Online List Update with Predictions

ARGHYA BHATTACHARYA, Stony Brook University, USA
SHAHIN KAMALI, York University, Canada
HELEN XU, Lawrence Berkeley National Laboratory, USA

We consider the classic list update problem under a prediction setting, where, upon accessing any item x, a possibly faulty oracle predicts the next two times (indices in the input sequence) where x is requested again. We present an algorithm named Penalty Distance Algorithm (PDA), which works by updating a list into a new configuration that minimizes the sum of "penalty", the deviation of a candidate configuration from optimal partial orderings of items when projected on lists of length 2, and "distance", which measures the cost of updating the list from its current state to the candidate configuration. We conjecture that PDA has a competitive ratio better than all existing algorithms, including offline algorithms when predictions are correct and can be combined with Move-To-Front (MTF) to offer a Pareto-efficient solution that is robust when predictions are faulty.

CCS Concepts: • **Theory of computation** → *Online learning algorithms.*

Additional Key Words and Phrases: ML Advice, List Update, Online Algorithm

1 INTRODUCTION

In the ***list update problem*** [12], the input is an online sequence of access requests to items in a linked list. To serve a request to an item at position i, an online algorithm makes a linear scan of the list until it finds the item; this linear scan has an access cost of i. After accessing the item, the algorithm can move the item to a position closer to the front using a ***free exchange*** that has no cost. Moreover, it can make any number of ***paid exchanges***, each swapping two consecutive items at a unit cost. Given an input request sequence, a list update algorithm's objective is to maintain the list in a way that minimizes the total cost (i.e., the sum of costs for all accesses and paid exchanges). List update has a long history of study (see, e.g., [9] for a survey), and was one of the two problems studied by Sleator and Tarjan in their seminal paper [12] that defines competitive analysis as a worst-case measure for the analysis of online algorithms. The other problem is the ***caching problem***, which asks for maintaining a fast but small memory (a cache) in a way to minimize the total number of requests to memory pages outside of the cache (number of faults). For the list update problem, ***Move To Front (MTF)***, which moves an accessed item to the front of the list (using a free exchange), is 2-competitive [12], and it is the best that a deterministic algorithm can achieve [8]. Relatably, the best deterministic algorithm for caching is ***Least Recently Used (LRU)***, which evicts the least recently used page from the cache whenever eviction is needed and has a competitive ratio of k for a cache of size k [12]. Essentially, LRU and MTF both prefer to keep the recently used items handy.

2 LIST UPDATE ALGORITHM WITH ML ADVISOR

We investigate the list update problem in light of the recent development for the caching problem under a prediction model. In this line of research, [7, 10, 11] upon a request to a page x, a prediction is made available concerning the next time x is requested in the input sequence, that is, the index of the next request to x in the input sequence. We study similar predictions for the list update problem. Our starting observation is that, unlike the paging problem where correct predictions about the next request ensure optimality via a simple algorithm that evicts the page that is requested furthest, the list update problem is NP-hard [2], and we cannot hope for optimal solutions even if predictions are correct.

Nevertheless, we get hopeful from an initial success: For lists of two items, correct predictions about the next request ensure optimality via a simple algorithm that only uses free exchanges: for a list formed by items a and b, whenever there is a request to an item, say b, in the second position, move it to the front if and only if the next request to b appears earlier than the next request to a. However, simple predictions about the next request reach their limit for lists of length 3. In particular, when there is a request to the item at the third position in a list of length 3, an optimal algorithm sometimes needs to use a paid exchange to swap the first two items and then move the requested item between the first two using a free exchange [1, 6]. To make such decisions, however, the algorithm requires the next *two* requests for each item. In other words, one can achieve optimality with correct predictions about the next two times each item is requested; but predictions about the next request is not sufficient.

2.1 PDA Algorithm

We design a prediction model where the list update algorithm has access to predictions concerning the next two occurrences of the requested item in the request sequence. We refer to this algorithm as ***Penalty Distance Algorithm (PDA)***. As each request is served, the list goes from one state to another through some paid exchanges and free exchanges. For that, PDA considers every possible **configuration** (ordering) of the list and computes two types of costs for each configuration, which we refer to as ***distance*** and ***penalty*** costs. PDA chooses the configuration that minimizes the sum of the penalty and the distance.

The idea behind penalty cost is to maintain an ideal configuration of the list that is aligned with an optimal ordering for lists of length 2 formed by the requested item and other items. Suppose b is the requested item, and let a be any item in the list. If the current ordering is (a, b), and the next three requests are predicted to be $\langle b, b, a, \cdots \rangle$, then a should be located after b in an ideal list. On the other hand, if the next three requests are predicted to be $\langle a, a, b, \cdots \rangle$, then a should be located before b in an ideal list. For any other continuation of the predicted input, the relative order of a and b does not matter. Any violation of the ideal ordering in the above scheme translates to one unit of "penalty" in the PDA algorithm.

Upon accessing an item x in the list, PDA colors all other items based on their relative order with x. For each possible configuration C of the list, it computes the number of items whose ordering with x in C is not ideal; the number of such items will be the *penalty* associated with x. PDA also computes the **distance** of C from the current configuration, defined as the minimum number of free and paid exchanges to convert the current configuration of the list to C. The algorithm moves to a configuration with a minimum sum of penalty and distance. We are working on proof to show the following upper bound for the competitive ratio of PDA. If correct, the algorithm improves the best existing approximation algorithm with an approximation factor of $1.6\bar{6}$ [6].

Conjecture 1. When the prediction is correct, using the proper potential function for the analysis, we show PDA's competitive ratio is ≤ 1.6.

2.2 Pareto-efficiency

We study the trade-off between *consistency* (competitive ratio when predictions are all correct) and *robustness* (competitive ratio when predictions are adversarial) for list update algorithms. PDA has good consistency, but it fails if predictions are incorrect. We will use techniques like that of [3–5] to PDA with MTF (which has optimal robustness) to achieve Pareto-efficient algorithms with good consistency and robustness trade-off.

REFERENCES

[1] Susanne Albers. 1998. A competitive analysis of the list update problem with lookahead. *Theoretical Computer Science* 197, 1 (1998), 95–109.

[2] Christoph Ambühl. 2000. Offline List Update is NP-hard. In *Algorithms - ESA 2000*, Mike S. Paterson (Ed.). Springer, Berlin, Heidelberg, 42–51.

[3] Spyros Angelopoulos, Christoph Dürr, Shendan Jin, Shahin Kamali, and Marc P. Renault. 2020. Online Computation with Untrusted Advice. In *Proceedings of the* 11^{th} *Innovations in Theoretical Computer Science Conference (ITCS) (LIPICS, Vol. 151)*, Thomas Vidick (Ed.). Dagstuhl Publishing, Germany, Seattle, WA, USA, 52:1–52:15.

[4] Antonios Antoniadis, Christian Coester, Marek Eliás, Adam Polak, and Bertrand Simon. 2020. Online metric algorithms with untrusted predictions. In *Proceedings of the* 37^{th} *International Conference on Machine Learning (ICML)*, Vol. 119. PMLR, Virtual, 345–355.

[5] Antonios Antoniadis, Christian Coester, Marek Eliáš, Adam Polak, and Bertrand Simon. 2023. Online Metric Algorithms with Untrusted Predictions. *ACM Trans. Algorithms* 19, 2, Article 19 (apr 2023), 34 pages.

[6] Joan Boyar, Shahin Kamali, Kim S. Larsen, and Alejandro López-Ortiz. 2017. On the list update problem with advice. *Information and Computation* 253 (2017), 411–423. LATA 2014.

[7] Jakub Chledowski, Adam Polak, Bartosz Szabucki, and Konrad Zolna. 2021. Robust Learning-Augmented Caching: An Experimental Study. In *Proceedings of the* 38^{th} *International Conference on Machine Learning (ICML)*. PMLR, Virtual, 14 pages.

[8] Sandy Irani. 1991. Two results on the list update problem. *Inform. Process. Lett.* 38, 6 (1991), 301–306.

[9] Shahin Kamali and Alejandro López-Ortiz. 2013. *A Survey of Algorithms and Models for List Update.* Lecture Notes in Comput. Sci., Vol. 8066. Springer, Berlin, Heidelberg. 251—-266 pages.

[10] Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. 2020. Online Scheduling via Learned Weights. In *Proc. of the Symposium on Discrete Algorithms (SODA)*. ACM-SIAM, Salt Lake City, Utah, USA, 1859–1877.

[11] Thodoris Lykouris and Sergei Vassilvitskii. 2018. Competitive Caching with Machine Learned Advice. In *Proc. of the* 35^{th} *International Conference on Machine Learning (ICML)*. PMLR, Stockholm, Sweden, 27 pages.

[12] Daniel D. Sleator and Robert E. Tarjan. 1985. Amortized efficiency of list update and paging rules. *Commun. ACM* 28 (1985), 202–208. Issue 2.