No-Regret Caching via Online Mirror Descent

IEEE International Conference on Communications (ICC) 2021

Tareq Si Salem [2,1] Giovanni Neglia[1,2] Stratis Ioannidis[3]

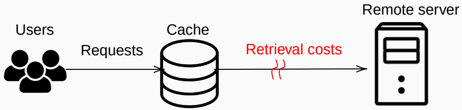
June 16, 2021

[1]Inria [2]Université Côte d'Azur

[3]Northeastern University

Caching

Caching



We can place a local cache with finite capacity to satisfy some of the requests.

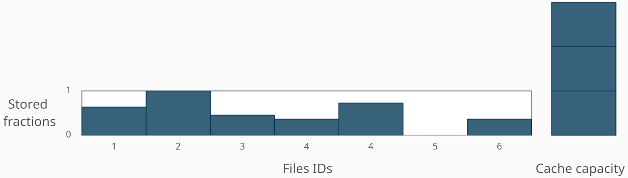
Classical Caching Policies

- Least recently used (LRU)
- Least frequently used (LFU)
- ... many more variants based on LRU and LFU

System

- Catalog of files $\mathcal{N} = \{1, 2, \ldots, N\}$
- $w_i \in \mathbb{R}_+$ cost to serve file i from the remote server
- $k \in \mathbb{N}$ is the cache capacity
- The cache can store fractions of files. The set of valid cache configurations is $\mathcal{X} = \left\{\mathrm{x} \in [0,1]^N : \sum_{i=1}^N x_i = k\right\}$

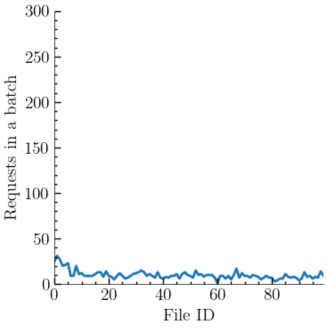
In a followup work we also consider integral caches.[1]



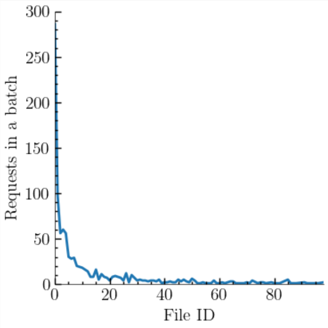
[1] T. Si Salem et al., *arXiv preprint arXiv:2101.12588* **2021**.

Requests

- We consider that requests may arrive in batches of size R, where a given file is requested at most h times



(a) Diverse request batch ($\frac{R}{h} \approx 35.7$)



(b) Concentrated request batch ($\frac{R}{h} \approx 3.4$)

- $\frac{R}{h}$ represented the diversity of the requests in a batch

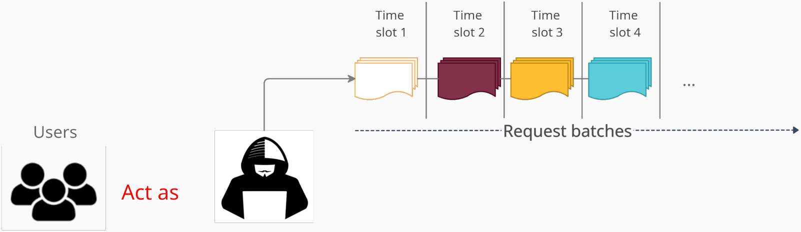
Cost

When a request batch r_t arrives, the cache incurs the following cost:

$$f_{\mathrm{r}_t}(\mathrm{x}_t) = \sum_{i=1}^{N} w_i r_{t,i}(1 - x_{t,i}).$$

Uncertain Environment $\approx$ Adversary

- Files popularity can change over time in an unpredictable manner.
- We target robust performance: the users can behave as adversaries selecting the request process to harm the system.



Regret as a Performance Metric

$$\begin{aligned}\text{Regret} &= \text{Total cost of the policy} - \text{Total cost of the static optimum} \\ &= \sum_{t=1}^{T} f_{\mathrm{r}_t}(\mathrm{x}_t) - \sum_{t=1}^{T} f_{\mathrm{r}_t}(\mathrm{x}_*)\end{aligned}$$

where x_* is the optimal static cache configuration.

- If the average regret (Regret$/T$) tends to 0 when T is large, then the policy experiences on average the same costs as the best static optimum!
- This happens when the regret is sublinear in T. Such policy is called a no-regret policy.

Online $\neq$ Offline

- One could get inspired from offline optimization and select a new state trying to minimize the aggregate cost. This can fail catastrophically, because of the adversarial nature of the requests!

Online Gradient Descent for Caching

Paschos et al.[2] proposed OGD for online caching, that updates the cache state as follows:

- When a request is received r_t the system incurs a cost $f_{\mathsf{r}_t}(\mathsf{x}_t)$
- Take a gradient update step $\tilde{\mathsf{x}}_{t+1} = \mathsf{x}_t - \eta \nabla f_{\mathsf{r}_t}(\mathsf{x}_t)$
- Project $\tilde{\mathsf{x}}_{t+1}$ to the set of valid cache configurations $\mathcal{X}$
- Obtain the new state $\mathsf{x}_{t+1} = \Pi_{\mathcal{X}}(\tilde{\mathsf{x}}_{t+1})$

[2] G. S. Paschos et al. in IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, **2019**, pp. 235–243.

Online Gradient Descent for Caching

- It is a no-regret policy!
- This result is proved in Paschos et al.[3] only for $R = h = 1$

[3] G. S. Paschos et al. in IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, **2019**, pp. 235–243.

Contributions

- First to apply the more general online mirror descent (OMD), a family of gradient methods that includes OGD as a special case
- Analyse OGD in the case $R \neq h \neq 1$. We also improve on the Paschos et al.'s bound (valid only for $R = h = 1$) by a factor at least $\sqrt{2}$
- Show that another policy in the OMD family has lower regret for diverse adversarial requests or small cache sizes (difficult scenarios), and enjoys lower computational cost
- Prove that the optimal OMD strategy depends on the diversity of the request batches
- Evaluate the policies on synthetic traces and a real CDN trace

Online Mirror Descent (OMD)

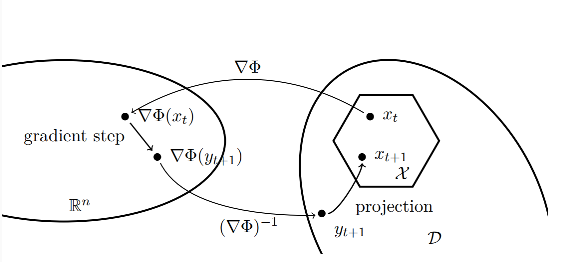


Figure 2: OMD update rule[4].

Different mirror maps Φ give different algorithms.

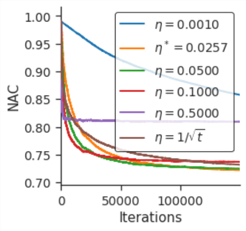
[4] S. Bubeck, *Found. Trends Mach. Learn.* **2015**, *8*, 231–357.

Negative Entropy Mirror Map

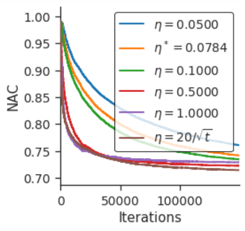
- We propose to use OMD with the negative entropy mirror map
$$\Phi(\mathbf{x}) = \sum_{i=1}^{N} x_i \log(x_i)$$
.
- OMD under the neg-entropy mirror map ($\mathrm{OMD}_{\mathrm{NE}}$) adapts the cache state via a multiplicative rule, as opposed to the additive rule of OGD
$$\tilde{x}_{t+1,i} = x_{t,i}\, e^{-\eta \frac{\partial f_{r_t}(x_t)}{\partial x_i}}.$$
- The mirror map is more suitable to the geometry of $\mathcal{X}$
- The projection is less costly than in OGD

$\mathcal{O}(k)$ vs. $\mathcal{O}(N)$ for $R = h = 1$.

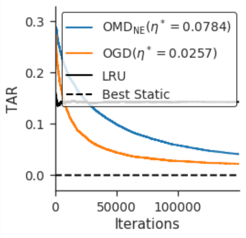
$\mathcal{O}(N + k\log k)$ vs. $\mathcal{O}\left(N^2\right)$ for general values of R, h.



(a) Normalized Time Averaged Cost of OGD



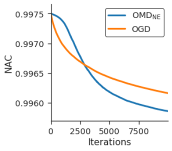
(b) Normalized Time Averaged Cost of $\mathrm{OMD}_{\mathrm{NE}}$



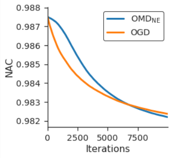
(c) Time-Averaged Regret

The learning rate denoted by η^* is the learning rate that gives the tightest worst-case regrets for OGD and $\mathrm{OMD}_{\mathrm{NE}}$. While this learning rate is selected to protect against any (adversarial) request sequence, it is not too pessimistic.

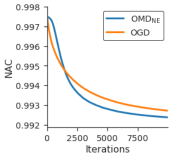
Effect of Diversity



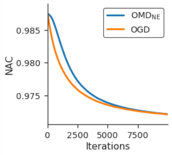
(a)
$k = 25, \alpha = 0.1$



(b)
$k = 125, \alpha = 0.1$



(c)
$k = 25, \alpha = 0.2$



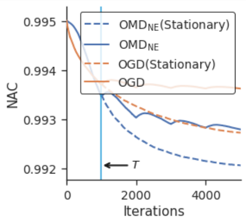
(d)
$k = 125, \alpha = 0.2$

Zipf(α) popularity distribution:

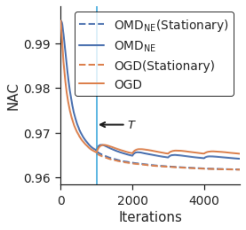
- Low α gives batches with high diversity
- High α gives batches with low diversity

$\mathrm{OMD}_{\mathrm{NE}}$ outperforms OGD in the high-diversity regime and small cache sizes.

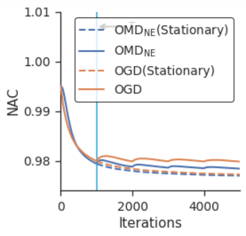
Robustness to Transient Requests



(a) $\alpha = 0.1$



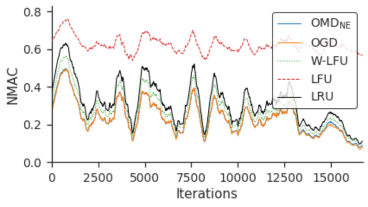
(b) $\alpha = 0.3$



(c) $\alpha = 0.4$

We observe that $\mathrm{OMD}_{\mathrm{NE}}$ is consistently more robust to popularity changes than OGD.

Akamai Trace



Normalized moving average cost of the different caching policies evaluated on the *Akamai Trace*. $\mathrm{OMD}_{\mathrm{NE}}$ and OGD provide consistently the best performance compared to W-LFU, LRU and LFU. OGD performs slightly better than OMD in some parts of the trace.

Thank you for your attention!