ne} , and C^{ne} , $\Delta FPM = [1, 1, 2, 2, 2, 2] \odot V$. By Table 1,

$$\begin{aligned} \sum_{y \prec z^*} \Delta \Phi_{z^*y} &\leq [\beta^{ne} - \alpha^d, \gamma^{ne} - \beta^d, \max\{\beta^d, \beta^o, \beta^{ne}\} - \alpha^{oe}, \alpha^{oe} - \beta^o, \alpha^d - \beta^{ne}, \alpha^d - \gamma^{ne}] \odot V \\ &= [\beta^{ne}, \gamma^{ne} - \beta^d, \max\{\beta^d, \beta^o\} - \alpha^{oe}, \alpha^{oe} - \beta^o, -\beta^{ne}, -\gamma^{ne}] \odot V, \end{aligned}$$

where the second equality follows as $\alpha^d = 0$ and $\beta^o \geq \beta^{ne}$ (by Assumption 10). Thus, the lemma holds for a partial move.

Next, assume FPM performs a full move. Then, $\Delta FPM = [2, 2, 2, 2, 2, 2] \odot V$. By Table 1,

$$\begin{aligned} \sum_{y \prec z^*} \Delta \Phi_{z^*y} &= [\beta^d - \alpha^d, \alpha^d - \beta^d, \beta^d - \alpha^{oe}, \alpha^d - \beta^o, \alpha^d - \beta^{ne}, \alpha^d - \gamma^{ne}] \odot V \\ &= [\beta^d, -\beta^d, \beta^d - \alpha^{oe}, -\beta^o, -\beta^{ne}, -\gamma^{ne}] \odot V, \end{aligned}$$

where in the second equality we used $\alpha^d = 0$ (by Assumption 10). Thus, the lemma holds for a full move as well. \blacktriangleleft

Recall that FPM is defined to choose the move that minimizes $\Delta FPM + \sum_{y \prec z^*} \Delta \Phi_{z^*y}$.

► **Corollary 17.** *It holds that $\Delta FPM + \sum_{y \prec z^*} \Delta \Phi_{z^*y} = \min\{G_{PM} \odot V, G_{FM} \odot V\}$.*

Finding Parameters. We may now prove Lemma 11, restated below for convenience.

► **Lemma 11.** *There exist parameters α^d , β^d , α^{oe} , β^o , β^{ne} and γ^{ne} , satisfying Assumption 10, such that for any step t , it holds that $\Delta FPM + \sum_{\{x,y\} \in \mathcal{P}_1} \Delta \Phi_{xy} \leq R \cdot \sum_{\{x,y\} \in \mathcal{P}_1} \Delta w^{xy}$.*

Proof. We choose the following values of the parameters:

$$\begin{aligned} \alpha^d &= 0 & \beta^d &= \frac{1}{16}(5 + 3\sqrt{17}) \approx 1.086 & \alpha^{oe} &= 2 \\ \beta^o &= \frac{1}{16}(1 + 7\sqrt{17}) \approx 1.866 & \beta^{ne} &= \frac{1}{16}(9 - \sqrt{17}) \approx 0.305 & \gamma^{ne} &= 2 \end{aligned}$$

It is straightforward to verify that these values satisfy the conditions of Assumption 10. We note that relation $\alpha^{oe} \leq \beta^{ne} + \frac{1}{2}R$ holds with equality.

By Corollary 17, $\Delta FPM + \sum_{\{x,y\} \in \mathcal{P}_1} \Delta \Phi_{xy} = \Delta FPM + \sum_{y \prec z^*} \Delta \Phi_{z^*y} = \min\{G_{PM} \odot V, G_{FM} \odot V\}$. On the other hand, by Lemma 15, $\sum_{\{x,y\} \in \mathcal{P}_1} \Delta w^{xy} = \sum_{y \prec z^*} \Delta w^{z^*y} = G_{OPT} \odot V$. Hence, it remains to show that $\min\{G_{PM} \odot V, G_{FM} \odot V\} \leq R \cdot G_{OPT} \odot V$.

We observe that

$$\begin{aligned} G_{PM} &= \frac{1}{16} \cdot [25 - \sqrt{17}, 43 - 3\sqrt{17}, 1 + 7\sqrt{17}, 63 - 7\sqrt{17}, 23 + \sqrt{17}, 0], \\ G_{FM} &= \frac{1}{16} \cdot [37 + 3\sqrt{17}, 27 - 3\sqrt{17}, 5 + 3\sqrt{17}, 31 - 7\sqrt{17}, 23 + \sqrt{17}, 0]. \end{aligned}$$

Let $c = \frac{1}{4}(\sqrt{17} - 1)$ and let $G_{COMB} = c \cdot G_{PM} + (1 - c) \cdot G_{FM}$. Then,

$$G_{COMB} = \frac{1}{16} \cdot [23 + \sqrt{17}, 23 + \sqrt{17}, 23 + \sqrt{17}, 23 + \sqrt{17}, 23 + \sqrt{17}, 0].$$

Now for any vector V ,

$$\begin{aligned} \min\{G_{PM} \odot V, G_{FM} \odot V\} &\leq c \cdot G_{PM} \odot V + (1 - c) \cdot G_{FM} \odot V \\ &= (c \cdot G_{PM} + (1 - c) \cdot G_{FM}) \odot V \\ &= G_{COMB} \odot V = R \cdot G_{OPT} \odot V, \end{aligned}$$

completing the proof. \blacktriangleleft

5 Final Remarks

The most intriguing question left open in our work is whether competitive ratio of 3 can be achieved. We have shown computationally (see the full version of the paper [14]) that 3-competitive algorithms exist for lists with up to 6 items.

However, even for short lists the definition of such 3-competitive algorithm remains elusive. For many online problems, the most natural candidate is the generic work function algorithm. This algorithm is 2-competitive in the LUP_S model [9]. However, our computer-aided calculation of its performance shows that its ratio is larger than 3 already for 5 items (see the full version of the paper [14]). It is 3-competitive for lists of length up to 4, though.

We do not know whether the analysis of FPM is tight. For the specific choice of parameters used in the paper, we verified that FPM is 3-competitive for lists of length 3, but not better than 3.04-competitive for lists of length 5 (see the full version of the paper [14]).

The focus of this paper is on the LUP_1 model (also known as P^1); we believe that the setting of $d = 1$ captures the essence and hardness of the deterministic variant. That said, extending the definition and analysis of FPM to the P^d model (for arbitrary d) is an interesting open problem that deserves further investigation.

References

- 1 Susanne Albers. A competitive analysis of the list update problem with lookahead. *Theoretical Computer Science*, 197(1):95–109, 1998. doi:10.1016/S0304-3975(97)00026-1.
- 2 Susanne Albers. Improved randomized on-line algorithms for the list update problem. *SIAM Journal on Computing*, 27(3):682–693, 1998. doi:10.1137/S0097539794277858.
- 3 Susanne Albers. Online list update. In *Encyclopedia of Algorithms*, pages 598–601. Springer, 2008. doi:10.1007/978-0-387-30162-4_266.
- 4 Susanne Albers and Maximilian Janke. New bounds for randomized list update in the paid exchange model. In *37th Int. Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 12:1–12:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.STACS.2020.12.
- 5 Susanne Albers and Sonja Lauer. On list update with locality of reference. *Journal of Computer and System Sciences*, 82(5):627–653, 2016. doi:10.1016/J.JCSS.2015.11.005.
- 6 Susanne Albers, Bernhard Von Stengel, and Ralph Werchner. A combined BIT and TIMES-TAMP algorithm for the list update problem. *Information Processing Letters*, 56(3):135–139, 1995. doi:10.1016/0020-0190(95)00142-Y.
- 7 Susanne Albers and Jeffery Westbrook. Self-organizing data structures. In *Online Algorithms, The State of the Art*, pages 13–51. Springer, 1996. doi:10.1007/BFb0029563.
- 8 Christoph Ambühl, Bernd Gärtner, and Bernhard von Stengel. Optimal lower bounds for projective list update algorithms. *ACM Transactions on Algorithms*, 9(4):1–18, 2013. doi:10.1145/2500120.
- 9 Eric J. Anderson, Kirsten Hildrum, Anna R. Karlin, April Rasala, and Michael E. Saks. On list update and work function algorithms. In *Proc. 7th Annual European Symposium on Algorithms (ESA)*, pages 289–300. Springer, 1999. doi:10.1007/3-540-48481-7_26.
- 10 Spyros Angelopoulos, Reza Dorrigiv, and Alejandro López-Ortiz. List update with locality of reference. In *Proc. 8th Latin American Symposium on Theoretical Informatics (LATIN)*, pages 399–410. Springer, 2008. doi:10.1007/978-3-540-78773-0_35.
- 11 Yossi Azar, Shahar Lewkowicz, and Varun Suriyanarayana. List update with prediction. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 15436–15444. AAAI Press, 2025. doi:10.1609/AAAI.V39I15.33694.
- 12 Yossi Azar, Shahar Lewkowicz, and Danny Vainstein. List update with delays or time windows. In *Proc. 51st Int. Colloquium on Automata, Languages, and Programming (ICALP)*, pages 15:1–15:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ICALP.2024.15.

- 13 Ran Bachrach, Ran El-Yaniv, and M. Reinstadtler. On the competitive theory and practice of online list accessing algorithms. *Algorithmica*, 32(2):201–245, 2002. doi:10.1007/S00453-001-0069-8.
- 14 Mateusz Basiak, Marcin Bienkowski, Martin Böhm, Marek Chrobak, Łukasz Jeż, Jiří Sgall, and Agnieszka Tatarczuk. A 3.3904-competitive online algorithm for list update with uniform costs. [arXiv:2503.17264](#). Full version.
- 15 Jon L. Bentley and Catherine C. McGeoch. Amortized analyses of self-organizing sequential search heuristics. *Communications of the ACM*, 28(4):404–411, 1985. doi:10.1145/3341.3349.
- 16 Joan Boyar, Shahin Kamali, Kim S. Larsen, and Alejandro López-Ortiz. On the list update problem with advice. *Information and Computation*, 253:411–423, 2017. doi:10.1016/J.IC.2016.06.007.
- 17 Reza Dorrigiv, Martin R. Ehmsen, and Alejandro López-Ortiz. Parameterized analysis of paging and list update algorithms. *Algorithmica*, 71(2):330–353, 2015. doi:10.1007/S00453-013-9800-5.
- 18 Alexander Golynski and Alejandro López-Ortiz. Optimal strategies for the list update problem under the MRM alternative cost model. *Information Processing Letters*, 112(6):218–222, 2012. doi:10.1016/J.IPL.2011.12.001.
- 19 Sandy Irani. Two results on the list update problem. *Information Processing Letters*, 38(6):301–306, 1991. doi:10.1016/0020-0190(91)90086-W.
- 20 Shahin Kamali. Online list update. In *Encyclopedia of Algorithms*, pages 1448–1451. Springer, 2016. doi:10.1007/978-1-4939-2864-4_266.
- 21 Shahin Kamali and Alejandro López-Ortiz. *A Survey of Algorithms and Models for List Update*, pages 251–266. Springer, 2013. doi:10.1007/978-3-642-40273-9_17.
- 22 Conrado Martinez and Salvador Roura. On the competitiveness of the move-to-front rule. *Theoretical Computer Science*, 242(1-2):313–325, 2000. doi:10.1016/S0304-3975(98)00264-3.
- 23 J. Ian Munro. On the competitiveness of linear search. In *Proc. 8th Annual European Symposium on Algorithms (ESA)*, pages 338–345. Springer, 2000. doi:10.1007/3-540-45253-2_31.
- 24 Nick Reingold, Jeffery Westbrook, and Daniel D Sleator. Randomized competitive algorithms for the list update problem. *Algorithmica*, 11(1):15–32, 1994. doi:10.1007/BF01294261.
- 25 Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985. doi:10.1145/2786.2793.
- 26 Boris Teia. A lower bound for randomized list update algorithms. *Information Processing Letters*, 47(1):5–9, 1993. doi:10.1016/0020-0190(93)90150-8.