Elastic-Sketch Under Random Stationary Streams: Limiting Behavior and Near-Optimal Configuration

Younes Ben Mazziane (University of Avignon)
Vinay Kumar B. R. (IIT Bombay)
Othmane Marfoq (Meta)

AVIGNON UNIVERSITÉ

Outline

1) Streaming Algorithms

2) Classical Streaming Algorithms

3) Elastic-Sktech

4) Motivation

5) Contributions

Streaming Algorithms

- Catalog of items $\mathscr{C}$, $|\mathscr{C}| = n_{\mathscr{C}}$
- Data Stream $\mathbf{R} = (R_1, \ldots, R_T) \in \mathscr{C}^T$
- Objective: Key features about $R_1, \ldots, R_t$ at each t
  - Example:
    - Count of item i: $N_i(t)$
    - Top k items
    - ϕ Heavy-Hitters
    - $\|\mathbf{N}(t)\|_p$

Streaming Algorithms

- Catalog of items $\mathscr{C}$, $|\mathscr{C}| = n_{\mathscr{C}}$
- Data Stream $\mathbf{R} = (R_1, \ldots, R_T) \in \mathscr{C}^T$
- Objective: Key features about $R_1, \ldots, R_t$ at each t
  - Example:
    - Count of item i: $N_i(t)$
    - Top k items
    - ϕ Heavy-Hitters
    - $\|\mathbf{N}(t)\|_p$
- Constraints: High arrival rate → $o(n_{\mathscr{C}})$ memory + one pass + fast processing
  → Settle for approximations
- Applications: Network monitoring and clickstream analytics

Outline

1) Streaming Algorithms

2) Classical Streaming Algorithms

3) Elastic-Sktech

4) Motivation

5) Contributions

Classical Streaming Algorithms

- Counter-Based: Maintain $(i, \hat{N}_i(t))_{i \in S_t}$: $|S_t| \ll |\mathscr{C}|$

Example: Space Saving [4], Frequent [1,2]

[1]Misra, S., & Gries, D. (1982). *Finding repeated elements*. **Science of Computer Programming**, 2(2), 143–152.
[2] Demaine, E. D., López-Ortiz, A., & Munro, J. I. (2002). *Frequency estimation of internet packet streams with limited space*. In **European Symposium on Algorithms (ESA)** (pp. 348–360).
[3] Charikar, M., Chen, K., & Farach-Colton, M. (2002). *Finding frequent items in data streams*. In **International Colloquium on Automata, Languages and Programming (ICALP)** (pp. 693–703).
[4] Metwally, A., Agrawal, D., & El Abbadi, A. (2005). *Efficient computation of frequent and top-k elements in data streams*. In **International Conference on Database Theory (ICDT)** (pp. 398–412).
[5] Cormode, G., & Muthukrishnan, S. (2005). *An improved data stream summary: The Count–Min sketch and its applications*. **Journal of Algorithms**, 55(1), 58–75.

Classical Streaming Algorithms

- Counter-Based: Maintain $(i, \hat{N}_i(t))_{i \in S_t} : \ |S_t| \ll |\mathscr{C}|$

  Example: Space Saving [4], Frequent [1,2]

- Sketches or hash-based data structures:

  1. Maintaining a vector $\mathbf{Y}(t)$ of size $m = o(n_{\mathscr{C}})$
  2. $\mathbf{Y}(t) = \mathbf{N}(t) \cdot \Pi$ (Random and sparse matrix)
  3. $\hat{N}_i(t) = f(Y_{h_1(i)}(t), \ldots, Y_{h_d(i)}(t)) : \ d \ll m$

  Example: Count-Sketch [3], Count-Min-Sketch [5]

[1]Misra, S., & Gries, D. (1982). *Finding repeated elements*. **Science of Computer Programming**, 2(2), 143–152.
[2] Demaine, E. D., López-Ortiz, A., & Munro, J. I. (2002). *Frequency estimation of internet packet streams with limited space*. In **European Symposium on Algorithms (ESA)** (pp. 348–360).
[3] Charikar, M., Chen, K., & Farach-Colton, M. (2002). *Finding frequent items in data streams*. In **International Colloquium on Automata, Languages and Programming (ICALP)** (pp. 693–703).
[4] Metwally, A., Agrawal, D., & El Abbadi, A. (2005). *Efficient computation of frequent and top-k elements in data streams*. In **International Conference on Database Theory (ICDT)** (pp. 398–412).
[5] Cormode, G., & Muthukrishnan, S. (2005). *An improved data stream summary: The Count–Min sketch and its applications*. **Journal of Algorithms**, 55(1), 58–75.

Count-Min Sketch (CMS)

- Generate Π uniformly at random such that: $\pi_{i,j} \in \{0,1\}, \sum_{j=1}^{m} \pi_{i,j} = d \ll m$
- Example ($N = 5$, $m = 4$, $d = 2$) $\mathbf{\Pi} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$
- At each step t, maintain $\mathbf{Y}(t) = \mathbf{N}(t) \cdot \Pi$

$\mathbf{N}(t) = [10\ 10\ 1\ 1\ 1]$ $\mathbf{Y}(t) = [21\ 2\ 12\ 11]$ (d estimators)

Count-Min Sketch (CMS)

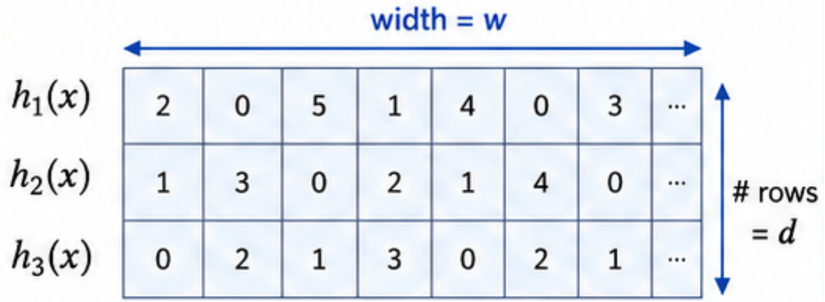
- Generate Π uniformly at random such that: $\pi_{i,j} \in \{0,1\}, \sum_{j=1}^{m} \pi_{i,j} = d \ll m$
- Example ($N = 5$, $m = 4$, $d = 2$) $\Pi = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$
- At each step t, maintain $\mathbf{Y}(t) = \mathbf{N}(t) \cdot \Pi$

$\mathbf{N}(t) = [10\ 10\ 1\ 1\ 1]$ $\mathbf{Y}(t) = [21\ 2\ 12\ 11]$ (d estimators)

- Query for an item i's count $\hat{N}_i(t) = \min_{c:\ \pi_{i,c}=1} Y_c(t)$
  - Example: $\hat{N}_1 = \min(21,12)$

Count-Min Sketch (CMS)

1 Data structure



d hash functions $h_1, h_2, \ldots, h_d$.

Each row is a hash table of counters.

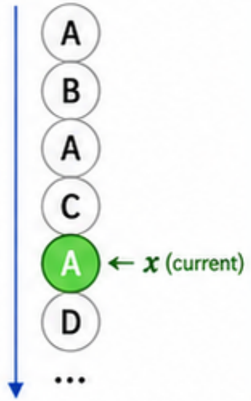
Count-Min Sketch (CMS)

1 Data structure

d hash functions $h_1, h_2, \ldots, h_d$.

Each row is a hash table of counters.

2 Update for an arriving item x



1. **Hash** x with d hash functions.
2. **Find the** d positions $h_1(x), h_2(x), \ldots, h_d(x)$.
3. **Increment** all corresponding counters by 1.

Example (for x):

Positions:	$h_1(x)$	$h_2(x)$	$h_3(x)$
Counters before:	4	6	4
	+1	+1	+1
Counters after:	5	7	5

All hashed counters are incremented.

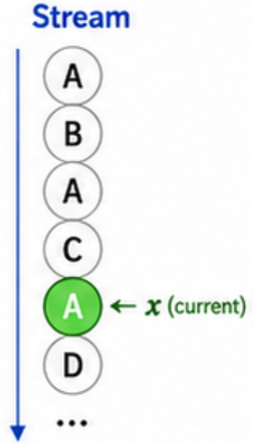
Count-Min Sketch (CMS)

1 Data structure

d hash functions $h_1, h_2, \ldots, h_d$.

Each row is a hash table of counters.

2 Update for an arriving item x



1. **Hash** x with d hash functions.
2. **Find the** d positions $h_1(x), h_2(x), \ldots, h_d(x)$.
3. **Increment** all corresponding counters by 1.

Example (for x):

Positions:	$h_1(x)$	$h_2(x)$	$h_3(x)$
Counters before:	4	6	4
	+1	+1	+1
Counters after:	5	7	5

All hashed counters are incremented.

3 Query

1. Hash x into the same d positions.
2. Read the d counters.
3. Return the minimum of these counters.

Example (for x):

Positions:	$h_1(x)$	$h_2(x)$	$h_3(x)$
Counters:	7	9	8

$\text{estimate}(x) = \min = 7$

The estimate is an upper bound on the true frequency of x.

Outline

1) Streaming Algorithms

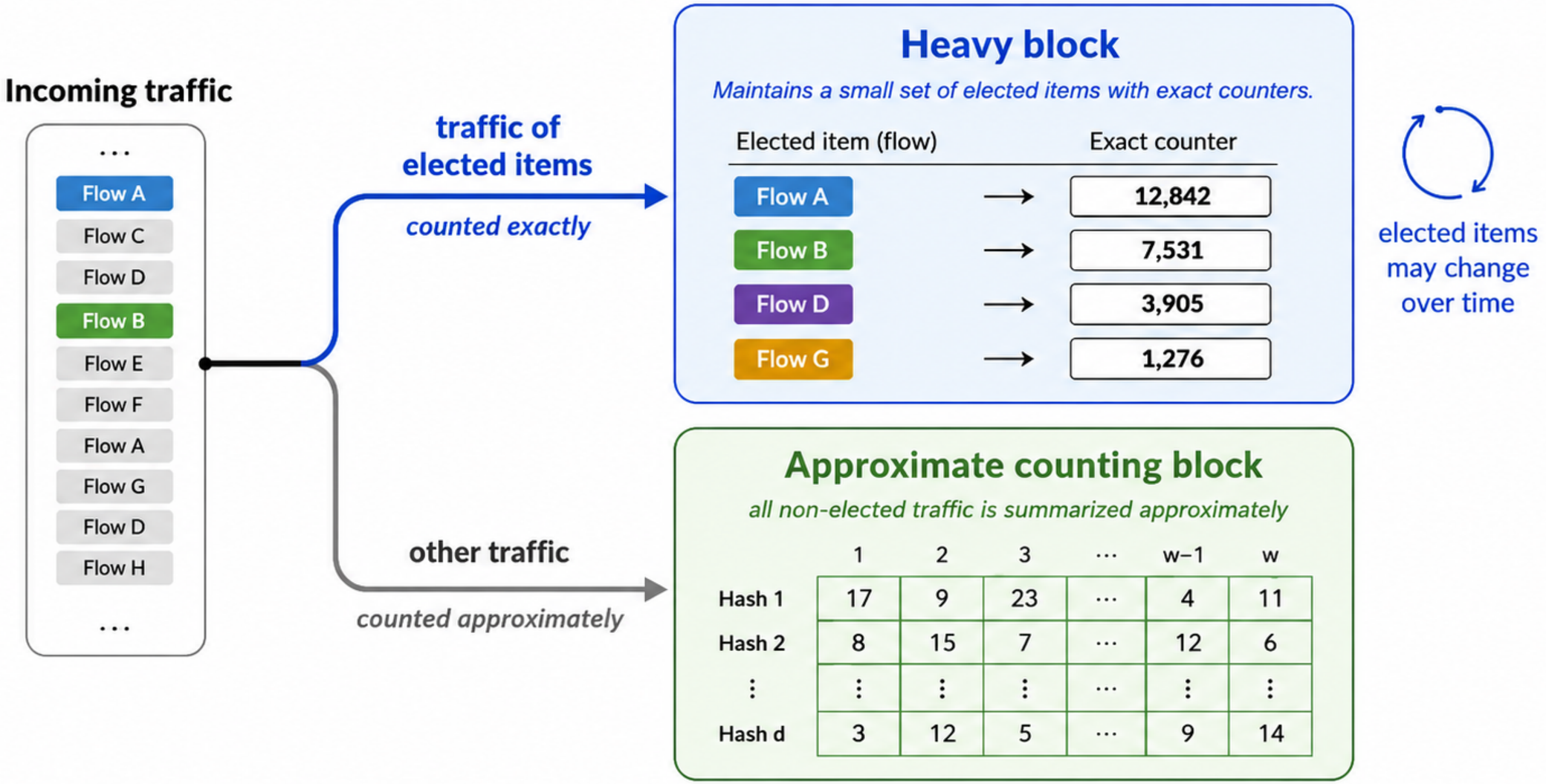
2) Classical Streaming Algorithms

3) Elastic-Sketch

4) Motivation

5) Contributions

Elastic-Sketch

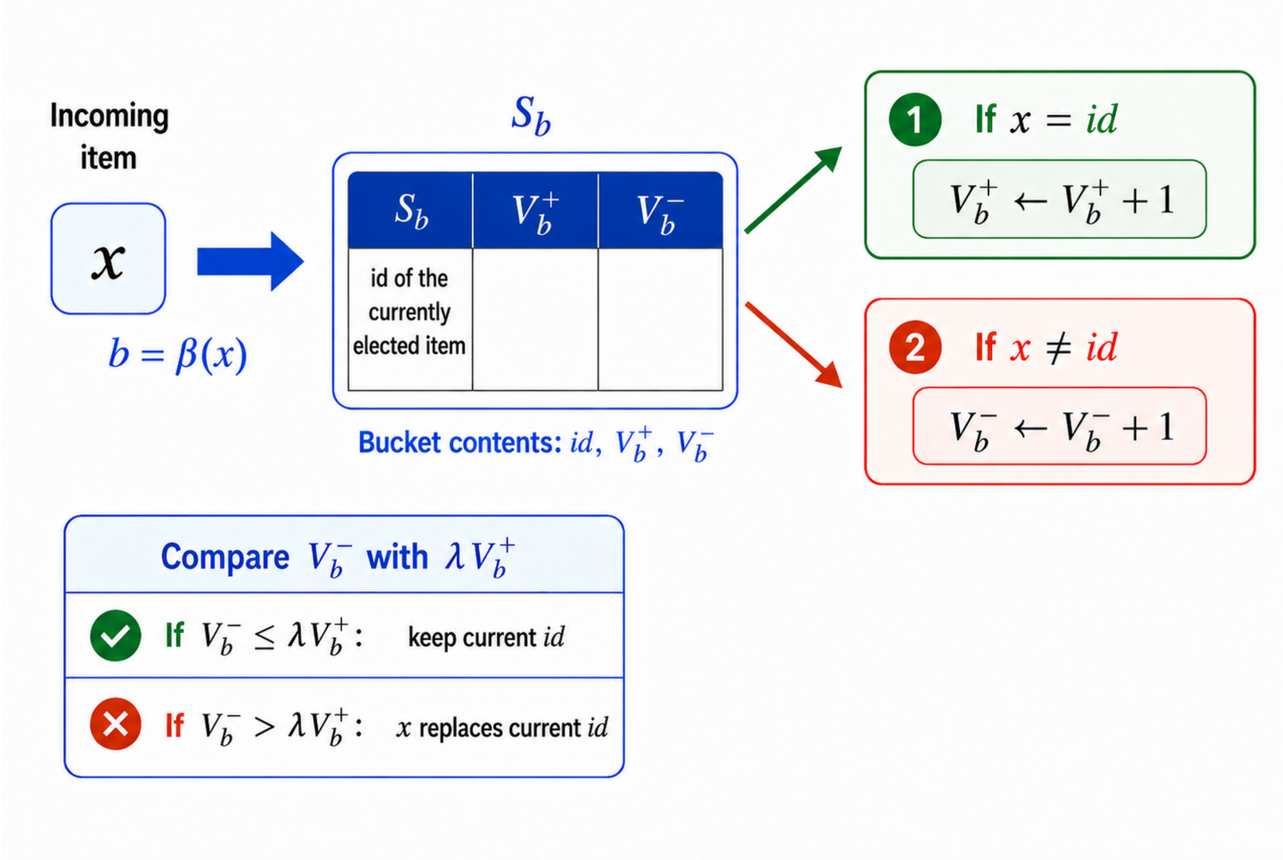


Elastic-Sketch

- m_1 buckets
- Hash function $\beta : \mathscr{C} \mapsto [m_1]$
- Eviction threshold $\lambda \in \mathbb{N}$

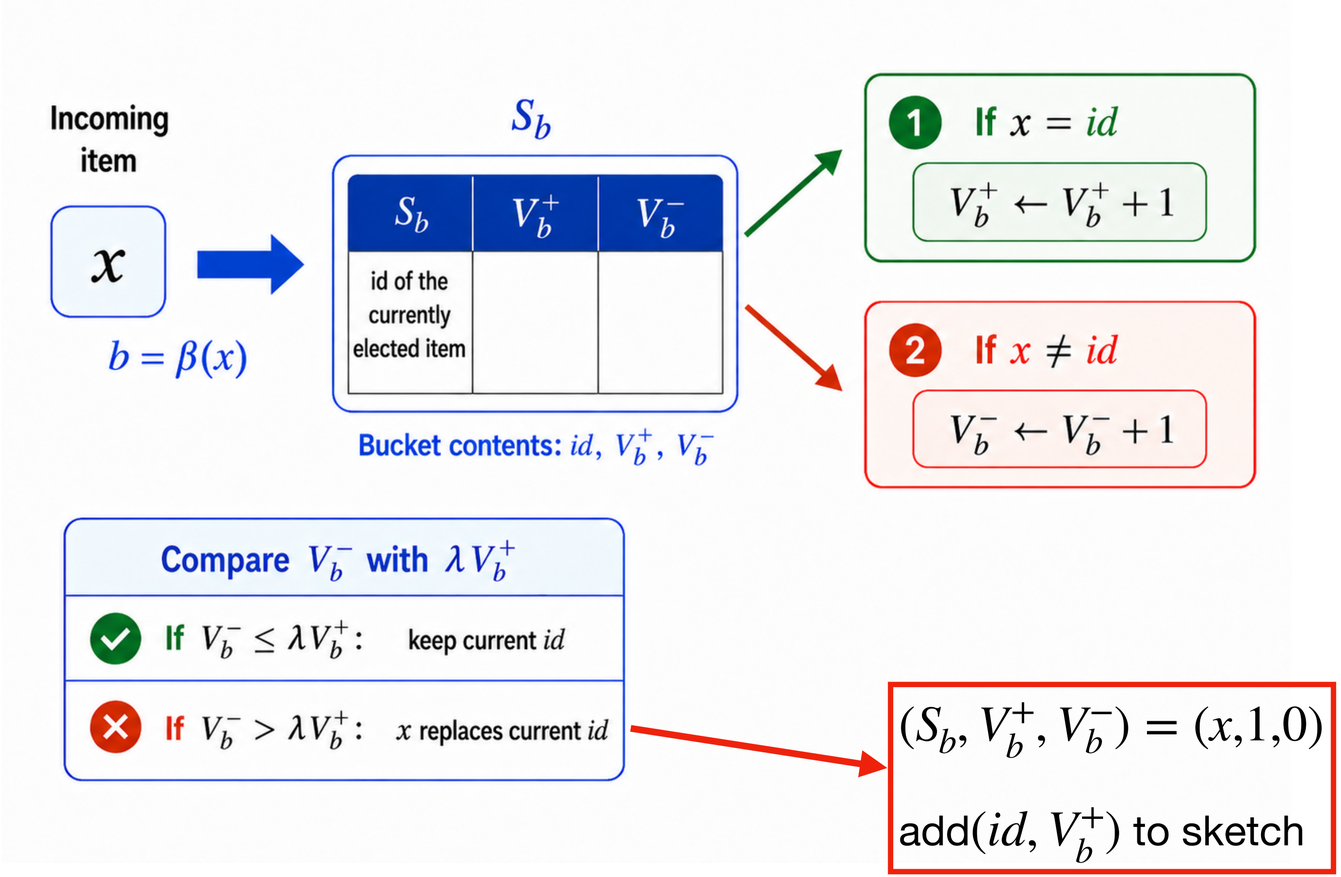
Elastic-Sketch

- m_1 buckets
- Hash function $\beta : \mathscr{C} \mapsto [m_1]$
- Eviction threshold $\lambda \in \mathbb{N}$



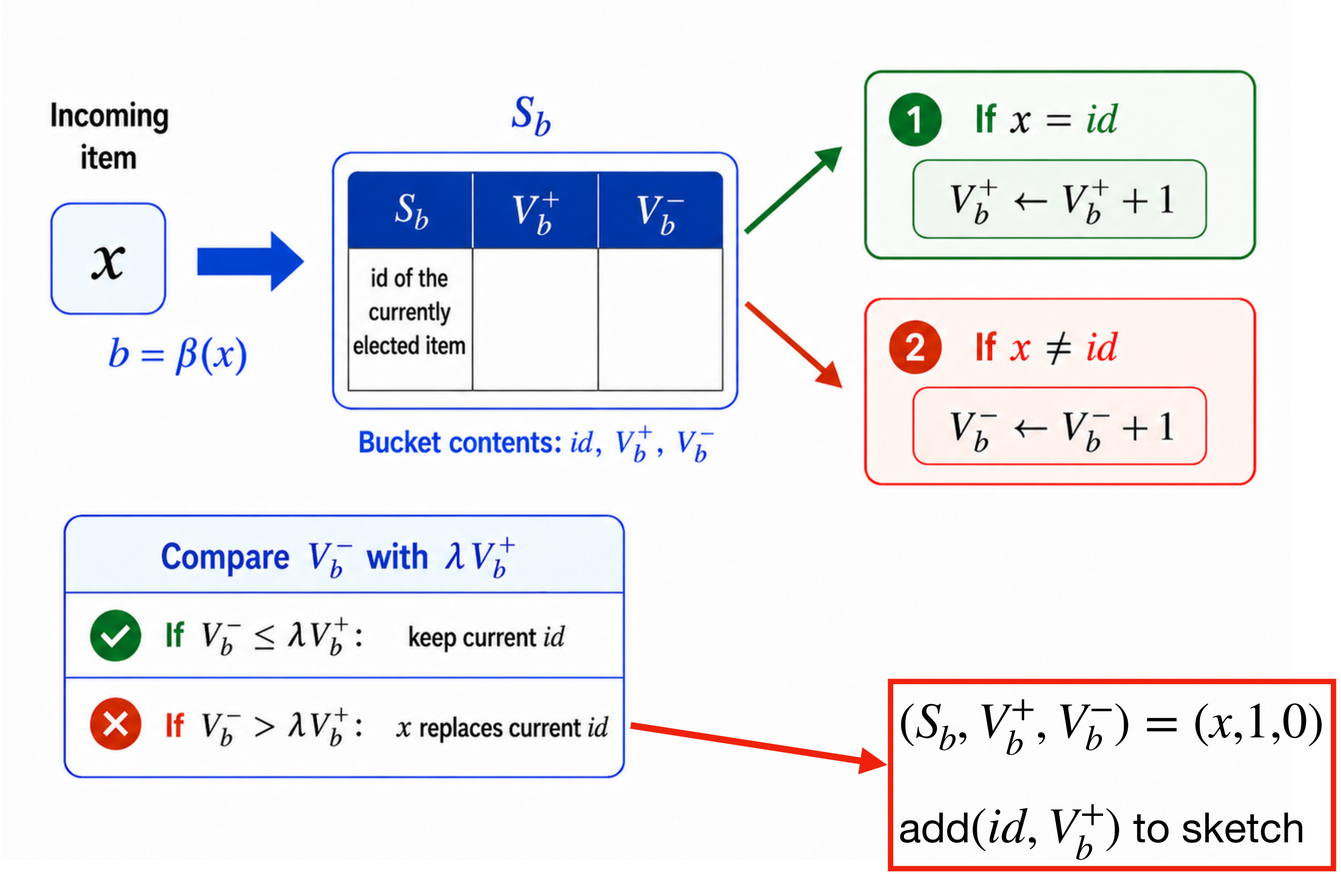
Elastic-Sketch

- m_1 buckets
- Hash function $\beta : \mathscr{C} \mapsto [m_1]$
- Eviction threshold $\lambda \in \mathbb{N}$



Elastic-Sketch

- m_1 buckets
- Hash function $\beta : \mathscr{C} \mapsto [m_1]$
- Eviction threshold $\lambda \in \mathbb{N}$



- Count estimation: $\hat{N}_i = 1_{\{S_{\beta(i)}=i\}} V^+_{\beta(i)} + \min_{l \in [d]} Y_{l,h_l(i)}$

Outline

1) Streaming Algorithms

2) Classical Streaming Algorithms

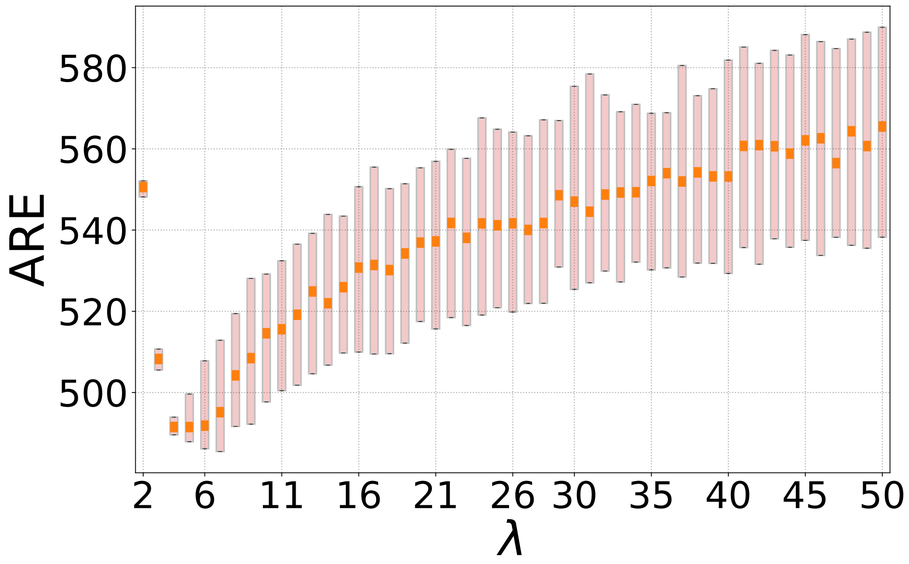
3) Elastic-Sketch

4) Motivation

5) Contributions

Motivation

- ARE: Average Relative Error
- Zipf streams with skew $\alpha = 1.2$
- 100 streams (Q1–Q3)

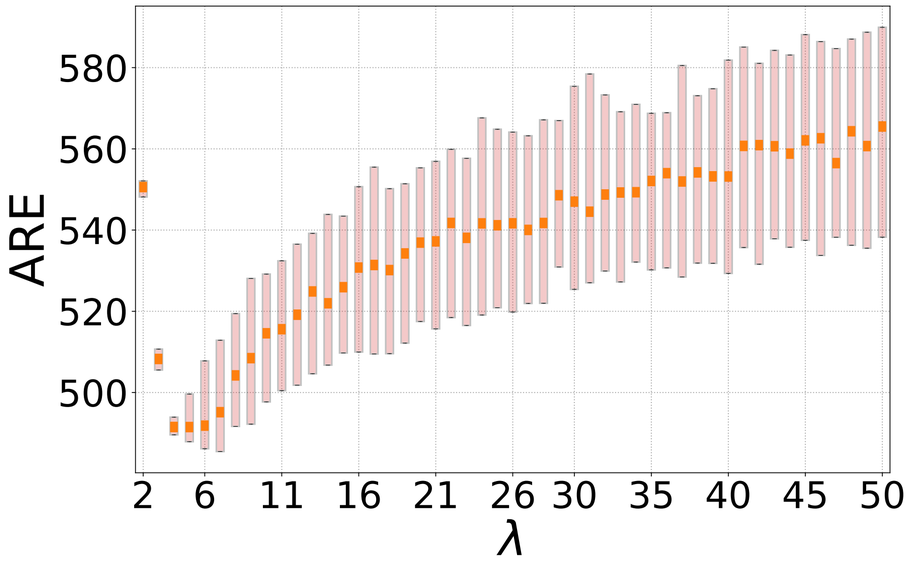


$m_1 = 50,\ w = 200,\ d = 1$

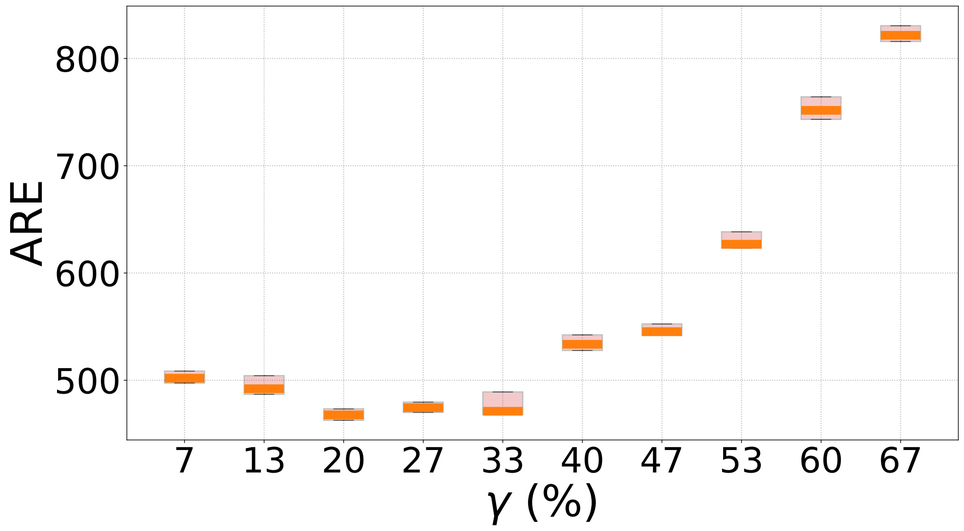
Motivation

- ARE: Average Relative Error
- Zipf streams with skew $\alpha = 1.2$
- 100 streams (Q1–Q3)

- γ: Fraction of memory for Heavy-block



$m_1 = 50, w = 200, d = 1$



$2m_1 + w \approx 300, \lambda = 5$

Motivation (Questions)

Given an arrival distribution $\mathbf{p} = (p_i)_{i \in \mathscr{C}}$ with support $n_{\mathscr{C}}$: $\Pr\left(R_t = i\right) = p_i$

- What is the memory-accuracy trade-off?
- How to choose λ and γ for $\mathbf{p}$?

Outline

1) Streaming Algorithms

2) Classical Streaming Algorithms

3) Elastic-Sketch

4) Motivation

5) Contributions

Contributions

5.1 Limiting behavior

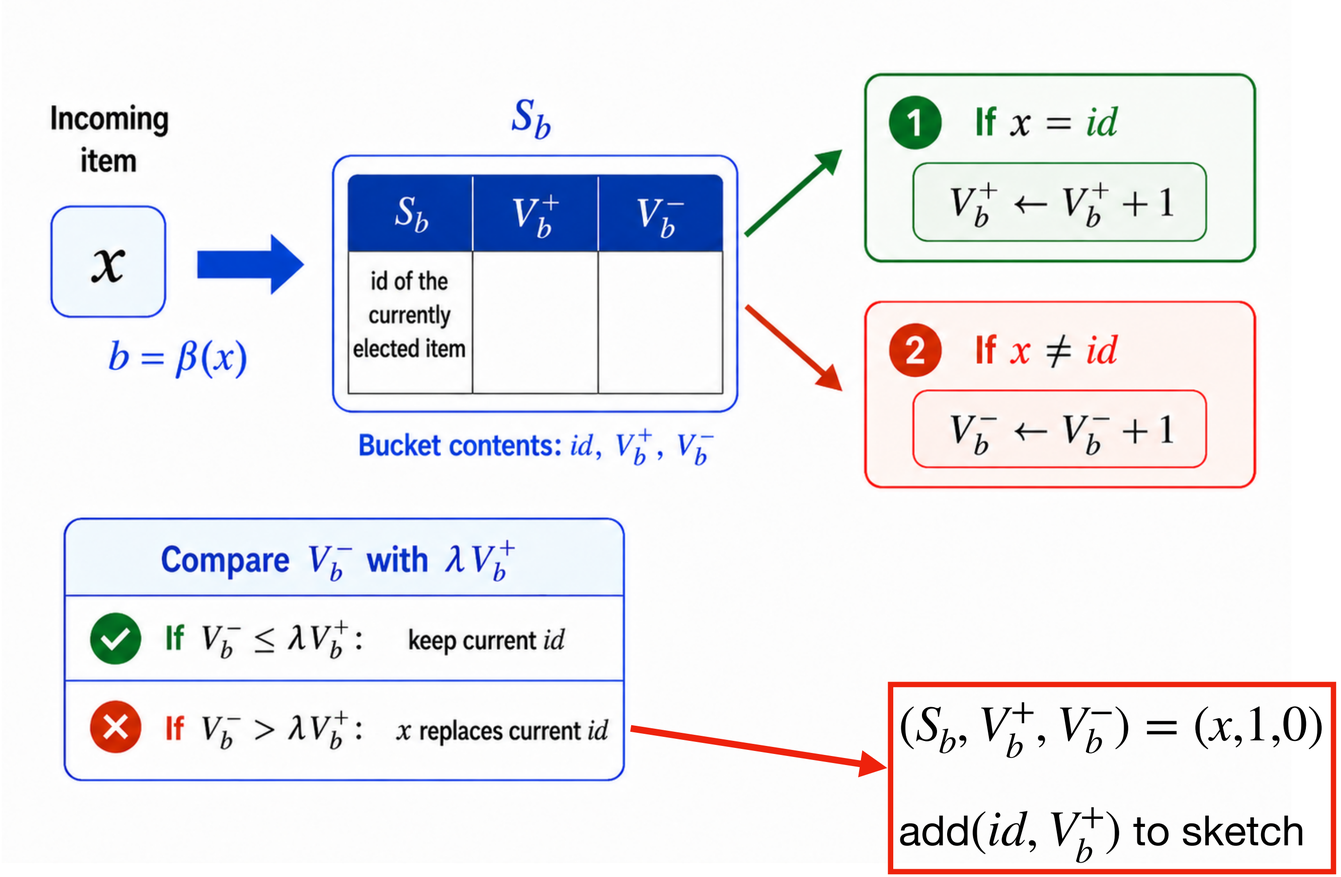
- 5.1.1 Heavy block
- 5.1.2 CM block and error

5.2 Near-Optimal Configuration

- 5.2.1 High probability upper bound on $\lambda^*(\beta)$
- 5.2.2 Efficient configuration

Elastic-Sketch (Recap)

- m_1 buckets
- Hash function $\beta : \mathscr{C} \mapsto [m_1]$
- Eviction threshold $\lambda \in \mathbb{N}$



- Count estimation: $\hat{n}_i = 1_{\{S_{\beta(i)}=i\}} V^+_{\beta(i)} + \min_{l \in [d]} Y_{l,h_l(i)}$

Limiting behavior (Heavy block)

Definition: Aggregate filtering/absorption: $G_\beta(\lambda) \triangleq \lim_{\tau\to\infty} \frac{1}{\tau} \sum_{b=1}^{m_1} V_b^+(\tau)$

Limiting behavior (Heavy block)

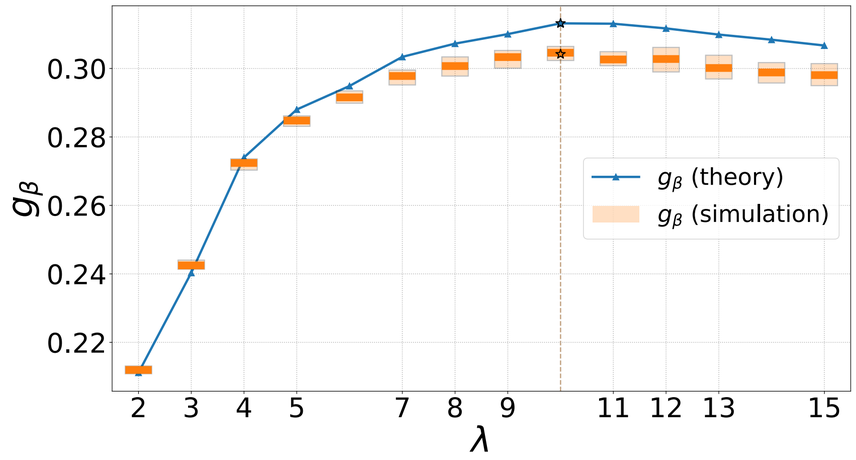
Definition: Aggregate filtering/absorption: $G_\beta(\lambda) \triangleq \lim_{\tau\to\infty} \frac{1}{\tau} \sum_{b=1}^{m_1} V_b^+(\tau)$

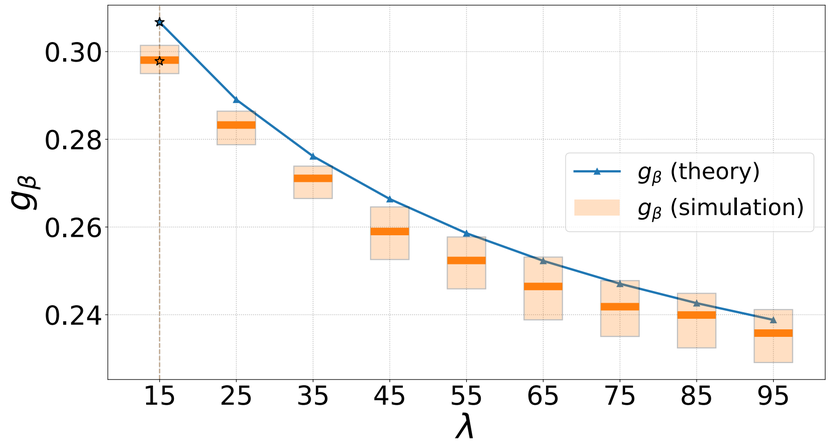
Theorem 1: $G_\beta(\lambda)$ admits a closed form expression and its expectation $g_\beta(\lambda)$ can be computed in $\mathcal{O}(n_{\mathscr{C}})$.

Limiting behavior (Heavy block)

Definition: Aggregate filtering/absorption: $G_\beta(\lambda) \triangleq \lim_{\tau\to\infty} \frac{1}{\tau} \sum_{b=1}^{m_1} V_b^+(\tau)$

Theorem 1: $G_\beta(\lambda)$ admits a closed form expression and its expectation $g_\beta(\lambda)$ can be computed in $\mathcal{O}(n_{\mathscr{C}})$.





50 Zipf samples with $\alpha = 0.8$, $m_1 = 200$, $n_{\mathscr{C}} = 10^4$, $\tau = 5 \times 10^5$

Limiting behavior (Proof)

Collisions set $\mathscr{C}_b = \left\{ i \in \mathscr{C} : \beta(i) = b \right\}$, and $\mu_b = \sum_{i \in \mathscr{C}_b} p_i$

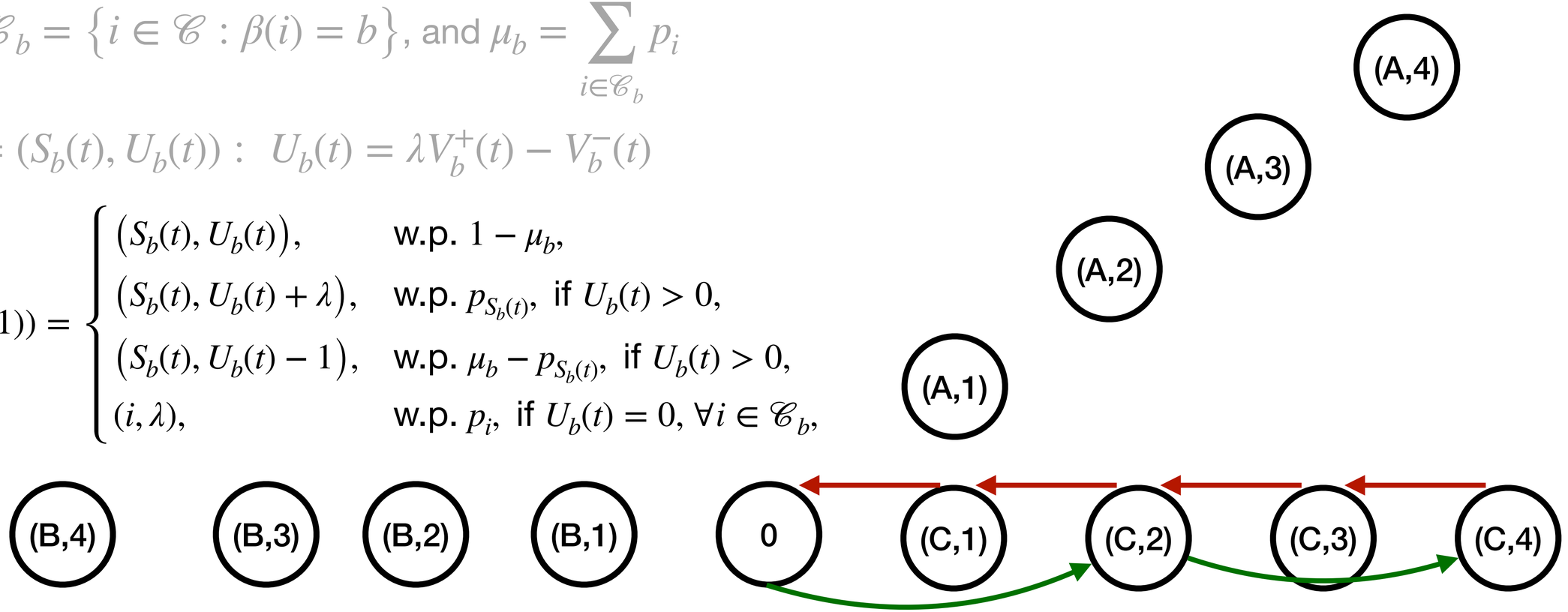
Define $M_b(t) = (S_b(t), U_b(t)) : \; U_b(t) = \lambda V_b^+(t) - V_b^-(t)$

Limiting behavior (Proof)

Collisions set $\mathscr{C}_b = \{i \in \mathscr{C} : \beta(i) = b\}$, and $\mu_b = \sum_{i \in \mathscr{C}_b} p_i$

Define $M_b(t) = (S_b(t), U_b(t)) : \ U_b(t) = \lambda V_b^+(t) - V_b^-(t)$

$$(S_b(t+1), U_b(t+1)) = \begin{cases} \big(S_b(t), U_b(t)\big), & \text{w.p. } 1-\mu_b, \\ \big(S_b(t), U_b(t)+\lambda\big), & \text{w.p. } p_{S_b(t)}, \text{ if } U_b(t) > 0, \\ \big(S_b(t), U_b(t)-1\big), & \text{w.p. } \mu_b - p_{S_b(t)}, \text{ if } U_b(t) > 0, \\ (i, \lambda), & \text{w.p. } p_i, \text{ if } U_b(t) = 0, \forall i \in \mathscr{C}_b, \end{cases}$$

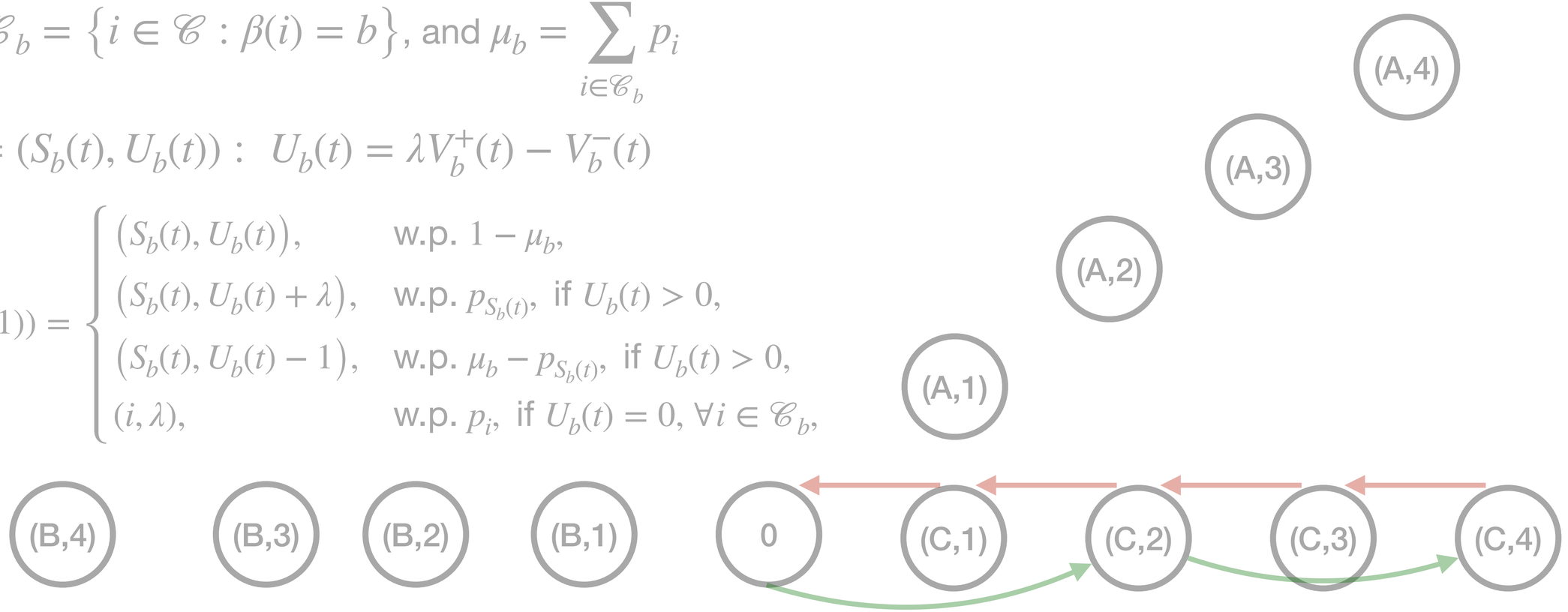


Limiting behavior (Proof)

Collisions set $\mathscr{C}_b = \{i \in \mathscr{C} : \beta(i) = b\}$, and $\mu_b = \sum_{i \in \mathscr{C}_b} p_i$

Define $M_b(t) = (S_b(t), U_b(t)) : \ U_b(t) = \lambda V_b^+(t) - V_b^-(t)$

$$(S_b(t+1), U_b(t+1)) = \begin{cases} \big(S_b(t), U_b(t)\big), & \text{w.p. } 1 - \mu_b, \\ \big(S_b(t), U_b(t) + \lambda\big), & \text{w.p. } p_{S_b(t)}, \text{ if } U_b(t) > 0, \\ \big(S_b(t), U_b(t) - 1\big), & \text{w.p. } \mu_b - p_{S_b(t)}, \text{ if } U_b(t) > 0, \\ (i, \lambda), & \text{w.p. } p_i, \text{ if } U_b(t) = 0, \forall i \in \mathscr{C}_b, \end{cases}$$



Define $\lambda_i(\beta) = \dfrac{\mu_b}{p_i} - 1$ and $\lambda_b^{(1)} = \min_{i \in \mathscr{C}_b} \lambda_i(\beta)$

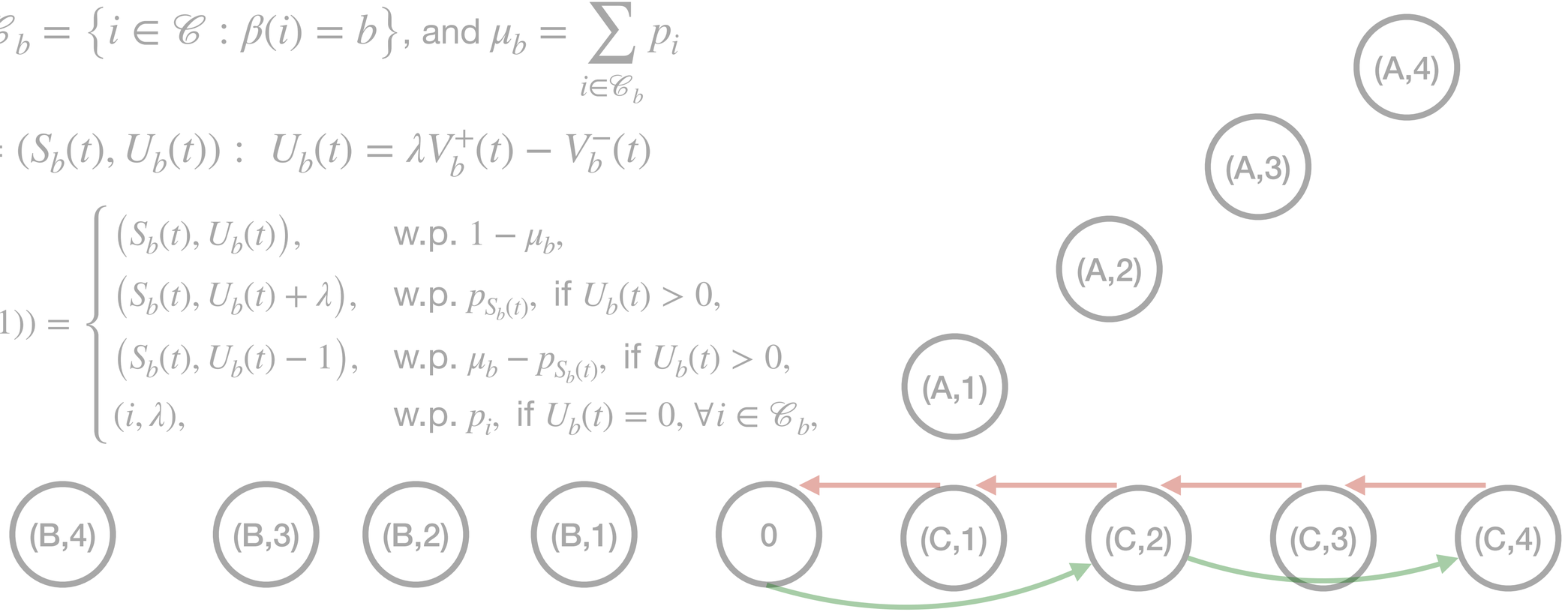
$$S_b(\infty) \begin{cases} \text{does not exist}, & \text{if } \lambda \le \lambda_b^{(1)} \\ \text{exists} & \text{otherwise} \end{cases}$$

Limiting behavior (Proof)

Collisions set $\mathscr{C}_b = \{i \in \mathscr{C} : \beta(i) = b\}$, and $\mu_b = \sum_{i \in \mathscr{C}_b} p_i$

Define $M_b(t) = (S_b(t), U_b(t)) : \; U_b(t) = \lambda V_b^+(t) - V_b^-(t)$

$$(S_b(t+1), U_b(t+1)) = \begin{cases} \big(S_b(t), U_b(t)\big), & \text{w.p. } 1 - \mu_b, \\ \big(S_b(t), U_b(t) + \lambda\big), & \text{w.p. } p_{S_b(t)}, \text{ if } U_b(t) > 0, \\ \big(S_b(t), U_b(t) - 1\big), & \text{w.p. } \mu_b - p_{S_b(t)}, \text{ if } U_b(t) > 0, \\ (i, \lambda), & \text{w.p. } p_i, \text{ if } U_b(t) = 0, \forall i \in \mathscr{C}_b, \end{cases}$$



Define $\lambda_i(\beta) = \dfrac{\mu_b}{p_i} - 1$ and $\lambda_b^{(1)} = \min_{i \in \mathscr{C}_b} \lambda_i(\beta)$

$$S_b(\infty) \begin{cases} \text{does not exist,} & \text{if } \lambda \le \lambda_b^{(1)} \\ \text{exists} & \text{otherwise} \end{cases} \Longrightarrow \Pr\left[S_b(\infty) = i\right] = a_{i,\beta}(\lambda) \Longrightarrow \lim_{\tau \to \infty} \frac{V_b^+(\tau)}{\tau} = \sum_{i \in \mathscr{C}_b} p_i \, 1_{\{S_b(\infty) = i\}}$$

Contributions

5.1 Limiting behavior

- 5.1.1 Heavy block
- 5.1.2 CM block and error

5.2 Near-Optimal Configuration

- 5.2.1 High probability upper bound on $\lambda^*(\beta)$
- 5.2.2 Efficient configuration

CM block and Error

Definition: $\mathrm{Err}_{(0)}(t) = \min_{l \in [d]} Y_{l,X_l}(t)$: (X_l) are i.i.d. uniform

Lemma: $\Pr\left(\mathrm{Err}_i(t) \geq x\right) \leq \Pr\left(\mathrm{Err}_{(0)}(t) \geq x\right)$

CM block and Error

Definition: $\mathrm{Err}_{(0)}(t) = \min_{l \in [d]} Y_{l,X_l}(t)$: (X_l) are i.i.d. uniform

Lemma: $\Pr\left(\mathrm{Err}_i(t) \geq x\right) \leq \Pr\left(\mathrm{Err}_{(0)}(t) \geq x\right)$

Definition: $\overline{\mathrm{Err}}_{(0)} = \lim_{\tau \to \infty} \frac{1}{\tau} \min_{l \in [d]} Y_{l,X_l}(\tau)$: (X_l) are i.i.d. uniform

Theorem 2: $\mathrm{E}\left[\overline{\mathrm{Err}}_0\right] = \frac{1}{m_2}\left(1 - \mathrm{E}[g_\beta(\lambda)]\right)$

Proxy for comparison in terms of λ and memory split between blocks

Contributions

5.1 Limiting behavior

- 5.1.1 Heavy block
- 5.1.2 CM block and error

5.2 Near-Optimal Configuration

- 5.2.1 High probability upper bound on $\lambda^*(\beta)$
- 5.2.2 Efficient configuration

Near-Optimal Configuration

Optimal configuration: $(\lambda^*, m_1^*, m_2^*) \in \operatorname{argmax}_{\lambda \in \mathbb{N}, \rho m_1 + m_2 \leq m} \frac{1}{m_2} \left(1 - \bar{g}(m_1, \lambda)\right)$

Near-Optimal Configuration

Optimal configuration: $(\lambda^*, m_1^*, m_2^*) \in \operatorname{argmax}_{\lambda \in \mathbb{N}, \rho m_1 + m_2 \leq m} \frac{1}{m_2} \left(1 - \bar{g}(m_1, \lambda)\right)$

Definition: $\lambda^*(\beta) = \operatorname{argmax}_{\lambda \in \mathbb{N}} g_\beta(\lambda)$

Definition: $\lambda_{m_1}^* \triangleq \operatorname{argmax}_{\lambda \in \mathbb{N}} \bar{g}(m_1, \lambda)$

Near-Optimal Configuration (High probability bound)

Theorem 3: If $m_1 \to \infty$ and $n_{\mathscr{C}} >> m_1$, w.h.p.,

$$\lambda^*(\beta) \leq \frac{n_{\mathscr{C}}}{m_1} + \mathcal{O}\left(\sqrt{\frac{n_{\mathscr{C}} \ln m_1}{m_1}}\right),$$

With equality under the uniform distribution

Near-Optimal Configuration (Efficient configuration)

Theorem 4 [Candidates set]: $\lambda^*_{m_1} \in \Lambda(m_1)$

Table 1. Number of distinct candidate values ($|\Lambda(m_1)|$): $n_{\text{samp}} = 10^3$, $n_{\mathcal{I}} = 10^4$.

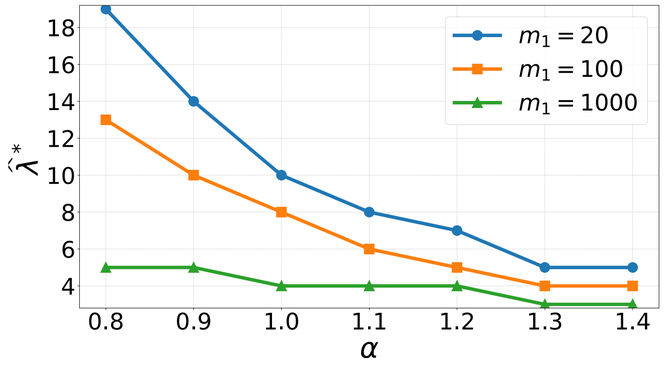
α	m_1						
	20	100	200	500	1000	2000	5000
0.8	55	35	27	18	13	9	8
1.0	38	29	24	16	12	9	7
1.2	26	24	21	15	11	9	7

Near-Optimal Configuration (Efficient configuration)

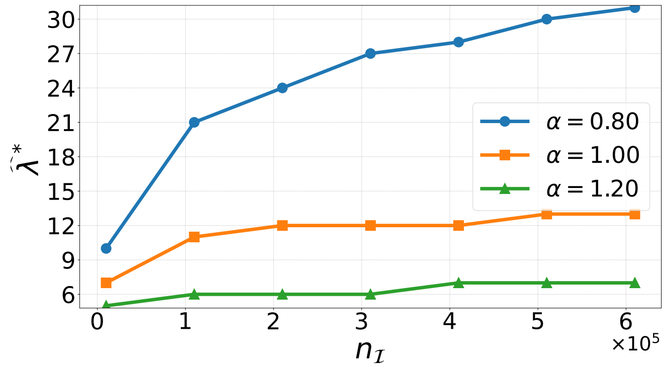
Theorem 4 [Candidates set]: $\lambda^*_{m_1} \in \Lambda(m_1)$

Table 1. Number of distinct candidate values ($|\Lambda(m_1)|$): $n_{\text{samp}} = 10^3$, $n_{\mathcal{I}} = 10^4$.

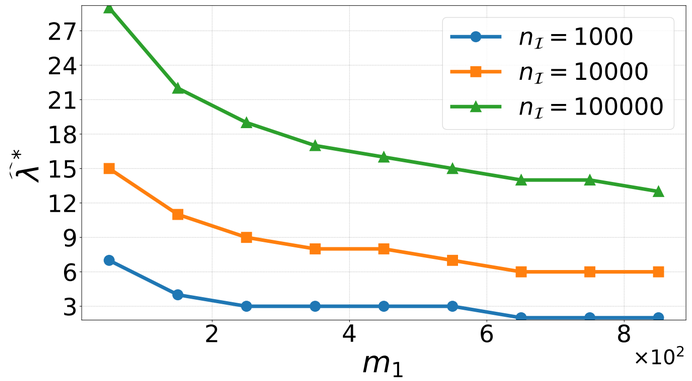
α	m_1						
	20	100	200	500	1000	2000	5000
0.8	55	35	27	18	13	9	8
1.0	38	29	24	16	12	9	7
1.2	26	24	21	15	11	9	7



$n_{\mathscr{C}} = 10^4$



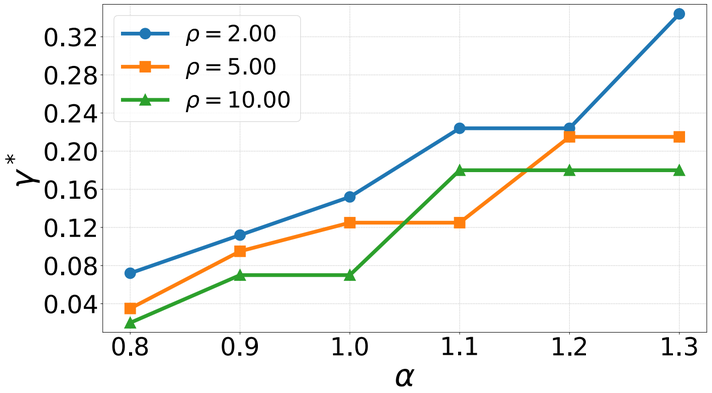
$m_1 = 200$



$\alpha = 0.8$

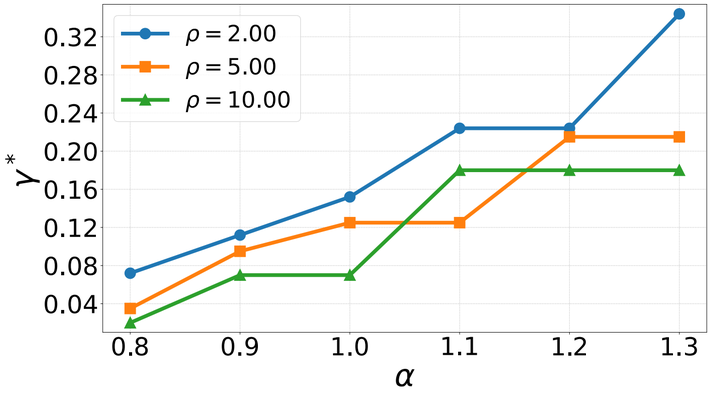
Near-Optimal Configuration (Efficient configuration)

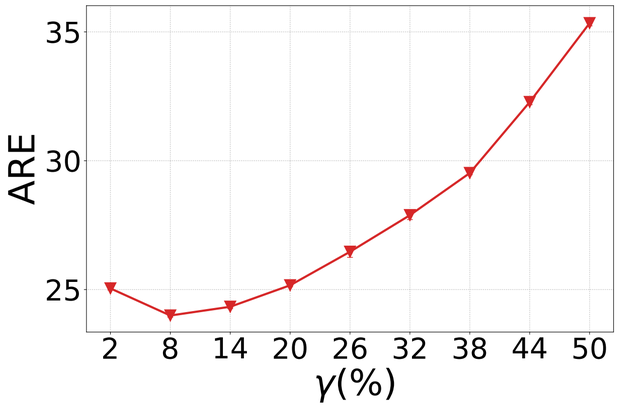
$m = 1000, n_{\mathscr{C}} = 10^4, 50$ samples



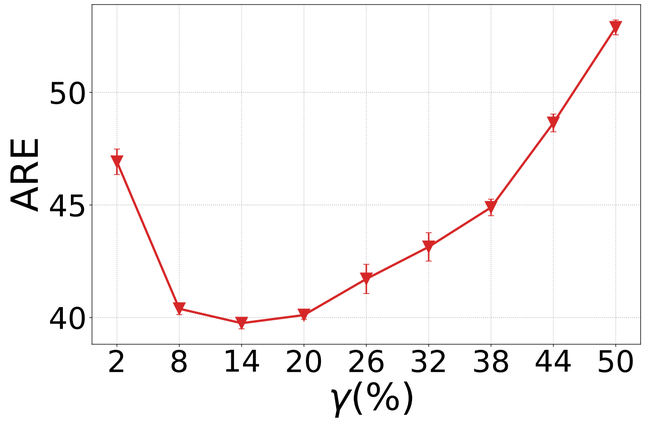
Near-Optimal Configuration (Efficient configuration)

$m = 1000, n_{\mathscr{C}} = 10^4, 50$ samples

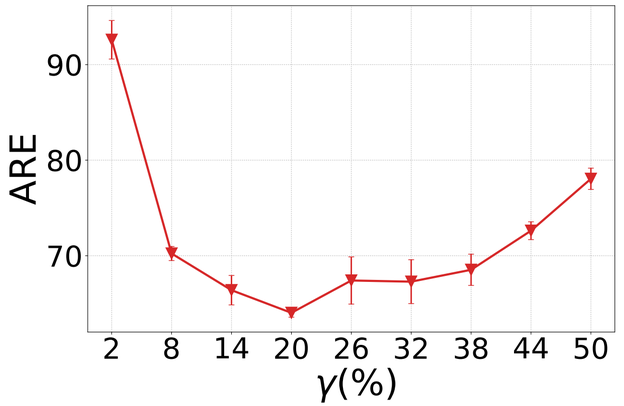




$\alpha = 0.8$



$\alpha = 1.0$



$\alpha = 1.2$

$\lambda = 5, \rho = 2$ (5 maps x 5 streams)

Future work

How does $\lambda^*(\beta)$, and the optimal memory split depend on Zipf(α)?

How can predictions be integrated in Elastic sketch?

Thank you