AÇAI: Ascent Similarity Caching with Approximate Indexes

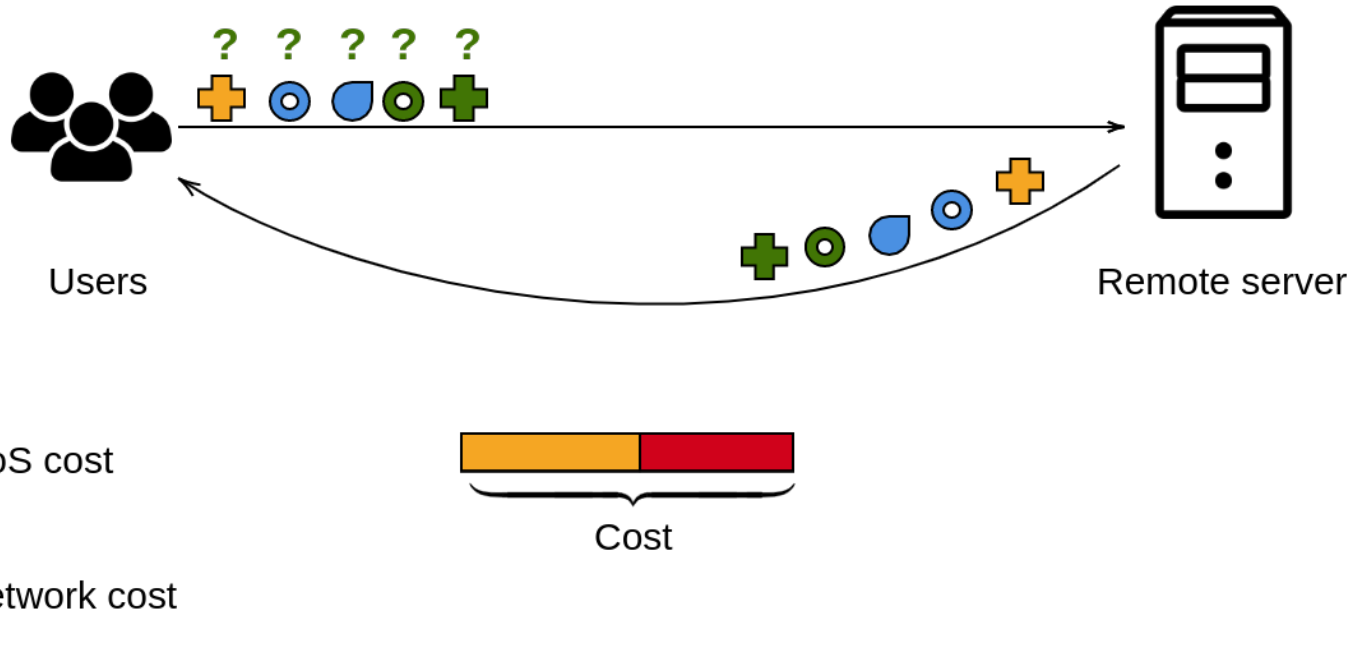
Tareq Si Salem[1] Giovanni Neglia[1] Damiano Carra[2]

[1]Inria, Université Côte d'Azur

[2]University of Verona

Context

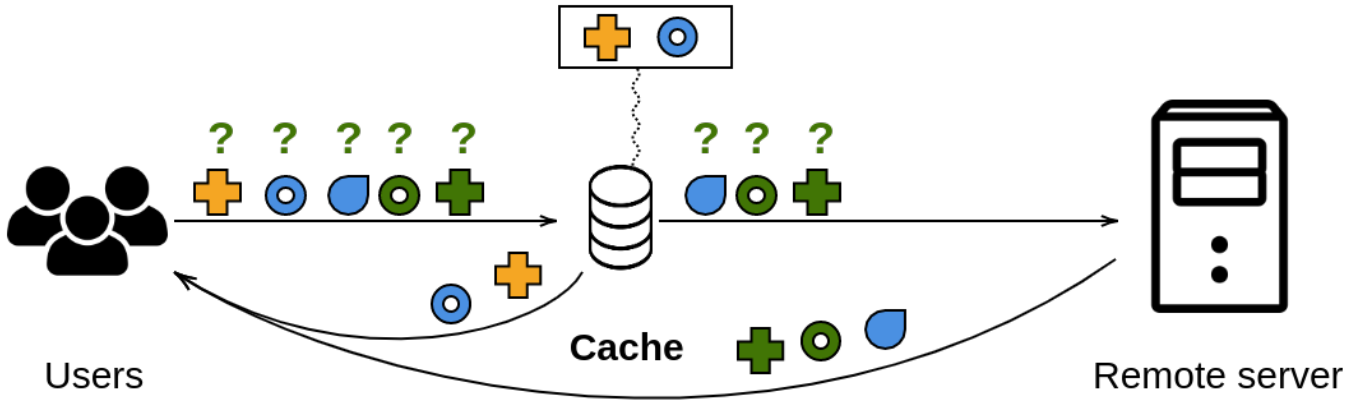
Classical Caching



Costs

- Quality of service cost incurred due to fetching latency
- Monetary cost for using the network infrastructure

Classical Caching



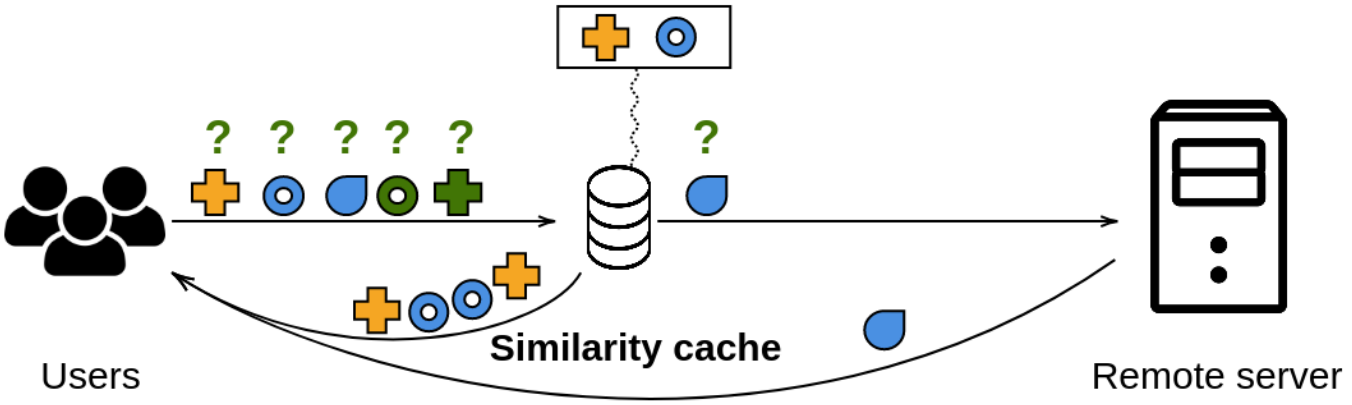

QoS cost

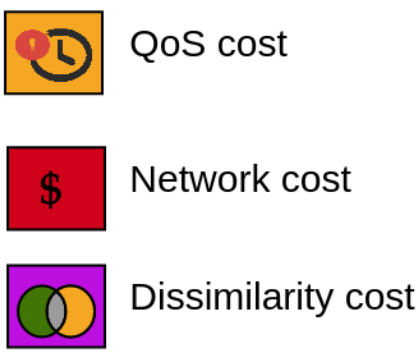
Network cost

A cache with finite capacity is put close to the users.

- Hit: the requested file is stored
- Miss: the request is forwarded to the server

Similarity Caching

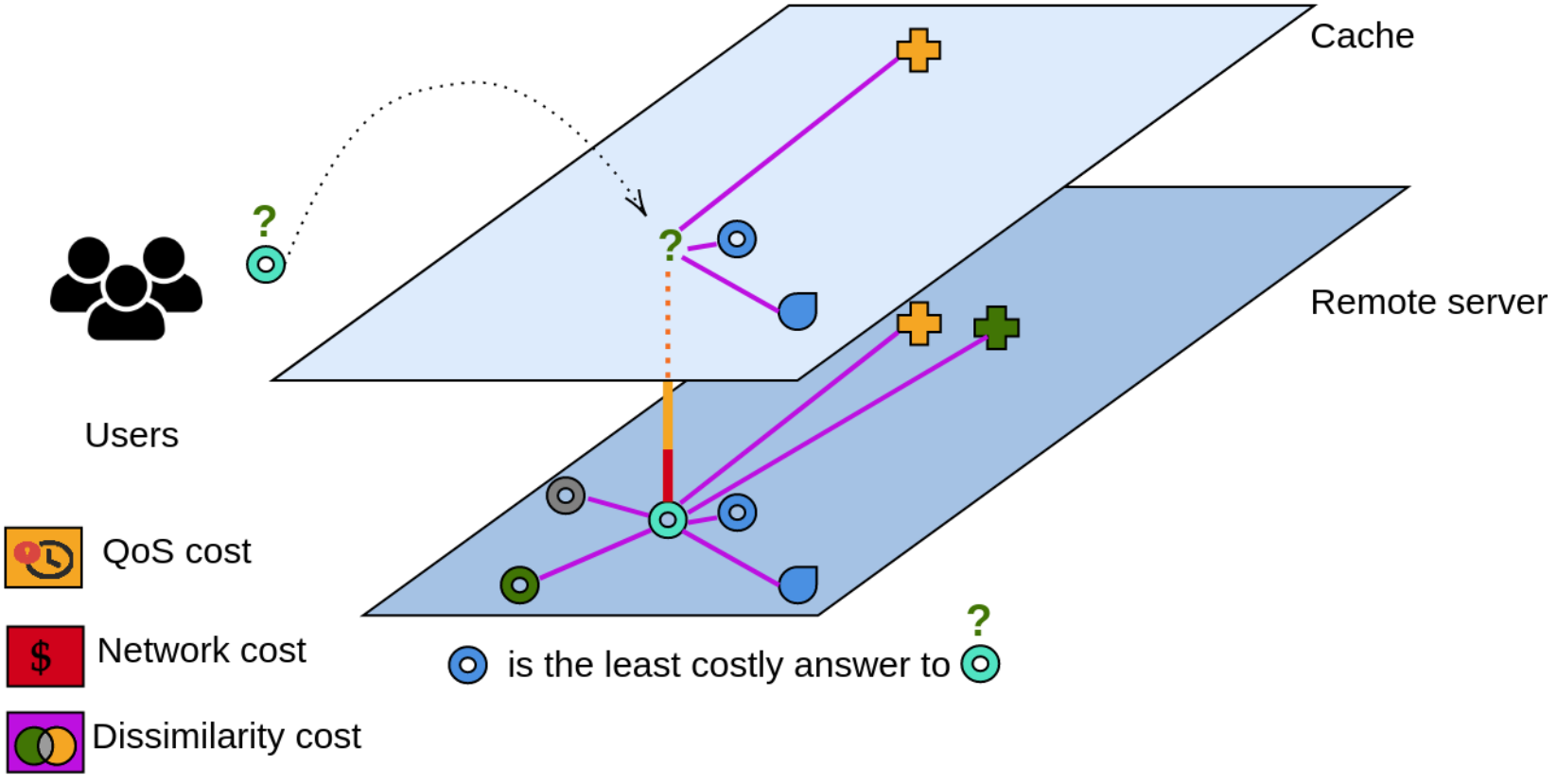




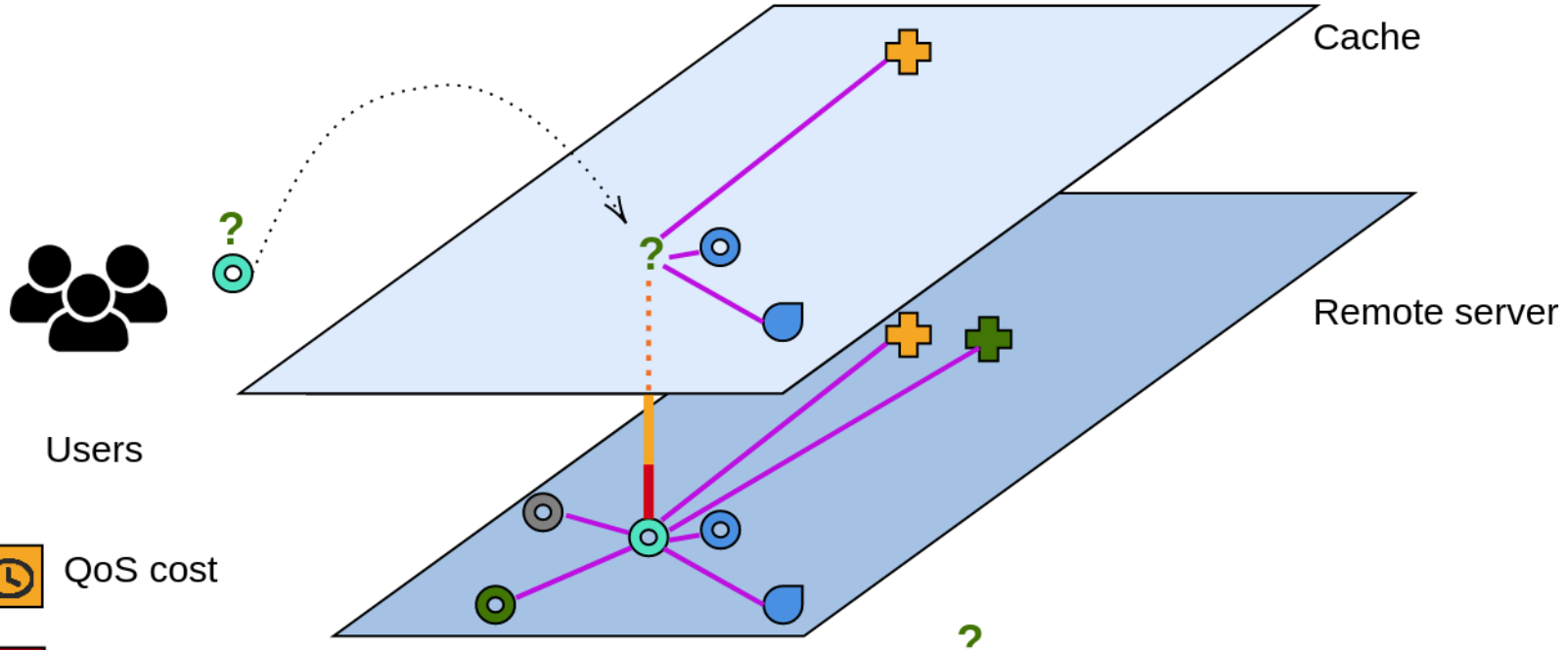
A cache with finite capacity is put close to the users.

- Hit: the cache serves the request object
- Miss: the cache forwards the request
- Approximate hit: serve with a similar object

Similarity Caching



Similarity Caching



Examples:

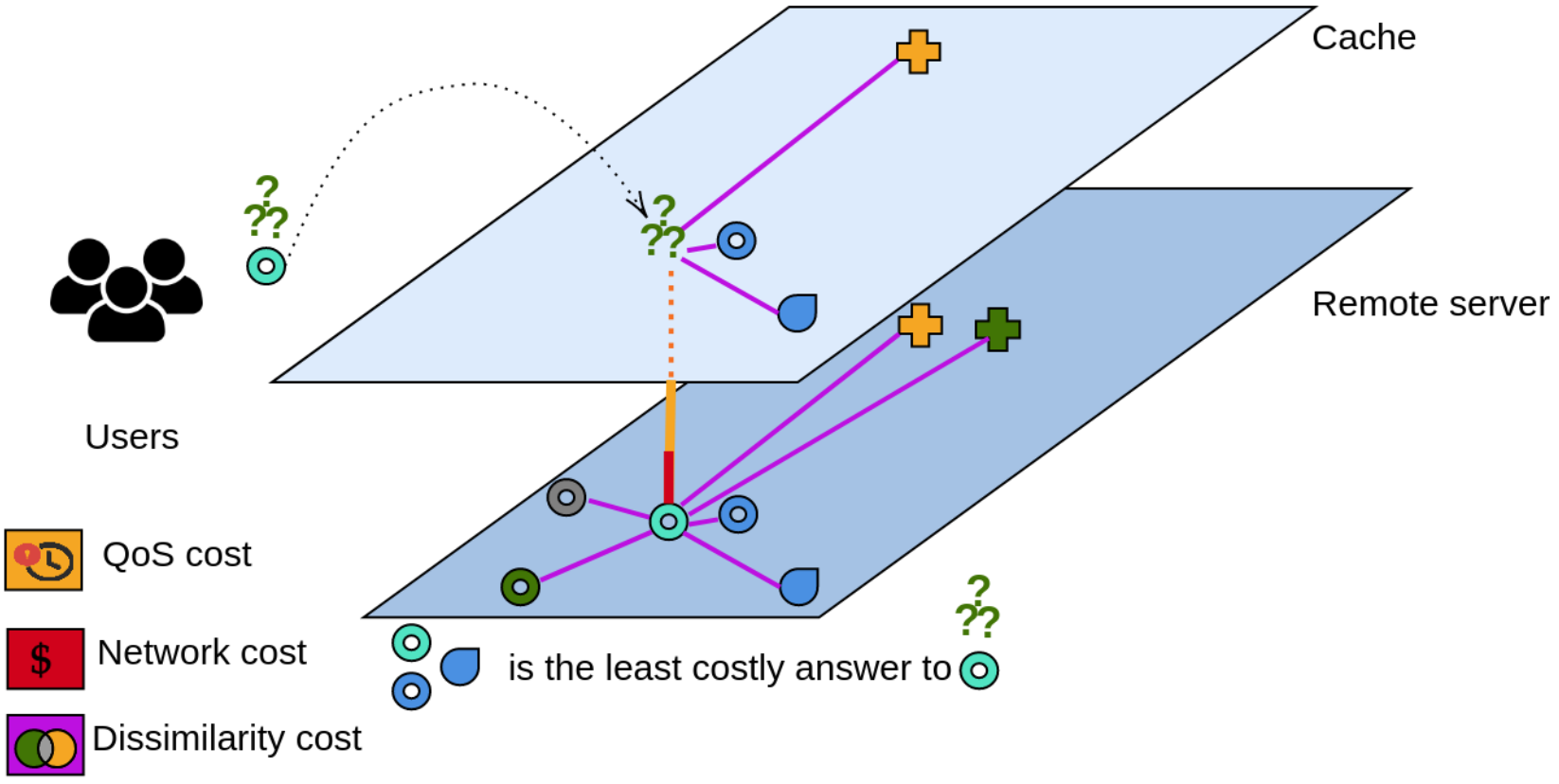
( , dog) ← answer to ( , ?)

image retrieval / classification results reuse

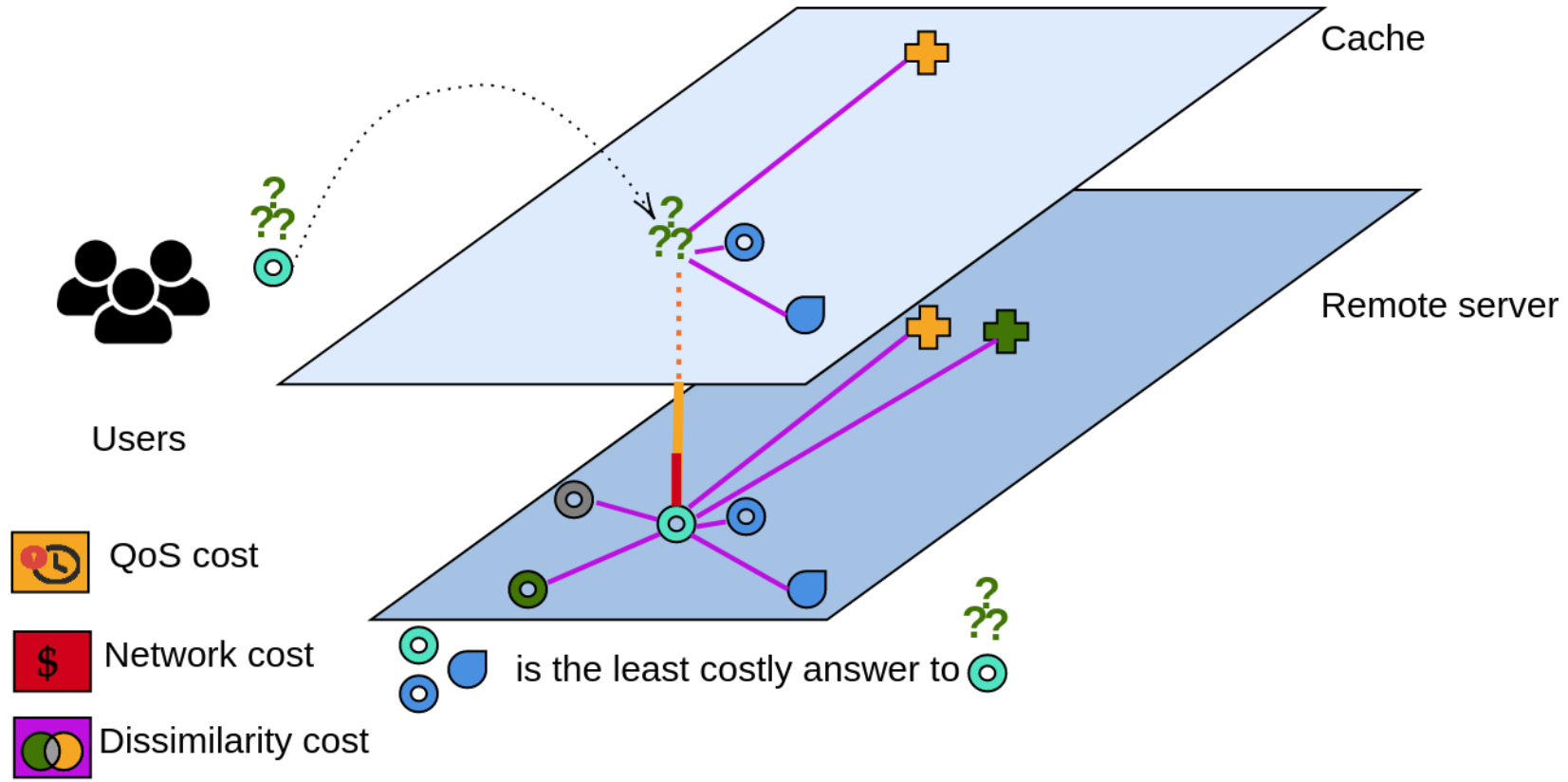
( , sd quality) ← answer to ( , any quality)

cost aware video caching

Similarity Caching with Recommendations (kNN Caching)



Similarity Caching with Recommendations (kNN Caching)



Some Applications

- Content recommendation[1]
- Image retrieval and contextual ads[2]
- Machine Learning serving[3]

---

[1] T. Spyropoulos et al. "Soft Cache Hits and the Impact of Alternative Content Recommendations on Mobile Edge Caching". In: *Proc. of the Eleventh ACM Workshop on Challenged Networks*. CHANTS '16. New York City, New York: Association for Computing Machinery, 2016, pp. 51–56.

[2] F. Falchi et al. "A metric cache for similarity search". In: *Proceedings of the 2008 ACM workshop on Large-Scale distributed systems for information retrieval*. 2008, pp. 43–50; S. Pandey et al. "Nearest-neighbor Caching for Content-match Applications". In: *Proc. of the 18th International Conference on World Wide Web*. WWW '09. Madrid, Spain: ACM, 2009, pp. 441–450.

[3] D. Crankshaw et al. "Clipper: A Low-Latency Online Prediction Serving System". In: *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. Boston, MA, 2017; U. Drolia et al. "Cachier: Edge-caching for recognition applications". In: *Proc. of the IEEE ICDCS*. IEEE. 2017, pp. 276–286; U. Drolia et al. "Precog: Prefetching for image recognition applications at the edge". In: *Proc. of ACM/IEEE Symposium on Edge Computing*. 2017, pp. 1–13; A. Kumar et al. "Accelerating deep learning inference via freezing". In: *11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19)*. 2019; S. Venugopal et al. "Shadow puppets: Cloud-level accurate AI inference at the speed and economy of edge". In: *USENIX HotEdge*. 2018.

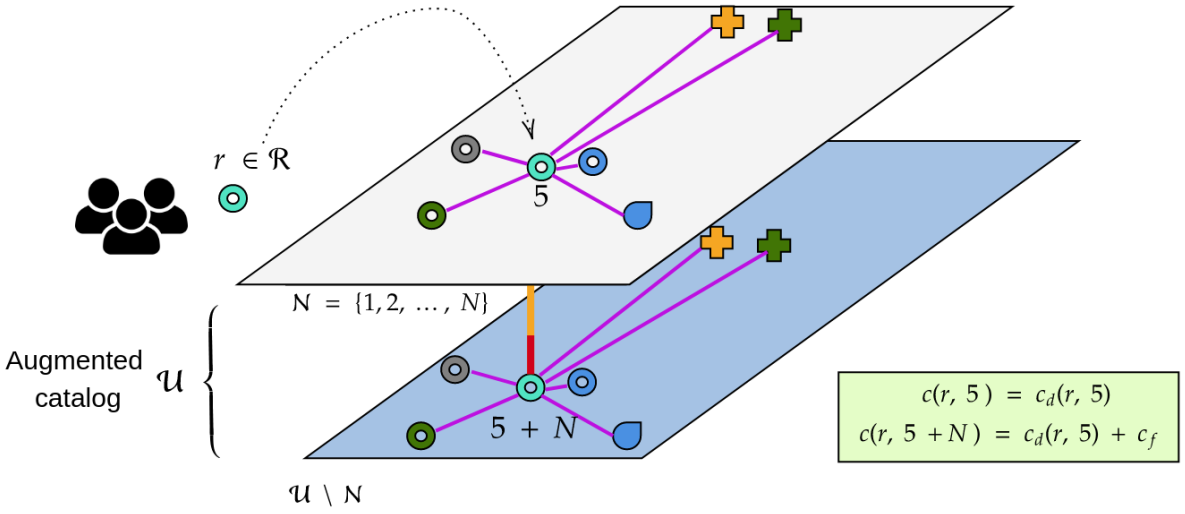
Contributions

Contributions

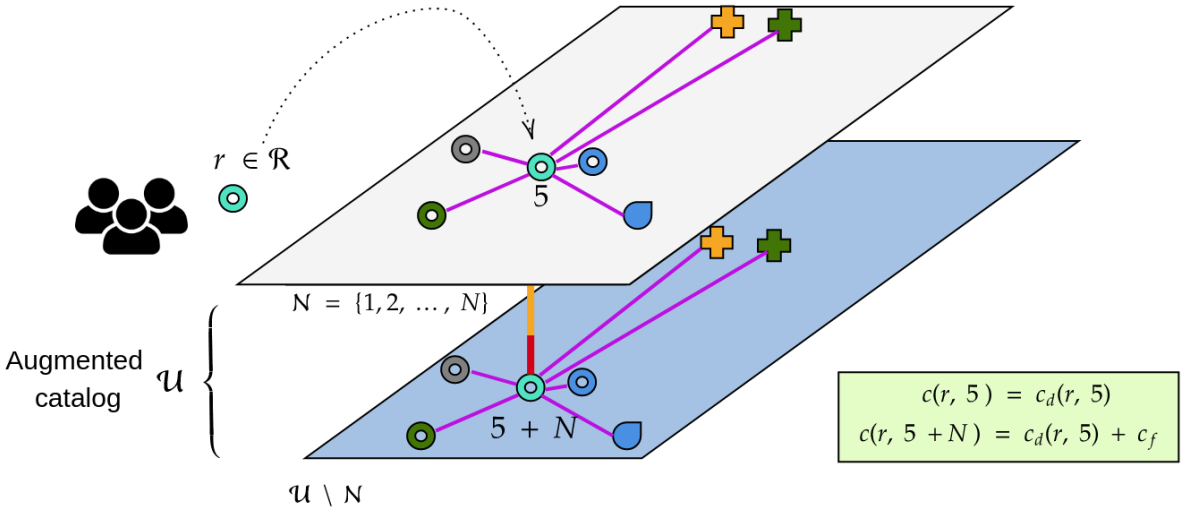
- Formulate the problem of kNN optimal caching
- Propose a new similarity online caching policy $\mathrm{A\text{Ç}AI}$ with strong performance guarantees
- $\mathrm{A\text{Ç}AI}$ consistently improves over state-of-the-art under realistic traces

Modeling

Caching Gain Modeling



Caching Gain Modeling



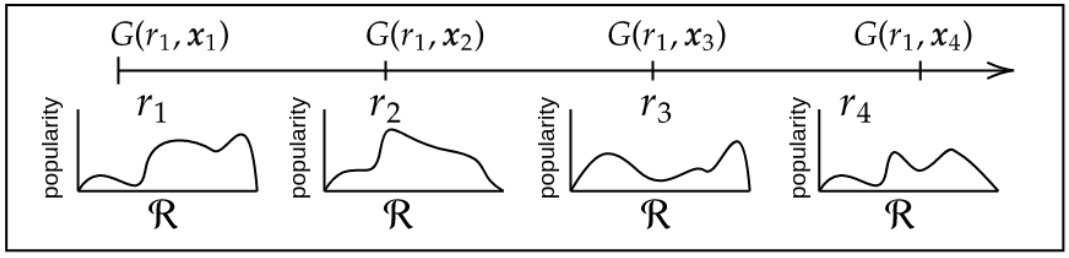
A cache with allocation vector $\boldsymbol{x} \in \{0,1\}^{2N}$ incurs the following cost when $r \in \mathcal{R}$ is received

$$C(r, \boldsymbol{x}) = \sum_{i=1}^{2N} c(r, \pi_i^r) x_{\pi_i^r} \mathbb{1}_{\left\{\sum_{j=1}^{i} x_{\pi_j^r} \leq k\right\}}.$$

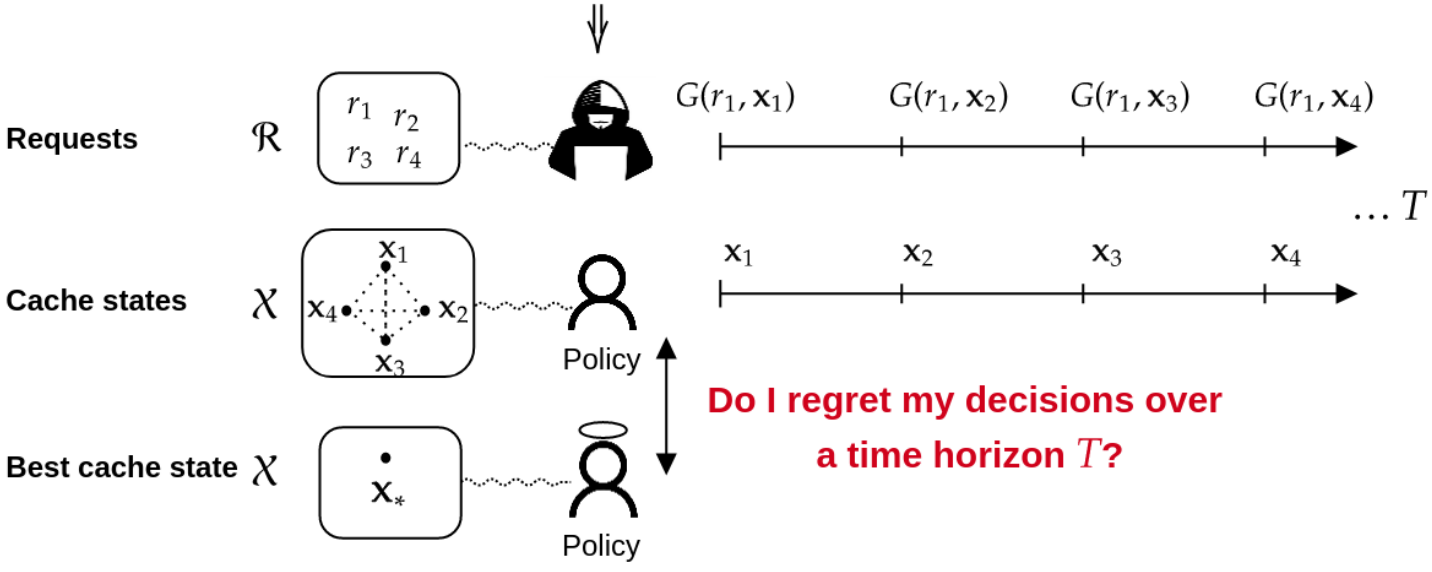
Our objective is to maximize the caching gain (cost savings) as the cache state $\boldsymbol{x}$ changes, given as

$$G(r, \boldsymbol{x}) \triangleq C(r, \text{empty cache}) - C(r, \boldsymbol{x}).$$

Setting and Performance Metric (1/2)



Noisy unpredictable environment can act as an **adversary** in the worst case scenario



Setting and Performance Metric (2/2)

Regret of a policy $\mathcal{P}$ with the (potentially random) cache states $\{\boldsymbol{x}_t\}_{t=1}^{T}$ is given by

$$\psi\text{-Regret}(\mathcal{P}) = \sup_{\{r_1, r_2, \ldots, r_T\} \in \mathcal{R}^T} \left\{ \psi \sum_{t=1}^{T} G(r_t, \boldsymbol{x}_*) - \mathbb{E}\left[\sum_{t=1}^{T} G(r_t, \boldsymbol{x}_t) \right] \right\}$$

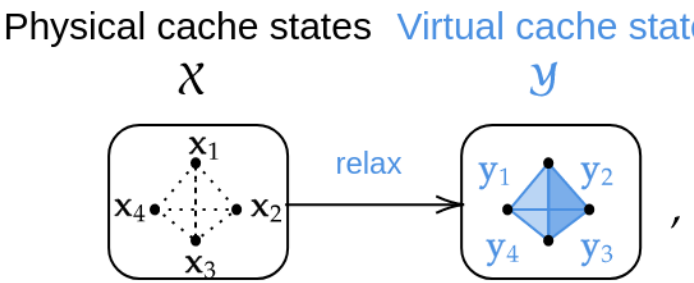
- The constant $\psi = 1 - 1/e$ is the best approximation ratio achievable in P to the NP-Hard static optimum
- When ψ-Regret$(\mathcal{P})$ is sublinear in T, the policy experiences no regret on average as $T \to \infty$

Online Convex Optimization (OCO)

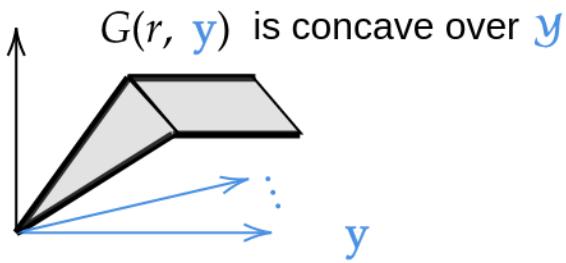
We prove that the caching gain can be expressed equivalently as

$$G(r, \boldsymbol{x}) = \sum_{i=1}^{K^r - 1} \alpha_i^r \min \left\{ k, \sum_{j=1}^{i} x_{\pi_j^r} \right\} + \sigma_i^r,$$

where α_i^r, σ_i^r, and K^r are constants.



,



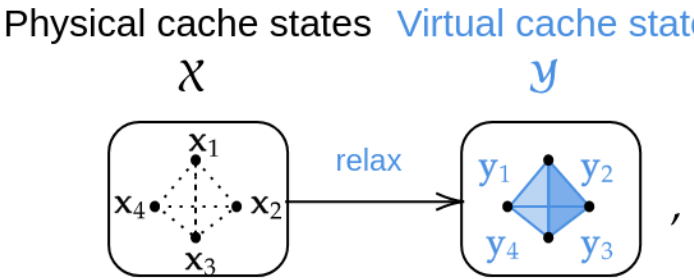
[4] E. Hazan. "Introduction to Online Convex Optimization". In: *Found. Trends Optim.* 2.3–4 (Aug. 2016), pp. 157–325.

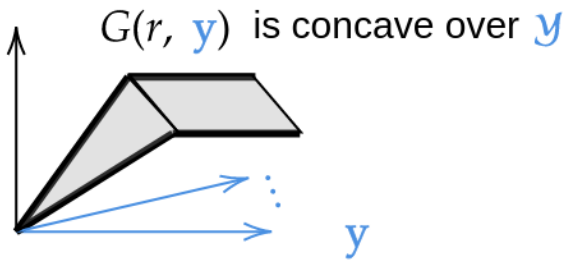
Online Convex Optimization (OCO)

We prove that the caching gain can be expressed equivalently as

$$G(r, \boldsymbol{x}) = \sum_{i=1}^{K^r - 1} \alpha_i^r \min \left\{ k, \sum_{j=1}^{i} x_{\pi_j^r} \right\} + \sigma_i^r,$$

where α_i^r, σ_i^r, and K^r are constants.





The fractionally relaxed problem can be cast in the framework of OCO.[5]

[5] E. Hazan. "Introduction to Online Convex Optimization". In: *Found. Trends Optim.* 2.3–4 (Aug. 2016), pp. 157–325.

Dissection of AÇAI

AÇAI: Core Algorithm - Online Mirror Ascent

Online Mirror Ascent (OMA) is a scheme used to generate no-regret policies.

- Requires as parameters the learning rate η and a mirror map Φ
- We select the negative entropy mirror map $\Phi(\boldsymbol{y}) = \sum_i y_i \log(y_i)$

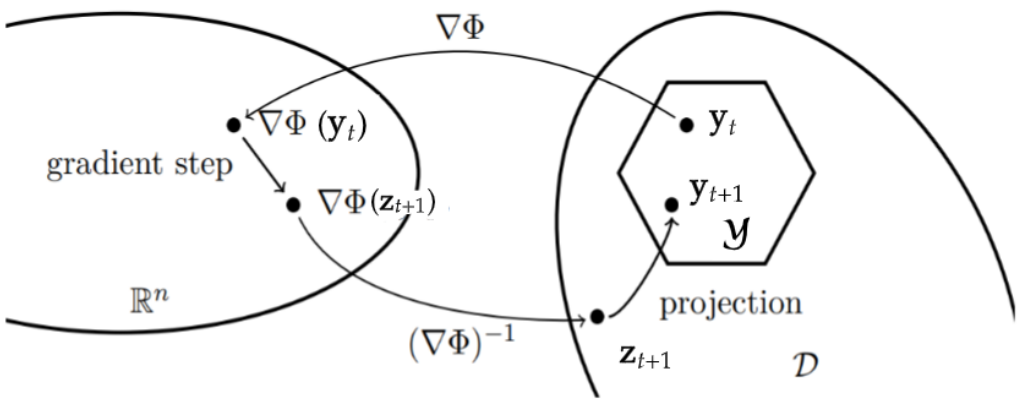
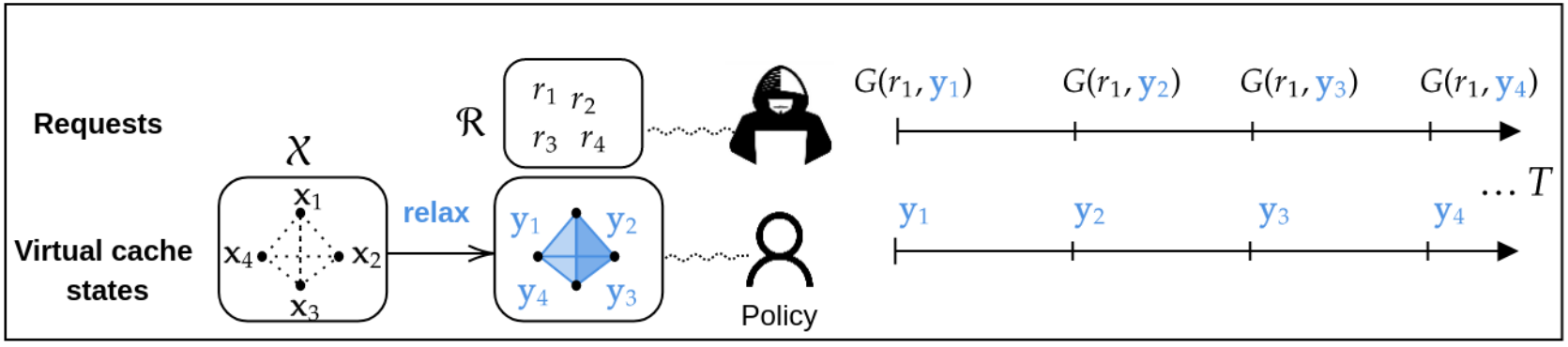


Figure 1: OMA update rule.[6]

[6] S. Bubeck. "Convex Optimization: Algorithms and Complexity". In: *Found. Trends Mach. Learn.* 8.3–4 (Nov. 2015), pp. 231–357.

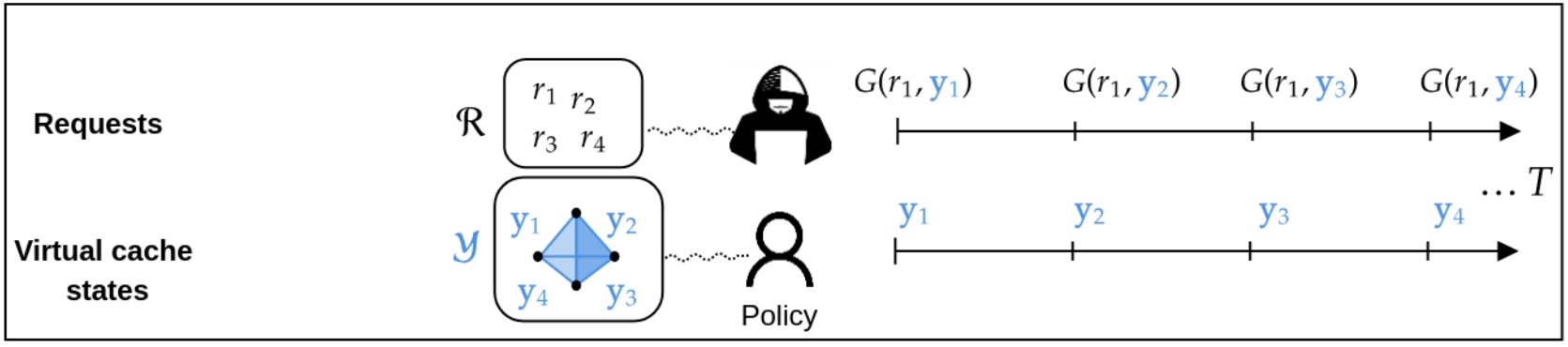
AÇAI: Virtual and Physical Cache States



AÇAI: Virtual and Physical Cache States

The regret guarentee over the fractional (relaxed) setting is

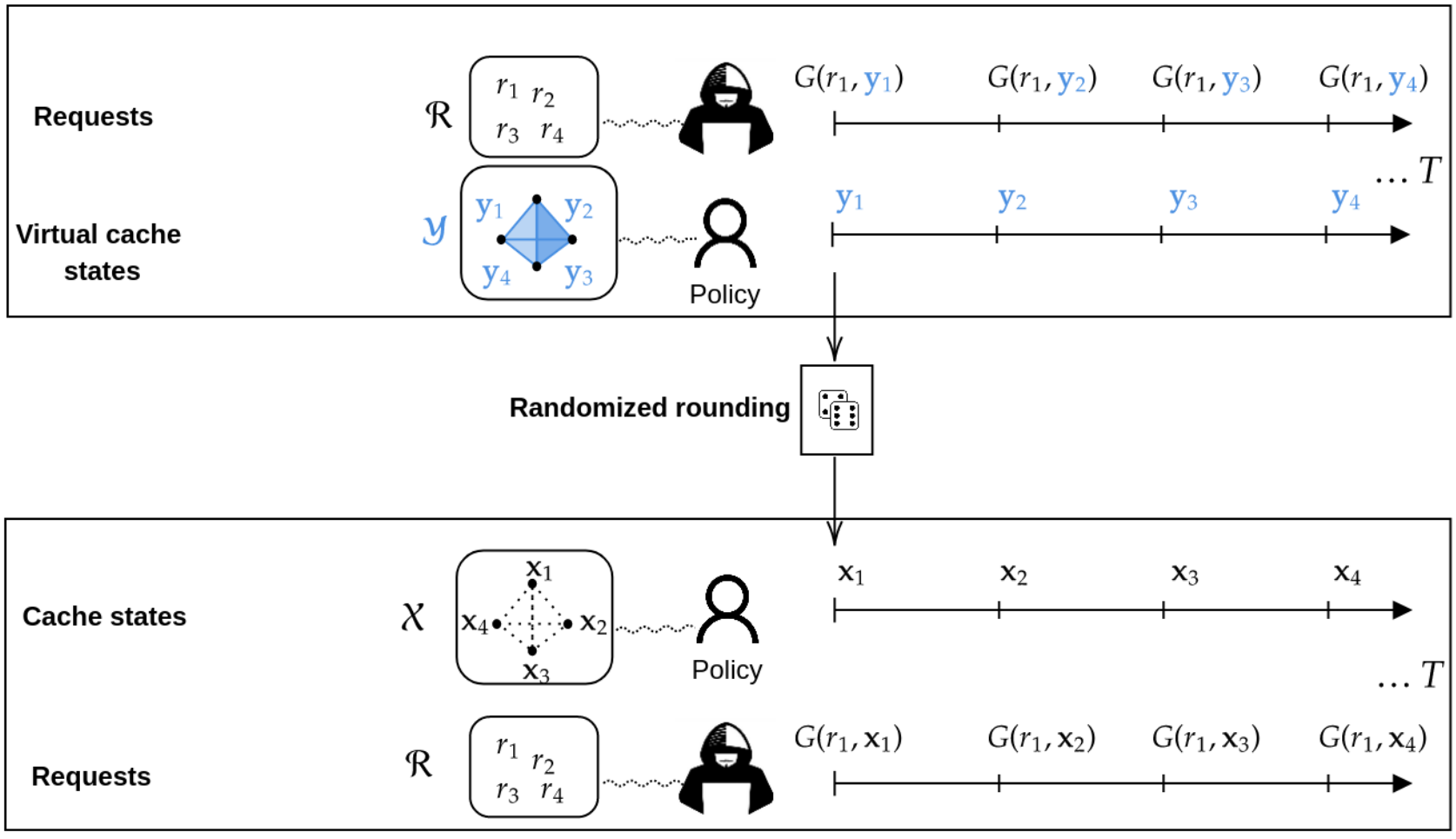
$$\text{Regret} = O\left(\sqrt{T}\right)$$



AÇAI: Virtual and Physical Cache States

The regret guarentee over the fractional (relaxed) setting is

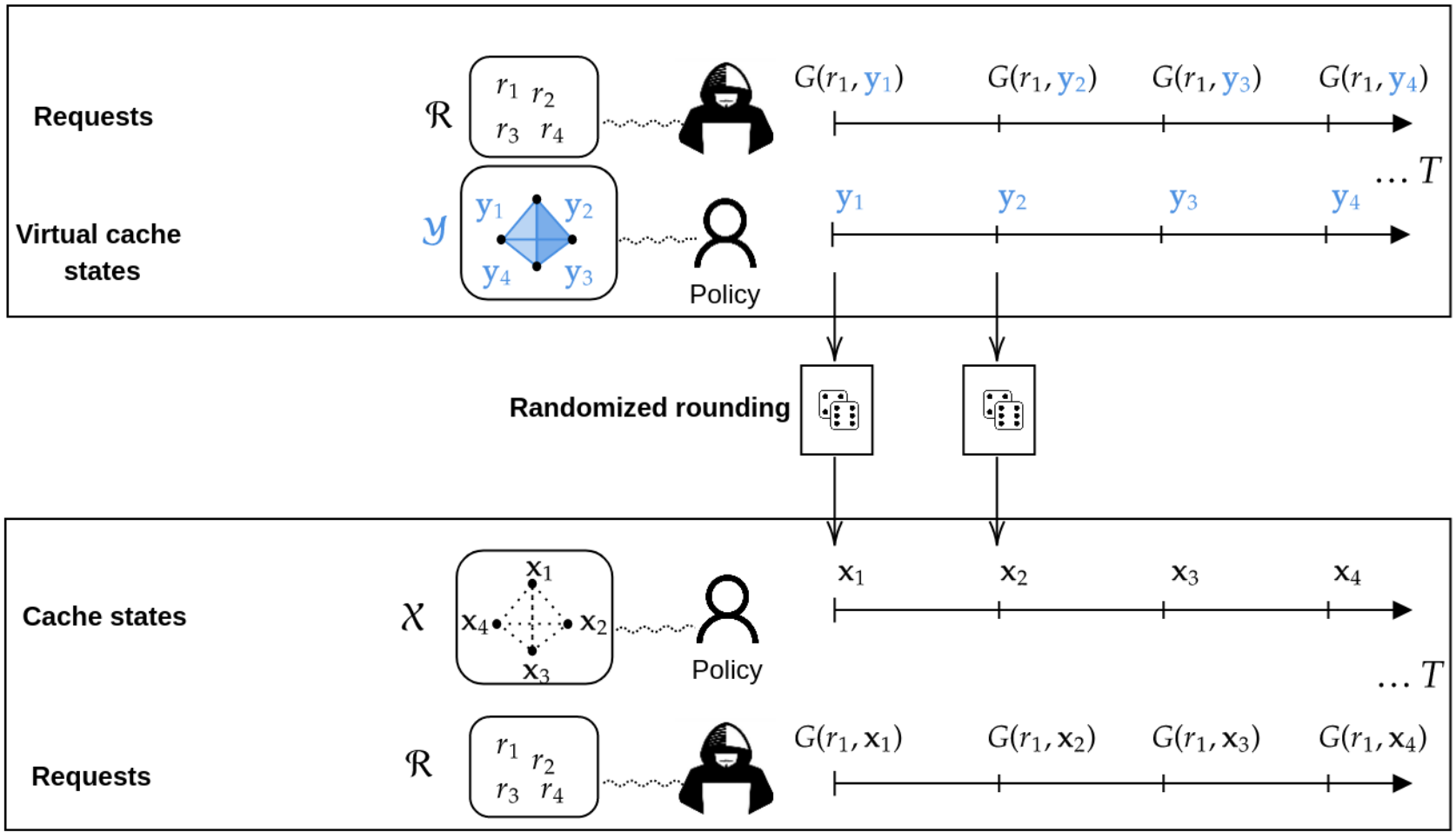
$$\text{Regret} = O\left(\sqrt{T}\right)$$



AÇAI: Virtual and Physical Cache States

The regret guarentee over the fractional (relaxed) setting is

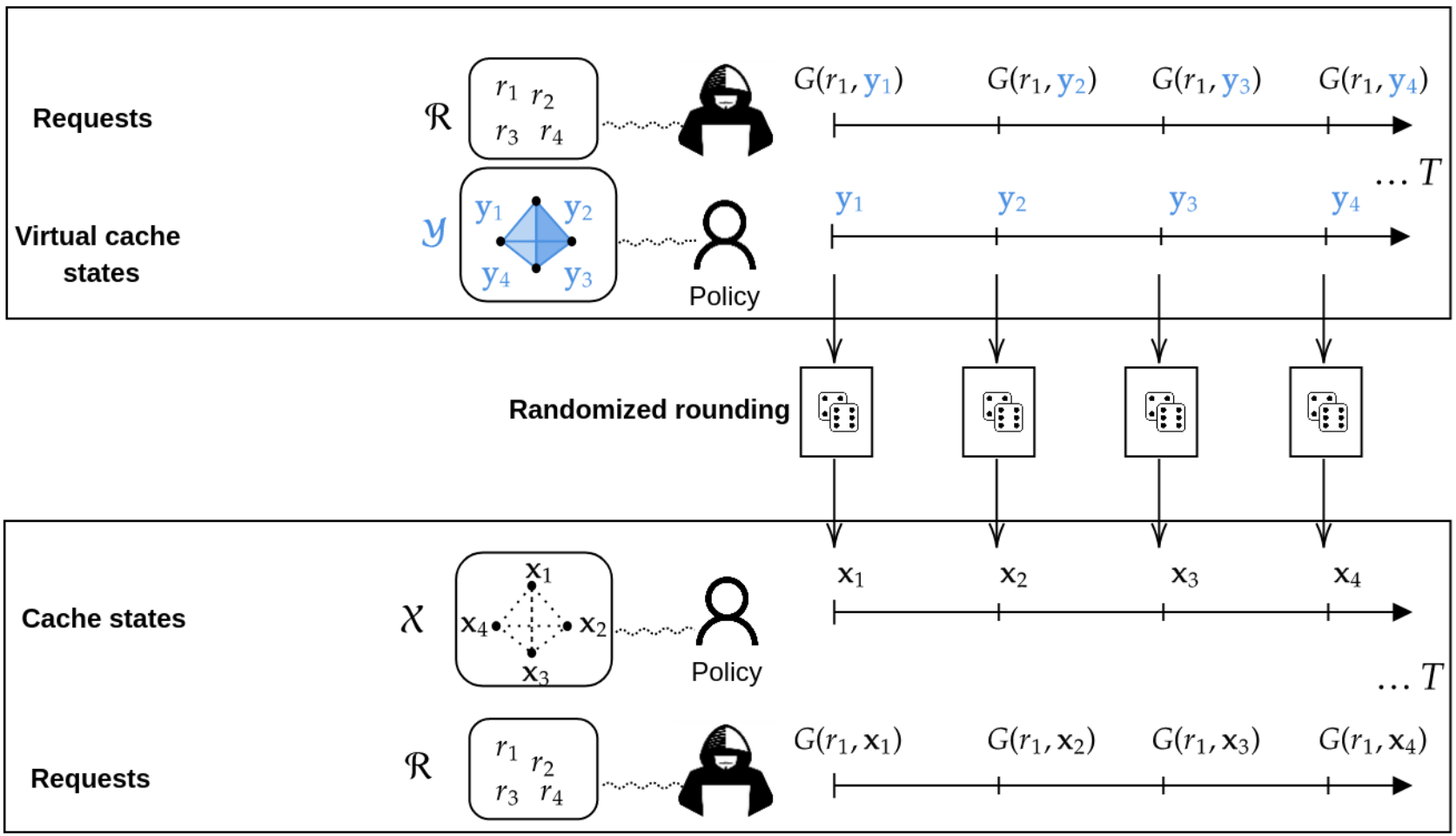
$$\text{Regret} = O\left(\sqrt{T}\right)$$



AÇAI: Virtual and Physical Cache States

The regret guarentee over the fractional (relaxed) setting is

$$\text{Regret} = O\left(\sqrt{T}\right)$$



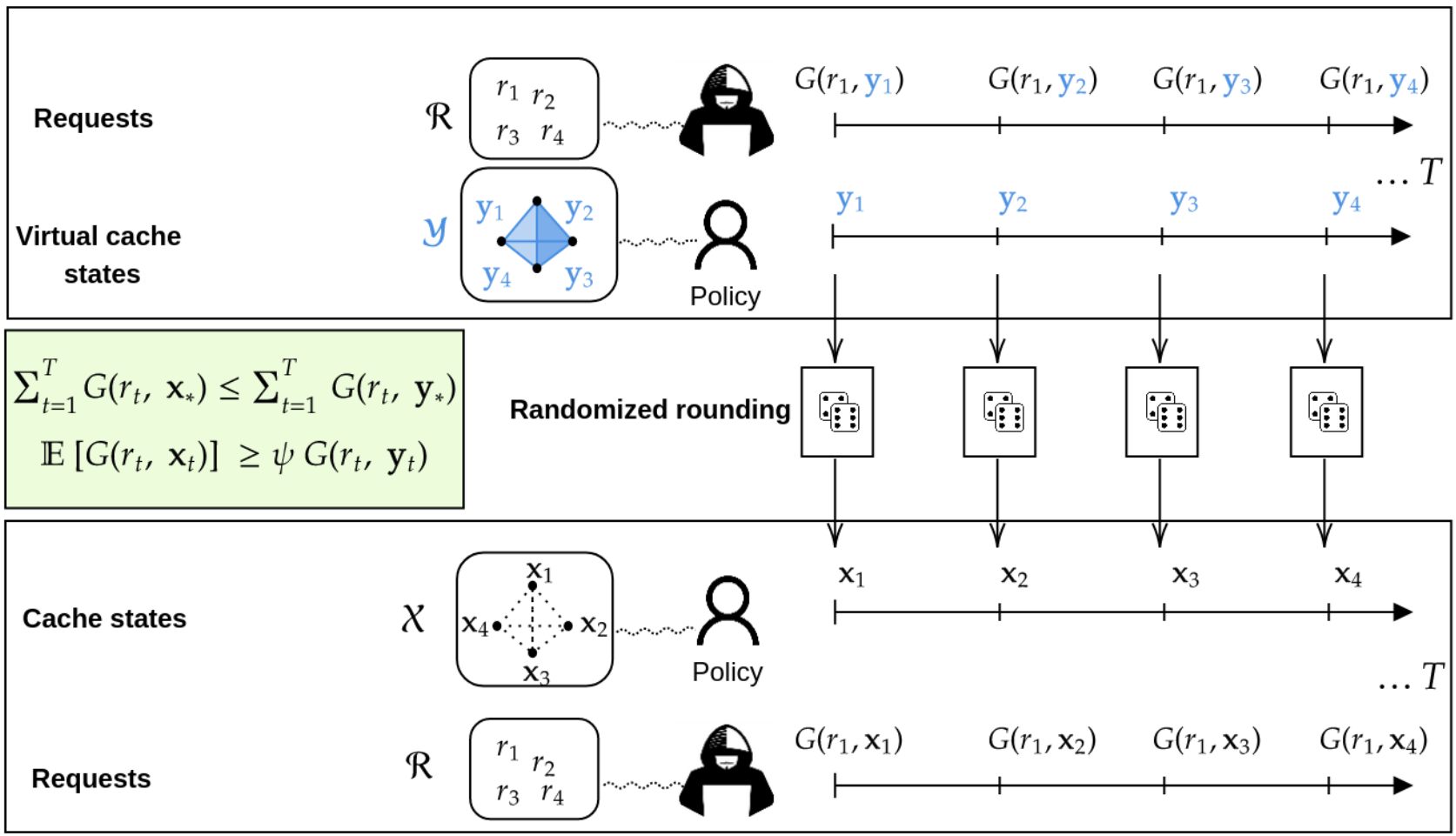
We transfer the guarentee from the fractional to the integral setting and obtain

$$\psi\text{-Regret} = O\left(\sqrt{T}\right)$$

AÇAI: Virtual and Physical Cache States

The regret guarentee over the fractional (relaxed) setting is

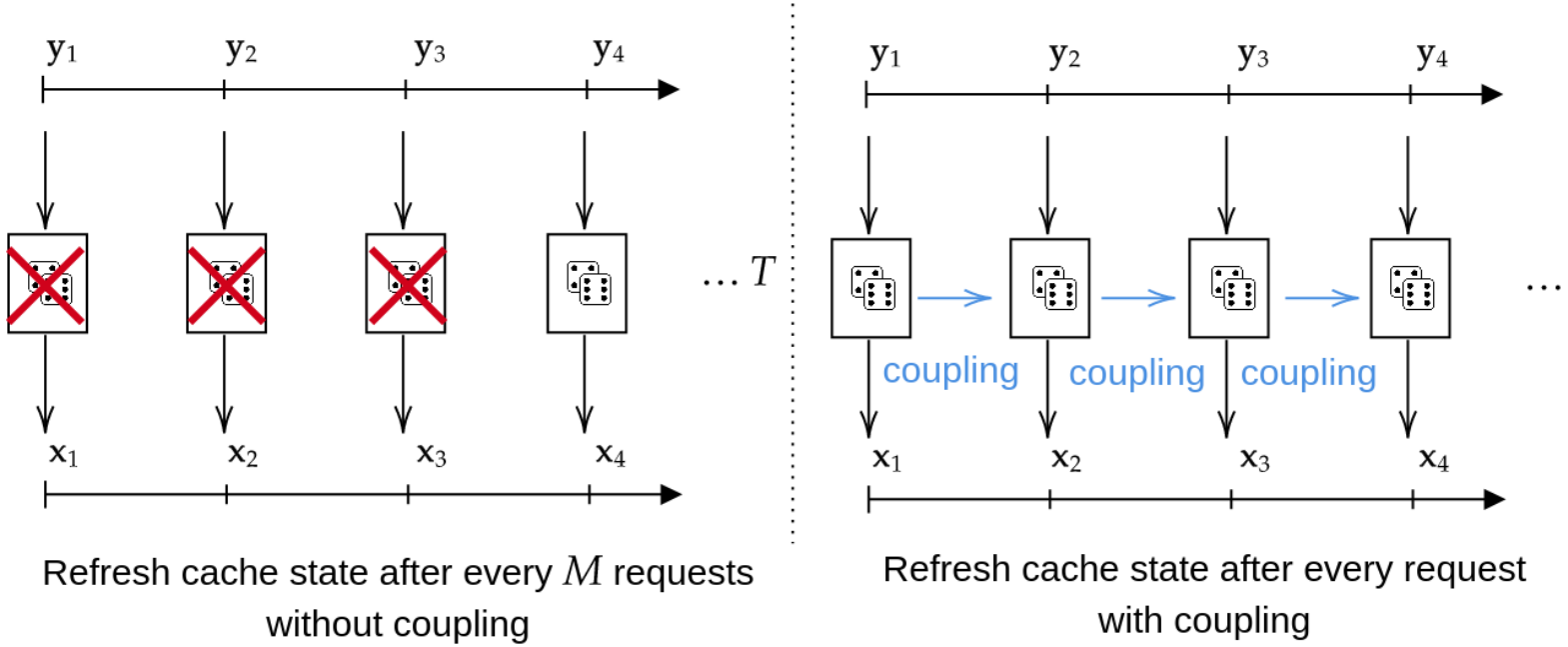
$$\text{Regret} = O\left(\sqrt{T}\right)$$



We transfer the guarentee from the fractional to the integral setting and obtain

$$\psi\text{-Regret} = O\left(\sqrt{T}\right)$$

AÇAI: Reducing Movement Costs



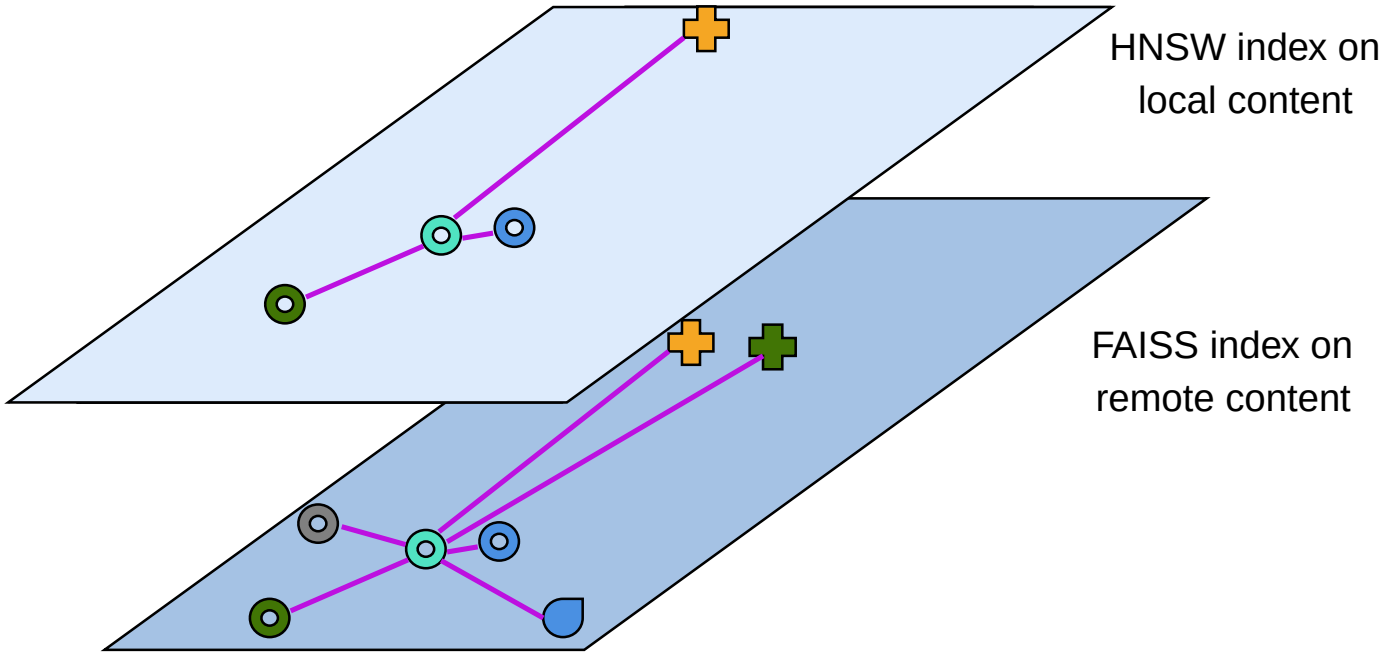
Refresh cache state after every M requests without coupling

Refresh cache state after every request with coupling

When the cache movements are costly or cannot be neglected, we propose two randomized rounding schemes:

- Refresh cache state every request with coupling
- Refresh cache state every M requests

AÇAI: Fast Approximate Indexes



AÇAI employs two approximate indexes:

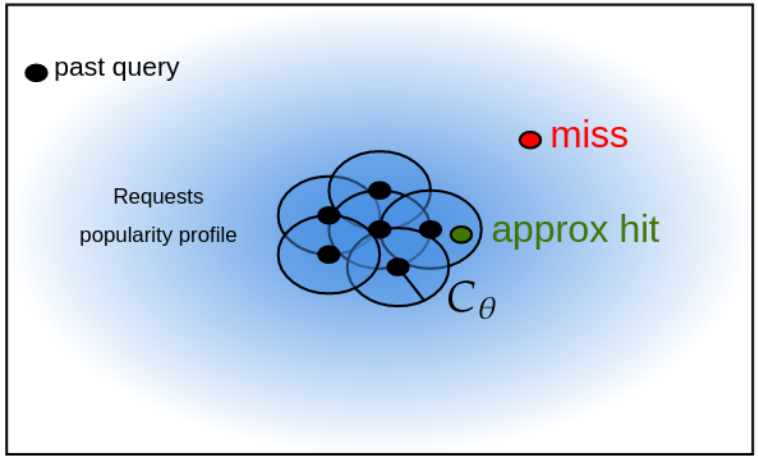
- one for content stored in the cache (HNSW)[7]
- one for the whole catalog $\mathcal{N}$ stored in the remote server (FAISS)[8]

[7] Y. A. Malkov et al. "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs". In: *IEEE trans. on pattern analysis and machine intelligence* (2018).

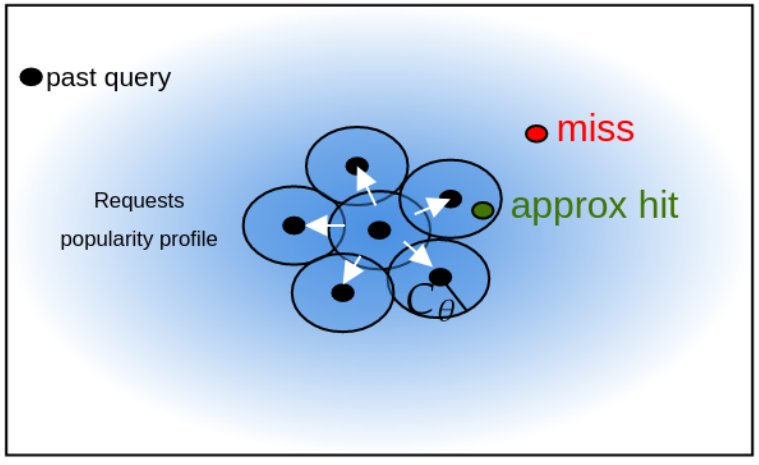
[8] J. Johnson et al. "Billion-scale similarity search with GPUs". In: *IEEE Transactions on Big Data* (2019).

SOTA Policies

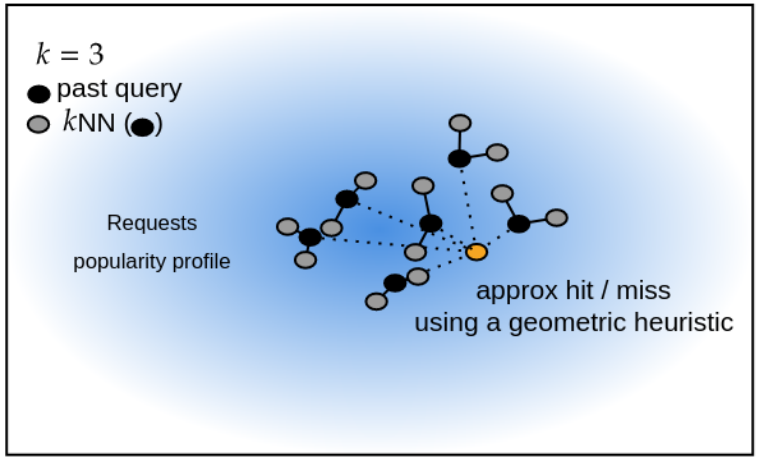
SOTA Policies for kNN Caching



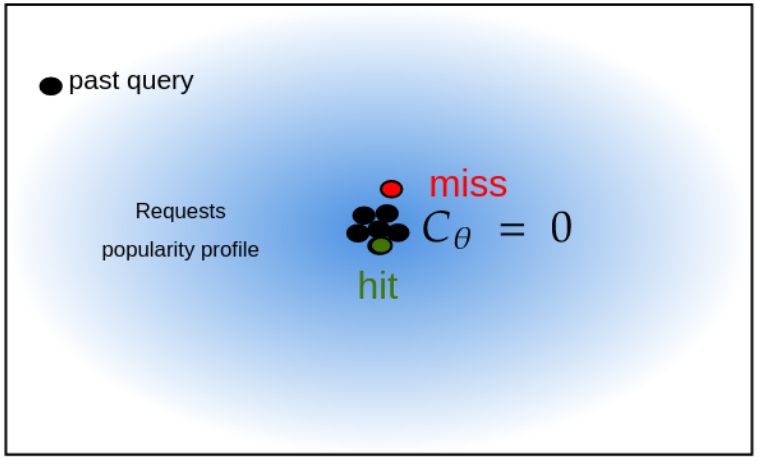
SIM-LRU



CLS-LRU



QCache



LRU

Experimental Validation

Experimental Validation

Datasets.

- SIFT1M trace: synthetic request process based on SIFT1M dataset[9]
- Amazon trace: request process is the timestamped reviews left by users[10,11]

Performance Metric. The normalized time-averaged gain over T requests:

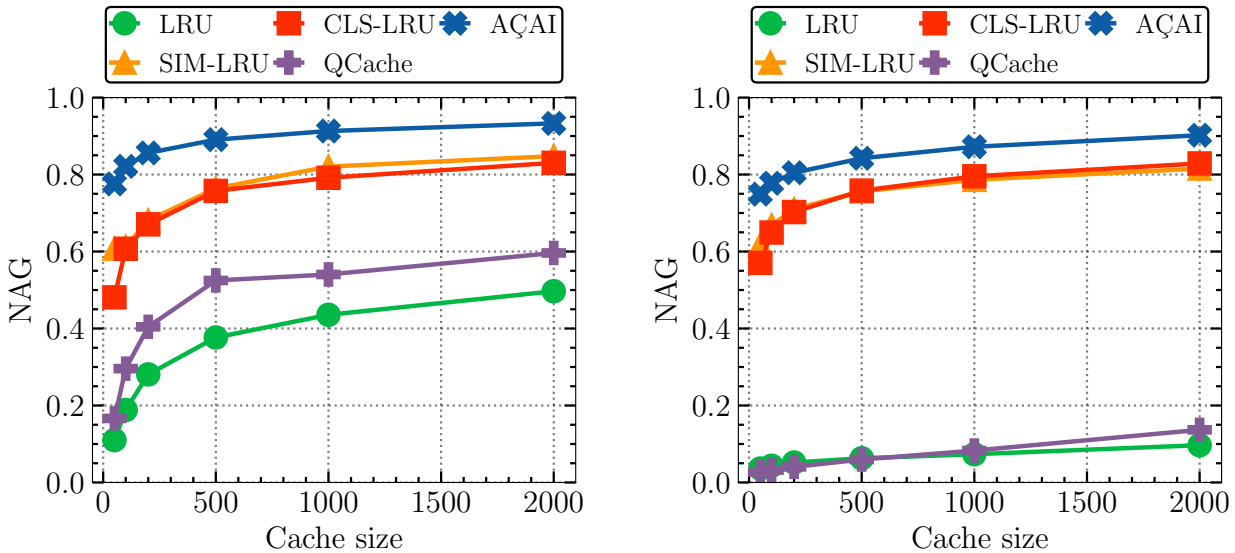
$$\mathrm{NAG}(\mathcal{P}) = \frac{1}{k c_f T} \sum_{t=1}^{T} G_{\mathcal{P}}(r_t, \boldsymbol{x}_t).$$

[9] H. Jegou et al. "Product quantization for nearest neighbor search". In: *IEEE trans. on pattern analysis and machine intelligence* 33.1 (2010), pp. 117–128.

[10] J. McAuley et al. "Image-based recommendations on styles and substitutes". In: *Proc. of the ACM SIGIR*. 2015, pp. 43–52.

[11] A. Sabnis et al. "GRADES: Gradient Descent for Similarity Caching". In: *IEEE Conference on Computer Communications (INFOCOM)*. 2021.

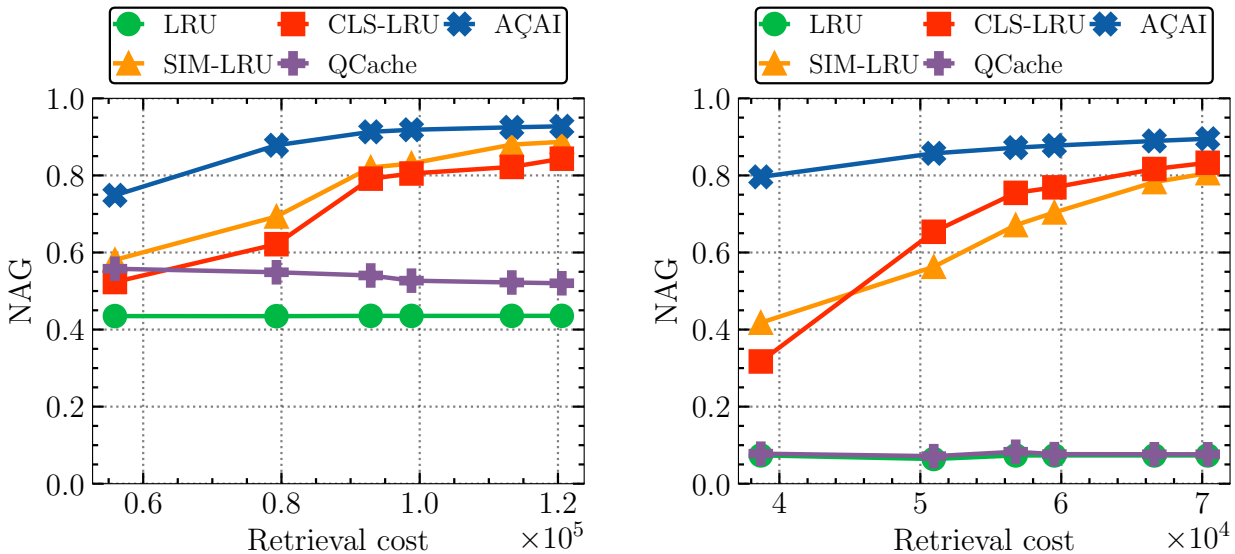
Experimental Validation - Different Capacities



(a) SIFT1M trace **(b)** Amazon trace

Figure 2: Caching gain for the different policies, for different cache sizes $h \in \{50, 100, 200, 500, 1000, 2000\}$ and $k = 10$.

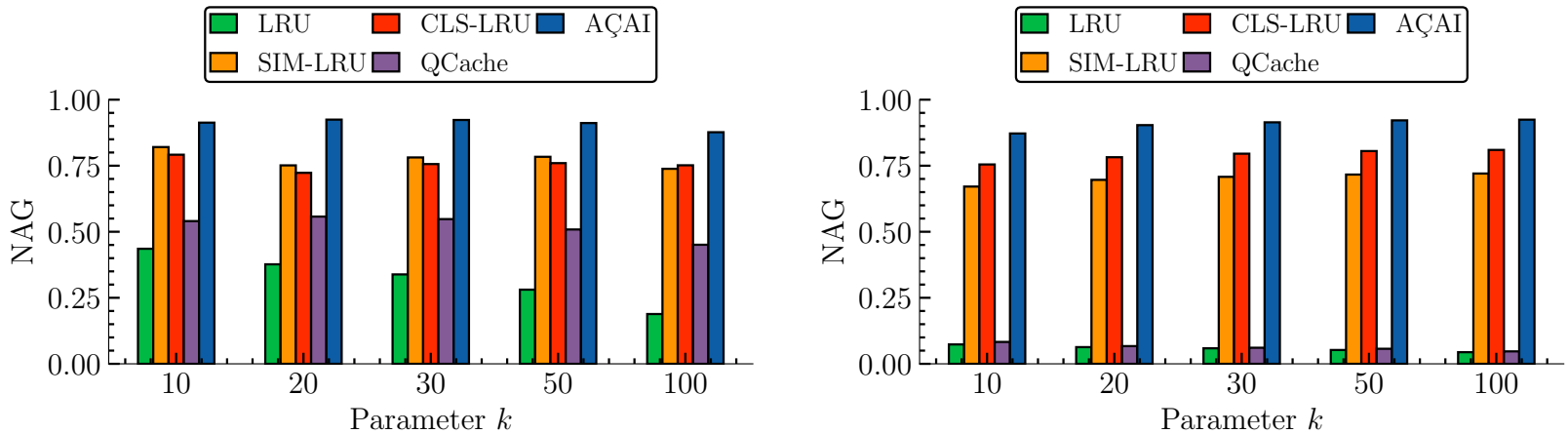
Experimental Validation - Different Retrieval Costs



(a) SIFT1M trace **(b)** Amazon trace

Figure 3: Caching gain for the different policies and different retrieval cost. The cache size is $h = 1000$ and $k = 10$.

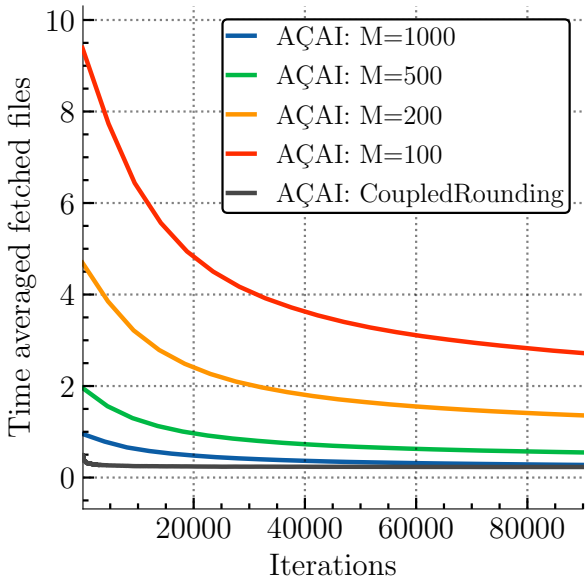
Experimental Validation - Different Recommendations k



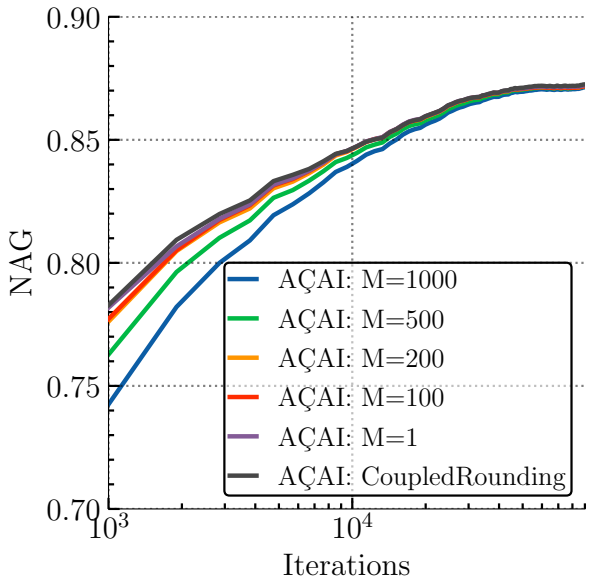
(a) SIFT1M trace **(b)** Amazon trace

Figure 4: Caching gain for the different policies. The cache size is $h = 1000$, and $k \in \{10, 20, 30, 50, 100\}$.

Experimental Validation - Movement Costs



(a) Time averaged fetched files



(b) Caching gain

Conclusion and Future Work

Conclusion and Future Work

We designed $\mathrm{A\text{Ç}AI}$, a content cache management policy that determines dynamically the best content to store on the edge server to reply to similarity search queries.

As future work, we plan to evaluate $\mathrm{A\text{Ç}AI}$ in the context of machine learning classification tasks[12]

[12] U. Khandelwal et al. "Generalization through memorization: Nearest neighbor language models". In: *Proc. of the ICLR*. 2020.

Thank you for your attention. Questions?

Extra Slides

Experimental Validation - Choice of Mirror Map

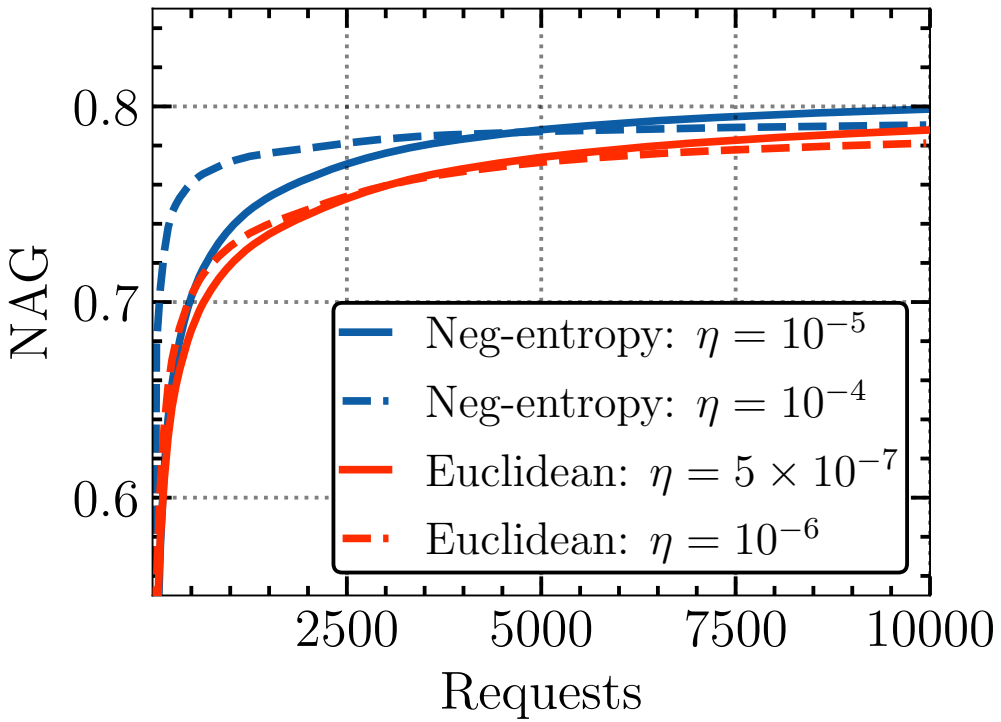


Figure 6: Caching gain for AÇAI configured with negative entropy and Euclidean maps (SIFT1M trace). The cache size is $h = 100$ and $k = 10$.