Balanced Allocations with Heterogeneous Bins: The Power of Memory

Dimitrios Los[1], Thomas Sauerwald[1], John Sylvester[2]

[1]University of Cambridge, UK, [2]University of Liverpool, UK

Balanced allocations: Background

Balanced allocations setting

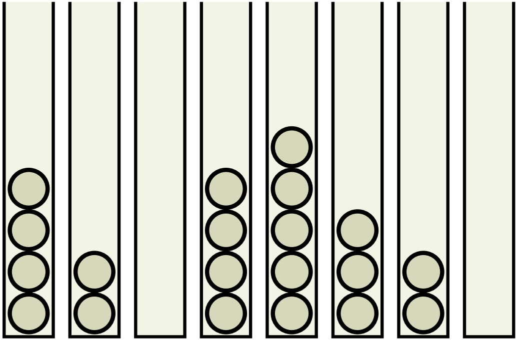
Allocate m tasks (balls) sequentially into n machines (bins).

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.

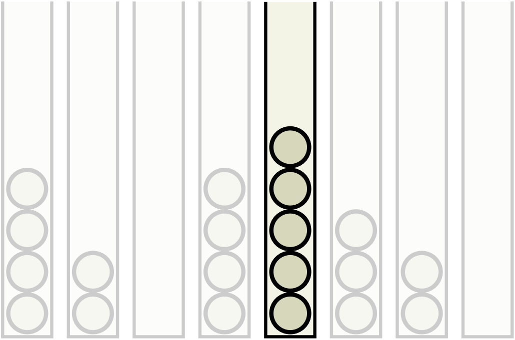
Balanced allocations setting

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.

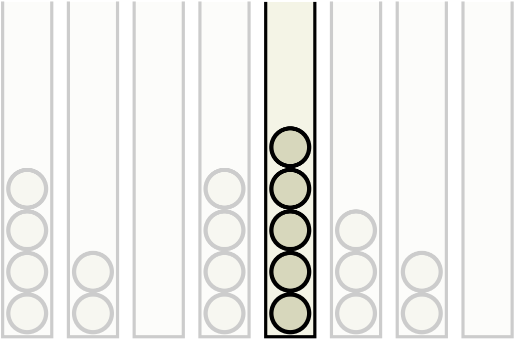


Balanced allocations setting

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

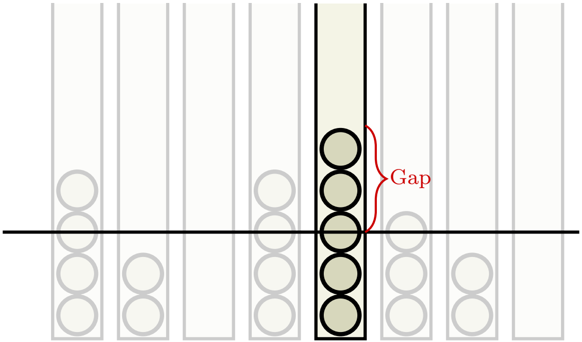
Goal: minimise the **maximum load** $\max_{i \in [n]} x_i^m$, where x^t is the load vector after ball t.

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



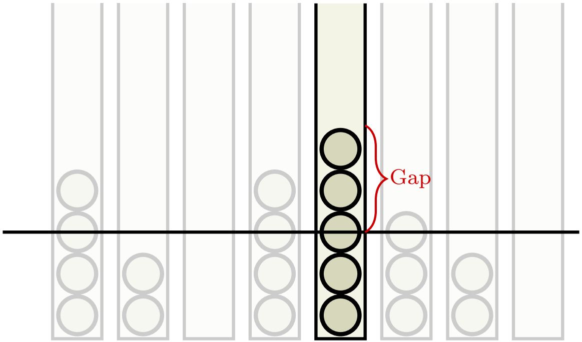
Balanced allocations setting

Allocate m tasks (balls) sequentially into n machines (bins).
<u>**Goal:**</u> 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 $\mathrm{Gap}(m) = \max_{i\in[n]}(x_i^m - m/n)$.



Balanced allocations setting

Allocate m tasks (balls) sequentially into n machines (bins).
Goal: minimise the **maximum load** $\max_{i \in [n]} x_i^m$, where x^t is the load vector after ball t.
$\Leftrightarrow$ minimise the **gap**, where $\mathrm{Gap}(m) = \max_{i \in [n]}(x_i^m - m/n)$.



- Applications in hashing [PR01], load balancing [Wie16] and routing [GKK88].

One-Choice and Two-Choice processes

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

One-Choice and Two-Choice processes

<u>One-Choice Process:</u>
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. $\mathrm{Gap}(n) = \Theta\left(\frac{\log n}{\log \log n}\right)$ [Gon81].

One-Choice and Two-Choice processes

<u>One-Choice Process:</u>
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 $c > 0$.

One-Choice and Two-Choice processes

<u>One-Choice Process:</u>
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. $\mathrm{Gap}(n) = \Theta\Big(\frac{\log n}{\log\log n}\Big)$ [Gon81].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\mathrm{Gap}(m) = \Theta\Big(\sqrt{\frac{m}{n} \cdot \log n}\Big)$ (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. $\mathrm{Gap}(n) = \Theta\left(\frac{\log n}{\log\log n}\right)$ [Gon81].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\mathrm{Gap}(m) = \Theta\left(\sqrt{\frac{m}{n} \cdot \log n}\right)$ (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.

One-Choice and Two-Choice processes

<u>One-Choice Process:</u>
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. $\mathrm{Gap}(n) = \Theta\left(\frac{\log n}{\log\log n}\right)$ [Gon81].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\mathrm{Gap}(m) = \Theta\left(\sqrt{\frac{m}{n} \cdot \log n}\right)$ (e.g. [RS98]).

<u>Two-Choice Process:</u>
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. $\mathrm{Gap}(n) = \log_2 \log n + \Theta(1)$ [KLMadH96, ABKU99].

One-Choice and Two-Choice processes

<u>One-Choice Process:</u>
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. $\mathrm{Gap}(n) = \Theta\left(\frac{\log n}{\log \log n}\right)$ [Gon81].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\mathrm{Gap}(m) = \Theta\left(\sqrt{\frac{m}{n} \cdot \log n}\right)$ (e.g. [RS98]).

<u>Two-Choice Process:</u>
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. $\mathrm{Gap}(n) = \log_2 \log n + \Theta(1)$ [KLMadH96, ABKU99].

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. $\mathrm{Gap}(n) = \Theta\left(\frac{\log n}{\log\log n}\right)$ [Gon81].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\mathrm{Gap}(m) = \Theta\left(\sqrt{\frac{m}{n} \cdot \log n}\right)$ (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. $\mathrm{Gap}(n) = \log_2 \log n + \Theta(1)$ [KLMadH96, ABKU99].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\mathrm{Gap}(m) = \log_2 \log n + \Theta(1)$ [BCSV06].

One-Choice and Two-Choice processes

<u>One-Choice Process:</u>
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. $\mathrm{Gap}(n) = \Theta\left(\frac{\log n}{\log\log n}\right)$ [Gon81].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\mathrm{Gap}(m) = \Theta\left(\sqrt{\frac{m}{n} \cdot \log n}\right)$ (e.g. [RS98]).

<u>Two-Choice Process:</u>
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. $\mathrm{Gap}(n) = \log_2 \log n + \Theta(1)$ [KLMadH96, ABKU99].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\mathrm{Gap}(m) = \log_2 \log n + \Theta(1)$ [BCSV06].

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.

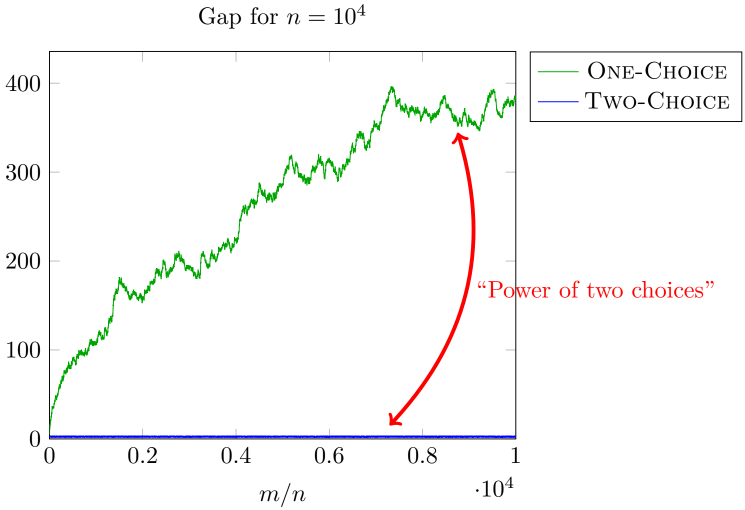
- In the lightly-loaded case ($m = n$), w.h.p. $\text{Gap}(n) = \Theta\left(\frac{\log n}{\log \log n}\right)$ [Gon81].
- In the heavily-loaded case ($m \gg n$), w.h.p. $\text{Gap}(m) = \Theta\left(\sqrt{\frac{m}{n} \cdot \log n}\right)$ (e.g. [RS98]).

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`



The MEMORY process

The MEMORY process

- Several different variants of d-CHOICE have been studied: $(1 + \beta)$ [PTW15], THINNING [FGGL21].

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.

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.

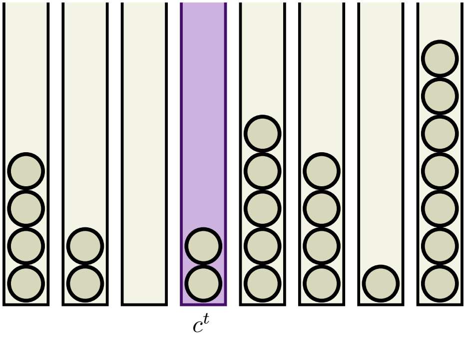
MEMORY Process ($M = 1$):

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

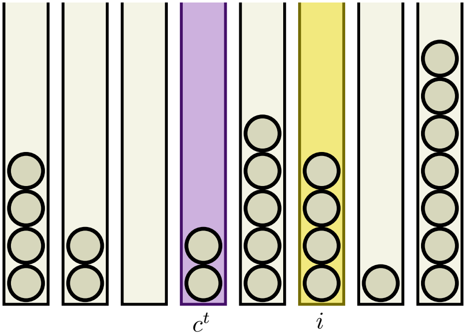
Iteration: For each step $t \geq 0$:

- Sample bins $i_1, \ldots, i_d$ uniformly at random.
- Allocate to bin $j = \operatorname{argmin}_{k \in \{c^t, i_1, \ldots, i_d\}} x_k^t$.
- Update the cache to $c^{t+1} = \operatorname{argmin}_{k \in \{c^t, i_1, \ldots, i_d\}} x_k^{t+1}$.

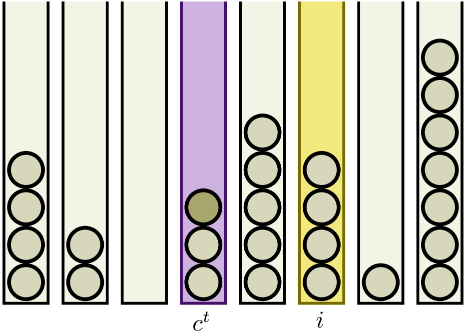
The MEMORY process



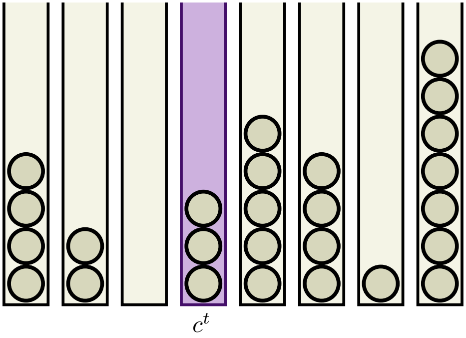
The MEMORY process



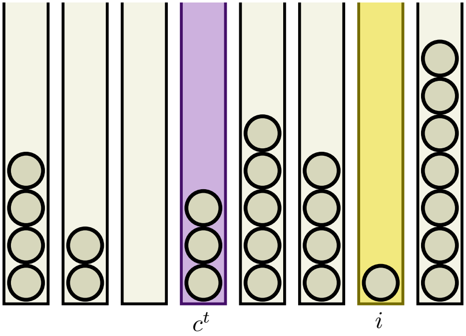
The Memory process



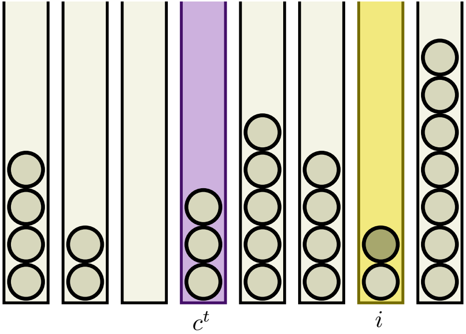
The Memory process



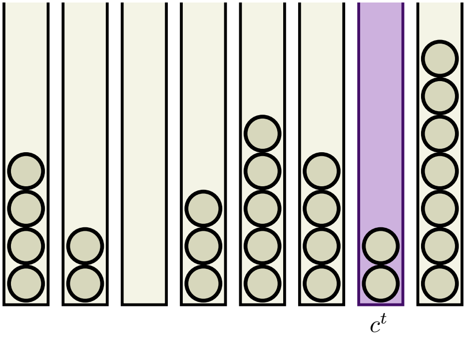
The Memory process



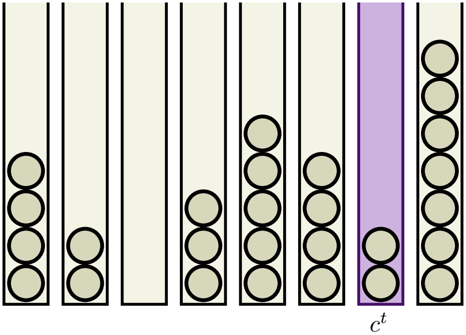
The MEMORY process



The MEMORY process

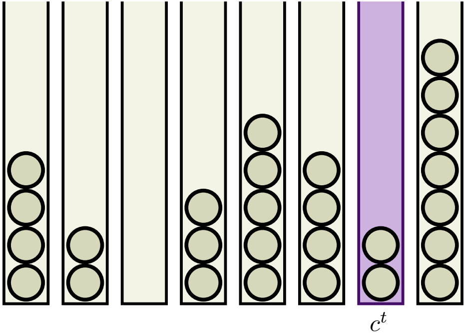


The MEMORY process



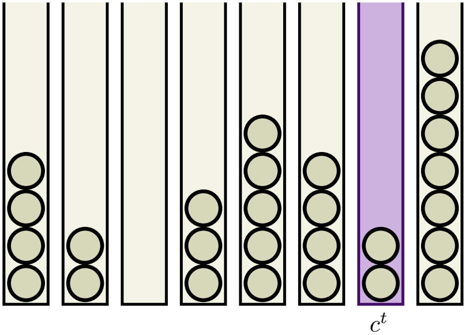
- In the lightly-loaded case, MEMORY with $d = 1$ w.h.p. achieves an $\mathcal{O}(\log \log n)$ gap [MPS02].

The Memory process



- In the lightly-loaded case, Memory with $d = 1$ 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 $f(d) \in (2d, 2d+1)$.

The MEMORY process



- In the lightly-loaded case, MEMORY with $d = 1$ 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

Heterogeneous sampling distributions

- Several different settings for d-Choice:

Heterogeneous sampling distributions

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

Heterogeneous sampling distributions

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

Heterogeneous sampling distributions

- Several different settings for d-Choice: outdated information [BCE^{+}12], graphical [BK22], adversarial noise [LS22b],

Heterogeneous sampling distributions

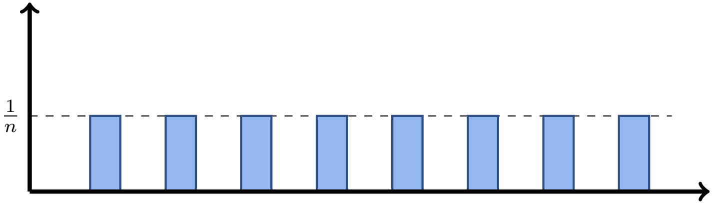
- Several different settings for d-Choice: outdated information [BCE+12], graphical [BK22], adversarial noise [LS22b], ...

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.

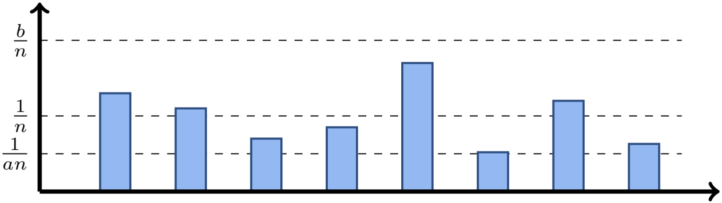
Heterogeneous sampling distributions

- 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.



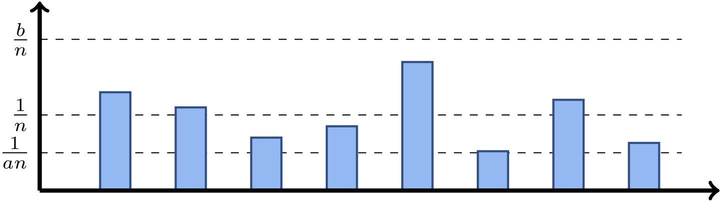
Heterogeneous sampling distributions

- 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.



Heterogeneous sampling distributions

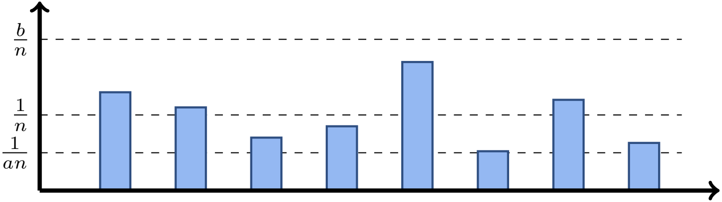
- Several different settings for d-Choice: outdated information [BCE$^+$12], graphical [BK22], adversarial noise [LS22b], ….
- Wieder [Wie07] studied d-Choice with non-uniform sampling distributions.



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

Heterogeneous sampling distributions

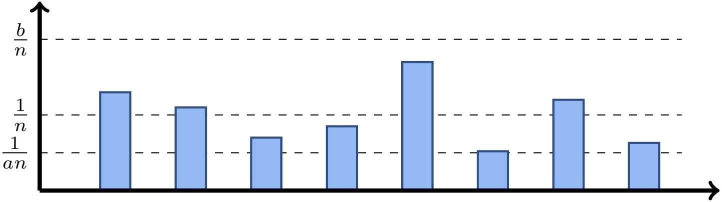
- Several different settings for d-Choice: outdated information [BCE$^+$12], graphical [BK22], adversarial noise [LS22b], ...
- Wieder [Wie07] studied d-Choice with non-uniform sampling distributions.



- 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 $\epsilon > 0$:

Heterogeneous sampling distributions

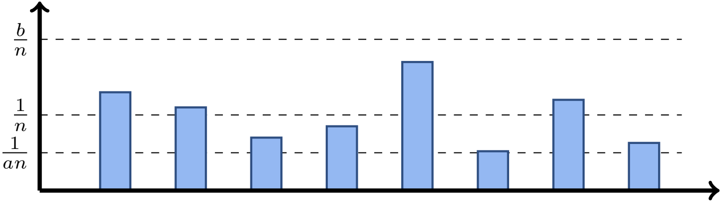
- Several different settings for d-Choice: outdated information [BCE+12], graphical [BK22], adversarial noise [LS22b], ...
- Wieder [Wie07] studied d-Choice with non-uniform sampling distributions.



- 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 $\epsilon > 0$:
  - For any $d \geq (1 + \epsilon) \cdot d'$, then d-Choice w.h.p. achieves $\text{Gap}(m) = \mathcal{O}(\log \log n)$.

Heterogeneous sampling distributions

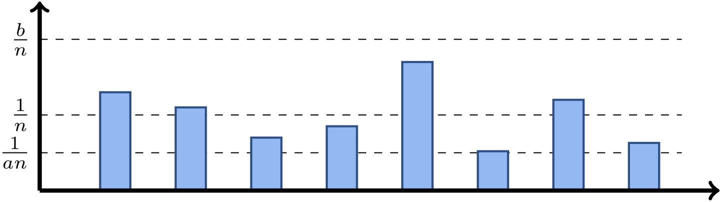
- Several different settings for d-Choice: outdated information [BCE+12], graphical [BK22], adversarial noise [LS22b], ...
- Wieder [Wie07] studied d-Choice with non-uniform sampling distributions.



- 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 $\epsilon > 0$:
  - For any $d \geq (1 + \epsilon) \cdot d'$, then d-Choice w.h.p. achieves $\text{Gap}(m) = \mathcal{O}(\log \log n)$.
  - For any $d \leq (1 - \epsilon) \cdot d'$, then d-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 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-Choice w.h.p. achieves $\text{Gap}(m) = \mathcal{O}(\log \log n)$.
  - 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?

Our results

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 $\mathrm{Gap}(m) = \mathcal{O}(\log \log n)$.

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 $m \geq n$.

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 $\mathrm{Gap}(m) = \mathcal{O}(\log \log n)$.

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

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

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. $\text{Gap}(m) = \mathcal{O}(\log \log n)$.
  $\leadsto$ 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. $\text{Gap}(m) = \mathcal{O}(\log \log n)$.
  $\rightsquigarrow$ 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. $\mathcal{O}(\log n)$. 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. $\text{Gap}(m) = \mathcal{O}(\log \log n)$.
  $\leadsto$ 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 MEMORY (with $d = M = 1$) achieves w.h.p. $\mathcal{O}(\log n)$. We improve this to $\mathrm{Gap}(m) = \mathcal{O}(\log \log n)$.

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

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

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. $\text{Gap}(m) = \mathcal{O}(\log \log n)$.
  $\leadsto$ 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 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. $\text{Gap}(m) = \mathcal{O}(\log \log n)$.
$\leadsto$ 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. $\text{Gap}(m) = \mathcal{O}(\log n)$

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. $\text{Gap}(m) = \mathcal{O}(\log \log n)$.
  $\leadsto$ In contrast to TWO-CHOICE, where the gap grows with m, for $a = b = 2$.

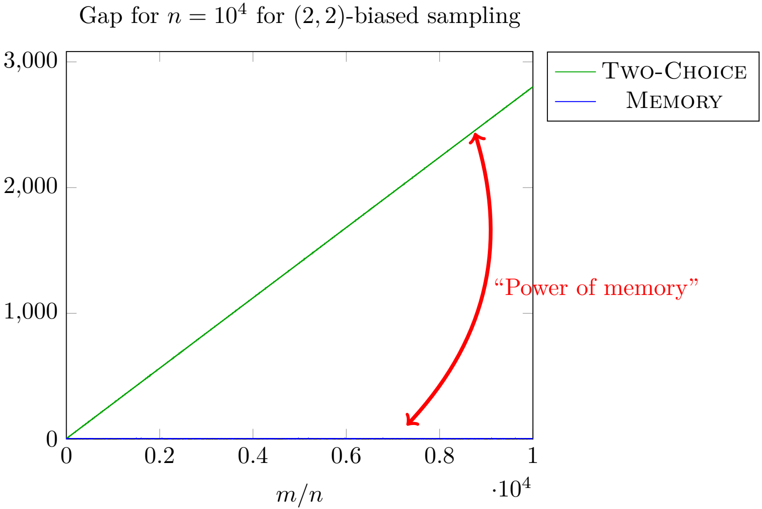
- For any $a := a(n)$ and $b := b(n)$, the gap is independent of m.

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

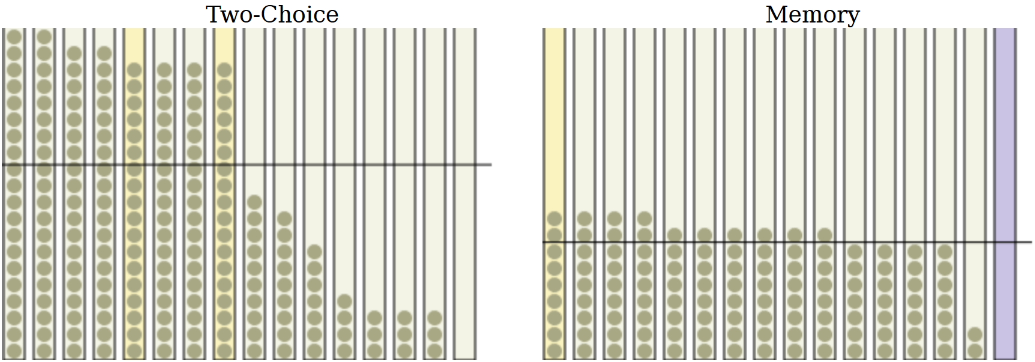
- d-RESET-MEMORY, a variant of MEMORY where the cache resets every d steps has w.h.p. $\text{Gap}(m) = \mathcal{O}(\log n)$, even in the presence of weights.

Power of memory: Visualisation

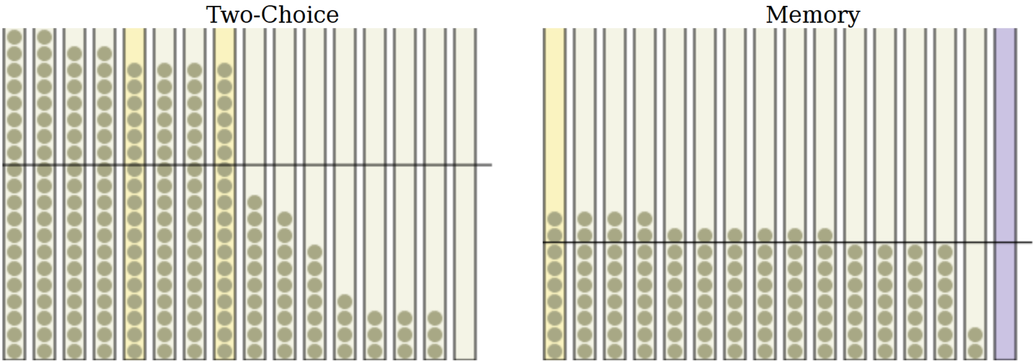
`Open visualiser`



Why MEMORY recovers?

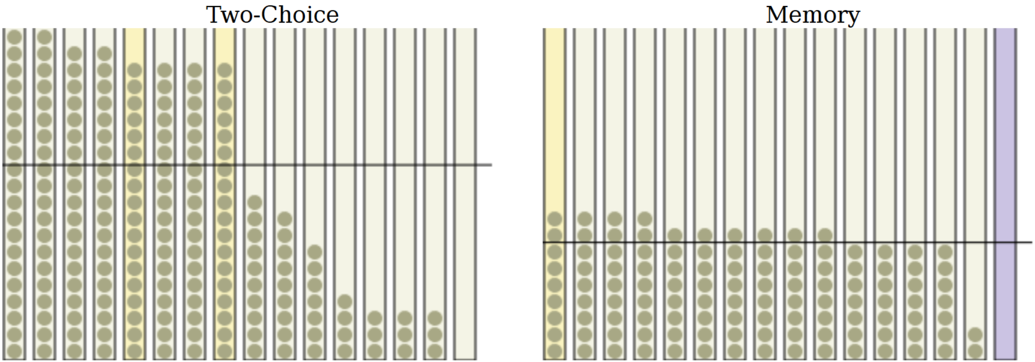


Why MEMORY recovers?



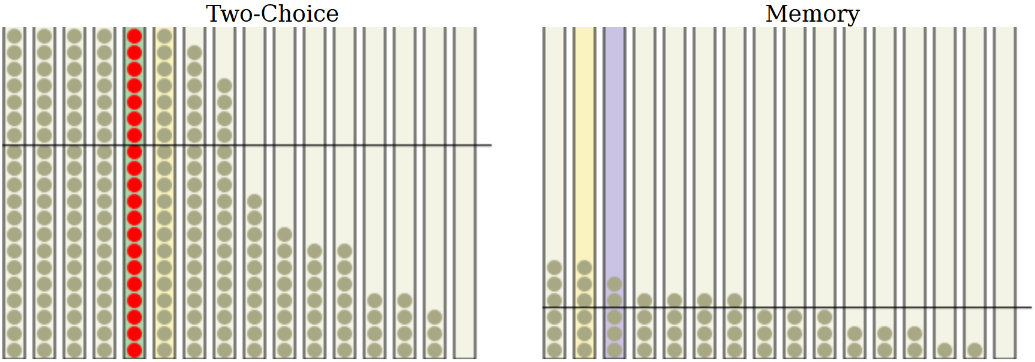
- In TWO-CHOICE, there is a set of bins that receives $> m/n$ balls in expectation.

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

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

- Define the **super-exponential potentials** Φ_j for $0 \leq j = \mathcal{O}(\log\log n)$,

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

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

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

- Define the **super-exponential potentials** Φ_j for $0 \leq j = \mathcal{O}(\log\log n)$,

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

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

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

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

- Define the **super-exponential potentials** Φ_j for $0 \leq j = \mathcal{O}(\log\log n)$,

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

  where $z_j := \frac{t}{n} + j \cdot z$ for constants $z > 0$, $\alpha \in (0,1)$ and $v > 1$.
- When $\Phi_j^t = \mathcal{O}(n)$, then $\mathrm{Gap}(t) = \mathcal{O}(j \cdot z + \frac{\log n}{\alpha \cdot v^j})$.
- For $j = \Theta(\log\log n)$, when $\Phi_j = \mathcal{O}(n)$, then $\mathrm{Gap}(m) = \Theta(\log\log n)$.

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

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

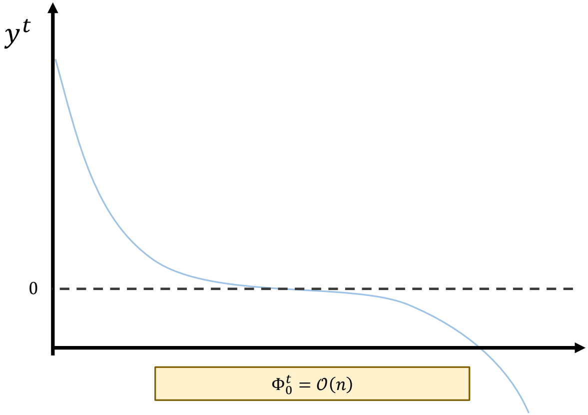
Outline for the $\mathcal{O}(\log\log n)$ bound

- Define the **super-exponential potentials** Φ_j for $0 \leq j = \mathcal{O}(\log\log n)$,
$$\Phi_j^t := \Phi_j^t(\alpha \cdot v^j, z_j) := \sum_{i : x_i^t \geq z_j} e^{\alpha \cdot v^j \cdot (x_i^t - z_j)},$$
where $z_j := \frac{t}{n} + j \cdot z$ for constants $z > 0$, $\alpha \in (0,1)$ and $v > 1$.
- When $\Phi_j^t = \mathcal{O}(n)$, then $\mathrm{Gap}(t) = \mathcal{O}(j \cdot z + \frac{\log n}{\alpha \cdot v^j})$.
- For $j = \Theta(\log\log n)$, when $\Phi_j = \mathcal{O}(n)$, then $\mathrm{Gap}(m) = \Theta(\log\log n)$.
- Further, when $\Phi_j^t = \mathcal{O}(n)$, then also number of bins with load at least z_{j+1} is at most $\mathcal{O}(n \cdot e^{-\alpha \cdot v^j \cdot z})$.
- We group steps into rounds (at most $e^{v^{j+2}} \cdot \log^3 n$ steps each) and show that
$$\mathbf{E}\left[\, \Phi_{j+1}^{r+1} \,\middle|\, \mathfrak{F}^r, \Phi_j^r = \mathcal{O}(n) \,\right] \leq \Phi_{j+1}^r \cdot \left(1 - \frac{e^{v^{j+2}}}{n}\right) + e^{-v^{j+1}/2}.$$

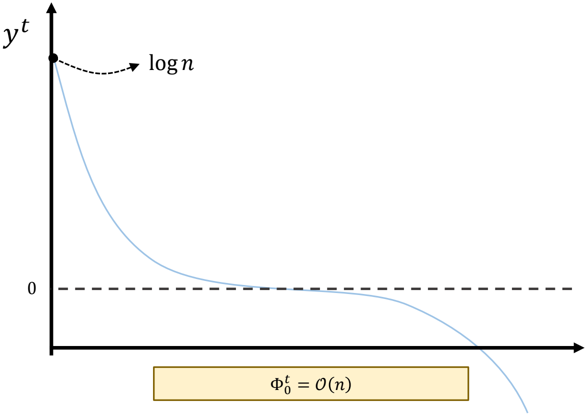
Outline for the $\mathcal{O}(\log\log n)$ bound

- Define the **super-exponential potentials** Φ_j for $0 \leq j = \mathcal{O}(\log\log n)$,
$$\Phi_j^t := \Phi_j^t(\alpha \cdot v^j, z_j) := \sum_{i : x_i^t \geq z_j} e^{\alpha \cdot v^j \cdot (x_i^t - z_j)},$$
where $z_j := \frac{t}{n} + j \cdot z$ for constants $z > 0$, $\alpha \in (0,1)$ and $v > 1$.
- When $\Phi_j^t = \mathcal{O}(n)$, then $\mathrm{Gap}(t) = \mathcal{O}(j \cdot z + \frac{\log n}{\alpha \cdot v^j})$.
- For $j = \Theta(\log\log n)$, when $\Phi_j = \mathcal{O}(n)$, then $\mathrm{Gap}(m) = \Theta(\log\log n)$.
- Further, when $\Phi_j^t = \mathcal{O}(n)$, then also number of bins with load at least z_{j+1} is at most $\mathcal{O}(n \cdot e^{-\alpha \cdot v^j \cdot z})$.
- We group steps into rounds (at most $e^{v^{j+2}} \cdot \log^3 n$ steps each) and show that
$$\mathbf{E}\left[\, \Phi_{j+1}^{r+1} \,\middle|\, \mathfrak{F}^r, \Phi_j^r = \mathcal{O}(n) \,\right] \leq \Phi_{j+1}^r \cdot \left(1 - \frac{e^{v^{j+2}}}{n}\right) + e^{-v^{j+1}/2}.$$
- The base case follows by an involved analysis of the hyperbolic cosine potential function [PTW15, LS22a].

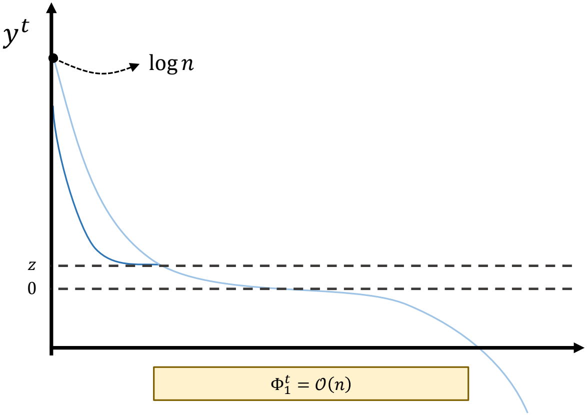
Layered induction over super-exponential potentials



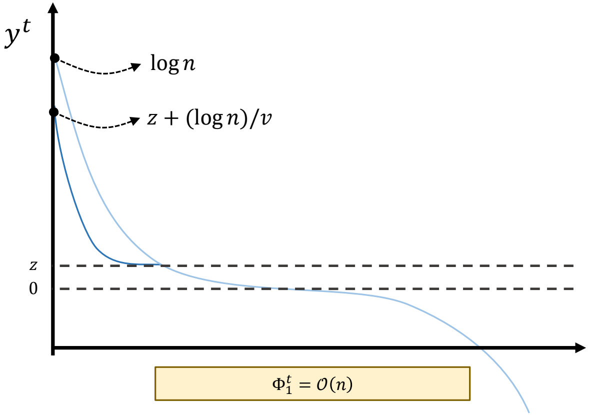
Layered induction over super-exponential potentials



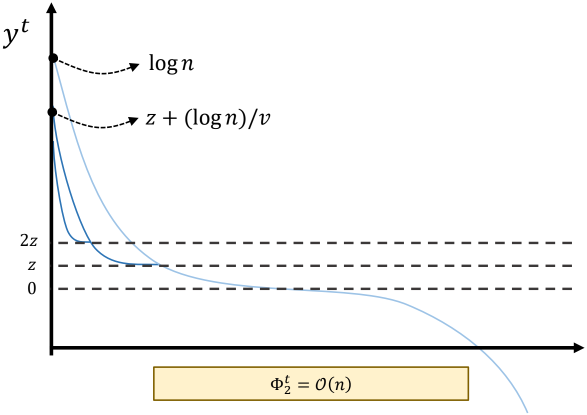
Layered induction over super-exponential potentials



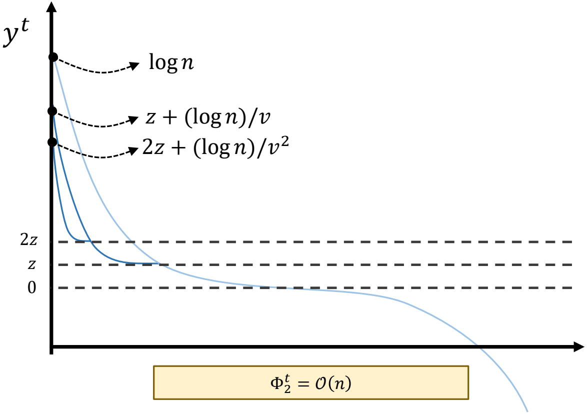
Layered induction over super-exponential potentials



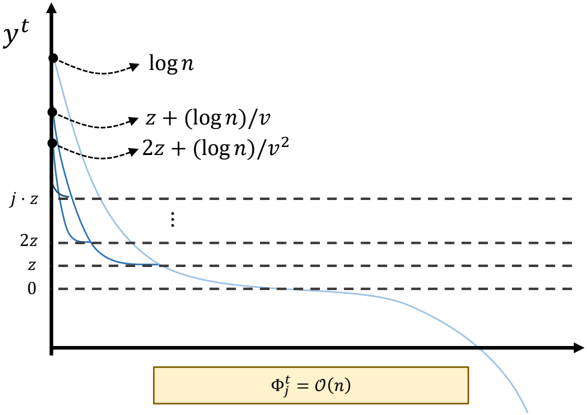
Layered induction over super-exponential potentials



Layered induction over super-exponential potentials

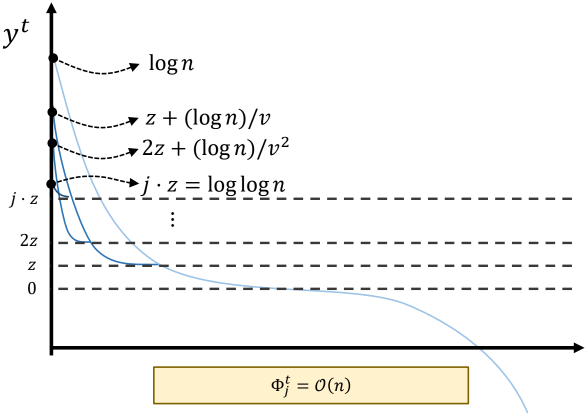


Layered induction over super-exponential potentials



$\Phi_j^t = \mathcal{O}(n)$

Layered induction over super-exponential potentials



Conclusion

Summary & Future work

Summary & Future work

We have shown that:

Summary & Future work

We have shown that:

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

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 $a, b > 1$.

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 $m \geq n$.

Summary & Future work

We have shown that:

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

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:

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. $\mathrm{Gap}(m) = \mathcal{O}(\log \log n)$.
- 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. $\mathrm{Gap}(m) = \mathcal{O}(\log n)$.

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. $\mathrm{Gap}(m) = \mathcal{O}(\log\log n)$.
- 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. $\mathrm{Gap}(m) = \mathcal{O}(\log n)$.

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

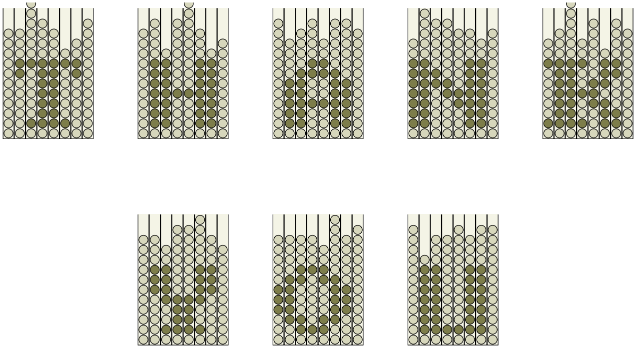
We have shown that:

- 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 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.

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.

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

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.
- For ONE-CHOICE, $p^t = \left(\frac{1}{n}, \quad \ldots \quad, \frac{1}{n}\right)$.
- For TWO-CHOICE,

$$p^t = \left(\frac{1}{n^2}, \quad \ldots \quad, \frac{2i-1}{n^2}, \quad \ldots \quad, \frac{2n-1}{n^2}\right).$$

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.
- For ONE-CHOICE, $p^t = \left(\frac{1}{n}, \quad \ldots \quad, \frac{1}{n}\right)$.
- For TWO-CHOICE,

$$p^t = \left(\frac{1}{n^2}, \quad \ldots \quad, \frac{2i-1}{n^2}, \quad \ldots \quad, \frac{2n-1}{n^2}\right).$$

- For MEMORY, if the cache is the k-th most loaded bin, then

$$p^t = \left(\underbrace{0, \quad \ldots \quad, 0}_{k-1 \text{ bins}}, \frac{k}{n}, \underbrace{\frac{1}{n}, \quad \ldots \quad, \frac{1}{n}}_{n-k \text{ bins}}\right).$$

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

$$p^t = \left(\frac{1}{n^2}, \quad \ldots \quad, \frac{2i-1}{n^2}, \quad \ldots \quad, \frac{2n-1}{n^2}\right).$$

- For Memory, if the cache is the k-th most loaded bin, then

$$p^t = \left(\underbrace{0, \quad \ldots \quad, 0}_{k-1 \text{ bins}}, \frac{k}{n}, \underbrace{\frac{1}{n}, \quad \ldots \quad, \frac{1}{n}}_{n-k \text{ bins}}\right).$$

For $k = 1$, this is like One-Choice.

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.
- For ONE-CHOICE, $p^t = \left(\frac{1}{n}, \quad \ldots \quad, \frac{1}{n}\right)$.
- For TWO-CHOICE,

$$p^t = \left(\frac{1}{n^2}, \quad \ldots \quad, \frac{2i-1}{n^2}, \quad \ldots \quad, \frac{2n-1}{n^2}\right).$$

- For MEMORY, if the cache is the k-th most loaded bin, then

$$p^t = \Bigg(\underbrace{0, \quad \ldots \quad, 0}_{k-1 \text{ bins}}, \frac{k}{n}, \underbrace{\frac{1}{n}, \quad \ldots \quad, \frac{1}{n}}_{n-k \text{ bins}}\Bigg).$$

- A probability vector p is (δ, ϵ)-smooth if majorized by

$$\Bigg(\underbrace{\frac{1-\epsilon}{n}, \quad \ldots \quad, \frac{1-\epsilon}{n}}_{\delta n \text{ bins}}, \underbrace{\frac{1+\tilde{\epsilon}}{n}, \quad \ldots \quad, \frac{1+\tilde{\epsilon}}{n}}_{(1-\delta)n \text{ bins}}\Bigg).$$

Hyperbolic cosine potential

Hyperbolic cosine potential

- Peres, Talwar and Wieder [PTW15] used the **hyperbolic cosine potential** Γ^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** Γ^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 $\text{Gap}(m) = \mathcal{O}\left(\frac{\log n}{\alpha}\right)$.

Hyperbolic cosine potential

- Peres, Talwar and Wieder [PTW15] used the **hyperbolic cosine potential** Γ^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 $\Gamma^m = \mathrm{poly}(n)$, then $\mathrm{Gap}(m) = \mathcal{O}\left(\frac{\log n}{\alpha}\right)$.
- They showed that for any (δ, ϵ)-smooth probability allocation vector p^t,
$$\mathbf{E}\left[\, \Gamma^{t+1} \mid \mathfrak{F}^t \,\right] \leq \Gamma^t \cdot \left(1 - \frac{\alpha\epsilon}{n}\right) + c.$$

Hyperbolic cosine potential

- Peres, Talwar and Wieder [PTW15] used the **hyperbolic cosine potential** Γ^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 $\Gamma^m = \mathrm{poly}(n)$, then $\mathrm{Gap}(m) = \mathcal{O}\left(\frac{\log n}{\alpha}\right)$.
- They showed that for any (δ, ϵ)-smooth probability allocation vector p^t,
$$\mathbf{E}\left[\,\Gamma^{t+1} \mid \mathfrak{F}^t\,\right] \leq \Gamma^t \cdot \left(1 - \frac{\alpha\epsilon}{n}\right) + c.$$
- By induction, this implies that $\mathbf{E}\left[\,\Gamma^m\,\right] \leq \frac{c}{\alpha\epsilon} \cdot n$.

Hyperbolic cosine potential

- Peres, Talwar and Wieder [PTW15] used the **hyperbolic cosine potential** Γ^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 $\text{Gap}(m) = \mathcal{O}\left(\frac{\log n}{\alpha}\right)$.
- They showed that for any (δ, ϵ)-smooth probability allocation vector p^t,
$$\mathbf{E}\left[\, \Gamma^{t+1} \mid \mathfrak{F}^t \,\right] \leq \Gamma^t \cdot \left(1 - \frac{\alpha\epsilon}{n}\right) + c.$$
- By induction, this implies that $\mathbf{E}\left[\, \Gamma^m \,\right] \leq \frac{c}{\alpha\epsilon} \cdot n$.
- And so, by Markov's inequality $\mathbf{Pr}\left[\, \Gamma^m \leq \frac{c}{\alpha\epsilon} \cdot n^3 \,\right] \geq 1 - n^{-2}$.

Hyperbolic cosine potential

- Peres, Talwar and Wieder [PTW15] used the **hyperbolic cosine potential** Γ^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 $\text{Gap}(m) = \mathcal{O}\left(\frac{\log n}{\alpha}\right)$.
- They showed that for any (δ, ϵ)-smooth probability allocation vector p^t,
$$\mathbf{E}\left[\, \Gamma^{t+1} \mid \mathfrak{F}^t \,\right] \leq \Gamma^t \cdot \left(1 - \frac{\alpha\epsilon}{n}\right) + c.$$
- By induction, this implies that $\mathbf{E}\left[\, \Gamma^m \,\right] \leq \frac{c}{\alpha\epsilon} \cdot n$.
- And so, by Markov's inequality $\mathbf{Pr}\left[\, \Gamma^m \leq \frac{c}{\alpha\epsilon} \cdot n^3 \,\right] \geq 1 - n^{-2}$.

Problem: p^t for MEMORY may not be (δ, ϵ)-smooth

A generalised drift inequality [LS22a]

A generalised drift inequality [LS22a]

- If for some (δ, ϵ)-smooth probability vector q,

$$\mathbf{E}\left[\, \Phi^{t+1} \,\middle|\, \mathfrak{F}^t \,\right] \leq \Phi^t + \sum_{i=1}^{n} \Phi_i^t \cdot \left(q_i^t - \frac{1}{n} \right) \cdot \alpha + \Phi^t \cdot C \cdot \frac{\alpha^2}{n},$$

$$\mathbf{E}\left[\, \Psi^{t+1} \,\middle|\, \mathfrak{F}^t \,\right] \leq \Psi^t + \sum_{i=1}^{n} \Psi_i^t \cdot \left(\frac{1}{n} - q_i^t \right) \cdot \alpha + \Psi^t \cdot C \cdot \frac{\alpha^2}{n}.$$

A generalised drift inequality [LS22a]

- If for some (δ, ϵ)-smooth probability vector q,

$$\mathbf{E}\left[\, \Phi^{t+1} \,\middle|\, \mathfrak{F}^t \,\right] \leq \Phi^t + \sum_{i=1}^{n} \Phi_i^t \cdot \left(q_i^t - \frac{1}{n} \right) \cdot \alpha + \Phi^t \cdot C \cdot \frac{\alpha^2}{n},$$

$$\mathbf{E}\left[\, \Psi^{t+1} \,\middle|\, \mathfrak{F}^t \,\right] \leq \Psi^t + \sum_{i=1}^{n} \Psi_i^t \cdot \left(\frac{1}{n} - q_i^t \right) \cdot \alpha + \Psi^t \cdot C \cdot \frac{\alpha^2}{n}.$$

- Then, for sufficiently small $\alpha > 0$,

$$\mathbf{E}\left[\, \Gamma^{t+1} \,\middle|\, \mathfrak{F}^t \,\right] \leq \Gamma^t \cdot \left(1 - \frac{\alpha\epsilon}{n} \right) + c.$$

A generalised drift inequality [LS22a]

- If for some (δ, ϵ)-smooth probability vector q,

$$\mathbf{E}\left[\, \Phi^{t+1} \,\middle|\, \mathfrak{F}^t \,\right] \leq \Phi^t + \sum_{i=1}^{n} \Phi_i^t \cdot \left(q_i^t - \frac{1}{n} \right) \cdot \alpha + \Phi^t \cdot C \cdot \frac{\alpha^2}{n},$$

Could be allocating more than one ball.

$$\mathbf{E}\left[\, \Psi^{t+1} \,\middle|\, \mathfrak{F}^t \,\right] \leq \Psi^t + \sum_{i=1}^{n} \Psi_i^t \cdot \left(\frac{1}{n} - q_i^t \right) \cdot \alpha + \Psi^t \cdot C \cdot \frac{\alpha^2}{n}.$$

- Then, for sufficiently small $\alpha > 0$,

$$\mathbf{E}\left[\, \Gamma^{t+1} \,\middle|\, \mathfrak{F}^t \,\right] \leq \Gamma^t \cdot \left(1 - \frac{\alpha\epsilon}{n} \right) + c.$$

A generalised drift inequality [LS22a]

- If for some (δ, ϵ)-smooth probability vector q, $\leftsquigarrow$ not always the prob allocation vector.

$$\mathbf{E}\left[\, \Phi^{t+1} \,\middle|\, \mathfrak{F}^t \,\right] \leq \Phi^t + \sum_{i=1}^{n} \Phi_i^t \cdot \left(q_i^t - \frac{1}{n} \right) \cdot \alpha + \Phi^t \cdot C \cdot \frac{\alpha^2}{n},$$

$$\mathbf{E}\left[\, \Psi^{t+1} \,\middle|\, \mathfrak{F}^t \,\right] \leq \Psi^t + \sum_{i=1}^{n} \Psi_i^t \cdot \left(\frac{1}{n} - q_i^t \right) \cdot \alpha + \Psi^t \cdot C \cdot \frac{\alpha^2}{n}.$$

- Then, for sufficiently small $\alpha > 0$,

$$\mathbf{E}\left[\, \Gamma^{t+1} \,\middle|\, \mathfrak{F}^t \,\right] \leq \Gamma^t \cdot \left(1 - \frac{\alpha\epsilon}{n} \right) + c.$$

A generalised drift inequality [LS22a]

- If for some (δ, ϵ)-smooth probability vector q, ⇜ not always the prob allocation vector.
$$\mathbf{E}\left[\,\Phi^{t+1} \,\middle|\, \mathfrak{F}^t\,\right] \leq \Phi^t + \sum_{i=1}^{n} \Phi_i^t \cdot \left(q_i^t - \frac{1}{n}\right) \cdot \alpha + \Phi^t \cdot C \cdot \frac{\alpha^2}{n},$$
$$\mathbf{E}\left[\,\Psi^{t+1} \,\middle|\, \mathfrak{F}^t\,\right] \leq \Psi^t + \sum_{i=1}^{n} \Psi_i^t \cdot \left(\frac{1}{n} - q_i^t\right) \cdot \alpha + \Psi^t \cdot C \cdot \frac{\alpha^2}{n}.$$
- Then, for sufficiently small $\alpha > 0$,
$$\mathbf{E}\left[\,\Gamma^{t+1} \,\middle|\, \mathfrak{F}^t\,\right] \leq \Gamma^t \cdot \left(1 - \frac{\alpha\epsilon}{n}\right) + c.$$
- For 2-Reset-Memory, q is the probability allocation vector of Two-Choice.

A generalised drift inequality [LS22a]

- If for some (δ, ϵ)-smooth probability vector q, not always the prob allocation vector.

$$\mathbf{E}\left[\,\Phi^{t+1} \,\middle|\, \mathfrak{F}^t\,\right] \leq \Phi^t + \sum_{i=1}^{n} \Phi_i^t \cdot \left(q_i^t - \frac{1}{n}\right) \cdot \alpha + \Phi^t \cdot C \cdot \frac{\alpha^2}{n},$$

$$\mathbf{E}\left[\,\Psi^{t+1} \,\middle|\, \mathfrak{F}^t\,\right] \leq \Psi^t + \sum_{i=1}^{n} \Psi_i^t \cdot \left(\frac{1}{n} - q_i^t\right) \cdot \alpha + \Psi^t \cdot C \cdot \frac{\alpha^2}{n}.$$

- Then, for sufficiently small $\alpha > 0$,

$$\mathbf{E}\left[\,\Gamma^{t+1} \,\middle|\, \mathfrak{F}^t\,\right] \leq \Gamma^t \cdot \left(1 - \frac{\alpha\epsilon}{n}\right) + c.$$

- 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

Handling heterogeneous distributions

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

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.

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. $\leftsquigarrow$ for sufficiently large d, beats the (a, b)-bias.

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. $\leftsquigarrow$ 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. $\leadsto$ for sufficiently large d, beats the (a, b)-bias.
  - Load comparisons are based on the last reset. $\leadsto$ 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. $\leftsquigarrow$ for sufficiently large d, beats the (a, b)-bias.
  - Load comparisons are based on the last reset. $\leftsquigarrow$ makes computation of q tractable.
- Moving probabilities between bins with almost the same load, introduces a small additive term in the bound,

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

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. $\leftsquigarrow$ for sufficiently large d, beats the (a, b)-bias.
  - Load comparisons are based on the last reset. $\leftsquigarrow$ makes computation of q tractable.
- Moving probabilities between bins with almost the same load, introduces a small additive term in the bound,

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

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. $\leadsto$ for sufficiently large d, beats the (a, b)-bias.
  - Load comparisons are based on the last reset. $\leadsto$ makes computation of q tractable.
- Moving probabilities between bins with almost the same load, introduces a small additive term in the bound,

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

since $\Phi_i^t - \Phi_j^t \leq \Phi_j^t \cdot (2\alpha d)$ and 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. $\leftsquigarrow$ for sufficiently large d, beats the (a, b)-bias.
  - Load comparisons are based on the last reset. $\leftsquigarrow$ makes computation of q tractable.
- Moving probabilities between bins with almost the same load, introduces a small additive term in the bound,

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

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

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. $\leadsto$ for sufficiently large d, beats the (a, b)-bias.
  - Load comparisons are based on the last reset. $\leadsto$ makes computation of q tractable.
- Moving probabilities between bins with almost the same load, introduces a small additive term in the bound,

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

  since $\Phi_i^t - \Phi_j^t \leq \Phi_j^t \cdot (2\alpha d)$ and probability of selecting a bin twice is at most $d^2 \cdot \frac{b}{n}$.
- Similarly for Ψ. So for sufficiently small $\alpha := \alpha(d) > 0$, $\mathbf{E}\left[\, \Gamma^m \,\right] = \mathcal{O}(n)$.

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. $\leftsquigarrow$ for sufficiently large d, beats the (a, b)-bias.
  - Load comparisons are based on the last reset. $\leftsquigarrow$ makes computation of q tractable.
- Moving probabilities between bins with almost the same load, introduces a small additive term in the bound,

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

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

- Similarly for Ψ. So for sufficiently small $\alpha := \alpha(d) > 0$, $\mathbf{E}\left[\, \Gamma^m \,\right] = \mathcal{O}(n)$.
- And so $\text{Gap}(m) = \mathcal{O}((\log n)/\alpha)$ gap.

Bibliography I

- Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal, *Balanced allocations*, SIAM J. Comput. **29** (1999), no. 1, 180–200. MR 1710347
- P. Berenbrink, A. Czumaj, M. Englert, T. Friedetzky, and L. Nagel, *Multiple-choice balanced allocation in (almost) parallel*, 16th International Workshop on Randomization and Computation (RANDOM'12) (Berlin Heidelberg), Springer-Verlag, 2012, pp. 411–422.
- P. Berenbrink, A. Czumaj, A. Steger, and B. Vöcking, *Balanced allocations: the heavily loaded case*, SIAM J. Comput. **35** (2006), no. 6, 1350–1385. MR 2217150
- N. Bansal and W. Kuszmaul, *Balanced allocations: The heavily loaded case with deletions*, 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'22), IEEE, 2022, pp. 801–812.
- O. N. Feldheim, O. Gurel-Gurevich, and J. Li, *Long-term balanced allocation via thinning*, 2021, arXiv:2110.05009.

Bibliography II

- R.J. Gibbens, F.P. Kelly, and P.B. Key, *Dynamic alternative routing – modelling and behavior*, Proceedings of the 12 International Teletraffic Congress, Torino, Italy, Elsevier, Amsterdam, 1988.
- G. H. Gonnet, *Expected length of the longest probe sequence in hash code searching*, J. Assoc. Comput. Mach. **28** (1981), no. 2, 289–304. MR 612082
- R. M. Karp, M. Luby, and F. Meyer auf der Heide, *Efficient PRAM simulation on a distributed memory machine*, Algorithmica **16** (1996), no. 4-5, 517–542. MR 1407587
- D. Los and T. Sauerwald, *Balanced allocations in batches: Simplified and generalized*, 34th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA'22) (New York, NY, USA), ACM, 2022, p. 389–399.
- ______, *Balanced allocations with the choice of noise*, 41st Annual ACM-SIGOPT Principles of Distributed Computing (PODC'22) (New York, NY, USA), ACM, 2022, p. 164–175.

Bibliography III

- D. Los, T. Sauerwald, and J. Sylvester, *Balanced Allocations: Caching and Packing, Twinning and Thinning*, 33rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'22) (Alexandria, Virginia), SIAM, 2022, pp. 1847–1874.
- M. Mitzenmacher, B. Prabhakar, and D. Shah, *Load balancing with memory*, 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'02), IEEE, 2002, pp. 799–808.
- R. Pagh and F. F. Rodler, *Cuckoo hashing*, Algorithms—ESA 2001 (Århus), Lecture Notes in Comput. Sci., vol. 2161, Springer, Berlin, 2001, pp. 121–133. MR 1913547
- Y. Peres, K. Talwar, and U. Wieder, *Graphical balanced allocations and the* $(1+\beta)$*-choice process*, Random Structures Algorithms **47** (2015), no. 4, 760–775. MR 3418914
- M. Raab and A. Steger, *"Balls into bins"—a simple and tight analysis*, 2nd International Workshop on Randomization and Computation (RANDOM'98), vol. 1518, Springer, 1998, pp. 159–170. MR 1729169

Bibliography IV

- D. Shah and B. Prabhakar, *The use of memory in randomized load balancing*, IEEE International Symposium on Information Theory (ISIT'02), 2002, p. 125.
- U. Wieder, *Balanced allocations with heterogenous bins*, 19th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA'07), ACM, 2007, pp. 188–193.
- ______, *Hashing, load balancing and multiple choice*, Found. Trends Theor. Comput. Sci. **12** (2016), no. 3-4, front matter, 276–379. MR 3683828