Assignment 2 - Balanced Allocations

Randomized Algorithms

Nicolás Zhao*

*Universitat Politècnica de Catalunya (UPC)
*nicolas.zhao@estudiantat.upc.edu

November 11, 2024

Source code available at `https://github.com/nicolasZhao1908/RA-balancedalloc`

1 Introduction

This study explores randomized allocation strategies for distributing a set of requests, represented as "balls," across a set of resources or "bins," aiming to minimize the maximum load difference—referred to as the *gap*—across bins. The core objective is to achieve balanced load distribution as the number of requests increases, ensuring that no single bin becomes overly burdened. The maximum gap, G_n, is defined as:

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

where $X_i(n)$ represents the load of bin B_i, indicating the deviation from the ideal average load $\frac{n}{m}$ after distributing n balls across m bins.

This investigation focuses on three primary allocation strategies, each employing different approaches to load data utilization and choice mechanisms:

- **d-choice ($d \geq 1$):** Each ball is assigned to the least-loaded of d randomly selected bins. This method improves load balance as d increases, though it may be affected by outdated information in batched allocations.
- **(1+β)-choice:** A hybrid method that combines one-choice and two-choice strategies. With probability β, a one-choice strategy (random bin selection) is applied, and with probability $1 - \beta$, a two-choice strategy is used, where the least-loaded of two bins is chosen. This strategy balances the need for load information with flexibility, making it suitable for distributed settings where real-time load monitoring may be limited.
- **Partial Information:** This approach uses minimal load information, such as threshold-based queries indicating whether a bin's load exceeds a certain level without specifying exact loads. This strategy is designed for environments where continuous, detailed load data is unavailable or costly to maintain.

We also examine the impact of *batch allocation*, where groups of balls are allocated simultaneously without updating load information during each batch. Non-batched allocation is treated as a special case with a batch size of 1. By analyzing scenarios with different batch sizes and load levels—ranging from light-loaded ($n = m$) to heavy-loaded ($n = m^2$)—we assess how each allocation strategy influences the maximum gap, G_n, in diverse settings.

This study provides insights relevant to load balancing in distributed computing and data center applications, where effective allocation strategies are crucial for maintaining performance

and managing resource distribution under varying demand conditions. By examining choice-based and data-limited methods, our findings offer guidance on selecting optimal strategies based on specific constraints in distributed environments.

2 Experiments

In this section, we evaluate the effectiveness of various load balancing strategies through a series of experiments, analyzing each method's performance under different configurations and constraints. These methods include the d-choices approach, the $(1+\beta)$-choices hybrid strategy, and the Partial Information approach, each designed to manage load distribution with varying levels of data availability and batch allocation sizes. Our experiments aim to assess the impact of choice-based and data-limited strategies on the overall balance of load distribution, particularly in settings with high request volume or communication limitations.

The d-choices method examines the effect of increasing choice options, where each allocation decision can sample multiple bins to choose the least-loaded option, effectively improving load balance as d increases. However, we observe that outdated load information can negatively affect performance in high-batch allocations, highlighting a key tradeoff between choice quantity and data freshness.

The $(1+\beta)$-choices method, which blends one-choice and two-choice strategies, is evaluated for its ability to balance load effectively with limited but timely load data, especially in networked environments where real-time load monitoring may not be feasible. This hybrid approach aims to minimize load imbalance while limiting the reliance on up-to-date data, making it particularly suitable for distributed systems with high request-to-server ratios.

Finally, the Partial Information strategy is assessed as a practical alternative for scenarios where access to complete load data is restricted. This method leverages basic load threshold queries, which provide approximate information on bin loads and guide allocation decisions without requiring continuous real-time data. Partial Information offers a balance between minimal data usage and effective load management, especially in distributed environments where inter-node communication is constrained.

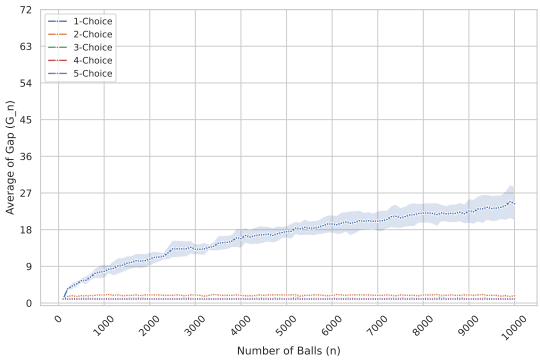
Through the analysis of each approach under varying batch sizes and load conditions, we aim to identify the strengths and limitations of each strategy. The following subsections present detailed experimental results, providing insights into the applicability of these methods across different distributed system settings and guiding the choice of allocation strategy based on specific environmental constraints.

2.1 d-Choices

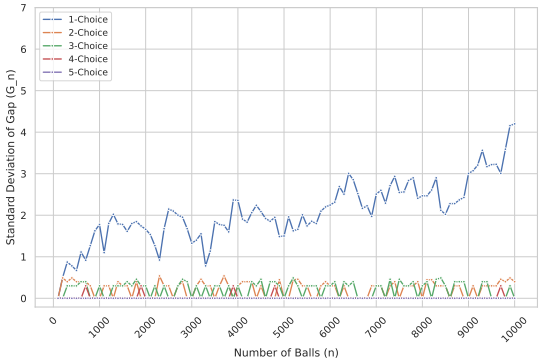
Figure 1 demonstrates how the average and standard deviation of gaps change as the number of choices d increases. The 1-choice method, which does not use bin load information, performs the worst in terms of load balancing. However, as dd increases, the allocation performance significantly improves, with lower average gaps, even for relatively low dd values compared to the number of bins mm.

For larger batch sizes, such as $b = 5m$ and $b = 20m$ (shown in Figures 2 and 3), higher dd-values lead to increased $\overline{G_n}$. This is due to the use of outdated load information, causing some bins to accumulate more load than others. Spikes in $\overline{G_n}$ for $d > 1$ in these figures reveal that once batch allocations begin, bins that initially have similar loads can end up disproportionately loaded by the batch's end. This occurs because each ball is allocated to what appears to be the least-loaded bin based on old data, leading to a "swinging" effect: one batch creates an imbalance, and in the following batch, bins mistakenly chosen as minimum in hte previous batch become less likely to be selected, and so on.

For cases where batch size is very large ($b \gg m$), as shown in Figure 4, higher d-choices

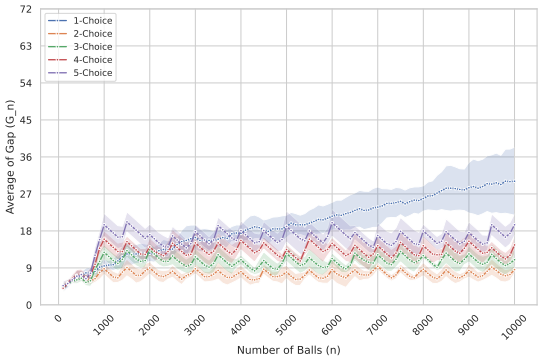


(a) Average - 1 batches

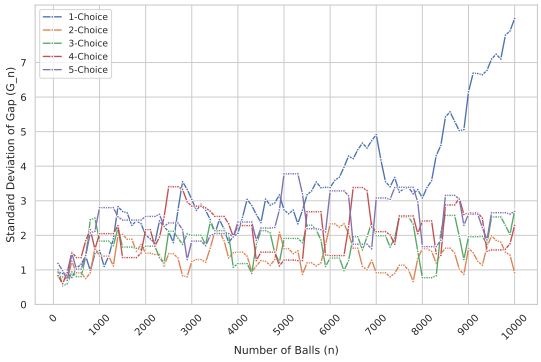


(b) Standard Deviation - 1 batches

Figure 1: d-choices performance for $n = 10000$ and $b = 1$

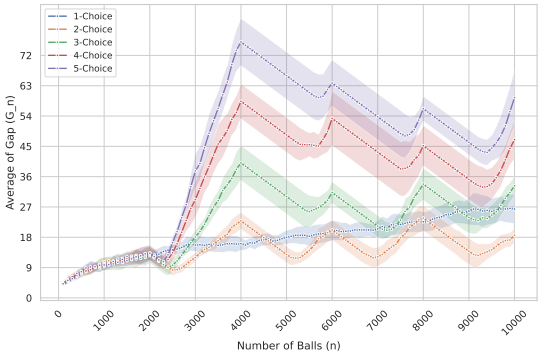


(a) Average - 500 batches

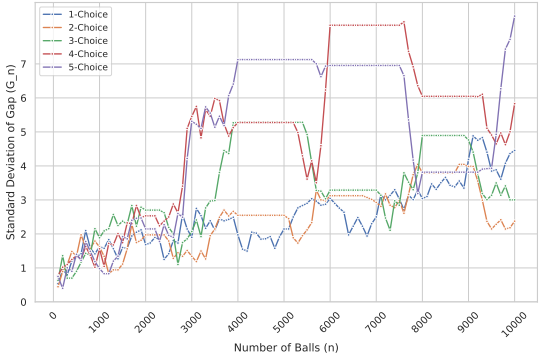


(b) Standard Deviation - 500 batches

Figure 2: d-choices performance for $n = 10000$ and $b = 500$



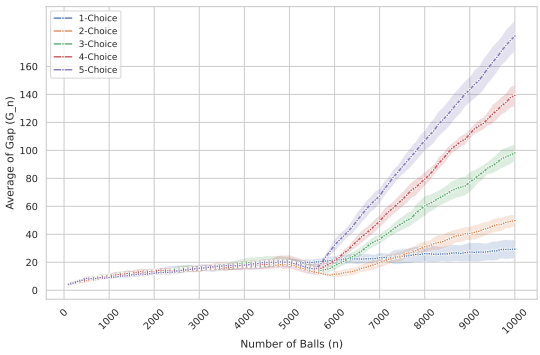
(a) Average - 2000 batches



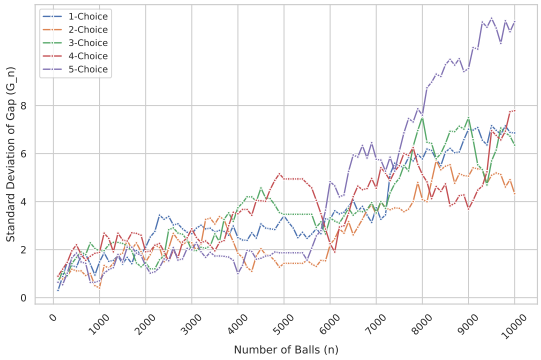
(b) Standard Deviation - 2000 batches

Figure 3: d-choices performance for $n = 10000$ and $b = 2000$

strategies eventually perform worse than the 1-choice strategy. This is because 1-choice always selecting bins randomly and do not rely on information of the chosen bins, thus it performs always the same independent of the presence of outdated information of bins.



(a) Average - 5000 batches



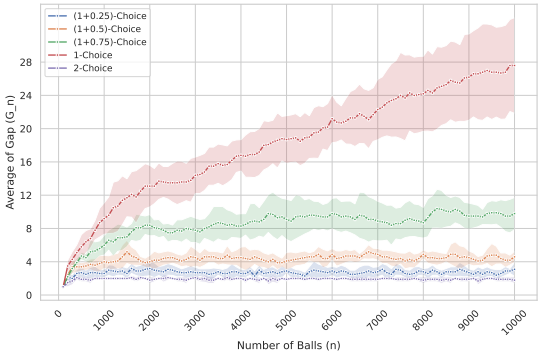
(b) Standard Deviation - 5000 batches

Figure 4: d-choices performance for $n = 10000$ and $b = 5000$

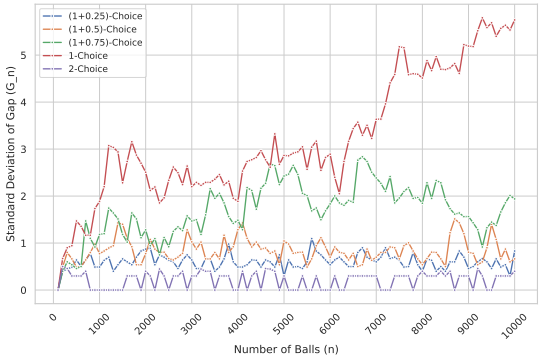
2.2 (1+β)Choices

In Figure 5, we compare $(1 + \beta)$-choice strategies with d-choices under sequential allocation ($b = 1$). Even a low frequency of two-choice decisions (e.g., $\beta = 0.75$) yields much better performance than the pure one-choice case, especially as n approaches m^2. This is significant, as full real-time knowledge of each bin's load may not be feasible in a networked system with limitations such as traffic contention, rate limits, or latency.

The effectiveness of this hybrid approach becomes even clearer in batched allocations with batch sizes $b = 5m$, $b = 20m$, and $b = 50m$ (Figures 6, 7, and 8). In environments with more requests (balls) than available servers (bins), distributing requests to servers at random but occasionally checking their current status (around 25% of the time with $\beta = 0.75$) significantly improves load balance.



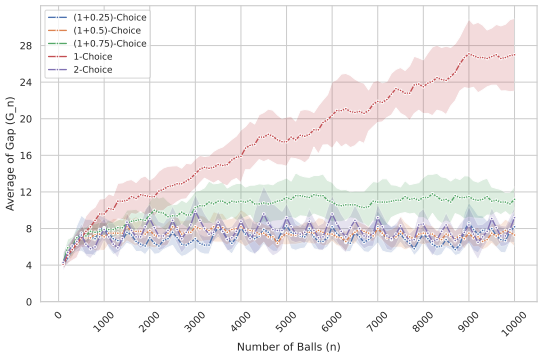
(a) Average - 1 batches



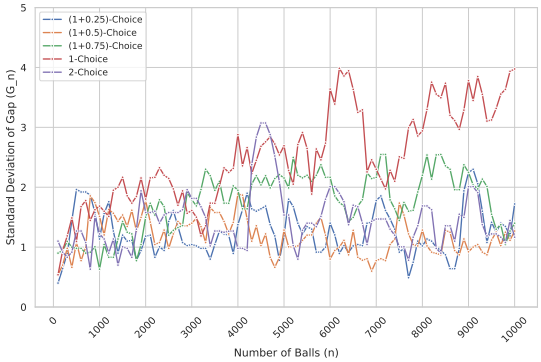
(b) Standard Deviation - 1 batches

Figure 5: d-choice vs β-choice performance for $n = 10000$ and $b = 1$

Across the tested values for $\beta \in \{0.25, 0.5, 0.75\}$, each outperforms traditional d-choice strategies in batched settings, but the case with $\beta = 0.75$ yields the best results in highly parallel allocations. This approach not only minimizes the average gap $\overline{G_n}$ for most values of n but also almost eliminates the swinging behavior seen in the 2-choice strategy, leading to more stable and balanced load distributions over time.

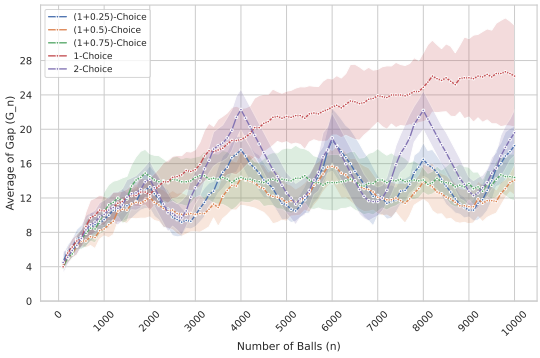


(a) Average - 500 batches

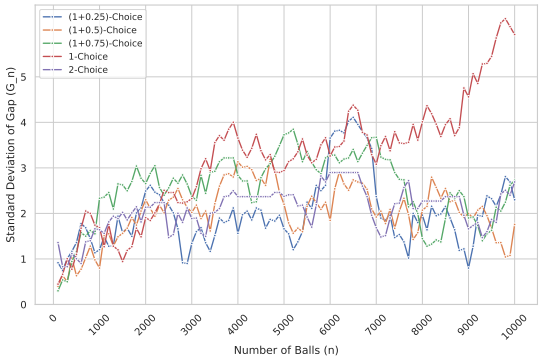


(b) Standard Deviation - 500 batches

Figure 6: d-choice vs β-choice performance for $n = 10000$ and $b = 500$

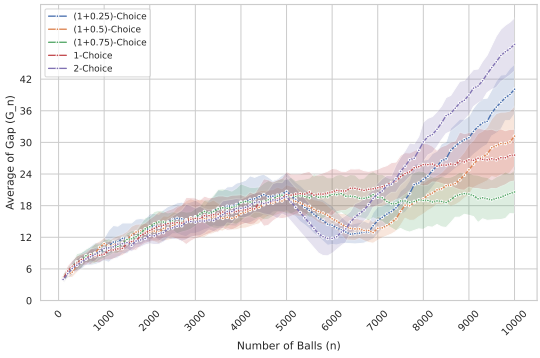


(a) Average - 2000 batches

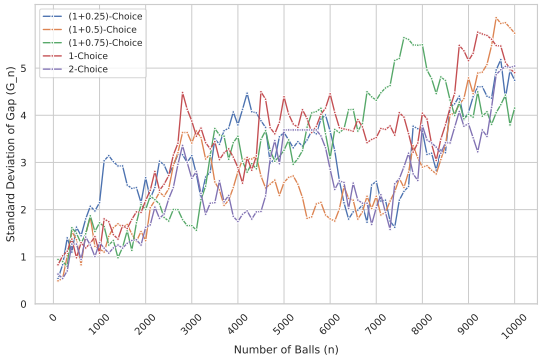


(b) Standard Deviation - 2000 batches

Figure 7: d-choice vs β-choice performance for $n = 10000$ and $b = 2000$



(a) Average - 5000 batches



(b) Standard Deviation - 5000 batches

Figure 8: d-choice vs β-choice performance for $n = 10000$ and $b = 5000$

2.3 Partial Information

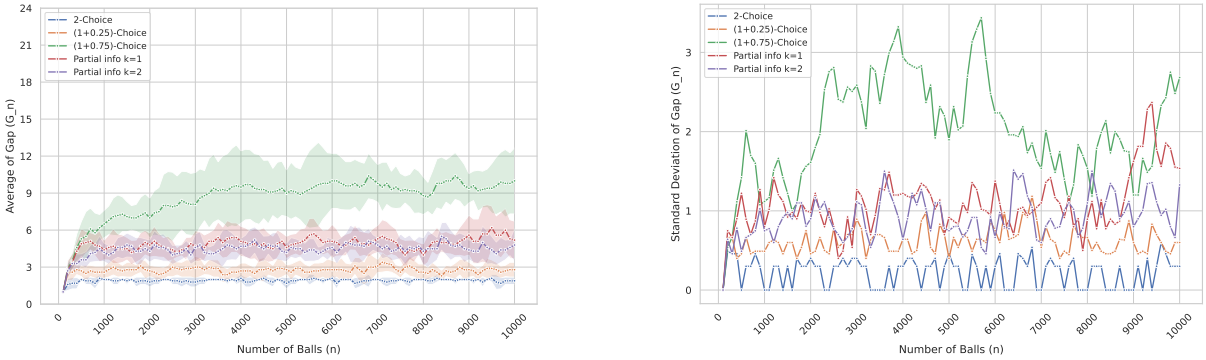
The Partial Information strategy enables allocation decisions to be made without precise knowledge of each bin's load, offering a practical approach when load data is constrained due to factors such as network latency, data access restrictions, or high communication costs. In scenarios where real-time, detailed load data is not available, this method provides an efficient alternative by relying on threshold queries to approximate load conditions.

Figures 9, 10, and 11 illustrate the performance of Partial Information under various batch sizes. Even with basic threshold checks (e.g., $k = 1$ or $k = 2$ queries, where k represents the number of threshold levels queried), this method achieves a load balance that is competitive with more data-intensive strategies like d-choice. Notably, Partial Information demonstrates higher stability, especially in high-load and batched scenarios, where each batch contains a large number of allocations without intermediate load updates.

This approach holds particular significance in distributed computing environments where access to comprehensive load information across all nodes is not feasible. In such systems, nodes typically lack detailed insights into each other's current load but may have access to global system metrics or basic threshold checks that allow them to make reasonable allocation choices. The Partial Information strategy models this real-world limitation, aligning well with scenarios where data on node states is restricted or sporadically available.

In practice, Partial Information not only reduces the communication overhead required to gather detailed load data but also achieves near-optimal load distribution by focusing on identifying heavily loaded bins and diverting allocations away from them. This method is especially beneficial in environments with high parallelism or limited inter-node communication, where more frequent data querying could otherwise lead to significant latency or congestion.

The impact of Partial Information is further evident in batched allocation experiments, where load data is updated only after each batch completes. In these cases, threshold-based decisions prevent severe imbalances that can arise from using outdated information, as seen in high-d-choice scenarios with larger batch sizes. As Figures 10 and 11 show, Partial Information maintains lower and more consistent gap values, making it particularly effective for applications in distributed systems where rapid, approximate load balancing decisions are essential.

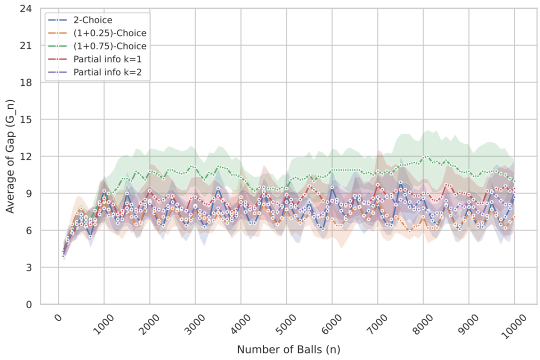


(a) Average - 1 batches

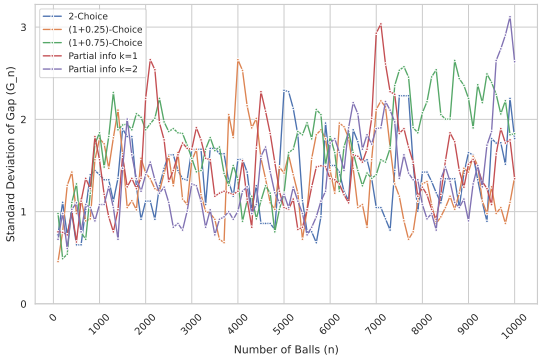
(b) Standard Deviation - 1 batches

Figure 9: Partial information vs others performance for $n = 10000$ and $b = 1$

Overall, the Partial Information strategy bridges the gap between information-intensive allocation methods and random strategies by leveraging limited load indicators to maintain efficient balance. It is particularly well suited for distributed environments that demand low communication overhead and high scalability.

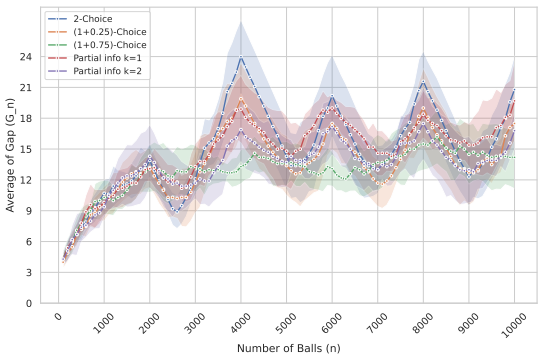


(a) Average - 500 batches

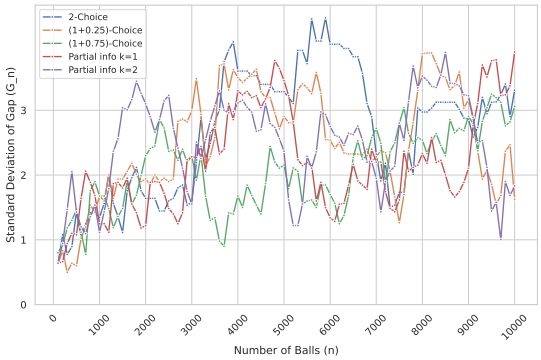


(b) Standard Deviation - 500 batches

Figure 10: Partial information vs others performance for $n = 10000$ and $b = 500$

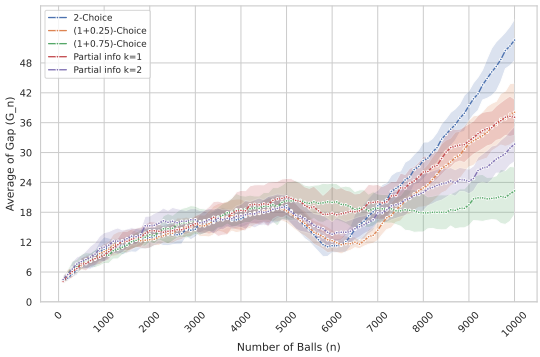


(a) Average - 2000 batches

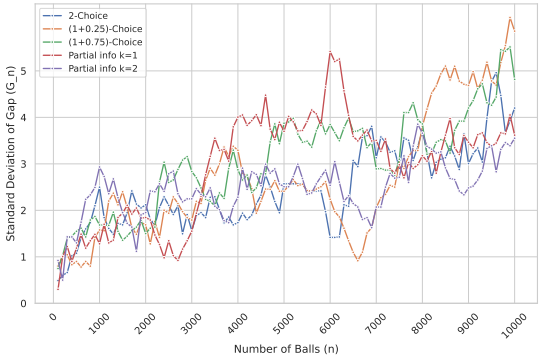


(b) Standard Deviation - 2000 batches

Figure 11: Partial information vs others performance for $n = 10000$ and $b = 2000$



(a) Average - 5000 batches



(b) Standard Deviation - 5000 batches

Figure 12: Partial information vs others performance for $n = 10000$ and $b = 5000$

3 Conclusion

This study provides insights into how various allocation methods affect the efficiency of load distribution in randomized settings. The main findings are summarized as follows:

1. **Choice-based methods**: Increasing the number of bin choices during allocation (i.e., higher d-values) typically enhances load balance, as it allows allocations to gravitate towards less-loaded bins. However, in batched allocations where load data may become outdated, an excessive d can lead to imbalances, as earlier choices are made based on stale information.
2. **Hybrid strategies**: The $(1 + \beta)$-choice approach offers robustness in high-demand or networked environments where real-time load information is limited. By incorporating a probabilistic mix of one-choice and two-choice strategies, this method strikes a balance between performance and information requirements, minimizing the risk of severe load spikes in highly concurrent environments.
3. **Partial Information utility**: In settings with restricted or delayed load data, the Partial Information strategy proves highly effective. By employing load threshold queries, this approach achieves competitive balance with minimal communication overhead, making it particularly valuable in distributed environments where nodes can only access limited information. This method aligns well with real-world distributed systems, where nodes generally lack comprehensive insights into other nodes' loads but may rely on global metrics or limited threshold checks to inform allocation decisions.

Overall, the selection of an allocation method should consider the specific constraints and demands of the environment. In distributed systems, combining the d-choice and $(1 + \beta)$-choice strategies may offer an optimal trade-off between load balancing efficiency and adaptability. For systems requiring minimal communication overhead, the Partial Information strategy serves as a viable option, ensuring effective load distribution with limited data access. Ultimately, a flexible approach that adapts to system constraints and information availability can provide the best results for balanced allocations in dynamic, high-demand settings.