**Universitat Politècnica de Catalunya**

Facultat d'Informàtica de Barcelona (FIB)

# ASSIGNMENT 2:
# BALANCED ALLOCATIONS

Randomized Algorithms

**IVÁN SÁNCHEZ GALA**
**SERGI GUIMERÀ ROIG**

Conrado Martínez Parra

Barcelona, Spain
Nov, 2025

# Contents

**1 Introduction** **3**
1.1 Allocation Approaches . . . . . 3
1.2 Batched and Uncertain Information Settings . . . . . 4
1.3 Goal . . . . . 5
1.4 Implementation . . . . . 5
1.5 Report structure . . . . . 5

**2 Experimentation** **6**
2.1 Analyzing the Effect of $\beta$ . . . . . 6
2.2 Analyzing the $1+\beta \times d$-choice . . . . . 9
2.3 Analyzing the Effects of Batch Size . . . . . 11
2.4 Analyzing the Effects of Uncertainty . . . . . 15
2.5 Analyzing the Effects of Uncertainty $d$ . . . . . 16
2.6 Analyzing the Effects of Uncertainty $\beta$ . . . . . 17

**3 Conclusions** **19**

# 1 Introduction

In this second assignment of *Randomized Algorithms (RA–MIRI)*, we are required to simulate some allocation strategies of balls into bins.

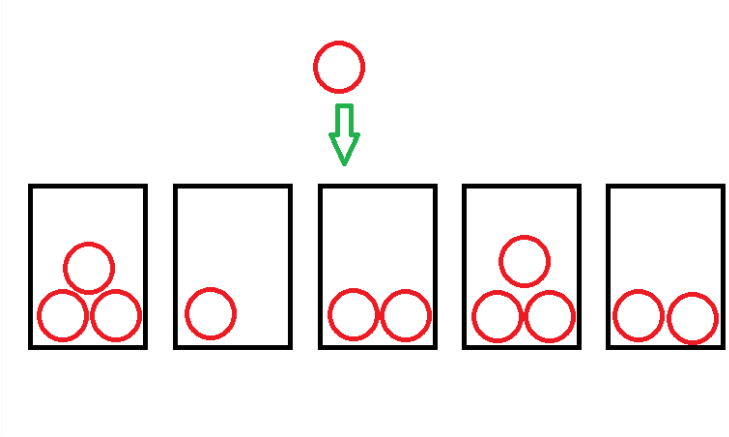


Figure 1: Schematic image of allocating balls into bins

The classical *balls and bins* framework is one of the fundamental models in the analysis of randomized algorithms. We consider a collection of $m$ bins and a sequence of $n$ balls that arrive one at a time or in batches. Each ball must be placed irrevocably into one of the bins. The load of a bin $B_i$ after $n$ balls have been allocated is denoted by $X_i(n)$, and by definition

$$\sum_{i=1}^{m} X_i(n) = n.$$

A central quantity of interest is the *gap*,

$$G_n = \max_{1 \le i \le m} \left\{ X_i(n) - \frac{n}{m} \right\},$$

which measures how far the most loaded bin deviates from the ideal perfectly balanced allocation $n/m$. Since a balanced system aims to keep all loads close to $n/m$, minimizing $G_n$ is a natural objective. In this assignment we empirically study the evolution of $G_n$ as $n$ grows, focusing on both the *light-loaded* regime ($n = m$) and the *heavy-loaded* regime ($n = m^2$).

## 1.1 Allocation Approaches

The assignment asks us to explore several classical allocation schemes:

**One-choice (standard).** For each ball, a single bin ($d = 1$) is selected uniformly at random, and the ball is placed in that bin. This baseline strategy yields a maximum load of order $\Theta(\frac{\log n}{\log \log n})$ in the light-loaded regime.

**Two-choice.** Here, $d = 2$ candidate bins are drawn independently and uniformly at random without replacement, and the ball is placed in the less loaded of the two (breaking ties uniformly at random). This simple modification, known as the *power of two choices*, dramatically improves load balance, reducing the gap to $\Theta(\log\log n)$.

**Standard $(1+\beta)$-choice.** This scheme interpolates between one- and two-choice strategies. Given $\beta \in (0,1)$, with probability $\beta$ the algorithm behaves like the two-choice scheme, and with probability $1-\beta$ it behaves like the one-choice scheme. When $\beta = 0$ we recover the one-choice strategy, and when $\beta = 1$ we recover the two-choice scheme. Intermediate values of $\beta$ allow a smooth trade-off between performance and cost.

**$d$-choice.** More generally, we may consider drawing $d \geq 1$ bins for each ball and allocating it to the least loaded among them. Larger $d$ yields better balancing but increases the required number of random accesses or queries.

**$(1+\beta \cdot d)$-choice.** In addition to the classical $d$-choice strategy, we also consider a mixed scheme that interpolates between the one-choice and $d$-choice allocations. In this $(1+\beta \cdot d)$-*choice* scheme, for each arriving ball:

- with probability $\beta$ we apply the $d$-choice rule, that is, $d$ candidate bins are sampled uniformly at random (with replacement) and the ball is placed into the bin of minimum load among them;
- with probability $1-\beta$ we fall back to the standard one-choice scheme, selecting a single bin uniformly at random.

When $\beta = 0$, the scheme coincides with the classical one-choice strategy. When $\beta = 1$, it becomes identical to the $d$-choice strategy. Thus the parameter $\beta$ provides a smooth interpolation between purely random allocation and the strong balancing effects of the power-of-$d$ choices.

## 1.2 Batched and Uncertain Information Settings

In addition to the sequential setting, we also consider the *b-batched* model, where balls arrive in batches of size $b$. The key constraint is that all allocation decisions within a batch must rely on the bin loads observed at the beginning of the batch. For large $b$, the system allocates many balls based on increasingly outdated information, potentially degrading performance.

We also explore an alternative model where full load information is not available, and only limited queries can be made. With $k = 1$ available query per bin, we may only ask whether the load of a candidate bin is above the median. With $k = 2$, we may additionally query whether a bin lies above the 75th percentile or below the 25th percentile of the load distribution. These partial-information strategies provide an interesting perspective on how little

information is needed to approximate the performance of multi-choice schemes. Moreover, for experimentation purposes instead of just considering $k = 1$ or $k = 2$, we made another beta ($\beta$) such that with probability $\beta$ the algorithm behaves like $k = 2$ and with probability $1 - \beta$ it behaves like $k = 1$.

### 1.3 Goal

The objective of this project is to simulate these allocation strategies, measure the evolution of the gap $G_n$, and compare their empirical behavior across the different scenarios:

- light-loaded ($n = m$) and heavy-loaded ($n = m^2$) regimes,
- sequential vs. $b$-batched allocation,
- full-information vs. partial-information settings.

Each experiment will be repeated $T = 5$ times to compute empirical averages and optionally to inspect variance.

### 1.4 Implementation

A Python program was developed to both simulate the Balls to Bins allocations and perform the statistical experiments automatically, as there the experiments are not deterministic, we make repetitions for each experiment configuration ($n$, $m$, batch size, $\beta$ and $d$ values) and to allow reproducibility the seeds were previously chosen.
The implementation stores the results ($m$, $n$, $\beta$, $d$, batch size, seed and gap) in a CSV file for further analysis, and generates comparative plots showing the gap as a function of $n$ and using other techniques such as coloring to add other variables to the plot.

The complete source code, together with setup instructions and generated plots, is available in the following GitHub repository:

`https://github.com/S3RXxX/RA_A2`

### 1.5 Report structure

The remainder of this report includes:

- **Experimental analysis** of the simulation results including all previous mentioned approaches of allocating the balls into the bins.
- **Conclusions** on the convergence of the empirical data to the theoretical framework.

## 2 Experimentation

For the experimentation we must specify the ranges of values used for each parameter. Unless explicitly stated otherwise, the following default sets will be used:

- Number of bins:
$$m \in \{2, 5, 10, 30, 50, 100\} \tag{1}$$
- Number of balls: For each chosen $m$, we consider a sequence beginning in the light-load regime and extending into the heavy-load regime:
$$n \in \left\{ \max\left(\frac{m}{4}, 1\right), \max\left(\frac{m}{2}, 1\right), m, 2m, \ldots, m^2 \right\}. \tag{2}$$
- Parameter of the $(1+\beta)$-choice scheme:
$$\beta \in \{0, 0.2, 0.4, 0.6, 0.8, 1\} \tag{3}$$
- Number of sampled bins:
$$d \in \{1, 2, 4, 6, 8\} \tag{4}$$
- Batch sizes (in the batched setting):
$$b \in \{1, m, 2m, 3m, \ldots, n\} \tag{5}$$

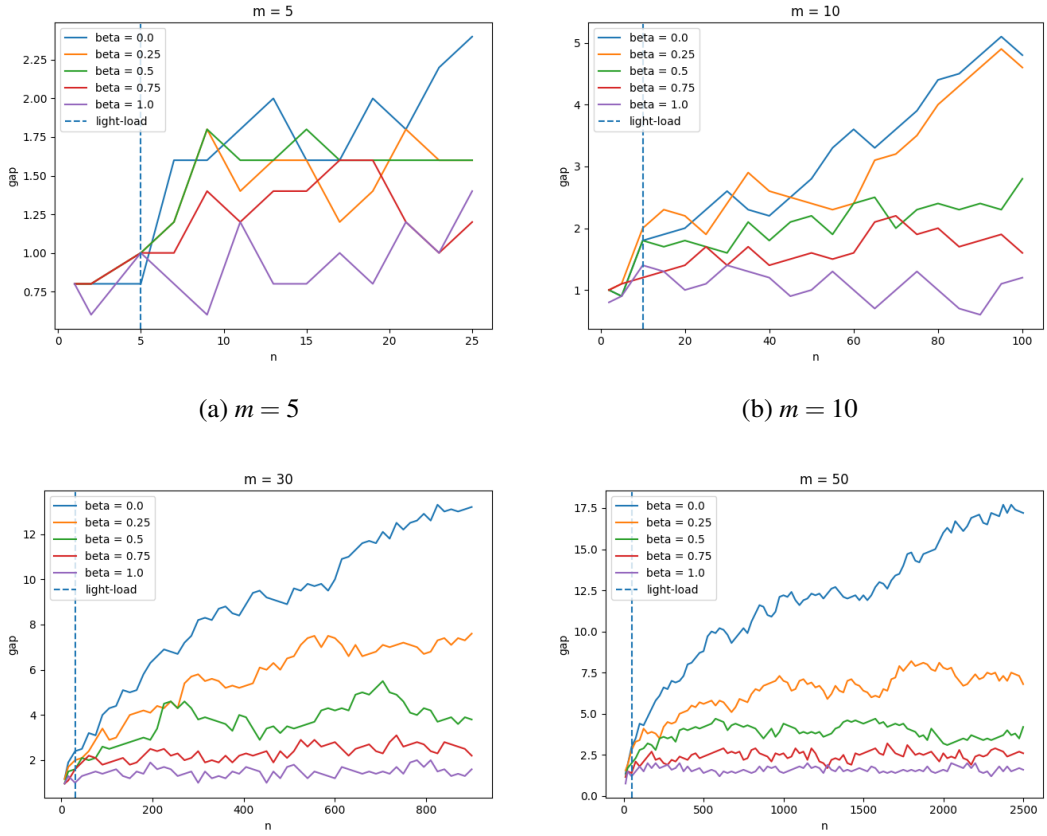
### 2.1 Analyzing the Effect of $\beta$

In the first experiment, we study the influence of the parameter $\beta$ on the evolution of the gap $G_n$. We focus on the standard (non-batched) setting, fixing:

$$d = 2, \qquad b = 1,$$

and varying $\beta$ over the values specified in Eq. (3). For each value of $m$, we observe how the gap evolves as $n$ increases from the light-load regime ($n = m$) up to the heavy-load regime ($n = m^2$). This experiment allows us to empirically understand how the $(1+\beta)$-choice strategy interpolates between the random (one-choice) case and the more balanced regimes obtained when $\beta$ is closer to 1.

For small values of $m$, the number of samples is not large enough to clearly separate the behaviour of the different $\beta$ values. However, from $m \geq 10$ onward, the effect becomes increasingly visible, and we can already distinguish how each value of $\beta$ influences the gap.




(a) $m = 5$

(b) $m = 10$

(c) $m = 30$

(d) $m = 50$

Figure 2: Evolution of the gap $G_n$ for different values of $\beta$ and several $m$.

For $m = 100$ ( Fig. 3), the trend becomes very clear: the larger the value of $\beta$, the better the balancing effect. However, the improvement is not linear. The gain from $\beta = 0$ to $\beta = 0.25$ is significantly larger than the improvement from $\beta = 0.25$ to $\beta = 0.5$, and so on. As $\beta$ approaches 1, the marginal benefit decreases, although the system still becomes progressively more balanced.

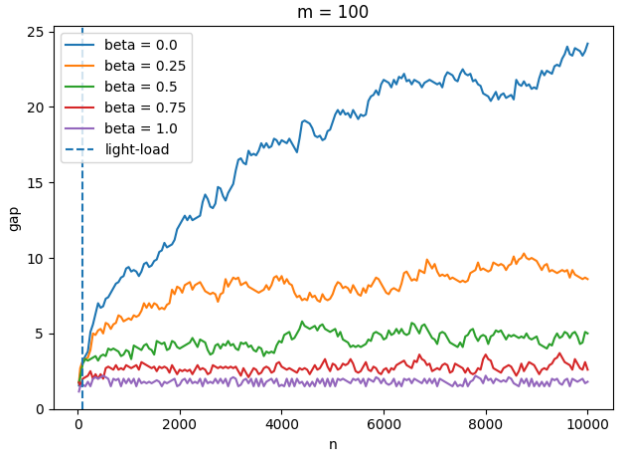



Figure 3: Evolution of the gap for $m = 100$ and various $\beta$.

Finally, when considering the standard deviation of the loads, we observe that using $d = 2$ choices dramatically reduces fluctuations compared to the purely random case. Since the process samples two bins instead of one, the probability that both sampled bins have large load is much smaller than the probability of selecting a single overloaded bin in the one-choice process. Therefore the system maintains configurations closer to the ideal average $n/m$, resulting in lower standard deviation and more predictable behaviour.

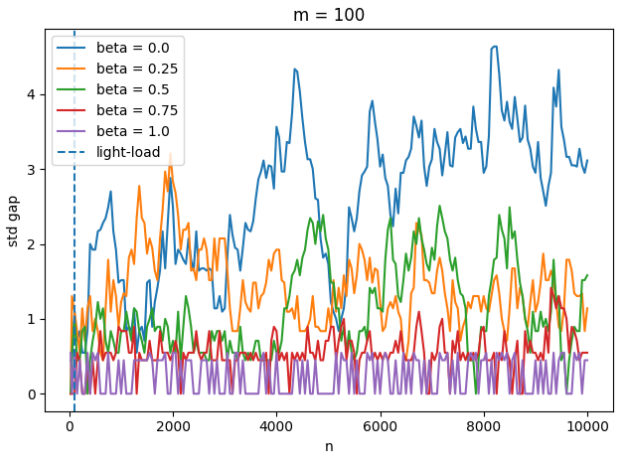



Figure 4: Evolution of the gap for $m = 100$ and various $\beta$.

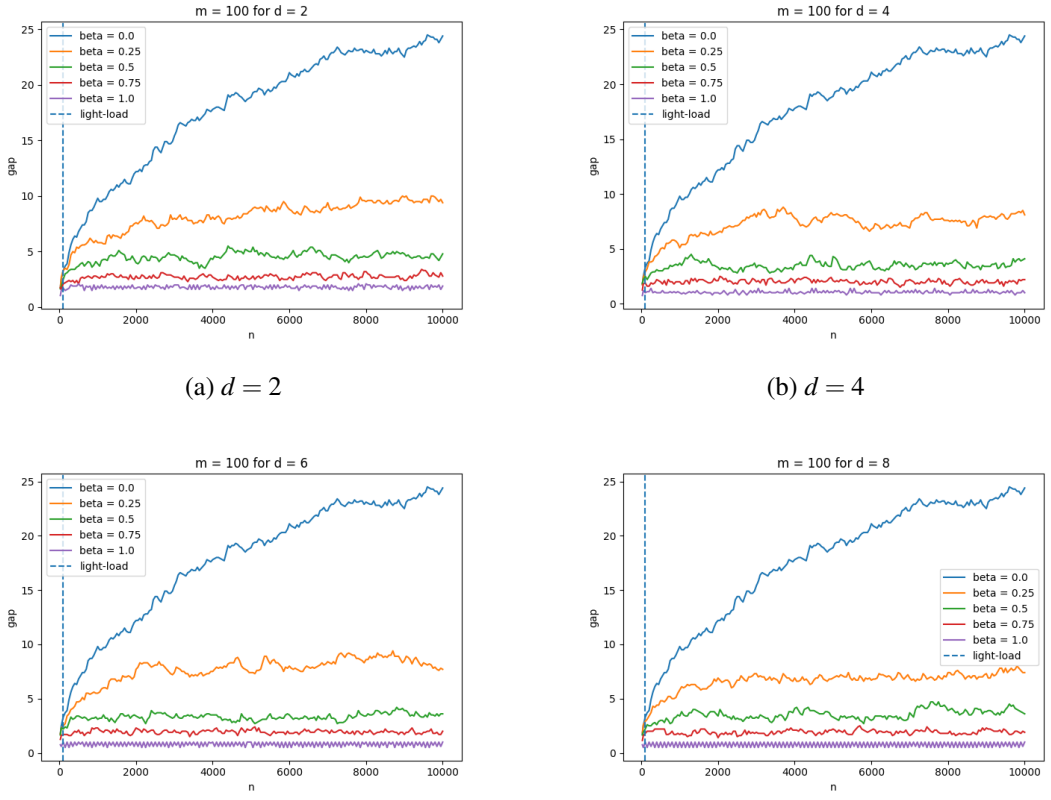
A particularly striking instance of this improvement from $d = 1$ to $d = 2$ is the one mentioned in Sec. 1.1. When $n = m$, the maximum load (whp) decreases from $\Theta\left(\frac{\log n}{\log\log n}\right)$ in the one-choice process to $\Theta(\log\log n)$ in the two-choice process, which constitutes an exponential improvement.

## 2.2 Analyzing the $1+\beta \times d$-choice

In the second experiment, we study the influence of the parameter $\beta$ and d on the evolution of the gap $G_n$. We focus on the standard (non-batched) setting, fixing:

$$b = 1,$$

varying $\beta$ and d over the values specified in Eq. (3) and Eq. (4)

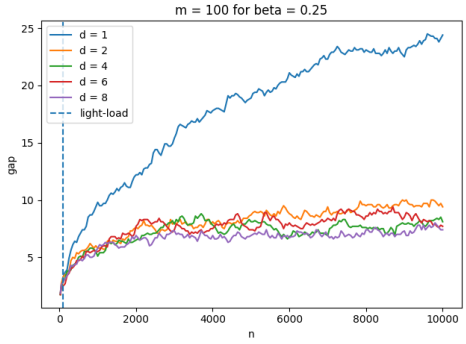



(a) $d = 2$ (b) $d = 4$

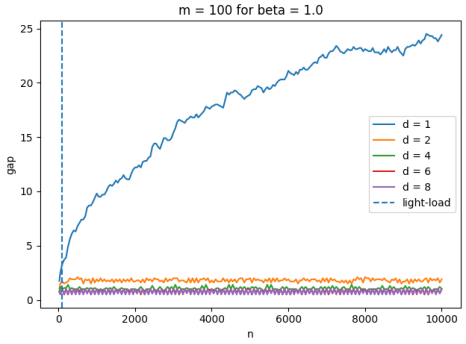
(c) $d = 6$ (d) $d = 8$

Figure 5: Evolution of the gap for $m = 100$ under different values of $d$.

Fig. 5 shows, for several values of $d$, how the different values of $\beta$ behave. As expected, the improvement from $d = 1$ to $d = 2$ is substantial (as already discussed in Sec. 1.1). However, once $d \geq 2$, the marginal improvement becomes much smaller. This effect becomes clearer when examining the evolution of the gap for fixed values of $\beta$.



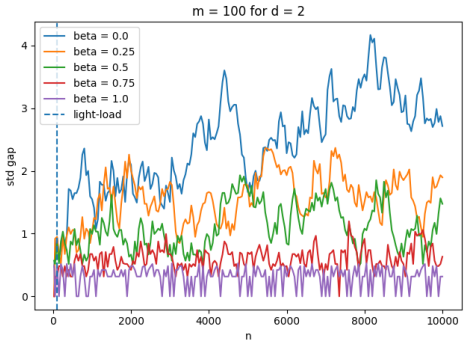

(a) $\beta = 0.25$



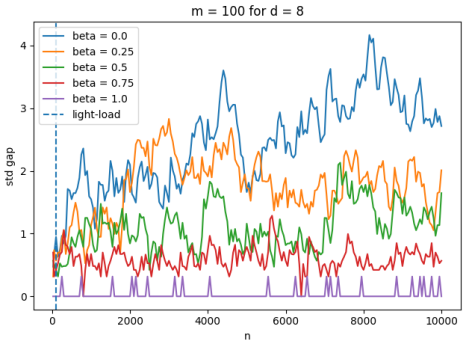

(b) $\beta = 1.0$

Figure 6: Evolution of the gap for $m = 100$ with fixed $\beta$ and varying $d$.

In Fig. 6, we observe that for larger values of $\beta$ (e.g., $\beta = 1$, corresponding to the pure $d$-choice scheme), the differences between values of $d$ become more pronounced, since randomness plays a smaller role, compared to the case when $\beta = 0.25$ where $d = 6$ can be even more loaded than $d = 2$. Nevertheless, the improvement ($\beta = 1$) from $d = 6$ to $d = 8$ is almost negligible compared to the improvement from $d = 2$ to $d = 4$.




(a) $d = 2$




(b) $d = 8$

Figure 7: Evolution of the standard deviation for $m = 100$ with fixed $d$ and varying $\beta$.

Finally, regarding the standard deviations, we observe ($\beta = 1$ in Fig. 7) that increasing $d$ consistently reduces the standard deviation of the gap. This is expected: as $d$ grows, it becomes increasingly unlikely that all $d$ sampled bins are heavily loaded, so the allocation becomes more stable and the gap fluctuates much less.

## 2.3 Analyzing the Effects of Batch Size

In the third experiment, we investigate how the batch size $b$ influences the evolution of the gap $G_n$ in the $b$-batched allocation model. Recall that in this model balls arrive in batches of size $b$, and all decisions within a batch are made using the loads observed at the beginning of the batch. This introduces uncertainty, since the allocation of earlier balls in the batch is not reflected when allocating the later ones.

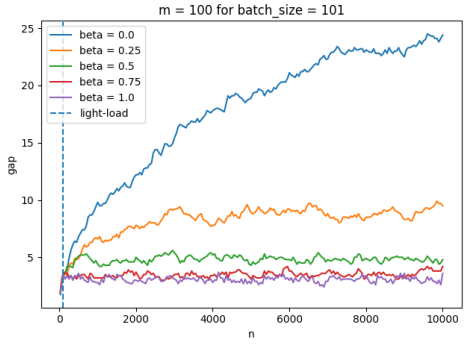
Throughout this experiment we fix

$$d = 2,$$

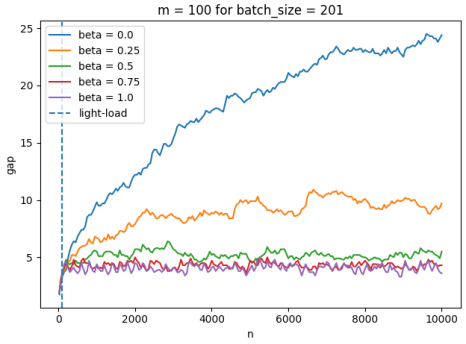
and vary the batch size $b$ over the set defined in Eq. (5), typically including values such as

$$b = 2,\ m,\ 2m,\ 5m,\ 10m,\ \ldots,\ 70m,\ \ldots, m^2.$$

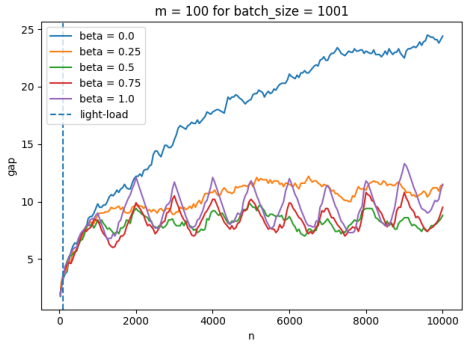
For each value of $b$, we observe how the gap evolves as $n$ grows from the light-load regime ($n = m$) to the heavy-load regime ($n = m^2$). Larger batch sizes introduce more outdated information, and therefore we expect the balancing performance to degrade as $b$ increases.



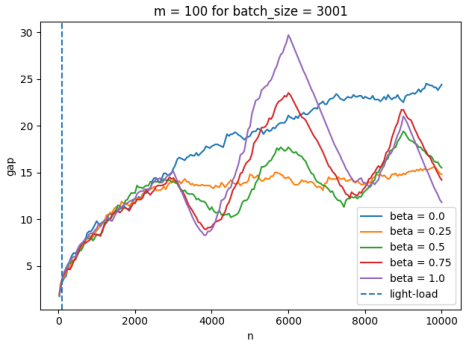

(a) $b = m$



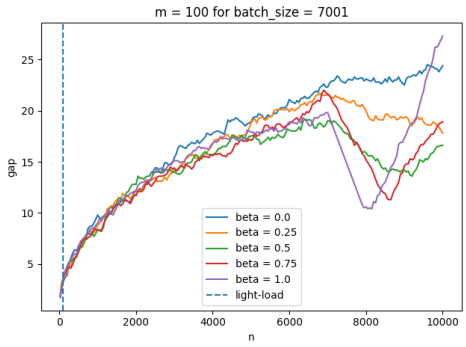

(b) $b = 2m$




(c) $b = 10m$




(d) $b = 30m$




(e) $b = 70m$

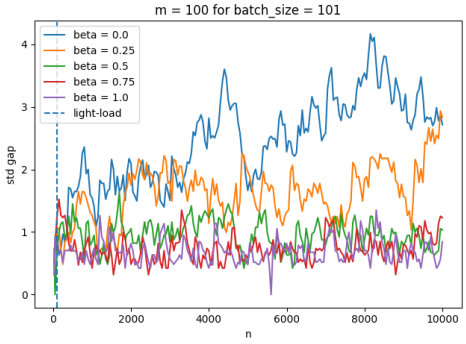
Figure 8: Evolution of the gap for $m = 100$ and $d = 2$ under different batch sizes $b$.

From Figure 8, we observe a clear degradation in performance as the batch size increases. When $b = m$, the system is still able to maintain reasonably balanced loads, since updates between batches occur frequently. However, as $b$ becomes large (e.g. $30m$ or $70m$), the algorithm is forced to make many decisions using severely outdated information, causing the gap to grow significantly. An interesting behaviour is that when $n \leq batch\_size$ the d-choice strategy behaves as if $d = 1$, because for the algorithm all the bins have 0 balls and one bin is selected at random.

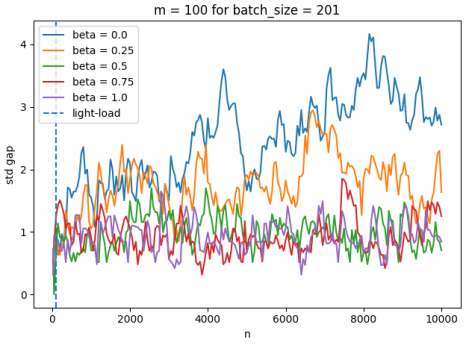
A striking phenomenon visible in the plots is the presence of large spikes in the gap when $\beta$ is close to 1. This effect is a direct consequence of the batch-based nature of the process. At the beginning of a batch, the algorithm identifies a set of bins that appear to be the least loaded. Because $\beta$ is near 1, the $(1+\beta)$-choice rule behaves almost like a pure $d$-choice process, so whenever these bins are sampled they are consistently preferred. As a result, many balls are allocated to the same initially-light bins, quickly overloading them. When the batch ends, new load information becomes available: the previously emptiest bins are now among the fullest, and a new group of bins appears to be the lightest. The next batch therefore overloads a new set of bins in exactly the same way, producing a repeated "spike" pattern. Since the true random (one-choice) strategy keeps the system roughly balanced, the bins that appear to be empty at the start of a batch are never dramatically lighter than the rest, so this systematic overfilling has an immediate and noticeable impact on the gap.

In contrast, when $\beta$ is small, the behaviour is much closer to random allocation. In this regime, wrong load information is less damaging, because most allocations ignore the (stale) recommendations of the $d$-choice rule. For short periods of time, this randomisation actually produces better behaviour than relying on incorrect load comparisons.

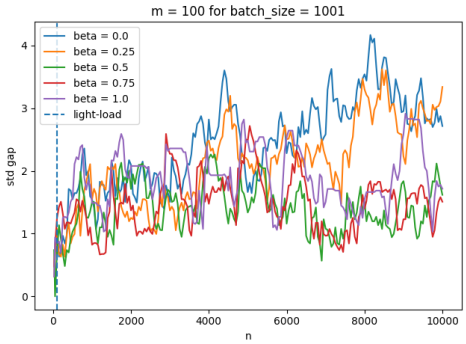
One practical takeaway is that when large batch sizes are unavoidable, a strategy that mixes randomness early and often may be beneficial. In particular, for the first few allocations after new information arrives, using $d = 2$ (i.e., a strong two-choice step) provides a significant advantage, but for the remainder of a long batch, switching to a more random behaviour (e.g., reducing $\beta$) can prevent the systematic overloading that produces the observed spikes. This suggests that adaptive or time-varying $(1+\beta)$-choice strategies may be particularly well suited for high-latency or high-batch settings.



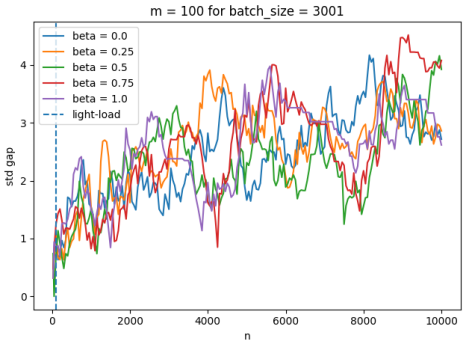

(a) $b = m$



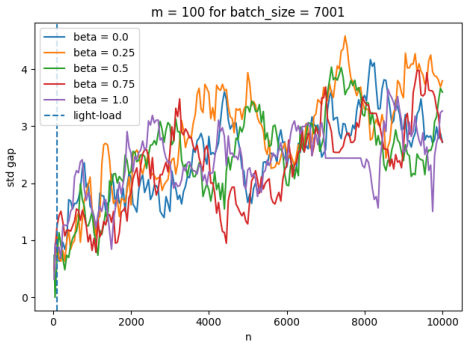

(b) $b = 2m$




(c) $b = 10m$




(d) $b = 30m$




(e) $b = 70m$

Figure 9: Evolution of the standard deviation from gap for $m = 100$ and $d = 2$ under different batch sizes $b$.

Furthermore, the variability of the gap is more similar to one-choice the larger is the batch size. Large batches introduce more randomness within each batch, since all $b$ balls are allocated based on the same initial snapshot. Consequently, small changes in bin loads amplify over the course of the batch, yielding larger fluctuations in the observed gap.

## 2.4 Analyzing the Effects of Uncertainty

In the fourth experiment, we study how partial information about bin loads affects the evolution of the gap $G_n$. We fix the number of candidate bins to

$$d = 2$$

and consider a non-batched scenario

$$b = 1,$$

so that each ball is allocated individually.

In this setting, when choosing among the two candidate bins, we do not observe their exact loads. Instead, we can ask a limited number of binary questions about the bins, such as whether a bin's load is above the median or among the top 25% or 75% of all loads.

We denote the number of allowed questions by $k$ and vary the parameter $\beta$ (probability of applying the $d$-choice strategy) as in previous experiments. This models a scenario where decisions are made with uncertainty, reflecting situations in which exact bin loads are costly or impossible to obtain.

For each choice of $m$, and $\beta \in \{0,1\}$ (having in mind that $k$ depends on $\beta$), we measure the evolution of the gap $G_n$ from the light-load regime ($n = m$) to the heavy-load regime ($n = m^2$). We expect that uncertainty increases the gap, as incomplete information may lead to suboptimal allocations. However, increasing $\beta$ reduces the impact of uncertainty, as it increases the probability of applying the two-choice strategy, improving the balance.

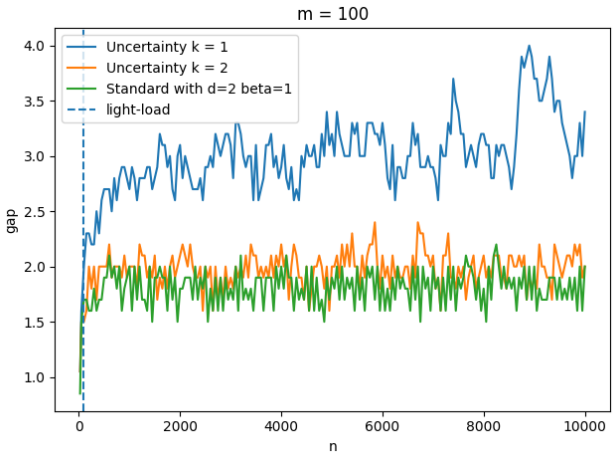



Figure 10: Evolution of the gap for $m = 100$ under uncertainty ($k = 1,2$) compared to the fully-informed case.

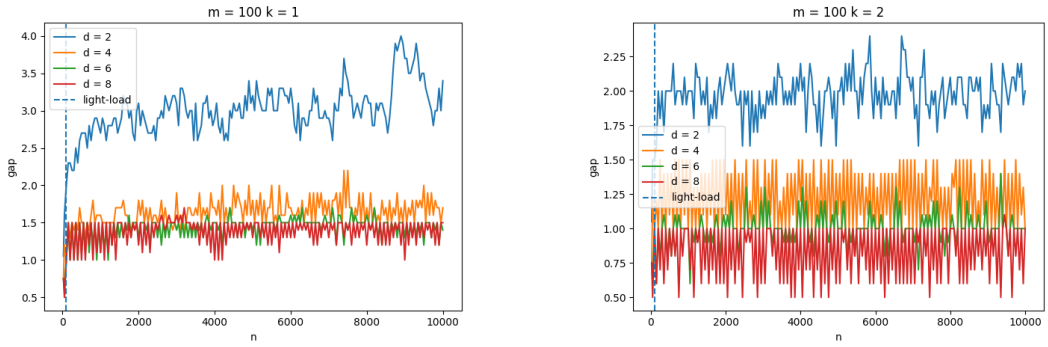
From Fig. 10, we observe that for $m = 100$ and $k = 1$, the gap exceeds 3 in the heavy-load regime. When increasing to $k = 2$, the gap drops significantly and remains close to 2, very similar (and only slightly worse) than the fully-informed case with $d = 2$ and $\beta = 1$. Thus, the improvement from $k = 1$ to $k = 2$ is substantial, while the difference between full certainty and $k = 2$ uncertainty is comparatively small.

## 2.5 Analyzing the Effects of Uncertainty $d$

In the fifth experiment, we study the effect of modifying d on the gap $G_n$ with partial observation. We focus on the standard (non-batched) setting, fixing:

$$m = 100, \qquad b = 1,$$

and varying $k \in \{1,2\}$ (the number of allowed queries) and $d$ over the values specified in Eq. (4).

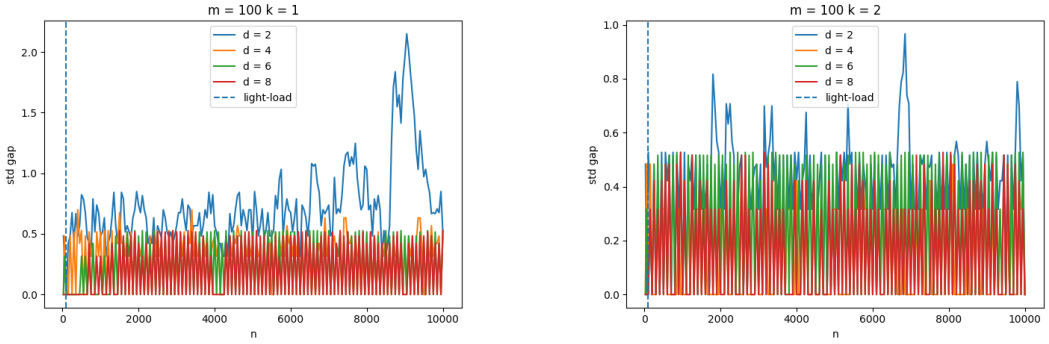



(a) $k = 1$ (b) $k = 2$

Figure 11: Evolution of the gap for $m = 100$ as $d$ varies, under $k = 1$ and $k = 2$ queries.

From Fig. 11, we observe that increasing $d$ consistently improves the gap, although, as discussed previously in Section 2.2, the marginal improvement diminishes as $d$ grows. In particular, the reduction in gap from $d = 2$ to $d = 4$ is substantially larger than the reduction from $d = 6$ to $d = 8$. This is expected: once the gap approaches 1, the system is already extremely well balanced, and further improvements become structurally limited. Indeed, if the gap drops below 1, moving any ball from one bin to another would immediately make the target bin among the most loaded, if not the most loaded, meaning the allocation cannot be made more balanced.

The improvements are more visible when $k = 1$. With fewer allowed queries, the algorithm operates with less information, and thus there is more room for improvement when increasing $d$. When $k = 2$, the selection mechanism is already significantly more accurate, so the additional information gained by sampling more bins is naturally smaller.




(a) $k = 1$ (b) $k = 2$

Figure 12: Standard deviation of the gap for $m = 100$ as $d$ varies, under $k = 1$ and $k = 2$ queries.

Finally, when examining the standard deviation in (Fig. 12) we see that increasing $d$ tends to reduce variability, although the effect is modest. However, because only five independent replicas were performed for each configuration, the resulting variance estimates are noisy, and the plots appear somewhat irregular and harder to interpret. Increasing the number of repetitions would produce smoother and more reliable curves.

## 2.6 Analyzing the Effects of Uncertainty $\beta$

In this last experiment, we study the effect of $\beta$ on the gap $G_n$ with partial observation. In this case, $\beta$ is the probability of using the selection method with k=2. We focus on the standard (non-batched) setting, fixing:

$$d = 2, \qquad m = 100, \qquad b = 1,$$

varying $\beta$ over the values specified in Eq. (3).

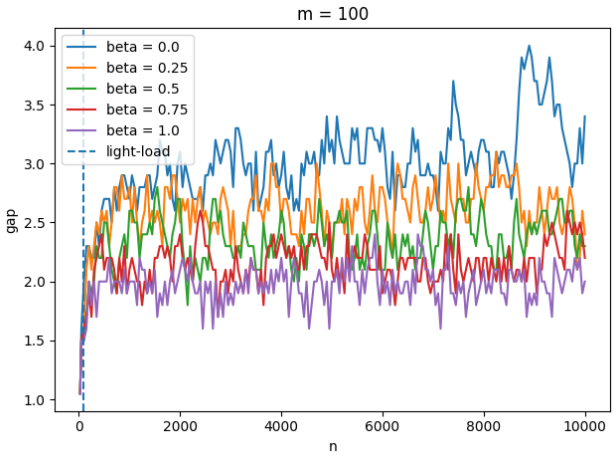



Figure 13: Evolution of the gap in an uncertainty scenario for $m = 100$ and various $\beta$.

In Fig. 13 shows how the gap improves as the algorithm relies more frequently on the $k = 2$ selection method. The extremes correspond to:

$$\beta = 0 \Rightarrow k = 1, \qquad \beta = 1 \Rightarrow k = 2.$$

As expected, increasing $\beta$ decreases the gap, since the algorithm operates with more accurate information more often. The improvement appears roughly linear in $\beta$, although with the limited number of replicas used, we cannot conclusively determine whether the scaling is perfectly linear or slightly concave.

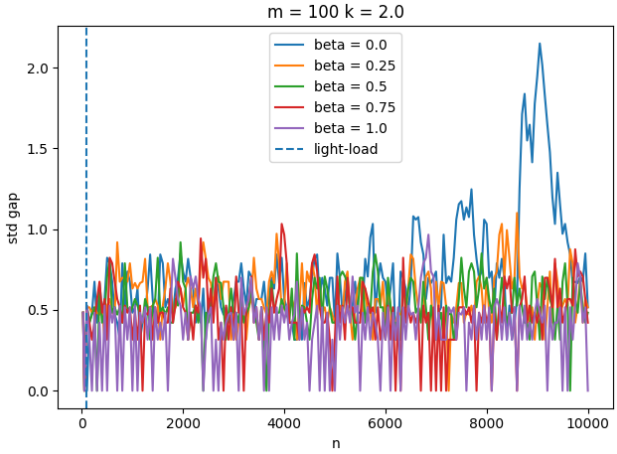



Figure 14: Evolution of the standard deviation of the gap for $m = 100$ as $\beta$ varies.

Lastly, when considering the standard deviation of the loads (Fig. 14), we observe that variability decreases as $\beta$ increases. This reflects the increased stability achieved when the selection mechanism behaves more like the $k = 2$ strategy, which uses more accurate information and thus produces more consistent allocations.

# 3 Conclusions

Across all experiments, several key conclusions emerged regarding balanced allocation under different strategies and information settings.

**Effects of $\beta$** As the number of bins $m$ increases, the influence of the parameter $\beta$ becomes significantly more pronounced. Higher values of $\beta$, corresponding to more frequent use of the $d$-choice strategy, consistently yield smaller gaps. In particular, when $m$ is moderately large (e.g., $m \geq 30$), choosing $\beta$ close to 1 results in dramatically improved balance, confirming the theoretical intuition that the $(1+\beta)$-choice scheme interpolates smoothly between purely random allocation ($\beta = 0$) and the highly efficient two-choice scheme ($\beta = 1$).

**Effects of $d$-choice** Our experiments reaffirm that the transition from $d = 1$ to $d = 2$ offers an enormous improvement in the gap—mirroring the classical jump from $\Theta(\log n / \log\log n)$ to $\Theta(\log\log n)$. However, increasing $d$ beyond 2 yields diminishing returns: improvements from 4 to 6 or 6 to 8 are minor, especially once the system is already well balanced. Thus, $d = 2$ is near-optimal in practice, providing excellent performance while keeping computation minimal.

**Effects of Batch Size** Batching introduces outdated information, which severely harms performance. When $b$ becomes large (e.g., $10m$ or $70m$), the algorithm repeatedly fills the same bins within the batch, producing large spikes in the gap. For large batches, a practical mitigation strategy is to:

- use $d$-choice at the beginning of the batch (when information is fresh), and
- switch to nearly random allocation (very small $\beta$) afterward,

or simply restrict the batch size to $b \leq 2m$ when possible. Overall, the best performance is always obtained with batch size $b = 1$, though this may not be feasible in large-scale or time-constrained systems.

**Effects of Partial Observability** When the exact loads cannot be queried, using two binary questions ($k = 2$) significantly reduces the gap compared to using only one ($k = 1$), and it is even of the same order of total observability for same $d$. The improvement changing the $k$ is substantial for small $d$ but less dramatic once the system is already well balanced. As in the fully observable setting, increasing $d$ still helps, though again with diminishing returns.

Using a probabilistic combination determined by $\beta$ (interpreted here as the probability of using the $k = 2$ rule) shows that:

$$\text{higher } \beta \;\Rightarrow\; \text{smaller gap but higher computational cost.}$$

Thus, when execution time is constrained (as in large $m$ or many repetitions), choosing an intermediate $\beta$ may offer a favorable trade-off between accuracy and runtime.