Supplementary material for: Competitive Caching with Machine Learned Advice

Thodoris Lykouris [1] **Sergei Vassilvitskii** [2]

A. Bad example for trusting the oracle

Recall that in Section 3.1 we showed that simply following the oracle's recommendations may lead to very high competitive ratios. The initial example relied on items whose predicted times have passed, but were predicted to be low enough that they are never evicted.

It is tempting to "fix" this approach by evicting elements whose predicted times have passed, but follow the same idea otherwise. Formally, let $h(j,t)$ denote the last prediction about z_j at or prior to time t. At time t this "fixed" approach evicts an arbitrary item from the set $S_t = \{j : h(j,t) < t\}$ if $S_t \neq \emptyset$ and $\arg\max_{z_i \in \text{Cache(t)}} h(i,t)$ otherwise. We show that the competitive ratio of this algorithm is also unbounded even when the average absolute loss is constant. Assume a cache of size $k = 3$ and four elements a, b, c, d. The initial configuration is a, b, c, when element d arrives. Let the total length of the sequence be T. The true sequence has element b at the last time T, element c at times $2^r + 1$ where r is odd, d at times $2^r + 1$ where r is even, and a at all other times. The oracle predicts the next appearance of a and b correctly but is incorrect on elements c and d and always predicts their next appearance time at $T + 1$. The optimal algorithm evicts b, and has two cache misses (when d arrives for the first time, and when b arrives at time T). On the other hand, the described algorithm keeps a, b always in the cache and incurs a cache miss for each appearance of c, d, for a total of $\log(T)$ misses. This is despite the fact that the average absolute loss is again a constant: $\eta_1/T = 3$ as the errors form a geometric series.

B. Proof of Theorem 1

Theorem 1 restated: For the caching problem, let A be an α-robust algorithm and B a γ-competitive algorithm. We can then create a black-box algorithm ALG that is both 9α-robust and 9γ-competitive.

Proof. We proceed by simulating A and B in parallel on the dataset, and maintaining the cache state and the number of misses incurred by each. Our algorithm switches between following the strategy of A and the strategy of B. Let $c_t(A)$ and $c_t(B)$ denote the cost (number of misses) of A and B up to time t. Without loss of generality, let ALG begin by following strategy of A; it will do so until a time t where $c_t(A) = 2 \cdot c_t(B)$. At this point ALG switches to following the eviction strategy of B, doing so until the simulated cost of B is double that of A: a time t' with $c_{t'}(B) = 2 \cdot c_{t'}(A)$. At this point it switches back to following eviction strategy of A, and so on. When ALG switches from A to B, the elements that A has in cache may not be the same as those that B has in the cache. In this case, it needs to reconcile the two. However, this can be done lazily (at the cost of an extra cache miss for every element that needs to be reconciled). To prove the bound on the performance of the algorithm, we need to show that $c_t(ALG) \leq 9 \cdot min(c_t(A), c_t(B))$ for all t. We decompose the cost incurred by ALG into that due to following the different algorithms, which we denote by $f_t(ALG)$, and that due to reconciling caches, $r_t(ALG)$.

We prove a bound on the following cost f_t by induction on the number of switches. Without loss of generality, suppose that at time t, ALG switched from A to B, and at time t' it switches from B back to A. By induction, suppose that $f_t(ALG) \leq 3\min(c_t(A), c_t(B)) = 3c_t(B)$, where the equality follows since ALG switched from A to B at time t. In both cases, assume that caches are instantly reconciled. Then:

$$\begin{aligned} f_{t'}(ALG) &= f_t(ALG) + (c_{t'}(B) - c_t(B)) \\ &= f_t(ALG) + 2c_{t'}(A) - 1/2 c_t(A) \\ &\leq 3c_t(B) + 2(c_{t'}(A) - c_t(A)) + 3/2 \cdot c_t(A) \\ &= 3c_t(A) + 2(c_{t'}(A) - c_t(A)) \\ &\leq 3c_{t'}(A) \\ &= 3\min(c_{t'}(A), c_{t'}(B)) \end{aligned}$$

What is left is to bound the following cost for the time since the last switch. Let s denote the time of the last switch and, assume without loss of generality that it was done from A

*Equal contribution [1]Cornell University, Ithaca, NY, USA [2]Google Research, New York, NY, USA. Correspondence to: Thodoris Lykouris <teddlyk@cs.cornell.edu>, Sergei Vassilvitskii <sergeiv@google.com>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

to B. Let s' denote the last time step. By the previous set of inequalities (changing the second equation to inequality) and the fact that the algorithm never switched back to A after s, it holds that $f_{s'}(ALG) \leq 3c_{s'}(A) \leq 6\min(c_{s'}(A), c_{s'}(B))$.

To bound the reconciliation cost, assume the switch at time t is from A to B. We charge reconciling each element in $B \setminus A$ to the cache miss when the element was last evicted by A. Therefore the overall reconciliation cost is bounded by $r_t(ALG) \leq c_t(A) + c_t(B) \leq 3\min(c_t(A), c_t(B)$. □

C. Proofs for Marker Algorithm

In this section, we provide the proofs of two crucial lemmas in the classical Marker algorithm analysis. The proofs are due to Fiat et al. (Fiat et al., 1991) and are stated for completeness.

Lemma 1 restated: Let L be the number of clean elements. Then the optimal algorithm suffers at least $L/2$ cache misses.

Proof. We denote by ℓ_r the number of clean elements at phase r. Let d_r^I be the number of elements that are in the cache of the optimal solution but not in the cache of the algorithm's solution at the beginning of phase r and let d_r^F be the number of elements that are in this set at the end of the phase. Then, the number of cache misses of the optimal solution are at least the difference $\ell_r - d_r^I$ (as any clean element that was not in the initial complement would cause one cache miss) and also at least d_r^F (since these many elements did not arise during the phase and the offline algorithm has for sure incurred these many misses). Hence, for each r, we know that:

$$\#\text{cache misses in opt during phase } r \leq \min\left(\ell_r - d_r^I, d_r^F\right) \leq \frac{\ell_r - d_r^I + d_r^F}{2}.$$

Telescoping and using that $d_0^I = 0$ (since both start with the same cache elements) concludes the lemma. □

Lemma 2 restated: Let L be the number of clean elements. Then the expected number of cache misses of the marker algorithm is $L \cdot H_k$ when randomly tie-breaking across unmarked elements.

Proof. We again denote by ℓ_r the number of clean elements at phase r. The worst-case scenario is that all the clean elements come in the beginning since the algorithm needs to remove more elements without knowing whether it should. After all ℓ_r elements have arrived, there are $k - \ell_r$ stale elements that the algorithm unfortunately does not know. The expected number of misses of the algorithm is the expected number of stale elements that are not in the cache when requested. Let's think of this probability for the i-th such element. Every clean element removes another element. If it is stale and before the position $k - i - \ell_r$, another element will be removed when its turn comes. So, for the i-th stale element, each clean element corresponds to the removal of one element in the last $k - i + 1$ slots and this is uniformly at random. Since there are ℓ_r of those, the probability that i-th element will be the one removed is at most $\frac{\ell_r}{k-i+1}$ for the i-th time where $i \in [1, \ell_r - 1]$. Summing over all those positions, the number of cache misses of Marker in this phase is at most $\ell_r H_k$. □

D. Proof of Lemma 3

In this section, we provide the proof of the lemma connecting spread to absolute and squared loss. Before doing so, we provide a useful auxiliary lemma.

Lemma 4 For odd $T = 2n + 1$, one pair (A_T, B_T) minimizing either absolute or squared loss subject to the constraints of the spread definition is $A_{2n+1} = (0 \dots 2n)$ and $B_T = (n \dots n)$.

Proof. First we show that there exists a B_T minimizing the loss with $b_i = b_j$ for all i, j. Assume otherwise; then there exist two subsequent i, j with $b'_i > b'_j$. Since $a_i < a_j + 1$ by the assumption on spread, $\min_{x \in \{b_i, b_j\}}\{\ell(a_i, x) + \ell(a_j, x)\} \leq \ell(a_i, b_i) + \ell(a_j, b_j)$. Applying this recursively, we conclude that such a B_T exists.

Second, we show that there exist an A_T that consists of elements $a_{i+1} = a_i + 1$. Since the elements of B_T are all equal to b, the sequence $\sum_{i=0}^{2n} \ell(a_i, b)$ is minimized for both absolute and squared loss when $a_i = b + i - n$.

Last, the exact value of b does not make a difference and therefore we can set it to be $b = n$ concluding the lemma. □

Lemma 3 restated: For absolute loss, $\ell_1(A, B) = \sum_i |a_i - b_i|$, the spread of ℓ_1 is $S_{\ell_1}(m) \leq \sqrt{4m + 1}$. For squared loss, $\ell_2(A, B) = \sum (a_i - b_i)^2$, the spread of ℓ_2 is $S_{\ell_2}(m) \leq \sqrt[3]{14m}$.

Proof. It will be easier to restrict ourselves to odd $T = 2n + 1$ and also assume that $T \geq 3$. This will give an upper bound on the spread (which is tight up to small constant factors). By Lemma 4, a pair of sequence minimizing absolute/squared loss is $A_T = (0, \dots, 2n)$ and $B_T = (n, \dots, n)$. We now provide bounds on the spread based on this sequence, that is we find a $T = 2n + 1$ that satisfies the inequality $\ell(A_T, B_T) \leq m$.

Absolute loss: The absolute loss of the above sequence is:

$$\ell(A_T, B_T) = 2 \cdot \sum_{j=1}^{n} j = 2 \cdot \frac{n(n+1)}{2} = n(n+1) = \\ = \frac{T-1}{2} \cdot \frac{T+1}{2} = \frac{T^2-1}{4}.$$

A T that makes $\ell(A_T, B_T) \geq m$ is $T = \sqrt{4m+1}$. Therefore, for absolute loss $S_\ell(m) \leq \sqrt{4m+1}$.

Squared loss: The squared loss of the above sequence is:

$$\ell(A_T, B_T) = 2 \cdot \sum_{j=1}^{n} j^2 = 2 \cdot \frac{n(n+1)(2n+1)}{6} \\ = \frac{(T^2-1) \cdot T}{12} = \frac{T^3-T}{12} \geq \frac{8T^3}{9 \cdot 12} = \frac{2T^3}{27}$$

where the inequality holds because $T \geq 3$.

A T that makes $\ell(A_T, B_T) \geq m$ is $T = \sqrt[3]{14m}$. Therefore, for squared loss $S_\ell(m) \leq \sqrt[3]{14m}$. □

References

Fiat, A., Karp, R. M., Luby, M., McGeoch, L. A., Sleator, D. D., and Young, N. E. Competitive paging algorithms. *J. Algorithms*, 12(4):685–699, December 1991. ISSN 0196-6774. doi: 10.1016/0196-6774(91)90041-V. URL http://dx.doi.org/10.1016/0196-6774(91)90041-V.