Balanced Allocations with Heterogeneous Bins: The Power of Memory

Dimitrios Los\textsuperscript{1}, Thomas Sauerwald\textsuperscript{1}, John Sylvester\textsuperscript{2}

\textsuperscript{1}University of Cambridge, UK, \textsuperscript{2}University of Liverpool, UK
Balanced allocations: Background
Balanced allocations setting

Allocate $m$ tasks (balls) sequentially into $n$ machines (bins).
Balanced allocations setting

Allocate $m$ tasks (balls) sequentially into $n$ machines (bins).

**Goal:** minimise the maximum load $\max_{i \in [n]} x_i^m$, where $x^t$ is the load vector after ball $t$. 


click here to view the full page

Balanced allocations: Background
Balanced allocations setting

Allocate $m$ tasks (balls) sequentially into $n$ machines (bins).

**Goal:** minimise the maximum load $\max_{i \in [n]} x_i^m$, where $x^t$ is the load vector after ball $t$. 
Balanced allocations setting

Allocate $m$ tasks (balls) sequentially into $n$ machines (bins).

**Goal:** minimise the maximum load $\max_{i \in [n]} x_i^m$, where $x^t$ is the load vector after ball $t$. 
Balanced allocations setting

Allocate \( m \) tasks (balls) sequentially into \( n \) machines (bins).

**Goal:** minimise the maximum load \( \max_{i \in [n]} x_i^m \), where \( x^t \) is the load vector after ball \( t \).

\[ \Leftrightarrow \text{minimise the gap, where } \text{Gap}(m) = \max_{i \in [n]} (x_i^m - m/n). \]
Balanced allocations setting

Allocate \( m \) tasks (balls) sequentially into \( n \) machines (bins).

**Goal:** minimise the maximum load \( \max_{i \in [n]} x^m_i \), where \( x^t \) is the load vector after ball \( t \).

\[ \Leftrightarrow \text{minimise the gap, where } \text{Gap}(m) = \max_{i \in [n]}(x^m_i - m/n). \]
Balanced allocations setting

Allocate $m$ tasks (balls) sequentially into $n$ machines (bins).

**Goal:** minimise the maximum load $\max_{i \in [n]} x_i^m$, where $x^t$ is the load vector after ball $t$.

$\Leftrightarrow$ minimise the gap, where $\text{Gap}(m) = \max_{i \in [n]} (x_i^m - m/n)$.

- Applications in hashing [PR01], load balancing [Wie16] and routing [GKK88].
One-Choice and Two-Choice processes

One-Choice Process:
Iteration: For each $t \geq 0$, sample one bin uniformly at random (u.a.r.) and place the ball there.

Two-Choice Process:
Iteration: For each $t \geq 0$, sample two bins independently u.a.r. and place the ball in the least loaded of the two.

In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \Theta(\log n \log \log n)$ [Gon81].

In the heavily-loaded case ($m \gg n$), w.h.p. $\text{Gap}(m) = \Theta(\sqrt{mn} \log n)$ (e.g. [RS98]).

In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \log_2 \log n + \Theta(1)$ [KLMadH96, ABKU99].

In the heavily-loaded case ($m \gg n$), w.h.p. $\text{Gap}(m) = \log_2 \log n + \Theta(1)$ [BCSV06].
One-Choice and Two-Choice processes

**One-Choice Process:**
Iteration: For each $t \geq 0$, sample one bin uniformly at random (u.a.r.) and place the ball there.

- In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \Theta\left(\frac{\log n}{\log \log n}\right)$ [Gon81].

- In the heavily-loaded case ($m \gg n$), w.h.p. $\text{Gap}(m) = \Theta\left(\sqrt{mn \cdot \log n}\right)$ (e.g. [RS98]).

**Two-Choice Process:**
Iteration: For each $t \geq 0$, sample two bins independently u.a.r. and place the ball in the least loaded of the two.

- In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \log_2 \log n + \Theta(1)$ [KLMadH96, ABKU99].

- In the heavily-loaded case ($m \gg n$), w.h.p. $\text{Gap}(m) = \log_2 \log n + \Theta(1)$ [BCSV06].
**One-Choice and Two-Choice processes**

**One-Choice Process:**

*Iteration:* For each $t \geq 0$, sample one bin uniformly at random (u.a.r.) and place the ball there.

- In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \Theta\left(\frac{\log n}{\log \log n}\right)$ [Gon81].

Meaning with probability at least $1 - n^{-c}$ for constant $c > 0$.

**Two-Choice Process:**

*Iteration:* For each $t \geq 0$, sample two bins independently u.a.r. and place the ball in the least loaded of the two.

- In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \log_2 \log n + \Theta(1)$ [KLMadH96, ABKU99].

- In the heavily-loaded case ($m \gg n$), w.h.p. $\text{Gap}(m) = \log_2 \log n + \Theta(1)$ [BCSV06].
**One-Choice and Two-Choice processes**

**One-Choice Process:**

**Iteration:** For each $t \geq 0$, sample one bin uniformly at random (u.a.r.) and place the ball there.

- In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \Theta\left(\frac{\log n}{\log \log n}\right)$ [Gon81].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\text{Gap}(m) = \Theta\left(\sqrt{\frac{m}{n}} \cdot \log n\right)$ (e.g. [RS98]).
**One-Choice and Two-Choice processes**

**One-Choice Process:**
*Iteration:* For each $t \geq 0$, sample one bin uniformly at random (u.a.r.) and place the ball there.

- In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \Theta\left(\frac{\log n}{\log \log n}\right)$ [Gon81].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\text{Gap}(m) = \Theta\left(\sqrt{\frac{m}{n}} \cdot \log n\right)$ (e.g. [RS98]).

**Two-Choice Process:**
*Iteration:* For each $t \geq 0$, sample two bins independently u.a.r. and place the ball in the least loaded of the two.

- In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \log_2 \log n + \Theta(1)$ [KLM96, ABKU99].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\text{Gap}(m) = \log_2 \log n + \Theta(1)$ [BCSV06].

Balanced allocations: Background
**One-Choice and Two-Choice processes**

**One-Choice Process:**

**Iteration:** For each $t \geq 0$, sample one bin uniformly at random (u.a.r.) and place the ball there.

- In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \Theta \left( \frac{\log n}{\log \log n} \right)$ [Gon81].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\text{Gap}(m) = \Theta \left( \sqrt{\frac{m}{n} \cdot \log n} \right)$ (e.g. [RS98]).

**Two-Choice Process:**

**Iteration:** For each $t \geq 0$, sample two bins independently u.a.r. and place the ball in the least loaded of the two.

- In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \log_2 \log n + \Theta(1)$ [KLMadH96, ABKU99].

---

Balanced allocations: Background
**One-Choice and Two-Choice processes**

**One-Choice Process:**

Iteration: For each $t \geq 0$, sample one bin uniformly at random (u.a.r.) and place the ball there.

- In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \Theta\left(\frac{\log n}{\log \log n}\right)$ [Gon81].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\text{Gap}(m) = \Theta\left(\sqrt{\frac{m}{n}} \cdot \log n\right)$ (e.g. [RS98]).

**Two-Choice Process:**

Iteration: For each $t \geq 0$, sample two bins independently u.a.r. and place the ball in the least loaded of the two.

- In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \log \log n + \Theta(1)$ [KLMadH96, ABKU99].

Balanced allocations: Background
One-Choice and Two-Choice processes

**One-Choice Process:**

*Iteration:* For each $t \geq 0$, sample one bin uniformly at random (u.a.r.) and place the ball there.

- In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \Theta\left(\frac{\log n}{\log \log n}\right)$ [Gon81].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\text{Gap}(m) = \Theta\left(\sqrt{\frac{m}{n} \cdot \log n}\right)$ (e.g. [RS98]).

**Two-Choice Process:**

*Iteration:* For each $t \geq 0$, sample two bins independently u.a.r. and place the ball in the least loaded of the two.

- In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \log_2 \log n + \Theta(1)$ [KLMadH96, ABKU99].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\text{Gap}(m) = \log_2 \log n + \Theta(1)$ [BCSV06].
**One-Choice and Two-Choice processes**

**One-Choice Process:**
Iteration: For each $t \geq 0$, sample one bin uniformly at random (u.a.r.) and place the ball there.

- In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \Theta\left(\frac{\log n}{\log \log n}\right)$ [Gon81].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\text{Gap}(m) = \Theta\left(\sqrt{\frac{m}{n}} \cdot \log n\right)$ (e.g. [RS98]).

**Two-Choice Process:**
Iteration: For each $t \geq 0$, sample two bins independently u.a.r. and place the ball in the least loaded of the two.

- In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \log_2 \log n + \Theta(1)$ [KLMadH96, ABKU99].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\text{Gap}(m) = \log_2 \log n + \Theta(1)$ [BCSV06].

Balanced allocations: Background
**One-Choice and d-Choice processes**

***One-Choice Process:***

**Iteration:** For each $t \geq 0$, sample one bin uniformly at random (u.a.r.) and place the ball there.

- In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \Theta\left(\frac{\log n}{\log \log n}\right)$ [Gon81].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\text{Gap}(m) = \Theta\left(\sqrt{\frac{m}{n}} \cdot \log n\right)$ (e.g. [RS98]).

***d-Choice Process:***

**Iteration:** For each $t \geq 0$, sample $d$ bins independently u.a.r. and place the ball in the least loaded of the two.

- In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \log_d \log n + \Theta(1)$ [KLMadH96, ABKU99].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\text{Gap}(m) = \log_d \log n + \Theta(1)$ [BCSV06].
Power of two choices: Visualisation

Open visualiser

Gap for \( n = 10^4 \)

Balanced allocations: Background
The Memory process

Initialization: Set the cache $c_0 = 1$.

Iteration: For each step $t \geq 0$:
- Sample bins $i_1, \ldots, i_d$ uniformly at random.
- Allocate to bin $j = \text{argmin } k \in \{c_t, i_1, \ldots, i_d\} x_t^k$.
- Update the cache to $c_{t+1} = \text{argmin } k \in \{c_t, i_1, \ldots, i_d\} x_{t+1}^k$.

Balanced allocations: Background
Several different variants of $d$-CHOICE have been studied: $(1 + \beta)$ \cite{PTW15}, \textsc{Thinning} \cite{FGGL21}.

**The MEMORY process**

Initialization: Set the cache $c_0 = 1$.

Iteration: For each step $t \geq 0$:

1. Sample bins $i_1, \ldots, i_d$ uniformly at random.
2. Allocate to bin $j = \arg\min_{k \in \{c_t, i_1, \ldots, i_d\}} x_t k$.
3. Update the cache to $c_{t+1} = \arg\min_{k \in \{c_t, i_1, \ldots, i_d\}} x_{t+1} k$. 

Balanced allocations: Background
The Memory process

- Several different variants of $d$-CHOICE have been studied: $(1 + \beta)$ [PTW15], \textsc{Thinning} [FGGL21].
- Shah and Prabhakar [SP02] introduced a variant of $d$-CHOICE maintaining $M$ cached bins.

Memory Process ($M = 1$):
- Initialization: Set the cache $c_0 = 1$.
- Iteration: For each step $t \geq 0$:
  - Sample bins $i_1, \ldots, i_d$ uniformly at random.
  - Allocate to bin $j = \arg\min_k \{ c_t, i_1, \ldots, i_d \} x_t^k$.
  - Update the cache to $c_{t+1} = \arg\min_k \{ c_t, i_1, \ldots, i_d \} x_{t+1}^k$.
The Memory process

- Several different variants of $d$-CHOICE have been studied: $(1 + \beta)$ [PTW15], Thinning [FGGL21].
- Shah and Prabhakar [SP02] introduced a variant of $d$-CHOICE maintaining $M$ cached bins.

**Memory Process ($M = 1$):**

**Initialization:** Set the cache $c^0 = 1$.

**Iteration:** For each step $t \geq 0$:
- Sample bins $i_1, \ldots, i_d$ uniformly at random.
- Allocate to bin $j = \arg\min_{k \in \{c^t, i_1, \ldots, i_d\}} x_k^t$.
- Update the cache to $c^{t+1} = \arg\min_{k \in \{c^t, i_1, \ldots, i_d\}} x_k^{t+1}$. 

Balanced allocations: Background
In the lightly-loaded case, Memory with $d = 1$ w.h.p. achieves an $O(\log \log n)$ gap [MPS02]. For general $d \geq 1$, the bound becomes $\log f(d) \log n + \Theta(1)$ for $f(d) \in (2^d, 2^d + 1)$.

What happens in the heavily-loaded case ($m \geq n$)?

Balanced allocations: Background
In the lightly-loaded case, Memory with $d = 1$ w.h.p. achieves an $O(\log \log n)$ gap [MPS02]. For general $d \geq 1$, the bound becomes $\log f(d) \log n + \Theta(1)$ for $f(d) \in (2^d, 2^d + 1)$.

What happens in the heavily-loaded case ($m \geq n$)?
The Memory process

In the lightly-loaded case, Memory with $d = 1$ w.h.p. achieves an $O(\log \log n)$ gap \cite{MPS02}. For general $d \geq 1$, the bound becomes $\log f(d) \log n + \Theta(1)$ for $f(d) \in (2^d, 2^d + 1)$.

What happens in the heavily-loaded case ($m \geq n$)?

Balanced allocations: Background
The Memory process

In the lightly-loaded case, Memory with $d = 1$ w.h.p. achieves an $O(\log \log n)$ gap [MPS02].

For general $d \geq 1$, the bound becomes $\log f(d) \log n + \Theta(1)$ for $f(d) \in (2^d, 2^d + 1)$.

What happens in the heavily-loaded case ($m \geq n$)?
The Memory process

In the lightly-loaded case, Memory with $d = 1$ w.h.p. achieves an $O(\log \log n)$ gap [MPS02].

For general $d \geq 1$, the bound becomes $\log f(d) \log n + \Theta(1)$ for $f(d) \in (2^d, 2^d + 1)$.

What happens in the heavily-loaded case ($m \geq n$)?

Balanced allocations: Background
The Memory process

In the lightly-loaded case, Memory with $d = 1$ w.h.p. achieves an $O(\log \log n)$ gap [MPS02]. For general $d \geq 1$, the bound becomes $\log f(d) \log n + \Theta(1)$ for $f(d) \in (2^d, 2^d + 1)$.

What happens in the heavily-loaded case ($m \geq n$)?

Balanced allocations: Background
The Memory process

In the lightly-loaded case, Memory with $d = 1$ w.h.p. achieves an $O(\log \log n)$ gap [MPS02].

For general $d \geq 1$, the bound becomes $\log f(d) \log n + \Theta(1)$ for $f(d) \in (2^d, 2^d + 1)$.

What happens in the heavily-loaded case ($m \geq n$)?
In the lightly-loaded case, MEMORY with \( d = 1 \) w.h.p. achieves an \( \mathcal{O}(\log \log n) \) gap [MPS02].
In the lightly-loaded case, MEMORY with $d = 1$ w.h.p. achieves an $O(\log \log n)$ gap [MPS02].

For general $d \geq 1$, the bound becomes $\log_{f(d)} \log n + \Theta(1)$ for $f(d) \in (2d, 2d + 1)$. 
The Memory process

- In the lightly-loaded case, MEMORY with $d = 1 \text{ w.h.p.}$ achieves an $\mathcal{O}(\log \log n)$ gap [MPS02].
- For general $d \geq 1$, the bound becomes $\log f(d) \log n + \Theta(1)$ for $f(d) \in (2d, 2d + 1)$.

What happens in the heavily-loaded case ($m \geq n$)?
Heterogeneous sampling distributions

Several different settings for $d$-Choice:
- Outdated information [BCE+12],
- Graphical [BK22],
- Adversarial noise [LS22b],...

Wieder [Wie07] studied $d$-Choice with non-uniform sampling distributions.

In particular, $a,b > 1$, Wieder showed that there exists $d' > 0$, such that for any $\epsilon > 0$:
- For any $d \geq (1 + \epsilon) \cdot d'$, then $d$-Choice w.h.p. achieves $\text{Gap}(m) = \tilde{O}(\log \log n)$.
- For any $d \leq (1 - \epsilon) \cdot d'$, then $d$-Choice has a gap that grows with $m$.

How does Memory deal with heterogeneous sampling distributions?
Heterogeneous sampling distributions

- Several different settings for $d$-Choice:
Heterogeneous sampling distributions

- Several different settings for $d$-CHOICE: outdated information [BCE+12],

- Wieder [Wie07] studied $d$-Choice with non-uniform sampling distributions.

- In particular, $(a, b)$-biased sampling distributions satisfy $a/n \leq s \leq b/n$.

- Given $a, b > 1$, Wieder showed that there exists $d' > 0$, such that for any $\epsilon > 0$:
  - For any $d \geq (1 + \epsilon) \cdot d'$, then $d$-Choice w.h.p. achieves $\text{Gap}(m) = O(\log \log n)$.
  - For any $d \leq (1 - \epsilon) \cdot d'$, then $d$-Choice has a gap that grows with $m$. 

How does Memory deal with heterogeneous sampling distributions?
Heterogeneous sampling distributions

- Several different settings for \textit{d-Choice}: outdated information [BCE$^+$12], graphical [BK22],

\begin{itemize}
  \item In particular, \((a, b)\) -biased sampling distributions satisfy \(a_n \leq s_i \leq b_n\).
  \item Given \(a, b > 1\), Wieder showed that there exists \(d' > 0\), such that for any \(\epsilon > 0\):
    \begin{itemize}
      \item For any \(d \geq (1 + \epsilon) \cdot d'\), then \textit{d-Choice} w.h.p. achieves \(\text{Gap}(m) = O(\log \log n)\).
      \item For any \(d \leq (1 - \epsilon) \cdot d'\), then \textit{d-Choice} has a gap that grows with \(m\).
    \end{itemize}
\end{itemize}

How does Memory deal with heterogeneous sampling distributions?
Heterogeneous sampling distributions

- Several different settings for \textit{d-CHOICE}: outdated information [BCE$^+12$], graphical [BK22], adversarial noise [LS22b],
Heterogeneous sampling distributions

Several different settings for \textit{d-Choice}: outdated information \cite{BCE12}, graphical \cite{BK22}, adversarial noise \cite{LS22b}, ...
Heterogeneous sampling distributions

- Several different settings for \textit{d-Choice}: outdated information [BCE$^+$12], graphical [BK22], adversarial noise [LS22b], ...
- Wieder [Wie07] studied \textit{d-Choice} with non-uniform sampling distributions.
Heterogeneous sampling distributions

- Several different settings for \(d\)-Choice: outdated information [BCE\(^+\)12], graphical [BK22], adversarial noise [LS22b], ...
- Wieder [Wie07] studied \(d\)-Choice with non-uniform sampling distributions.

\[ \frac{1}{n} \]

Balanced allocations: Background
Heterogeneous sampling distributions

- Several different settings for $d$-CHOICE: outdated information [BCE$^+$12], graphical [BK22], adversarial noise [LS22b], ....
- Wieder [Wie07] studied $d$-CHOICE with non-uniform sampling distributions.

In particular, $(a, b)$-biased sampling distributions satisfy $\frac{1}{an} \leq s_i \leq \frac{1}{n}$.

Given $a, b > 1$, Wieder showed that there exists $d' > 0$, such that for any $\epsilon > 0$:

- For any $d \geq (1 + \epsilon) \cdot d'$, then $d$-Choice w.h.p. achieves $\text{Gap}(m) = O(\log \log n)$.
- For any $d \leq (1 - \epsilon) \cdot d'$, then $d$-Choice has a gap that grows with $m$.

How does Memory deal with heterogeneous sampling distributions?
Heterogeneous sampling distributions

- Several different settings for \textit{d-Choice}: outdated information \cite{BCE+12}, graphical \cite{BK22}, adversarial noise \cite{LS22b}, ...
- Wieder \cite{Wie07} studied \textit{d-Choice} with non-uniform sampling distributions.

- In particular, \((a, b)\)-biased sampling distributions \(s\) satisfy \(\frac{1}{an} \leq s_i \leq \frac{b}{n}\).
Heterogeneous sampling distributions

- Several different settings for $d$-CHOICE: outdated information [BCE$^+$12], graphical [BK22], adversarial noise [LS22b], ....
- Wieder [Wie07] studied $d$-CHOICE with non-uniform sampling distributions.

- In particular, $(a,b)$-biased sampling distributions $s$ satisfy $\frac{1}{an} \leq s_i \leq \frac{b}{n}$.
- Given $a,b > 1$, Wieder showed that there exists $d' > 0$, such that for any $\epsilon > 0$: 

$$\text{For any } d \geq (1 + \epsilon) \cdot d', \text{ then } d\text{-Choice w.h.p. achieves } \text{Gap}(m) = O(\log \log n).$$

$$\text{For any } d \leq (1 - \epsilon) \cdot d', \text{ then } d\text{-Choice has a gap that grows with } m.$$
Heterogeneous sampling distributions

- Several different settings for $d$-CHOICE: outdated information [BCE+12], graphical [BK22], adversarial noise [LS22b], ....
- Wieder [Wie07] studied $d$-CHOICE with non-uniform sampling distributions.

In particular, $(a, b)$-biased sampling distributions satisfy $\frac{1}{an} \leq s_i \leq \frac{b}{n}$.

Given $a, b > 1$, Wieder showed that there exists $d' > 0$, such that for any $\epsilon > 0$:
  - For any $d \geq (1 + \epsilon) \cdot d'$, then $d$-CHOICE w.h.p. achieves $\text{Gap}(m) = O(\log \log n)$. 

How does Memory deal with heterogeneous sampling distributions?
Heterogeneous sampling distributions

- Several different settings for \textit{d-CHOICE}: outdated information \[BCE^+12\], graphical \[BK22\], adversarial noise \[LS22b\], ....
- Wieder \[Wie07\] studied \textit{d-CHOICE} with non-uniform sampling distributions.

\begin{itemize}
  \item In particular, \((a,b)\)-biased sampling distributions \(s\) satisfy \(\frac{1}{an} \leq s_i \leq \frac{b}{n}\).
  \item Given \(a,b > 1\), Wieder showed that there exists \(d' > 0\), such that for any \(\epsilon > 0\):
    \begin{itemize}
      \item For any \(d \geq (1 + \epsilon) \cdot d'\), then \textit{d-CHOICE} w.h.p. achieves \(\text{Gap}(m) = \mathcal{O}(\log \log n)\).
      \item For any \(d \leq (1 - \epsilon) \cdot d'\), then \textit{d-CHOICE} has a gap that grows with \(m\).
    \end{itemize}
\end{itemize}
Heterogeneous sampling distributions

- Several different settings for \(d\text{-CHOICE}:\) outdated information [BCE\(^+\)12], graphical [BK22], adversarial noise [LS22b], ...
- Wieder [Wie07] studied \(d\text{-CHOICE}\) with non-uniform sampling distributions.

In particular, \((a, b)\)-biased sampling distributions \(s\) satisfy \(\frac{1}{an} \leq s_i \leq \frac{b}{n}\).

Given \(a, b > 1\), Wieder showed that there exists \(d' > 0\), such that for any \(\epsilon > 0\):
- For any \(d \geq (1 + \epsilon) \cdot d'\), then \(d\text{-CHOICE}\) w.h.p. achieves \(\text{Gap}(m) = \mathcal{O}(\log \log n)\).
- For any \(d \leq (1 - \epsilon) \cdot d'\), then \(d\text{-CHOICE}\) has a gap that grows with \(m\).

How does MEMORY deal with heterogeneous sampling distributions?
Our results

In the heavily-loaded case ($m \geq n$), [LSS22] proved that $\text{Memory}(d = M = 1)$ achieves w.h.p. $O(\log n)$.

We improve this to $\text{Gap}(m) = O(\log \log n)$.

Further, we show that w.h.p. $\text{Gap}(m) = \Omega(\log \log n)$ for any $m \geq n$.

For $(a,b)$-biased distributions with any $a,b > 1$, w.h.p. $\text{Gap}(m) = O(\log \log n)$.

In contrast to Two-Choice, where the gap grows with $m$, for $a = b = 2$.

For any $a = a(n)$ and $b = b(n)$, the gap is independent of $m$.

Challenges: (i) long-term dependencies due to cache and (ii) biased sampling.

$d$-Reset-Memory, a variant of Memory where the cache resets every $d$ steps has w.h.p. $\text{Gap}(m) = O(\log n)$, even in the presence of weights.
Our results

- In the heavily-loaded case \( m \geq n \), [LSS22] proved that MEMORY (with \( d = M = 1 \)) achieves w.h.p. \( \mathcal{O}(\log n) \).
Our results

- In the heavily-loaded case ($m \geq n$), [LSS22] proved that MEMORY (with $d = M = 1$) achieves w.h.p. $\mathcal{O}(\log n)$. We improve this to $\text{Gap}(m) = \mathcal{O}(\log \log n)$.

- Further, we show that w.h.p. $\text{Gap}(m) = \Omega(\log \log n)$ for any $m \geq n$.

- For $(a,b)$-biased distributions with any constant $a, b > 1$, w.h.p. $\text{Gap}(m) = \mathcal{O}(\log \log n)$.

- In contrast to Two-Choice, where the gap grows with $m$, for $a = b = 2$.

- For any $a := a(n)$ and $b := b(n)$, the gap is independent of $m$.

Challenges:

(i) long-term dependencies due to cache and
(ii) biased sampling.

- $d$-Reset-Memory, a variant of MEMORY where the cache resets every $d$ steps has w.h.p. $\text{Gap}(m) = \mathcal{O}(\log n)$, even in the presence of weights.
Our results

- In the heavily-loaded case ($m \geq n$), [LSS22] proved that MEMORY (with $d = M = 1$) achieves w.h.p. $O(\log n)$. We improve this to $\text{Gap}(m) = O(\log \log n)$.

- Further, we show that w.h.p. $\text{Gap}(m) = \Omega(\log \log n)$ for any $m \geq n$.

For $(a,b)$-biased distributions with any const $a, b > 1$, w.h.p. $\text{Gap}(m) = O(\log \log n)$.

In contrast to Two-Choice, where the gap grows with $m$, for $a = b = 2$.

For any $a := a(n)$ and $b := b(n)$, the gap is independent of $m$.

Challenges:

(i) long-term dependencies due to cache
(ii) biased sampling.

$d$-Reset-Memory, a variant of MEMORY where the cache resets every $d$ steps has w.h.p. $\text{Gap}(m) = O(\log n)$, even in the presence of weights.
Our results

- In the heavily-loaded case \( m \geq n \), [LSS22] proved that MEMORY (with \( d = M = 1 \)) achieves w.h.p. \( \mathcal{O}(\log n) \). We improve this to \( \text{Gap}(m) = \mathcal{O}(\log \log n) \).

- Further, we show that w.h.p. \( \text{Gap}(m) = \Omega(\log \log n) \) for any \( m \geq n \).

- For \((a, b)\)-biased distributions with any const \( a, b > 1 \), w.h.p. \( \text{Gap}(m) = \mathcal{O}(\log \log n) \).
Our results

- In the heavily-loaded case \((m \geq n)\), [LSS22] proved that MEMORY (with \(d = M = 1\)) achieves w.h.p. \(\mathcal{O}(\log n)\). We improve this to \(\text{Gap}(m) = \mathcal{O}(\log \log n)\).

- Further, we show that w.h.p. \(\text{Gap}(m) = \Omega(\log \log n)\) for any \(m \geq n\).

- For \((a, b)\)-biased distributions with any const \(a, b > 1\), w.h.p. \(\text{Gap}(m) = \mathcal{O}(\log \log n)\).

  \(\Leftarrow\) In contrast to TWO-CHOICE, where the gap grows with \(m\), for \(a = b = 2\).
Our results

- In the heavily-loaded case \( m \geq n \), [LSS22] proved that MEMORY (with \( d = M = 1 \)) achieves w.h.p. \( \mathcal{O}(\log n) \). We improve this to \( \text{Gap}(m) = \mathcal{O}(\log \log n) \).

- Further, we show that w.h.p. \( \text{Gap}(m) = \Omega(\log \log n) \) for any \( m \geq n \).

- For \((a,b)\)-biased distributions with any const \( a, b > 1 \), w.h.p. \( \text{Gap}(m) = \mathcal{O}(\log \log n) \). 
  \( \Leftarrow \) In contrast to TWO-CHOICE, where the gap grows with \( m \), for \( a = b = 2 \).

- For any \( a := a(n) \) and \( b := b(n) \), the gap is independent of \( m \).
Our results

- In the heavily-loaded case \((m \geq n)\), [LSS22] proved that MEMORY (with \(d = M = 1\)) achieves w.h.p. \(O(\log n)\). We improve this to \(\text{Gap}(m) = O(\log \log n)\).

- Further, we show that w.h.p. \(\text{Gap}(m) = \Omega(\log \log n)\) for any \(m \geq n\).

- For \((a, b)\)-biased distributions with any const \(a, b > 1\), w.h.p. \(\text{Gap}(m) = O(\log \log n)\).
  \(\Leftrightarrow\) In contrast to TWO-CHOICE, where the gap grows with \(m\), for \(a = b = 2\).

- For any \(a := a(n)\) and \(b := b(n)\), the gap is independent of \(m\).

Challenges:
Our results

■ In the heavily-loaded case \((m \geq n)\), [LSS22] proved that \textsc{Memory} (with \(d = M = 1\)) achieves w.h.p. \(O(\log n)\). We improve this to \(\text{Gap}(m) = O(\log \log n)\).

■ Further, we show that w.h.p. \(\text{Gap}(m) = \Omega(\log \log n)\) for any \(m \geq n\).

■ For \((a, b)\)-biased distributions with any const \(a, b > 1\), w.h.p. \(\text{Gap}(m) = O(\log \log n)\).

\(\iff\) In contrast to \textsc{Two-Choice}, where the gap grows with \(m\), for \(a = b = 2\).

■ For any \(a := a(n)\) and \(b := b(n)\), the gap is independent of \(m\).

\begin{itemize}
  \item Challenges: (i) long-term dependencies due to cache
\end{itemize}
Our results

- In the heavily-loaded case \((m \geq n)\), [LSS22] proved that MEMORY (with \(d = M = 1\)) achieves w.h.p. \(O(\log n)\). We improve this to \(\text{Gap}(m) = O(\log \log n)\).

- Further, we show that w.h.p. \(\text{Gap}(m) = \Omega(\log \log n)\) for any \(m \geq n\).

- For \((a, b)\)-biased distributions with any const \(a, b > 1\), w.h.p. \(\text{Gap}(m) = O(\log \log n)\).

- In contrast to TWO-CHOICE, where the gap grows with \(m\), for \(a = b = 2\).

- For any \(a := a(n)\) and \(b := b(n)\), the gap is independent of \(m\).

**Challenges:** (i) long-term dependencies due to cache and (ii) biased sampling.
Our results

- In the heavily-loaded case \((m \geq n)\), [LSS22] proved that \textsc{Memory} (with \(d = M = 1\)) achieves w.h.p. \(O(\log n)\). We improve this to \(\text{Gap}(m) = O(\log \log n)\).

- Further, we show that w.h.p. \(\text{Gap}(m) = \Omega(\log \log n)\) for any \(m \geq n\).

- For \((a, b)\)-biased distributions with any const \(a, b > 1\), w.h.p. \(\text{Gap}(m) = O(\log \log n)\).
  
  \(\Leftarrow\) In contrast to \textsc{Two-Choice}, where the gap grows with \(m\), for \(a = b = 2\).

- For any \(a := a(n)\) and \(b := b(n)\), the gap is independent of \(m\).

**Challenges:** (i) long-term dependencies due to cache and (ii) biased sampling.

- \textit{d-Reset-Memory}, a variant of \textsc{Memory} where the cache resets every \(d\) steps has w.h.p. \(\text{Gap}(m) = O(\log n)\)
Our results

- In the heavily-loaded case \((m \geq n)\), [LSS22] proved that MEMORY (with \(d = M = 1\)) achieves w.h.p. \(\mathcal{O}(\log n)\). We improve this to \(\text{Gap}(m) = \mathcal{O}(\log \log n)\).

- Further, we show that w.h.p. \(\text{Gap}(m) = \Omega(\log \log n)\) for any \(m \geq n\).

- For \((a, b)\)-biased distributions with any const \(a, b > 1\), w.h.p. \(\text{Gap}(m) = \mathcal{O}(\log \log n)\).

  \(\Leftrightarrow\) In contrast to TWO-CHOICE, where the gap grows with \(m\), for \(a = b = 2\).

- For any \(a := a(n)\) and \(b := b(n)\), the gap is independent of \(m\).

**Challenges:**

\((i)\) long-term dependencies due to cache and \((ii)\) biased sampling.

- **d-Reset-Memory**, a variant of MEMORY where the cache resets every \(d\) steps has w.h.p. \(\text{Gap}(m) = \mathcal{O}(\log n)\), even in the presence of weights.
Power of memory: Visualisation

Open visualiser
Gap for $n = 10^4$ for (2, 2)-biased sampling

“Power of memory”
Why \textbf{MEMORY} recovers?

\textbf{Two-Choice}: In two-choice, there is a set of bins that receives $\geq m/n$ balls in expectation.

\textbf{Memory}: W.h.p. we sample every bin roughly every $\log n$ steps.
Why Memory recovers?

In Two-Choice, there is a set of bins that receives $> m/n$ balls in expectation.
Why **MEMORY** recovers?

- In **Two-Choice**, there is a set of bins that receives $> m/n$ balls in expectation.
- In **MEMORY**, w.h.p. we sample every bin roughly every $an \log n$ steps.
Why **Memory** recovers?

- In **Two-Choice**, there is a set of bins that receives $> m/n$ balls in expectation.
- In **Memory**, w.h.p. we sample every bin roughly every $an \log n$ steps.
Upper Bound for MEMORY
Outline for the $\mathcal{O}(\log \log n)$ bound

Define the super-exponential potentials $\Phi_j$ for $0 \leq j = \mathcal{O}(\log \log n)$, where $z_j := n + j \cdot z$ for constants $z > 0$, $\alpha \in (0, 1)$ and $v > 1$.

When $\Phi_t j = \mathcal{O}(n)$, then $\text{Gap}(t) = \mathcal{O}(j \cdot z + \log n \alpha \cdot v_j)$. For $j = \Theta(\log \log n)$, when $\Phi_j = \mathcal{O}(n)$, then $\text{Gap}(m) = \Theta(\log \log n)$.

Further, when $\Phi_t j = \mathcal{O}(n)$, then also the number of bins with load at least $z_j + 1$ is at most $\mathcal{O}(n \cdot e^{-\alpha \cdot v_j \cdot z})$.

We group steps into rounds (at most $e^{v_j + 2 \cdot \log 3} n$ steps each) and show that $\mathbb{E}[\Phi_{r+1} j | F_r, \Phi_r j = \mathcal{O}(n)] \leq \Phi_r j + 1 \cdot (1 - e^{v_j + 2 n}) + e^{-v_j + 1 / 2}$.

The base case follows by an involved analysis of the hyperbolic cosine potential function [PTW15, LS22a].
Outline for the $O(\log \log n)$ bound

- Define the super-exponential potentials $\Phi_j$ for $0 \leq j = O(\log \log n)$,

$$
\Phi_j^t := \Phi_j^t(\alpha \cdot v^j, z_j) := \sum_{i: x_i^t \geq z_j} e^{\alpha \cdot v^j \cdot (x_i^t - z_j)},
$$

where $z_j := \frac{t}{n} + j \cdot z$ for constants $z > 0$, $\alpha \in (0, 1)$ and $v > 1$. 

---

Upper Bound for MEMORY
Outline for the $O(\log \log n)$ bound

Define the super-exponential potentials $\Phi_j$ for $0 \leq j = O(\log \log n)$,

$$
\Phi_j^t := \Phi_j^t(\alpha \cdot v^j, z_j) := \sum_{i: x_i^t \geq z_j} e^{\alpha \cdot v^j \cdot (x_i^t - z_j)},
$$

where $z_j := \frac{t}{n} + j \cdot z$ for constants $z > 0, \alpha \in (0, 1)$ and $v > 1$.

When $\Phi_j^t = O(n)$, then $\text{Gap}(t) = O(j \cdot z + \frac{\log n}{\alpha \cdot v^j})$. 
Outline for the $O(\log \log n)$ bound

- Define the **super-exponential potentials** $\Phi_j$ for $0 \leq j = O(\log \log n)$,
  \[
  \Phi_j(t) := \Phi_j^t(\alpha \cdot v^j, z_j) := \sum_{i: x_i^t \geq z_j} e^{\alpha \cdot v^j \cdot (x_i^t - z_j)},
  \]
  where $z_j := \frac{t}{n} + j \cdot z$ for constants $z > 0$, $\alpha \in (0, 1)$ and $v > 1$.
- When $\Phi_j = O(n)$, then $\text{Gap}(t) = O(j \cdot z + \frac{\log n}{\alpha \cdot v^j})$.
- For $j = \Theta(\log \log n)$, when $\Phi_j = O(n)$, then $\text{Gap}(m) = \Theta(\log \log n)$.
Outline for the $\mathcal{O}(\log \log n)$ bound

- Define the **super-exponential potentials** $\Phi_j$ for $0 \leq j = \mathcal{O}(\log \log n)$,
  \[
  \Phi_j^t := \Phi_j^t(\alpha \cdot v^j, z_j) := \sum_{i: x_i^t \geq z_j} e^{\alpha \cdot v^j (x_i^t - z_j)},
  \]
  where $z_j := \frac{t}{n} + j \cdot z$ for constants $z > 0$, $\alpha \in (0, 1)$ and $v > 1$.

- When $\Phi_j^t = \mathcal{O}(n)$, then $\text{Gap}(t) = \mathcal{O}(j \cdot z + \frac{\log n}{\alpha \cdot v^j})$.

- For $j = \Theta(\log \log n)$, when $\Phi_j = \mathcal{O}(n)$, then $\text{Gap}(m) = \Theta(\log \log n)$.

- Further, when $\Phi_j^t = \mathcal{O}(n)$, then also number of bins with load at least $z_{j+1}$ is at most $\mathcal{O}(n \cdot e^{-\alpha \cdot v^j \cdot z})$. 

---

**Upper Bound for Memory**

14
Outline for the $\mathcal{O}(\log \log n)$ bound

- Define the **super-exponential potentials** $\Phi_j$ for $0 \leq j = \mathcal{O}(\log \log n)$,
  \[
  \Phi^t_j := \Phi^t_j(\alpha \cdot v^j, z_j) := \sum_{i : x_i^t \geq z_j} e^{\alpha \cdot v^j \cdot (x_i^t - z_j)},
  \]
  where $z_j := \frac{t}{n} + j \cdot z$ for constants $z > 0$, $\alpha \in (0, 1)$ and $v > 1$.

- When $\Phi^t_j = \mathcal{O}(n)$, then $\text{Gap}(t) = \mathcal{O}(j \cdot z + \frac{\log n}{\alpha \cdot v^j})$.

- For $j = \Theta(\log \log n)$, when $\Phi_j = \mathcal{O}(n)$, then $\text{Gap}(m) = \Theta(\log \log n)$.

- Further, when $\Phi^t_j = \mathcal{O}(n)$, then also number of bins with load at least $z_{j+1}$ is at most $\mathcal{O}(n \cdot e^{-\alpha \cdot v^j \cdot z})$.

- We group steps into rounds (at most $e^{v^j + 2} \cdot \log^3 n$ steps each) and show that
  \[
  \mathbb{E} \left[ \Phi^r_{j+1} \mid \mathcal{I}^r, \Phi^r_j = \mathcal{O}(n) \right] \leq \Phi^r_{j+1} \cdot \left(1 - \frac{e^{v^j + 2}}{n} \right) + e^{-v^j + 1} / 2.
  \]
Outline for the $O(\log \log n)$ bound

- Define the super-exponential potentials $\Phi_j$ for $0 \leq j = O(\log \log n)$,
  \[
  \Phi^t_j := \Phi^t_j(\alpha \cdot v^j, z_j) := \sum_{i: x^t_i \geq z_j} e^{\alpha \cdot v^j \cdot (x^t_i - z_j)},
  \]
  where $z_j := \frac{t}{n} + j \cdot z$ for constants $z > 0$, $\alpha \in (0, 1)$ and $v > 1$.

- When $\Phi^t_j = O(n)$, then $\text{Gap}(t) = O(j \cdot z + \log n / \alpha \cdot v^j)$.

- For $j = \Theta(\log \log n)$, when $\Phi_j = O(n)$, then $\text{Gap}(m) = \Theta(\log \log n)$.

- Further, when $\Phi^t_j = O(n)$, then also number of bins with load at least $z_{j+1}$ is at most $O(n \cdot e^{-\alpha \cdot v^j \cdot z})$.

- We group steps into rounds (at most $e^{v^j+2} \cdot \log^3 n$ steps each) and show that
  \[
  \mathbb{E} \left[ \Phi^r_{j+1} \mid \mathcal{F}^r, \Phi^r_j = O(n) \right] \leq \Phi^r_{j+1} \cdot \left(1 - \frac{e^{v^j+2}}{n}\right) + e^{-v^j+1/2}.
  \]

- The base case follows by an involved analysis of the hyperbolic cosine potential function [PTW15, LS22a].
Layered induction over super-exponential potentials

\[ \phi_0^t = \mathcal{O}(n) \]
Layered induction over super-exponential potentials

\[ y_t \]

\[ \log n \]

\[ \Phi_0(t) = O(n) \]
Layered induction over super-exponential potentials

\[ \log n \]

\[ \Phi_1^t = O(n) \]
Layered induction over super-exponential potentials

\[ y^t \]

\[ \log n \]

\[ z + \frac{(\log n)}{\nu} \]

\[ z \]

\[ 0 \]

\[ \Phi_1^t = \mathcal{O}(n) \]
Layered induction over super-exponential potentials

\[ y_t = O(n) \]

Upper Bound for Memory $\phi_z = O(n)$
Layered induction over super-exponential potentials

\[ y_t^0 \leq \frac{2z + (\log n)/v^2}{\frac{\log n}{v}} \leq \Phi^t_{\frac{1}{2}} = \mathcal{O}(n) \]
Layered induction over super-exponential potentials

$\Phi_j^t = O(n)$

Upper Bound for Memory
Layered induction over super-exponential potentials

$$\Phi_j^t = \mathcal{O}(n)$$
Conclusion
We have shown that:

- Memory with $d = M = 1$ has w.h.p. $\text{Gap}(m) = O(\log \log n)$.
- Same upper bound for $(a, b)$-biased sampling distributions with any constant $a, b > 1$.
- A matching lower bound holds for any $m \geq n$.
- $d$-Reset-Memory has w.h.p. $\text{Gap}(m) = O(\log n)$.

Several avenues for future work:

- What is the gap for the optimal caching strategy at step $m$?
- Are there any weighted settings where Memory is superior to $d$-Choice?
- Obtaining tight bounds for $(a, b)$-biased distributions for non-constant $a, b$.
- Obtaining tight bounds up to lower order terms (as in [MPS02]).
- Analyse Memory in settings with outdated or noisy information.
Summary & Future work

We have shown that:

- Memory with \( d = M = 1 \) has w.h.p. \( \text{Gap}(m) = O(\log \log n) \).
- Same upper bound for \((a,b)\)-biased sampling distributions with any const \( a,b > 1 \).
- A matching lower bound holds for any \( m \geq n \).
- \( d\)-Reset-Memory has w.h.p. \( \text{Gap}(m) = O(\log n) \).

Several avenues for future work:

- What is the gap for the optimal caching strategy at step \( m \)?
- Are there any weighted settings where Memory is superior to \( d\)-Choice?
- Obtaining tight bounds for \((a,b)\)-biased distributions for non-const \( a,b \).
- Obtaining tight bounds up to lower order terms (as in [MPS02]).
- Analyse Memory in settings with outdated or noisy information.
Summary & Future work

We have shown that:

- **MEMORY** with $d = M = 1$ has w.h.p. $\text{Gap}(m) = \mathcal{O}(\log \log n)$.

Several avenues for future work:

- What is the gap for the optimal caching strategy at step $m$?
- Are there any weighted settings where MEMORY is superior to $d$-Choice?
- Obtaining tight bounds for $(a,b)$-biased distributions for non-const $a,b$.
- Obtaining tight bounds up to lower order terms (as in [MPS02]).
- Analyse MEMORY in settings with outdated or noisy information.
Summary & Future work

We have shown that:

- **MEMORY** with \( d = M = 1 \) has w.h.p. \( \text{Gap}(m) = \mathcal{O}(\log \log n) \).
- Same upper bound for \((a, b)\)-biased sampling distributions with any const \( a, b > 1 \).

Several avenues for future work:

- What is the gap for the optimal caching strategy at step \( m \)?
- Are there any weighted settings where MEMORY is superior to \( d \)-Choice?
- Obtaining tight bounds for \((a, b)\)-biased distributions for non-const \( a, b \).
- Obtaining tight bounds up to lower order terms (as in [MPS02]).
- Analyse MEMORY in settings with outdated or noisy information.
Summary & Future work

We have shown that:

- **MEMORY** with $d = M = 1$ has w.h.p. $\text{Gap}(m) = \mathcal{O}(\log \log n)$.
- Same upper bound for $(a, b)$-biased sampling distributions with any const $a, b > 1$.
- A matching lower bound holds for any $m \geq n$.

Several avenues for future work:

- What is the gap for the optimal caching strategy at step $m$?
- Are there any weighted settings where **MEMORY** is superior to $d$-Choice?
- Obtaining tight bounds for $(a, b)$-biased distributions for non-const $a, b$.
- Obtaining tight bounds up to lower order terms (as in [MPS02]).
- Analyse **MEMORY** in settings with outdated or noisy information.
Summary & Future work

We have shown that:

- **MEMORY** with \( d = M = 1 \) has w.h.p. \( \text{Gap}(m) = \mathcal{O}(\log \log n) \).
- Same upper bound for \((a, b)\)-biased sampling distributions with any const \( a, b > 1 \).
- A matching lower bound holds for any \( m \geq n \).
- **d-RESET-MEMORY** has w.h.p. \( \text{Gap}(m) = \mathcal{O}(\log n) \).
Summary & Future work

We have shown that:

- **MEMORY** with $d = M = 1$ has w.h.p. $\text{Gap}(m) = \mathcal{O}(\log \log n)$.
- Same upper bound for $(a, b)$-biased sampling distributions with any const $a, b > 1$.
- A matching lower bound holds for any $m \geq n$.
- **$d$-RESET-MEMORY** has w.h.p. $\text{Gap}(m) = \mathcal{O}(\log n)$.

Several avenues for future work:

- What is the gap for the optimal caching strategy at step $m$?
- Are there any weighted settings where MEMORY is superior to $d$-Choice?
- Obtaining tight bounds for $(a, b)$-biased distributions for non-const $a, b$.
- Obtaining tight bounds up to lower order terms (as in [MPS02]).
- Analyse MEMORY in settings with outdated or noisy information.
Summary & Future work

We have shown that:

- **MEMORY** with $d = M = 1$ has w.h.p. $\text{Gap}(m) = \mathcal{O}(\log \log n)$.
- Same upper bound for $(a, b)$-biased sampling distributions with any const $a, b > 1$.
- A matching lower bound holds for any $m \geq n$.
- **$d$-Reset-Memory** has w.h.p. $\text{Gap}(m) = \mathcal{O}(\log n)$.

Several avenues for future work:

- What is the gap for the optimal caching strategy at step $m$?
Summary & Future work

We have shown that:

- **MEMORY** with $d = M = 1$ has w.h.p. $\text{Gap}(m) = \mathcal{O}(\log \log n)$.
- Same upper bound for $(a, b)$-biased sampling distributions with any const $a, b > 1$.
- A matching lower bound holds for any $m \geq n$.
- **d-RESET-MEMORY** has w.h.p. $\text{Gap}(m) = \mathcal{O}(\log n)$.

Several avenues for future work:

- What is the gap for the optimal caching strategy at step $m$?
- Are there any weighted settings where **MEMORY** is superior to **d-CHOICE**?
Summary & Future work

We have shown that:

- **MEMORY** with \(d = M = 1\) has w.h.p. \(\text{Gap}(m) = \mathcal{O}(\log \log n)\).
- Same upper bound for \((a, b)\)-biased sampling distributions with any const \(a, b > 1\).
- A matching lower bound holds for any \(m \geq n\).
- **d-Reset-Memory** has w.h.p. \(\text{Gap}(m) = \mathcal{O}(\log n)\).

Several avenues for future work:

- What is the gap for the optimal caching strategy at step \(m\)?
- Are there any weighted settings where **MEMORY** is superior to **d-Choice**?
- Obtaining tight bounds for \((a, b)\)-biased distributions for **non-const** \(a, b\).
Summary & Future work

We have shown that:

- **MEMORY** with $d = M = 1$ has w.h.p. $\text{Gap}(m) = \mathcal{O}(\log \log n)$.
- Same upper bound for $(a,b)$-biased sampling distributions with any const $a, b > 1$.
- A matching lower bound holds for any $m \geq n$.
- $d$-Reset-Memory has w.h.p. $\text{Gap}(m) = \mathcal{O}(\log n)$.

Several avenues for future work:

- What is the gap for the optimal caching strategy at step $m$?
- Are there any weighted settings where **MEMORY** is superior to $d$-CHOICE?
- Obtaining tight bounds for $(a,b)$-biased distributions for **non-const** $a, b$.
- Obtaining tight bounds up to lower order terms (as in [MPS02]).
Summary & Future work

We have shown that:

- **MEMORY** with \( d = M = 1 \) has w.h.p. \( \text{Gap}(m) = \mathcal{O}(\log \log n) \).
- Same upper bound for \((a, b)\)-biased sampling distributions with any const \( a, b > 1 \).
- A matching lower bound holds for any \( m \geq n \).
- **d-RESET-MEMORY** has w.h.p. \( \text{Gap}(m) = \mathcal{O}(\log n) \).

Several avenues for future work:

- What is the gap for the optimal caching strategy at step \( m \)?
- Are there any weighted settings where \( \text{MEMORY} \) is superior to \( d\)-CHOICE?
- Obtaining tight bounds for \((a, b)\)-biased distributions for non-const \( a, b \).
- Obtaining tight bounds up to lower order terms (as in [MPS02]).
- Analyse \( \text{MEMORY} \) in settings with outdated or noisy information.
Questions?

More visualisations: dimitrioslos.com/soda23
Probability allocation vectors

Some processes induce a probability allocation vector $p^t$, where $p_i^t$ gives the probability to allocate to the $i$-th most loaded bin.
Some processes induce a **probability allocation vector** \( p^t \), where \( p^t_i \) gives the probability to allocate to the \( i \)-th most loaded bin.

For **One-Choice**, \( p^t = \left( \frac{1}{n}, \ldots, \frac{1}{n} \right) \).
Probability allocation vectors

- Some processes induce a **probability allocation vector** $p^t$, where $p^t_i$ gives the probability to allocate to the $i$-th most loaded bin.
- For **One-Choice**, $p^t = \left( \frac{1}{n}, \ldots, \frac{1}{n} \right)$.
- For **Two-Choice**, 
  \[
  p^t = \left( \frac{1}{n^2}, \ldots, \frac{2i-1}{n^2}, \ldots, \frac{2n-1}{n^2} \right).
  \]
Probability allocation vectors

- Some processes induce a probability allocation vector $p^t$, where $p^t_i$ gives the probability to allocate to the $i$-th most loaded bin.
- For One-Choice, $p^t = \left( \frac{1}{n}, \ldots, \frac{1}{n} \right)$.
- For Two-Choice,
  \[
  p^t = \left( \frac{1}{n^2}, \ldots, \frac{2i-1}{n^2}, \ldots, \frac{2n-1}{n^2} \right).
  \]
- For Memory, if the cache is the $k$-th most loaded bin, then
  \[
  p^t = \left( 0, \ldots, 0, \frac{k}{n}, \frac{1}{n}, \ldots, \frac{1}{n} \right).
  \]
Probability allocation vectors

- Some processes induce a **probability allocation vector** \( p^t \), where \( p^t_i \) gives the probability to allocate to the \( i \)-th most loaded bin.

- For **One-Choice**, \( p^t = \left( \frac{1}{n}, \ldots, \frac{1}{n} \right) \).

- For **Two-Choice**, \( p^t = \left( \frac{1}{n^2}, \ldots, \frac{2i-1}{n^2}, \ldots, \frac{2n-1}{n^2} \right) \).

- For **Memory**, if the cache is the \( k \)-th most loaded bin, then

\[
p^t = \left( 0, \ldots, 0, \frac{k}{n}, \frac{1}{n}, \ldots, \frac{1}{n} \right)
\]

For \( k = 1 \), this is like **One-Choice**.
Probability allocation vectors

■ Some processes induce a probability allocation vector $p^t$, where $p^t_i$ gives the probability to allocate to the $i$-th most loaded bin.

■ For **One-Choice**, $p^t = \left( \frac{1}{n}, \ldots, \frac{1}{n} \right)$.

■ For **Two-Choice**, 
  
  $$p^t = \left( \frac{1}{n^2}, \ldots, \frac{2i-1}{n^2}, \ldots, \frac{2n-1}{n^2} \right).$$

■ For **Memory**, if the cache is the $k$-th most loaded bin, then
  
  $$p^t = \left( 0, \ldots, 0, \frac{k}{n}, \frac{1}{n}, \ldots, \frac{1}{n} \right).$$

■ A probability vector $p$ is $(\delta, \epsilon)$-smooth if majorized by
  
  $$\left( \frac{1 - \epsilon}{n}, \ldots, \frac{1 - \epsilon}{n}, \frac{1 + \tilde{\epsilon}}{n}, \ldots, \frac{1 + \tilde{\epsilon}}{n} \right).$$
Hyperbolic cosine potential

Peres, Talwar and Wieder [PTW15] used the hyperbolic cosine potential \( \Gamma_t \), defined as

\[
\Gamma_t := \Phi_t + \Psi_t = \sum_{i=1}^{n} e^{\alpha (x_t^i - t/n)} + \sum_{i=1}^{n} e^{-\alpha (x_t^i - t/n)}.
\]

When \( \Gamma_m = \text{poly}(n) \), then \( \text{Gap}(m) = O(\log n \alpha) \).

They showed that for any \((\delta, \epsilon)\)-smooth probability allocation vector \( p_t \),

\[
E[\Gamma_{t+1} | F_t] \leq \Gamma_t \cdot (1 - \alpha \epsilon n) + c.
\]

By induction, this implies that \( E[\Gamma_m] \leq c \alpha \epsilon \cdot n \).

And so, by Markov's inequality

\[
\Pr[\Gamma_m \leq c \alpha \epsilon \cdot n^3] \geq 1 - n^{-2}.
\]

Problem: \( p_t \) for Memory may not be \((\delta, \epsilon)\)-smooth.
Hyperbolic cosine potential

- Peres, Talwar and Wieder [PTW15] used the hyperbolic cosine potential $\Gamma^t$, defined as

$$\Gamma^t := \Phi^t + \Psi^t := \sum_{i=1}^{n} e^{\alpha(x_i^t - t/n)} + \sum_{i=1}^{n} e^{-\alpha(x_i^t - t/n)}.$$
Hyperbolic cosine potential

- Peres, Talwar and Wieder [PTW15] used the **hyperbolic cosine potential** $\Gamma^t$, defined as

$$\Gamma^t := \Phi^t + \Psi^t := \sum_{i=1}^{n} e^{\alpha(x_i^t - t/n)} + \sum_{i=1}^{n} e^{-\alpha(x_i^t - t/n)}.$$

- When $\Gamma^m = \text{poly}(n)$, then $\text{Gap}(m) = \mathcal{O}\left(\frac{\log n}{\alpha}\right)$. 

- Problem:

  $p_t$ for Memory may not be $(\delta, \epsilon)$-smooth.
Hyperbolic cosine potential

- Peres, Talwar and Wieder [PTW15] used the **hyperbolic cosine potential** $\Gamma^t$, defined as
  \[
  \Gamma^t := \Phi^t + \Psi^t := \sum_{i=1}^{n} e^{\alpha(x^t_i - t/n)} + \sum_{i=1}^{n} e^{-\alpha(x^t_i - t/n)}.
  \]

- When $\Gamma^m = \text{poly}(n)$, then $\text{Gap}(m) = O\left(\frac{\log n}{\alpha}\right)$.

- They showed that for any $(\delta, \epsilon)$-smooth probability allocation vector $p^t$,
  \[
  \mathbb{E}\left[ \Gamma^{t+1} \mid \delta^t \right] \leq \Gamma^t \cdot \left(1 - \frac{\alpha\epsilon}{n}\right) + c.
  \]
Peres, Talwar and Wieder [PTW15] used the hyperbolic cosine potential $\Gamma^t$, defined as

$$\Gamma^t := \Phi^t + \Psi^t := \sum_{i=1}^{n} e^{x_i - t/n} + \sum_{i=1}^{n} e^{-x_i / n}.$$

When $\Gamma^m = \text{poly}(n)$, then $\text{Gap}(m) = O\left(\frac{\log n}{\alpha}\right)$.

They showed that for any $(\delta, \epsilon)$-smooth probability allocation vector $p^t$,

$$\mathbb{E}\left[\Gamma^{t+1} \mid \mathbf{S}^t\right] \leq \Gamma^t \cdot \left(1 - \frac{\alpha \epsilon}{n}\right) + c.$$

By induction, this implies that $\mathbb{E}[\Gamma^m] \leq \frac{c}{\alpha \epsilon} \cdot n$. 
Hyperbolic cosine potential

- Peres, Talwar and Wieder [PTW15] used the hyperbolic cosine potential $\Gamma^t$, defined as

$$\Gamma^t := \Phi^t + \Psi^t := \sum_{i=1}^{n} e^{\alpha(x^t_i - t/n)} + \sum_{i=1}^{n} e^{-\alpha(x^t_i - t/n)}.$$

- When $\Gamma^m = \text{poly}(n)$, then $\text{Gap}(m) = \mathcal{O}\left(\frac{\log n}{\alpha}\right)$.

- They showed that for any $(\delta, \epsilon)$-smooth probability allocation vector $p^t$,

$$E[\Gamma^{t+1} | \tilde{\mathcal{G}}^t] \leq \Gamma^t \cdot \left(1 - \frac{\alpha \epsilon}{n}\right) + c.$$

- By induction, this implies that $E[\Gamma^m] \leq \frac{c}{\alpha \epsilon} \cdot n$.

- And so, by Markov’s inequality $\Pr[\Gamma^m \leq \frac{c}{\alpha \epsilon} \cdot n^3] \geq 1 - n^{-2}$. 
Peres, Talwar and Wieder [PTW15] used the **hyperbolic cosine potential** $\Gamma^t$, defined as

$$\Gamma^t := \Phi^t + \Psi^t := \sum_{i=1}^{n} e^{\alpha(x^t_i - t/n)} + \sum_{i=1}^{n} e^{-\alpha(x^t_i - t/n)}.$$

When $\Gamma^m = \text{poly}(n)$, then $\text{Gap}(m) = \mathcal{O}\left(\frac{\log n}{\alpha}\right)$.

They showed that for any $(\delta, \epsilon)$-smooth probability allocation vector $p^t$,

$$E\left[\Gamma^{t+1} \mid \tilde{\Omega}^t\right] \leq \Gamma^t \cdot \left(1 - \frac{\alpha\epsilon}{n}\right) + c.$$

By induction, this implies that $E[\Gamma^m] \leq \frac{c}{\alpha\epsilon} \cdot n$.

And so, by Markov’s inequality $\Pr[\Gamma^m \leq \frac{c}{\alpha\epsilon} \cdot n^3] \geq 1 - n^{-2}$.

**Problem:** $p^t$ for Memory may not be $(\delta, \epsilon)$-smooth
A generalised drift inequality [LS22a]
A generalised drift inequality [LS22a]

- If for some $(\delta, \epsilon)$-smooth probability vector $q$,

\[
\mathbb{E} \left[ \Phi^{t+1} \mid \mathcal{F}^t \right] \leq \Phi^t + \sum_{i=1}^{n} \Phi_i^t \cdot \left( q_i^t - \frac{1}{n} \right) \cdot \alpha + \Phi^t \cdot C \cdot \frac{\alpha^2}{n},
\]

\[
\mathbb{E} \left[ \Psi^{t+1} \mid \mathcal{F}^t \right] \leq \Psi^t + \sum_{i=1}^{n} \Psi_i^t \cdot \left( \frac{1}{n} - q_i^t \right) \cdot \alpha + \Psi^t \cdot C \cdot \frac{\alpha^2}{n}.
\]
A generalised drift inequality [LS22a]

If for some \((\delta, \epsilon)\)-smooth probability vector \(q\),
\[
\mathbb{E} \left[ \Phi^{t+1} \mid \tilde{s}^t \right] \leq \Phi^t + \sum_{i=1}^{n} \Phi_i^t \cdot \left(q_i^t - \frac{1}{n}\right) \cdot \alpha + \Phi^t \cdot C \cdot \frac{\alpha^2}{n},
\]
\[
\mathbb{E} \left[ \Psi^{t+1} \mid \tilde{s}^t \right] \leq \Psi^t + \sum_{i=1}^{n} \Psi_i^t \cdot \left(1 - q_i^t\right) \cdot \alpha + \Psi^t \cdot C \cdot \frac{\alpha^2}{n}.
\]

Then, for sufficiently small \(\alpha > 0\),
\[
\mathbb{E} \left[ \Gamma^{t+1} \mid \tilde{s}^t \right] \leq \Gamma^t \cdot \left(1 - \frac{\alpha \epsilon}{n}\right) + c.
\]
A generalised drift inequality [LS22a]

- If for some \((\delta, \epsilon)\)-smooth probability vector \(q\),

\[
\mathbb{E} \left[ \Phi^{t+1} \mid \mathcal{F}^t \right] \leq \Phi^t + \sum_{i=1}^{n} \Phi_i^t \cdot \left( q_i^t - \frac{1}{n} \right) \cdot \alpha + \Phi^t \cdot C \cdot \frac{\alpha^2}{n},
\]

Could be allocating more than one ball.

\[
\mathbb{E} \left[ \Psi^{t+1} \mid \mathcal{F}^t \right] \leq \Psi^t + \sum_{i=1}^{n} \Psi_i^t \cdot \left( \frac{1}{n} - q_i^t \right) \cdot \alpha + \Psi^t \cdot C \cdot \frac{\alpha^2}{n}.
\]

- Then, for sufficiently small \(\alpha > 0\),

\[
\mathbb{E} \left[ \Gamma^{t+1} \mid \mathcal{F}^t \right] \leq \Gamma^t \cdot \left( 1 - \frac{\alpha \epsilon}{n} \right) + c.
\]
A generalised drift inequality [LS22a]

- If for some \((\delta, \epsilon)\)-smooth probability vector \(q, \leftrightarrow\) not always the prob allocation vector,

\[
\mathbb{E} \left[ \Phi^{t+1} \mid \tilde{s}^t \right] \leq \Phi^t + \sum_{i=1}^{n} \Phi_i^t \cdot \left( q_i^t - \frac{1}{n} \right) \cdot \alpha + \Phi^t \cdot C \cdot \frac{\alpha^2}{n},
\]

\[
\mathbb{E} \left[ \Psi^{t+1} \mid \tilde{s}^t \right] \leq \Psi^t + \sum_{i=1}^{n} \Psi_i^t \cdot \left( \frac{1}{n} - q_i^t \right) \cdot \alpha + \Psi^t \cdot C \cdot \frac{\alpha^2}{n}.
\]

- Then, for sufficiently small \(\alpha > 0\),

\[
\mathbb{E} \left[ \Gamma^{t+1} \mid \tilde{s}^t \right] \leq \Gamma^t \cdot \left( 1 - \frac{\alpha \epsilon}{n} \right) + c.
\]
A generalised drift inequality [LS22a]

■ If for some \((\delta, \epsilon)\)-smooth probability vector \(q\), not always the prob allocation vector.

\[
\begin{align*}
\mathbb{E} \left[ \Phi^{t+1} \mid \tilde{\sigma}^t \right] & \leq \Phi^t + \sum_{i=1}^{n} \Phi_i^t \cdot \left( q_i^t - \frac{1}{n} \right) \cdot \alpha + \Phi^t \cdot C \cdot \frac{\alpha^2}{n}, \\
\mathbb{E} \left[ \Psi^{t+1} \mid \tilde{\sigma}^t \right] & \leq \Psi^t + \sum_{i=1}^{n} \Psi_i^t \cdot \left( \frac{1}{n} - q_i^t \right) \cdot \alpha + \Psi^t \cdot C \cdot \frac{\alpha^2}{n}.
\end{align*}
\]

■ Then, for sufficiently small \(\alpha > 0\),

\[
\mathbb{E} \left[ \Gamma^{t+1} \mid \tilde{\sigma}^t \right] \leq \Gamma^t \cdot \left( 1 - \frac{\alpha \epsilon}{n} \right) + c.
\]

■ For 2-Reset-Memory, \(q\) is the probability allocation vector of Two-Choice.
A generalised drift inequality [LS22a]

- If for some \((\delta, \epsilon)\)-smooth probability vector \(q\), \(\not\sim\) not always the prob allocation vector,

\[
\mathbb{E} \left[ \Phi^{t+1} \mid \mathcal{F}_t \right] \leq \Phi^t + \sum_{i=1}^{n} \Phi_i \cdot \left( q_i^t - \frac{1}{n} \right) \cdot \alpha + \Phi^t \cdot C \cdot \frac{\alpha^2}{n},
\]

\[
\mathbb{E} \left[ \Psi^{t+1} \mid \mathcal{F}_t \right] \leq \Psi^t + \sum_{i=1}^{n} \Psi_i \cdot \left( \frac{1}{n} - q_i^t \right) \cdot \alpha + \Psi^t \cdot C \cdot \frac{\alpha^2}{n}.
\]

- Then, for sufficiently small \(\alpha > 0\),

\[
\mathbb{E} \left[ \Gamma^{t+1} \mid \mathcal{F}_t \right] \leq \Gamma^t \cdot \left( 1 - \frac{\alpha \epsilon}{n} \right) + c.
\]

- For 2-Reset-Memory, \(q\) is the probability allocation vector of Two-Choice.
- which is \((1/4, 1/2)\)-smooth, implying an \(\mathcal{O}(\log n)\) gap for Memory.
Handling heterogeneous distributions

To analyze a heterogeneous sampling distribution \( s \leq \sum_{i} \leq b \), we make two further reductions:

- Cache resets every \( d \) steps. For sufficiently large \( d \), beats the \((a,b)\)-bias.
- Load comparisons are based on the last reset. \( f \) makes computation of \( q \) tractable.

Moving probabilities between bins with almost the same load, introduces a small additive term in the bound,

\[
\mathbb{E}[\Phi_{t+1}|F_t] \leq \Phi_t + n \sum_{i=1}^{\Phi_t} \Phi_t i \cdot \left( q_t i - \frac{1}{n} \right) \cdot \alpha + \Phi_t \cdot C \cdot \alpha^2 n + O(\Phi_t \cdot \alpha^2 n \cdot (2d^3b))
\]

since \( \Phi_t i - \Phi_t j \leq \Phi_t j \cdot (2\alpha d) \) and probability of selecting a bin twice is at most \( d^2 b n \).

Similarly for \( \Psi \).

So for sufficiently small \( \alpha := \alpha(d) > 0 \), \( \mathbb{E}[\Gamma_m] = O(n) \).

And so \( \text{Gap}(m) = O((\log n) / \alpha) \text{ gap} \).
Handling heterogeneous distributions

To analyze a heterogeneous sampling distribution \( s \left( \frac{1}{an} \leq s_i \leq \frac{b}{n} \right) \), we make two further reductions:

- Cache resets every \( d \) steps. For sufficiently large \( d \), beats the \((a,b)\)-bias.
- Load comparisons are based on the last reset. \( f \) makes computation of \( q \) tractable.

■ Moving probabilities between bins with almost the same load, introduces a small additive term in the bound,

\[
E \left[ \Phi_{t+1} \mid F_t \right] \leq \Phi_t + \sum_{i=1}^{\Phi_t} \Phi_t \cdot \alpha \cdot \left( q - \frac{1}{n} \right) \cdot \alpha + \Phi_t \cdot C \cdot \alpha^2 + O\left( \Phi_t \cdot \alpha^2 \cdot \left( 2^d \cdot b \right)^3 \right),
\]

since \( \Phi_t - \Phi_j \leq \Phi_j \cdot (2^\alpha d^3) \) and probability of selecting a bin twice is at most \( d^2 \cdot b^n \).

■ Similarly for \( \Psi \).

So for sufficiently small \( \alpha := \alpha(d) > 0 \),

\[
E \left[ \Gamma_m \right] = O\left( n \right).
\]

■ And so \( \text{Gap}(m) = O\left( \frac{\log n}{\alpha} \right) \) gap.
Handling heterogeneous distributions

- To analyze a heterogeneous sampling distribution \( s \) \( \left( \frac{1}{an} \leq s_i \leq \frac{b}{n} \right) \), we make two further reductions:
  - Cache resets every \( d \) steps.

  - Load comparisons are based on the last reset. \( f \) makes computation of \( q \) tractable.

- Moving probabilities between bins with almost the same load, introduces a small additive term in the bound, 
  \[ E \left[ \Phi_{t+1} \bigg| F_t \right] \leq \Phi_t + \sum_{i=1}^{n} \Phi_t^i \cdot \left( q_t^i - 1 \right) n \cdot \alpha + \Phi_t \cdot C \cdot \alpha^2 n + O \left( \Phi_t \cdot \alpha^2 n \cdot \left( 2^d b^3 \right) \right), \]

  - Since \( \Phi_t^i - \Phi_t^j \leq \Phi_t^j \cdot (2^\alpha d) \) and probability of selecting a bin twice is at most \( d^2 \cdot b n \).

  - Similarly for \( \Psi \).

- So for sufficiently small \( \alpha = \alpha(d) > 0 \), 
  \[ E \left[ \Gamma_m \right] = O \left( n \right). \]

- And so \( \text{Gap}(m) = O \left( \frac{\log(n)}{\alpha} \right) \) gap.
Handling heterogeneous distributions

To analyze a heterogeneous sampling distribution $s \left( \frac{1}{an} \leq s_i \leq \frac{b}{n} \right)$, we make two further reductions:

- Cache resets every $d$ steps. $\leftrightarrow$ for sufficiently large $d$, beats the $(a, b)$-bias.

- Moving probabilities between bins with almost the same load, introduces a small additive term in the bound,

$$E \left[ \Phi_{t+1} \mid |F_t| \right] \leq \Phi_t + \sum_{i=1}^{n} \Phi_t i \cdot \left( q_t i - 1 \right) n \cdot \alpha + \Phi_t \cdot C \cdot \alpha^2 n + O \left( \Phi_t \cdot \alpha^2 n \cdot \left( 2d^3 b \right) \right),$$

since $\Phi_t i - \Phi_t j \leq \Phi_t j \cdot \left( 2\alpha d \right)$ and probability of selecting a bin twice is at most $d^2 \cdot b n$.

- Similarly for $\Psi$.

So for sufficiently small $\alpha := \alpha (d) > 0$,

$$E \left[ \Gamma_m \right] = O \left( n \right).$$

And so $\text{Gap}(m) = O \left( \frac{\log n}{\alpha} \right)$.
Handling heterogeneous distributions

To analyze a heterogeneous sampling distribution \( s \) \((\frac{1}{an} \leq s_i \leq \frac{b}{n})\), we make two further reductions:

- Cache resets every \( d \) steps. \( \Leftrightarrow \) for sufficiently large \( d \), beats the \((a, b)\)-bias.
- Load comparisons are based on the last reset.
Handling heterogeneous distributions

To analyze a heterogeneous sampling distribution $s (\frac{1}{an} \leq s_i \leq \frac{b}{n})$, we make two further reductions:

- Cache resets every $d$ steps. $\iff$ for sufficiently large $d$, beats the $(a, b)$-bias.
- Load comparisons are based on the last reset. $\iff$ makes computation of $q$ tractable.
Handling heterogeneous distributions

- To analyze a heterogeneous sampling distribution \( s (\frac{1}{an} \leq s_i \leq \frac{b}{n}) \), we make two further reductions:
  - Cache resets every \( d \) steps. \( \iff \) for sufficiently large \( d \), beats the \((a,b)\)-bias.
  - Load comparisons are based on the last reset. \( \iff \) makes computation of \( q \) tractable.

- Moving probabilities between bins with almost the same load, introduces a small additive term in the bound,

\[
\mathbb{E} \left[ \Phi^{t+1} \mid \bar{\mathcal{S}}^t \right] \leq \Phi^t + \sum_{i=1}^{n} \Phi_i^t \cdot \left( q_i^t - \frac{1}{n} \right) \cdot \alpha + \Phi^t \cdot C \cdot \frac{\alpha^2}{n} + \mathcal{O} \left( \Phi^t \cdot \frac{\alpha^2}{n} \cdot (2d^3b) \right),
\]
Handling heterogeneous distributions

To analyze a heterogeneous sampling distribution $s \left( \frac{1}{an} \leq s_i \leq \frac{b}{n} \right)$, we make two further reductions:

- Cache resets every $d$ steps. ✗ for sufficiently large $d$, beats the $(a, b)$-bias.
- Load comparisons are based on the last reset. ✗ makes computation of $q$ tractable.

Moving probabilities between bins with almost the same load, introduces a small additive term in the bound,

$$
E \left[ \Phi^{t+1} \mid \mathcal{F}^t \right] \leq \Phi^t + \sum_{i=1}^{n} \Phi_i^t \cdot \left( q_i^t - \frac{1}{n} \right) \cdot \alpha + \Phi^t \cdot C \cdot \frac{\alpha^2}{n} + O \left( \Phi^t \cdot \frac{\alpha^2}{n} \cdot (2d^3 b) \right),
$$

since $\Phi_i^t - \Phi_j^t \leq \Phi_j^t \cdot (2\alpha d)$ and
Handling heterogeneous distributions

■ To analyze a heterogeneous sampling distribution \( s \) \((\frac{1}{an} \leq s_i \leq \frac{b}{n})\), we make two further reductions:
  - Cache resets every \( d \) steps. \( \leftrightarrow \) for sufficiently large \( d \), beats the \((a, b)\)-bias.
  - Load comparisons are based on the last reset. \( \leftrightarrow \) makes computation of \( q \) tractable.

■ Moving probabilities between bins with almost the same load, introduces a small additive term in the bound,

\[
E[\Phi^{t+1} | \bar{g}^t] \leq \Phi^t + \sum_{i=1}^{n} \Phi_i^t \cdot \left( q_i^t - \frac{1}{n} \right) \cdot \alpha + \Phi^t \cdot C \cdot \frac{\alpha^2}{n} + O\left( \Phi^t \cdot \frac{\alpha^2}{n} \cdot (2d^3b) \right),
\]

since \( \Phi_i^t - \Phi_j^t \leq \Phi_j^t \cdot (2\alpha d) \) and probability of selecting a bin twice is at most \( d^2 \cdot \frac{b}{n} \).
Handling heterogeneous distributions

To analyze a heterogeneous sampling distribution $s$ ($\frac{1}{an} \leq s_i \leq \frac{b}{n}$), we make two further reductions:

- Cache resets every $d$ steps. $\Leftrightarrow$ for sufficiently large $d$, beats the $(a,b)$-bias.
- Load comparisons are based on the last reset. $\Leftrightarrow$ makes computation of $q$ tractable.

Moving probabilities between bins with almost the same load, introduces a small additive term in the bound,

$$E \left[ \Phi^{t+1} \mid \mathcal{G}^t \right] \leq \Phi^t + \sum_{i=1}^{n} \Phi^t_i \cdot \left( q^t_i - \frac{1}{n} \right) \cdot \alpha + \Phi^t \cdot C \cdot \frac{\alpha^2}{n} + O \left( \Phi^t \cdot \frac{\alpha^2}{n} \cdot (2d^3b) \right),$$

since $\Phi^t_i - \Phi^t_j \leq \Phi^t_j \cdot (2\alpha d)$ and probability of selecting a bin twice is at most $d^2 \cdot \frac{b}{n}$.

Similarly for $\Psi$. 

Handling heterogeneous distributions

To analyze a heterogeneous sampling distribution $s \ (\frac{1}{an} \leq s_i \leq \frac{b}{n})$, we make two further reductions:

- Cache resets every $d$ steps. $\Leftrightarrow$ for sufficiently large $d$, beats the $(a, b)$-bias.
- Load comparisons are based on the last reset. $\Leftrightarrow$ makes computation of $q$ tractable.

Moving probabilities between bins with almost the same load, introduces a small additive term in the bound,

$$E \left[ \Phi^{t+1} \mid \mathcal{S}^t \right] \leq \Phi^t + \sum_{i=1}^{n} \Phi_i^t \cdot \left( q_i^t - \frac{1}{n} \right) \cdot \alpha + \Phi^t \cdot C \cdot \frac{\alpha^2}{n} + O \left( \Phi^t \cdot \frac{\alpha^2}{n} \cdot (2d^3b) \right),$$

since $\Phi_i^t - \Phi_j^t \leq \Phi_j^t \cdot (2\alpha d)$ and probability of selecting a bin twice is at most $d^2 \cdot \frac{b}{n}$.

Similarly for $\Psi$. So for sufficiently small $\alpha := \alpha(d) > 0$, $E[\Gamma^m] = O(n)$. 
Handling heterogeneous distributions

- To analyze a heterogeneous sampling distribution $s \left( \frac{1}{an} \leq s_i \leq \frac{b}{n} \right)$, we make two further reductions:
  - Cache resets every $d$ steps. ⇐ for sufficiently large $d$, beats the $(a, b)$-bias.
  - Load comparisons are based on the last reset. ⇐ makes computation of $q$ tractable.

- Moving probabilities between bins with almost the same load, introduces a small additive term in the bound,

$$
E \left[ \Phi^{t+1} | \Phi^t \right] \leq \Phi^t + \sum_{i=1}^{n} \Phi_i^t \cdot \left( q_i^t - \frac{1}{n} \right) \cdot \alpha + \Phi^t \cdot C \cdot \frac{\alpha^2}{n} + O \left( \Phi^t \cdot \frac{\alpha^2}{n} \cdot (2d^3b) \right),
$$

since $\Phi_i^t - \Phi_j^t \leq \Phi_j^t \cdot (2\alpha d)$ and probability of selecting a bin twice is at most $d^2 \cdot \frac{b}{n}$.

- Similarly for $\Psi$. So for sufficiently small $\alpha := \alpha(d) > 0$, $E \left[ \Gamma^m \right] = O(n)$.

- And so $\text{Gap}(m) = O((\log n)/\alpha)$ gap.
Bibliography I


Bibliography II


Bibliography III


