Towards Optimal Robustness in Learning-Augmented Paging

Peng Chen[1] **(Presenter)**, Hailiang Zhao[1*],
Xueyan Tang[2], Yixuan Wang[3], Shuiguang Deng[1*]

[1] Zhejiang University [2] Nanyang Technological University
[3] Nanjing University of Aeronautics and Astronautics

Motivation and the Questions We Ask

- ML for systems is everywhere: ML-based paging, scheduling, resource allocation.
- In production, the hard part is often not high average accuracy — it is coping with the inevitable failures of predictions after deployment.
- *Algorithms with predictions* (ALPS) treat predictions as helpful but fallible signals, while keeping worst-case guarantees.
- We study **learning-augmented paging**, a prototypical ALPS problem.

Two questions

1. How can we **approach optimal robustness**, and at what cost?
2. What is the **essence** of establishing bounded robustness around a classical baseline's competitive ratio?

Online Paging and Classical Optimality

- **Setup:** cache of size k; serve a request stream online; a **hit** is free, a **miss** loads the page (evicting one if full). Minimize #misses.
- **Offline optimum (OPT):** Belady's rule — evict the page used furthest in the future.
- **Online** quality is the **competitive ratio** vs. OPT.

Best-possible competitive ratios

- Deterministic: k (LRU is k-competitive).
- Randomized: Marker is $(2H_k - 1)$-competitive; the optimum is H_k.
- $H_k = \sum_{i=1}^{k} \frac{1}{i} \approx \ln k$, achieved by Partition, Equitable, K_Equitable, OnlineMin via the *work function*.

Learning-Augmented Paging

- Predict future reuse — each page's **next-arrival time** — to mimic BELADY when predictions are good.

Three metrics:

Consistency γ competitive ratio under perfect predictions (ideal: $\gamma = 1$).

Robustness δ competitive ratio under adversarial predictions (smaller is better).

Smoothness $f(\eta)$ how the ratio degrades with prediction error η (ℓ_1 error).

Goal: *near-optimal when accurate, safe worst-case guarantee regardless.*

The Landscape, and a Persistent Gap

Algorithm	Consistency	Robustness
BlindOracle	1	unbounded
PredictiveMarker	$> 1^{\dagger}$	$4H_k$
LMarker	$> 1^{\dagger}$	$2H_k + 4$
BlindOracle&LRU	2	$2k$
F&R	1	$\mathcal{O}(\log k)^{\ddagger}$
RPB-OM **(ours)**	**1**	$\mathbf{H_k} + \mathcal{O}(\mathbf{1})$

† not 1-consistent (consistency > 1); many *other* Marker-based methods stay $\geq$ 2-competitive even with perfect predictions.

‡ non-optimal leading constant; needs heavy recomputation.

Trust&Doubt reaches consistency close to 1 with more careful Marker-style ideas.

- Marker's marking inherently caps robustness at $2H_k + \mathcal{O}(1)$.
- Prior randomized robustness sits at $2H_k + \mathcal{O}(1)$ — a gap to the optimal H_k.

Our Contributions

1. **Online optimality, revisited.** Insights into online-optimal paging and a new property of OnlineMin's H_k-competitiveness that guides learning-augmented design.
2. **Relative Prediction Budget (RPB).** A unifying primitive capturing *how* prior algorithms obtain robustness — revealing that they overuse or underutilize predictions.
3. **A new framework RPB-OnOPT.** Its instantiation RPB-OM achieves

$$\Big(\underbrace{1}_{\text{consistency}}, \underbrace{H_k + \mathcal{O}(1)}_{\text{robustness}} \Big), \quad \mathcal{O}(\log k) \text{ per request,}$$

optimal up to an additive constant. Experiments confirm strong practice.

Work Functions and Valid Configurations

- The **work function** ω tracks the *current* optimal cost; it is kept as $k+1$ ordered layers $(L_0, L_1, \ldots, L_k)$.
- Support $S(\omega) = L_1 \cup \cdots \cup L_k$; L_0 holds pages outside the support.
- **Valid configuration:** X with $\omega(X) - \min(\omega) = 0$ — a state the offline optimum *could currently be in*.

Two facts we lean on

- OPT incurs a miss iff the requested page lies in L_0.
- Support splits into **revealed** $R(\omega)$ (suffix singleton layers) and **unrevealed** $N(\omega)$; $U(\omega) = k - |R(\omega)|$ counts unrevealed layers.

Observations 1 & 2: Uncertainty

Let $\mathcal{V}$ be the set of valid configurations: *everywhere* OPT *might be*. Its size $|\mathcal{V}|$ measures uncertainty.

Observation 1 A request to L_0 resets the support $\Rightarrow |\mathcal{V}|$ jumps (uncertainty peaks). A request to $N(\omega)$ reveals a page $\Rightarrow |\mathcal{V}|$ shrinks. A request to $R(\omega)$ leaves $\mathcal{V}$ unchanged.

Observation 2 $\mathcal{V}$ is algorithm-independent and evolves using only past requests $\Rightarrow$ an inherent offline/online gap. **Predictions can narrow** $\mathcal{V}$, reducing uncertainty and guiding evictions.

Observations 1 & 2: Uncertainty

Let $\mathcal{V}$ be the set of valid configurations: *everywhere* OPT *might be*. Its size $|\mathcal{V}|$ measures uncertainty.

Observation 1 A request to L_0 resets the support $\Rightarrow |\mathcal{V}|$ jumps (uncertainty peaks). A request to $N(\omega)$ reveals a page $\Rightarrow |\mathcal{V}|$ shrinks. A request to $R(\omega)$ leaves $\mathcal{V}$ unchanged.

Observation 2 $\mathcal{V}$ is algorithm-independent and evolves using only past requests $\Rightarrow$ an inherent offline/online gap. **Predictions can narrow** $\mathcal{V}$, reducing uncertainty and guiding evictions.

Online-Optimal Algorithms

Equitable maintains a *distribution* over all valid configurations; H_k-competitive, but its transition kernel costs $\mathcal{O}(k^2)$.

OnlineMin (OM) keeps a *single* configuration via random priorities, drawn from the same distribution as Equitable, in $\mathcal{O}(\log k)$ per request.

OnOPT a class generalizing OnlineMin with flexible priorities (OnlineMin is one instance); it always stays in a valid configuration and can be H_k-competitive.

Prior work: Equitable [Achlioptas et al., 2000], OnlineMin [Brodal et al., 2015], OnOPT [Moruz et al., 2012].

Observation 3

OnlineMin / OnOPT track valid configurations *online*, which provides the natural hook to inject predictions for eviction.

The Relative Prediction Budget (RPB)

- Predictions are extremely helpful when accurate, yet arbitrarily harmful when adversarial $\Rightarrow$ prediction-driven actions must be **gated**.

RPB primitive

Fix a robust baseline $\mathcal{A}$ and keep a nonnegative budget B_t. Deviate from $\mathcal{A}$ to a prediction-driven action only when you can pay from B_t; budget is *earned relative to* $\mathcal{A}$.

Prior algorithms implicitly do this — they differ only in their *earning rules*.

Prior Algorithms Through the RPB Lens

BlindOracle&LRU earns budget globally (follow whichever of BlindOracle/LRU is cheaper on the prefix). Coarse ⇒ extra factor 2 ⇒ $2k$ robustness.

F&R earns when its step cost matches Belady — but detecting this needs excessive recomputation.

PredictiveMarker earns H_k per clean request (aggressive) ⇒ $4H_k$ robustness.

LMarker earns 1 per clean request (conservative) ⇒ $2H_k+4$ robustness, but underuses good predictions.

Observation 4 and Our Principle

Limitation 1 **Coarse evaluation.** Global-only comparison (BlindOracle&LRU) over/under-uses predictions ⇒ an extra multiplicative factor.

Limitation 2 **Fixed thresholds.** Marker-based thresholds (H_k or 1) do not adapt to prediction effectiveness ⇒ overuse ($4H_k$) or underuse (weak in practice).

Principle

Adaptively correlate the budget with **both** prediction effectiveness **and** the baseline's performance ⇒ robustness within an *additive* $\mathcal{O}(1)$ of the baseline.

Observation 4 and Our Principle

Limitation 1 **Coarse evaluation.** Global-only comparison ($\textsc{BlindOracle\&LRU}$) over/under-uses predictions $\Rightarrow$ an extra multiplicative factor.

Limitation 2 **Fixed thresholds.** $\textsc{Marker}$-based thresholds (H_k or 1) do not adapt to prediction effectiveness $\Rightarrow$ overuse ($4H_k$) or underuse (weak in practice).

Principle

Adaptively correlate the budget with **both** prediction effectiveness **and** the baseline's performance $\Rightarrow$ robustness within an *additive* $\mathcal{O}(1)$ of the baseline.

Observation 4 and Our Principle

Limitation 1 **Coarse evaluation.** Global-only comparison (BlindOracle&LRU) over/under-uses predictions ⇒ an extra multiplicative factor.

Limitation 2 **Fixed thresholds.** Marker-based thresholds (H_k or 1) do not adapt to prediction effectiveness ⇒ overuse ($4H_k$) or underuse (weak in practice).

Principle

Adaptively correlate the budget with **both** prediction effectiveness **and** the baseline's performance ⇒ robustness within an *additive* $\mathcal{O}(1)$ of the baseline.

From Predictions to 1-Consistency

- Use predictions exactly at an L_0-miss, where uncertainty (Observation 1) peaks.
- *Perfect predictions* rank, for eviction, the pages OPT evicts before their next request; *following predictions* evicts by that order (e.g., largest predicted next-arrival).

Theorem (1-consistency, informal)

With perfect predictions, an algorithm that on every L_0-miss evicts following the predictions is optimal — hence 1-consistent.

This is also the **minimal** prediction usage: 1-consistency *requires* using predictions on every L_0-miss (Appendix).

The RPB-OnOPT Framework

On a cache miss to page p:

1. $p \in L_0$ **(uncertainty peaks):** evict following predictions; reset $B \leftarrow \tau$ ($\tau = \mathcal{O}(1)$). $\rightarrow$ 1-*consistency*
2. $p \in L_i,\ i > 0$ **(a lazy request):**
   - build OnlineMin's eviction-candidate set V;
   - **earn:** if the gate $\mathcal{G}_{RPB}(Y, \omega)$ holds, $B \leftarrow B + 1$;
   - **spend:** if $B > 0$, evict following predictions, $B \leftarrow B - 1$; else evict by the self-defined priority. $\rightarrow$ *robustness*
   - update $Y \leftarrow \mathcal{U}_{RPB}(\omega)$.

The RPB-OnOPT Framework

On a cache miss to page p:

1. $p \in L_0$ **(uncertainty peaks):** evict following predictions; reset $B \leftarrow \tau$ ($\tau = \mathcal{O}(1)$). $\rightarrow$ 1-*consistency*
2. $p \in L_i,\ i > 0$ **(a lazy request):**
   - build OnlineMin's eviction-candidate set V;
   - **earn:** if the gate $\mathcal{G}_{RPB}(Y, \omega)$ holds, $B \leftarrow B + 1$;
   - **spend:** if $B > 0$, evict following predictions, $B \leftarrow B - 1$; else evict by the self-defined priority. $\rightarrow$ *robustness*
   - update $Y \leftarrow \mathcal{U}_{RPB}(\omega)$.

RPB-OM: Instantiating the Framework

Set the base to OnlineMin (self-defined priority = OnlineMin priority), and

$$\mathcal{G}_{RPB}(Y,\omega) = \Big(U(\omega) \le \tfrac{Y+2}{e} - 2 \Big), \qquad \mathcal{U}_{RPB}(\omega) = U(\omega),$$

where $U(\omega) = k - |R(\omega)|$ is the number of unrevealed layers.

Why this gate

- Charge budget only when previous evictions did well relative to the robust baseline (effectiveness $\leftrightarrow$ budget).
- The gate uses a *worst-case lower bound* on OnlineMin's accumulated expected cost — no need to simulate OnlineMin's cache.

Main Results

Theorem (Consistency)

RPB-OnOPT *is* 1*-consistent; hence so is* RPB-OM.

Theorem (Robustness)

RPB-OM *is* $(H_k + \mathcal{O}(1))$*-robust* — *optimal up to an additive constant*.

Theorem (Smoothness)

RPB-OM *is* $\mathcal{O}(1 + \log(\eta_1 / \mathit{cost}(\mathrm{OPT})))$*-smooth, matching the best known (*LMarker*).*

- $\mathcal{O}(\log k)$ per request (same as OnlineMin).
- Why not exactly H_k? That would require following the H_k-competitive *online* algorithm with no predictions — outside the learning-augmented setting.

A New Property of OnlineMin (Contribution 1)

Let $\phi(\omega)$ be OnlineMin's potential: its expected cost on a lazy adversary strategy from ω.

Potential changes

- *(prior)* On a request to $p \in L_0$: $\Delta\phi \leq H_k - 1$.
- New lemma — on a lazy adversary request to $p \in N(\omega)$:

$$\Delta\phi \leq -\frac{1}{U(\omega)+1}.$$

This per-step *strict drop* (proved by induction on $U(\omega)$) is what lets RPB-OM earn budget cheaply against OnlineMin and reach robustness near H_k.

Robustness Proof Sketch

Potential combining three terms (OnlineMin potential + cache gap + budget):

$$\Phi(\omega) = \phi(\omega) + D(\omega) + B(\omega).$$

- **Cache-difference lemma:** on a lazy-adversary miss, $\Delta D \le -1 + x$ if RPB-OM evicts by OnlineMin's priorities, else $\Delta D \le x$ ($x =$ OnlineMin's miss prob.).
- **Per-request amortization:**

$$\underbrace{\Delta cost + \Delta\Phi \le (H_k + \mathcal{O}(1))\,\Delta cost(\mathrm{OPT})}_{L_0\text{-miss}}, \quad \underbrace{\Delta cost + \Delta\Phi \le 0}_{\text{lazy request}}$$

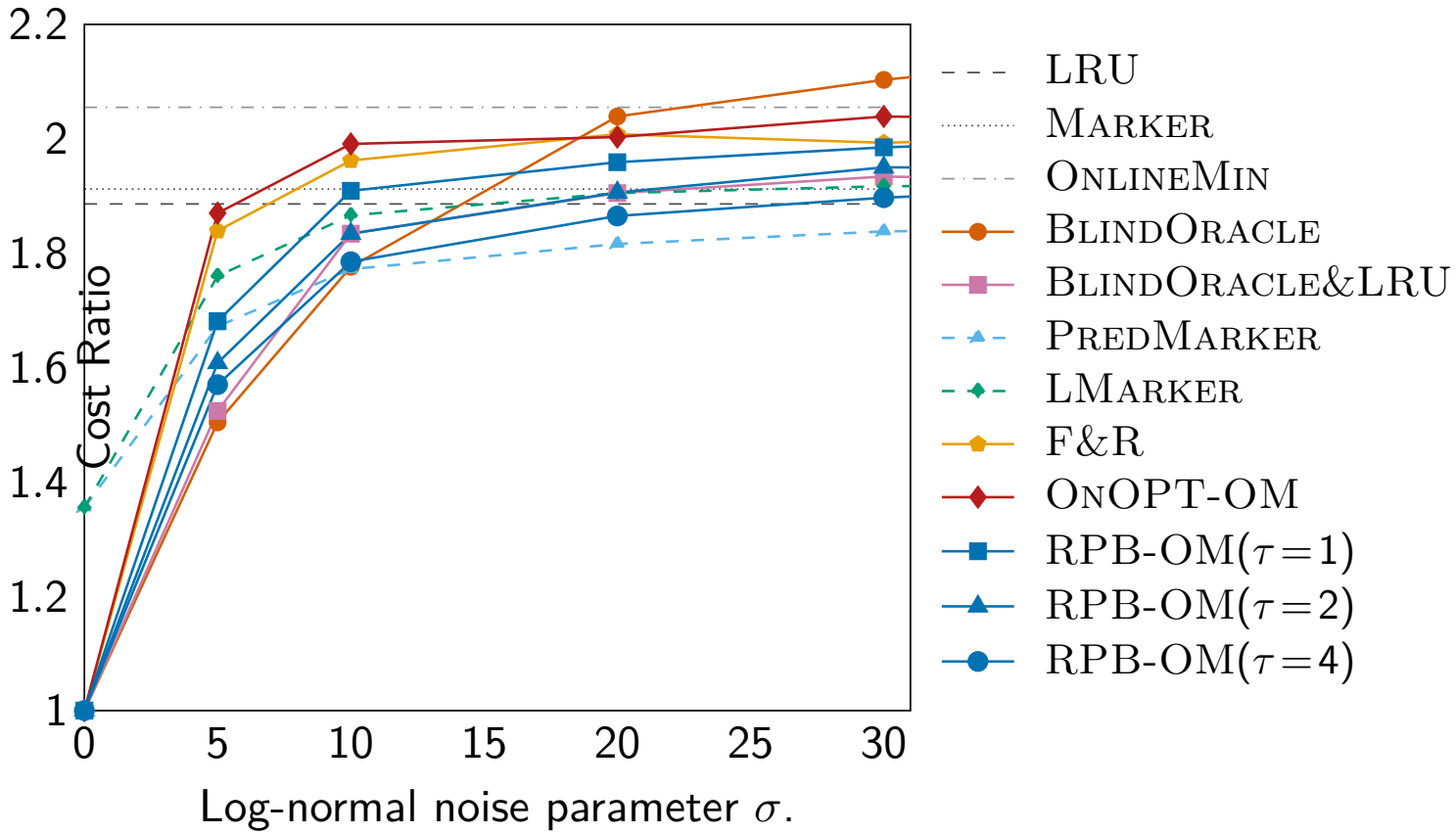
  (the gate-pass case uses $-\ln \frac{Y+2}{U(\omega)+2} \le -1$).
- **Telescope** ($\Phi_{\text{init}} = 0,\ \Phi_{\text{final}} \ge 0$):
  $cost(\text{RPB-OM}) \le (H_k + \mathcal{O}(1))\, cost(\mathrm{OPT})$.

Experimental Setup

- **Benchmark:** real-world traces from **SPEC CPU 2006**; cache 2MB, 16-way, 64B lines $\Rightarrow$ 2,048 sets of 16 lines, i.e. $k = 16$.
- **Synthetic predictions:** log-normal noise added to true next-arrival times (varying noise levels).
- **Real predictors:** PLECO (probability-based) and POPU (frequency-based), each predicting next request after $1/p$ steps.

Cost ratio = #misses relative to OPT (lower is better); hit ratio in % (higher is better).

Synthetic Predictions (*bzip*)



Cost ratio vs. log-normal noise σ. RPB-OM tracks OPT when predictions are good and degrades gracefully as noise grows; BlindOracle lacks robustness; PredictiveMarker/LMarker lack the ideal 1-consistency.

Real Predictors + Ablation (SPEC CPU 2006)

Algorithm	**Cost Ratio ↓**	**Hit Ratio (%) ↑**
OPT	1.000	34.62
LRU / Marker / OnlineMin	1.478 / 1.392 / 1.396	14.3 / 16.5 / 16.3
PLECO predictor		
BlindOracle	1.404	15.92
BlindOracle&LRU	1.286	20.53
OnOPT-OM	1.253	22.42
RPB-OM (τ=1)	**1.249**	**22.75**
POPU predictor		
BlindOracle&LRU	1.230	23.10
OnOPT-OM	1.237	23.76
RPB-OM (τ=4)	**1.213**	**24.55**

OnOPT-OM augments OnlineMin with predictions on L_0-misses (*without* RPB). Omitted baselines (F&R, LMarker, PredictiveMarker) are all worse than the rows shown.

Conclusion

- We close a gap: $\mathrm{RPB\text{-}OnOPT}$ enables best-possible robustness $H_k + \mathcal{O}(1)$; $\mathrm{RPB\text{-}OM}$ attains $(1, H_k + \mathcal{O}(1))$ with $\mathcal{O}(\log k)$ per request and matching smoothness.
- The *relative prediction budget* is a general lens — it pinpoints precisely where prior methods over- or under-trust predictions.
- **Future work:** retrofit RPB-style budgeting into existing learning-augmented algorithms.

Thank You

Code available at:
`https://github.com/Natureal/ICML-Cache-Coliseum`

Contact:
`naturechenpeng@gmail.com`

Backup — Hit-Credit Variant (RPB-OM-HC)

An alternative way to *earn* the prediction budget inside the RPB-OnOPT framework.

- **Idea:** every cache hit signals that past evictions kept useful pages — so earn budget *directly from hits* rather than from the gate.
- The hit-credit accumulates *cumulatively* across phases (not reset at L_0-misses), tracking the long-run effectiveness of predictions.
- **Same guarantees** as RPB-OM (1-consistent, $(H_k + \mathcal{O}(1))$-robust, smooth), and slightly better in practice at small τ when predictions are good.