Randomized Algorithms: Balanced Allocations

Vasco Gouveia

November 11, 2024

1 Introduction to Balanced Allocations

This assignment studies various allocation strategies for distributing balls into bins to achieve balanced loads across bins. Given m bins and n balls, the goal is to minimize the maximum load among bins. For this purpose, we investigate different allocation schemes that focus on minimizing the load gap G_n, particularly under light-loaded ($n = m$) and heavy-loaded ($n = m^2$) scenarios.

1.1 Allocation Schemes

We will compare several allocation strategies:

- **One-choice scheme:** Each ball is allocated to a randomly selected bin.
- **Two-choice scheme:** Each ball selects two bins randomly, and it is placed in the bin with the fewer balls. If the two bins have the same load, the ball is allocated to one of the bins chosen randomly.
- **(1 + β)-choice scheme:** With probability β, a ball is allocated using one-choice; otherwise, we use two-choice.

1.2 Batched Allocations

In the batched allocation, balls arrive in batches of size b. Each batch of b balls is allocated based on the load information available at the start of the batch. The load of each bin is only updated between batches, meaning that the decisions within a batch are based on outdated data. This delayed updating can affect the balance of the load distribution, particularly in larger batches.

The experiments using the batched allocation will consider values of b such as $m, 2m, 10m, 50m, 80m$, allowing us to analyze the impact of batch size on the load gap G_n and observe how increasing batch size affects load imbalance in both light-loaded and heavy-loaded scenarios.

1.3 Partial Information

In scenarios where only partial information about bin loads is available, the allocation decisions rely on 1 or 2 comparisons per decision:

- For $k = 1$: For every two candidate bins, a single query determines if each bin's load is above the median load. If one of them is below the median, the ball will go to that bin; if both answer the same thing, it chooses between them randomly.
- For $k = 2$: Along with the median query, we have a second query that, for each bin, tests whether its load is among the top 25% or the tail of 75% of the heaviest loads. The test refines the decision when both bins are below or above the median, making the choice to balance their allocations.

These partial information strategies introduce uncertainty in bin load knowledge, making it challenging to achieve perfectly balanced allocations.

2 Implementation of Allocation Schemes

For each allocation strategy, the program simulates the process of placing balls into bins. We track the evolution of G_n as the number of balls increases.

Algorithm 1 `d_choice` Allocation Strategy

```
function d_choice(bins, d)
    if d = 1 then
        return a random bin from bins
    end if
    sample ← randomly select d bins from bins
    min_value ← minimum load in sample
    min_bins ← bins in sample with load equal to min_value
    return a random bin from min_bins
end function
```

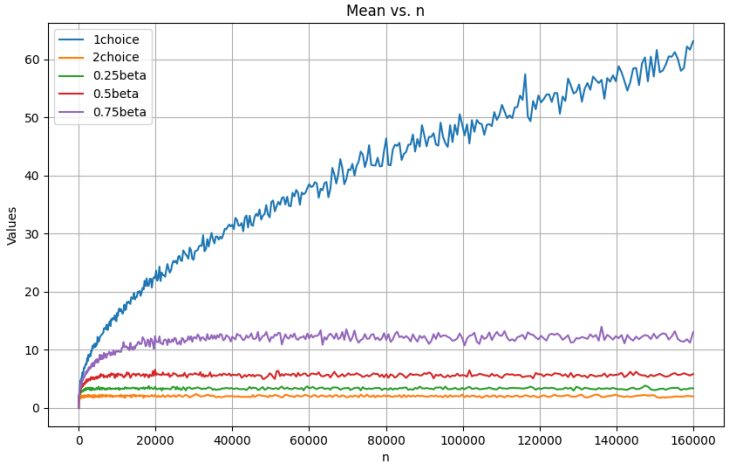
Algorithm 2 `beta_choice` Allocation Strategy

```
function beta_choice(bins, β)
    r ← random number from uniform distribution in [0, 1]
    if r < β then
        d ← 1
    else
        d ← 2
    end if
    return d_choice(bins , d)
end function
```

3 Experiments and Results

Empirical results are gathered for each scheme, comparing the evolution of G_n in both light-loaded and heavy-loaded scenarios. Plots highlight the trends and illustrate the effectiveness of each strategy in balancing loads. All scenarios are tested with m = 400 and T = 20

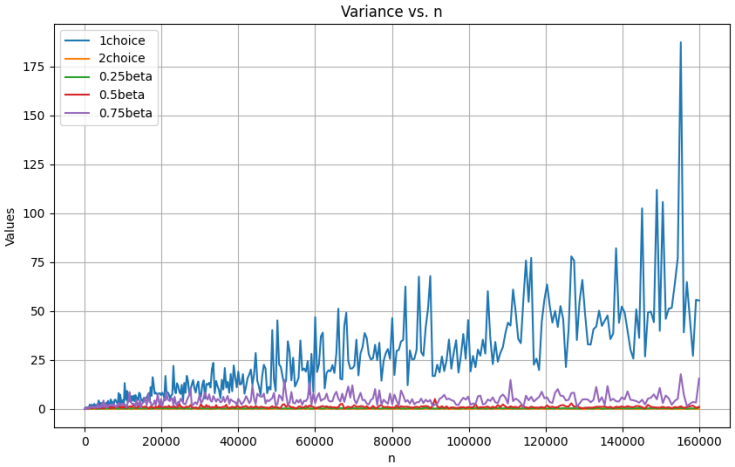
3.1 Standard Allocations

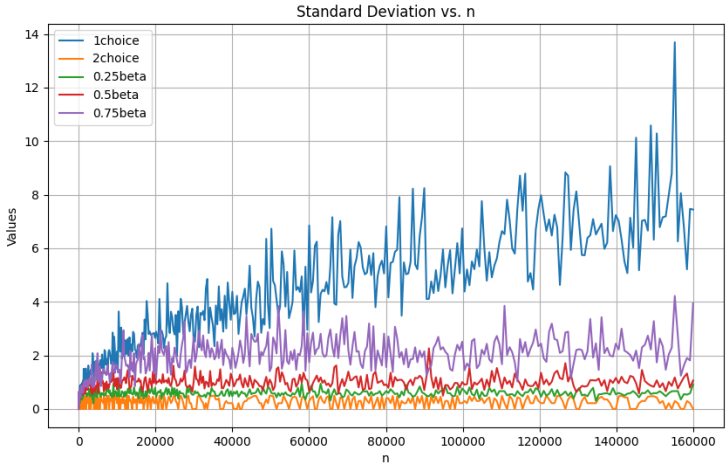


Gap	1 Choice	2 Choice	0.25 Beta	0.5 Beta	0.75 Beta
light-load	3.75	1.85	2.45	3.0	3.45
high-load	63.15	2.0	3.4	5.85	13.05

Table 1: Highlighted loads

- **Light-Load Scenario:** The Two-Choice scheme achieves the smallest load gap of 1.85, indicating strong load balancing under minimal load conditions.
- **High-Load Scenario:** In the high-load scenario, Two-Choice maintains an exceptionally low gap of 2.0, significantly outperforming all other strategies. This reinforces that Two-Choice is the most effective allocation method for minimizing load imbalance, regardless of load intensity.

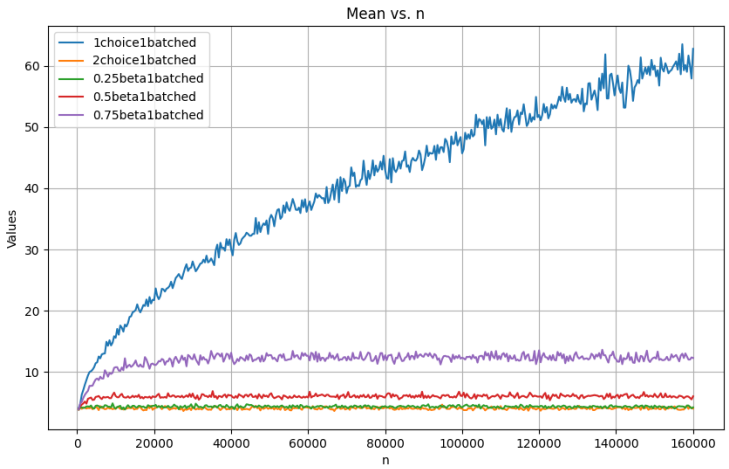




- **Standard Deviation and Variance:** One-Choice shows the highest standard deviation and variance, which increase significantly with n. This reflects substantial load imbalances and variability, confirming that One-Choice is ineffective for load balancing

3.2 Batched Allocations

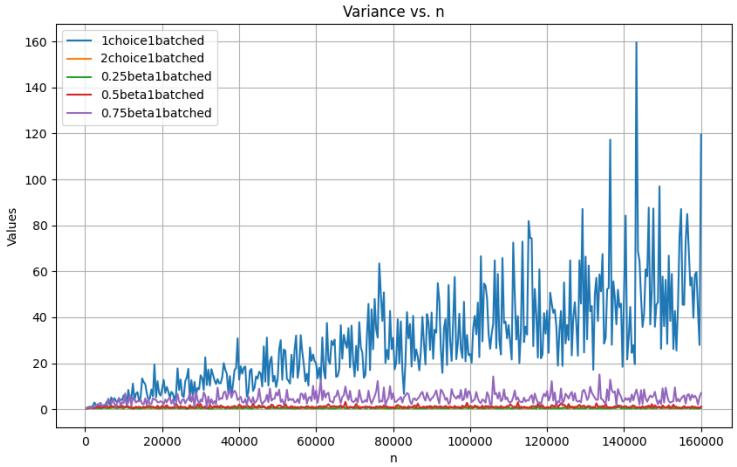
3.2.1 b=1*m

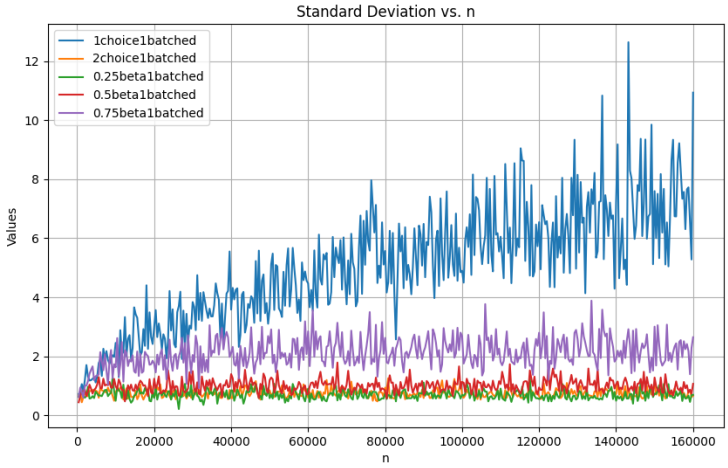


Gap	1 Choice	2 Choice	0.25 Beta	0.5 Beta	0.75 Beta
light-load	4.15	3.85	3.9	4.0	3.9
high-load	62.75	4.05	4.25	6.05	12.3

Table 2: Highlighted loads

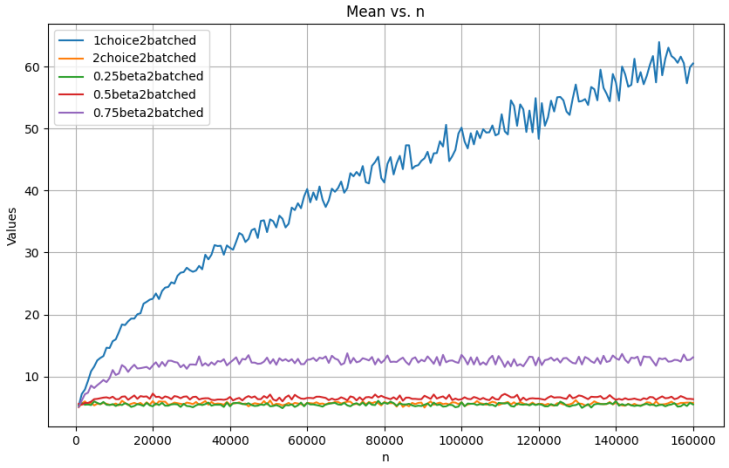
- **Light-Load Scenario:** Although the Two-Choice scheme achieves the smallest load gap again of 3.85, all the schemes showed really close results.
- **High-Load Scenario:** Two-Choice with a gap of 4.05, again still outperforms, but 0.25 Beta shows closer results.





- **Standard Deviation and Variance:** Doesn't seem to have a significant change in regards to the previous experiment.

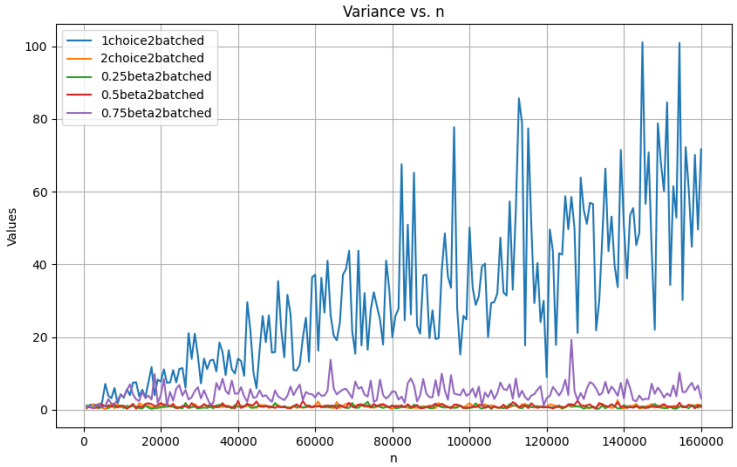
3.2.2 b=2*m

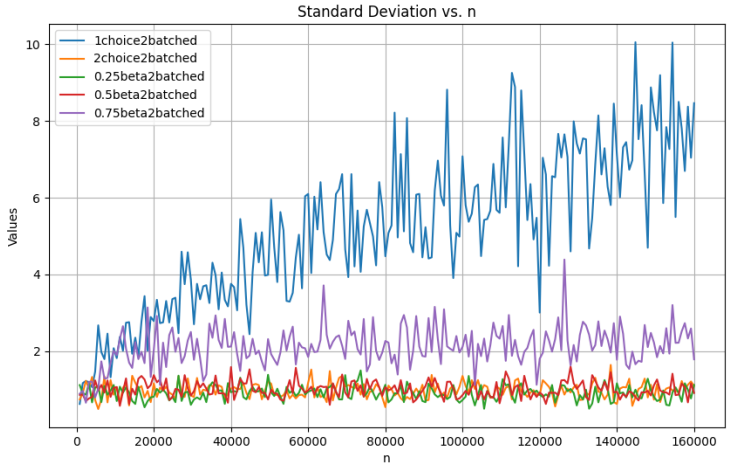


Gap	1 Choice	2 Choice	0.25 Beta	0.5 Beta	0.75 Beta
light-load	None	None	None	None	None
high-load	60.2	5.75	5.5	6.35	13.1

Table 3: Highlighted loads

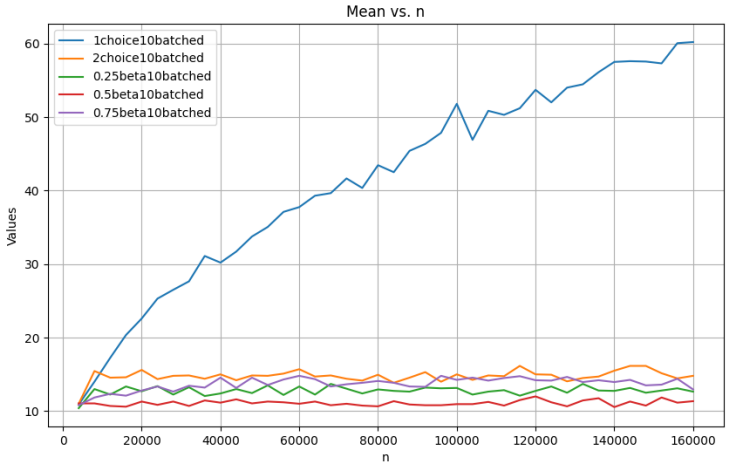
- **Light-Load Scenario:** There is no light-load information due to the batches being bigger than m.
- **High-Load Scenario:** In this experiment 0.25 Beta resulted a better load balance than 2 Choice





- **Standard Deviation and Variance:** Doesn't seem to have a significant change in regards to the previous experiment.

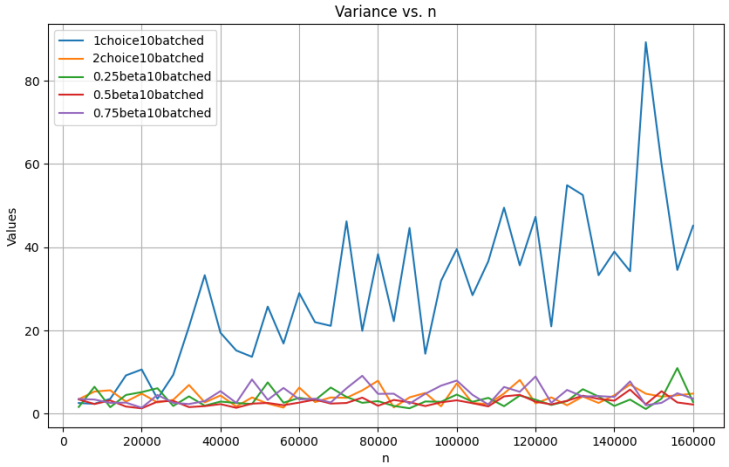
3.2.3 b=10*m

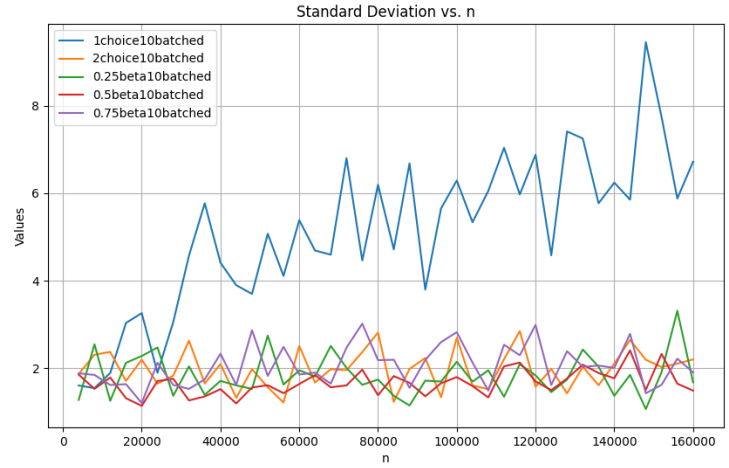


Gap	1 Choice	2 Choice	0.25 Beta	0.5 Beta	0.75 Beta
light-load	None	None	None	None	None
high-load	60.2	14.8	12.65	11.35	12.95

Table 4: Highlighted loads

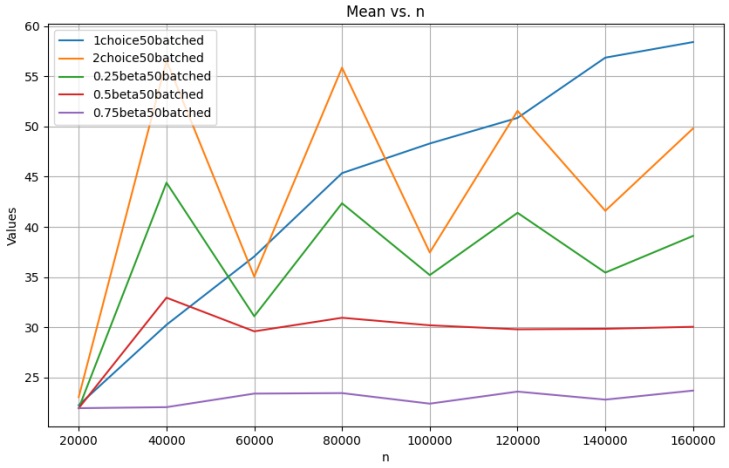
- **Light-Load Scenario:** There is no light-load information due to the batches being bigger than m.
- **High-Load Scenario:** In this experiment we get 0.5 Beta having the best load balance with a gap of 11.35. A good thing to note is 0.75 Beta remained stable in comparison to last experiment (b = 2*m)





- **Standard Deviation and Variance:** Doesn't seem to have a significant change in regards to the previous experiment.

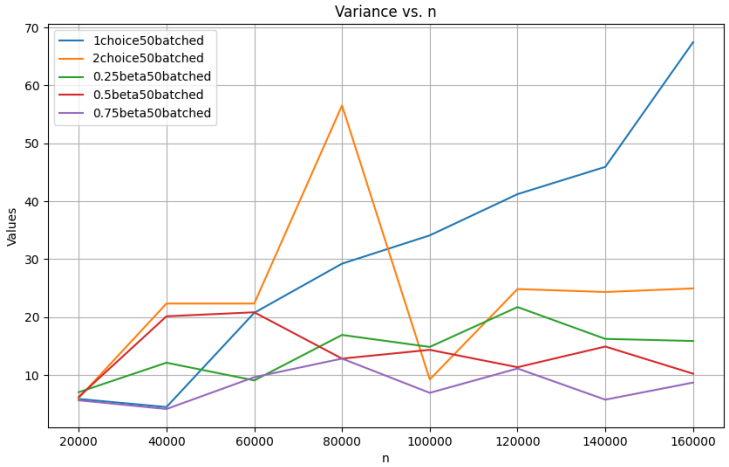
3.2.4 b=50*m

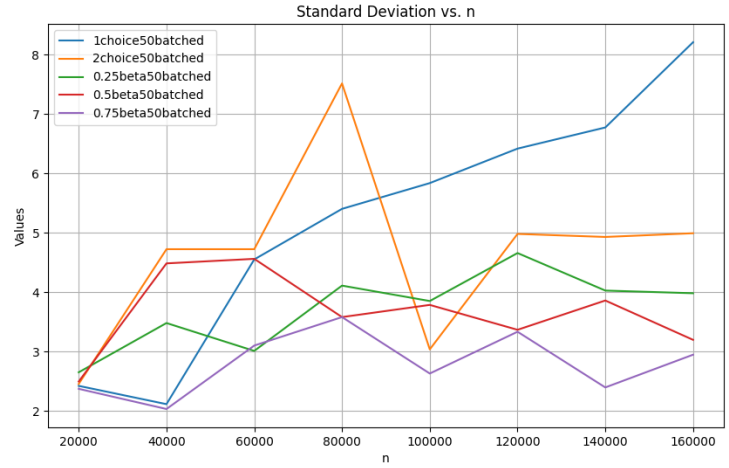


Gap	1 Choice	2 Choice	0.25 Beta	0.5 Beta	0.75 Beta
light-load	None	None	None	None	None
high-load	58.4	49.8	39.1	30.05	23.7

Table 5: Highlighted loads

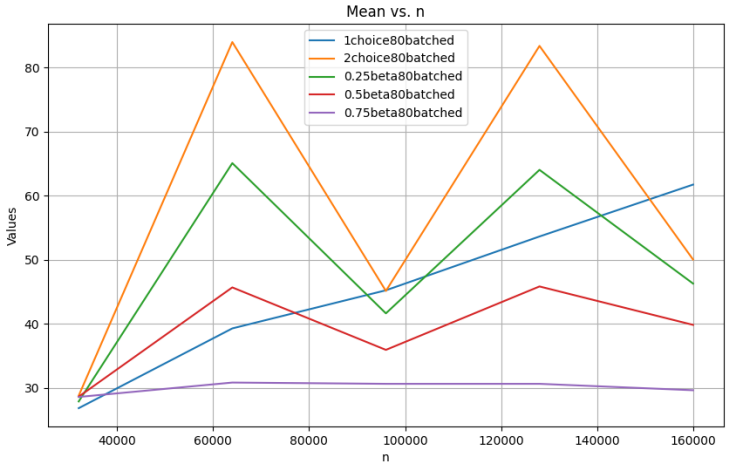
- **Light-Load Scenario:** There is no light-load information due to the batches being bigger than m.
- **High-Load Scenario:** 0.75 Beta has a greater load balance compared to any of the other allocation schemes. Furthermore, for the first time, we start to notice that for some loads all the other schemes under perform the one-choice.





- **Standard Deviation and Variance:** Though the mean of 0.75 Beta is very sable, it has still a bit of variance.

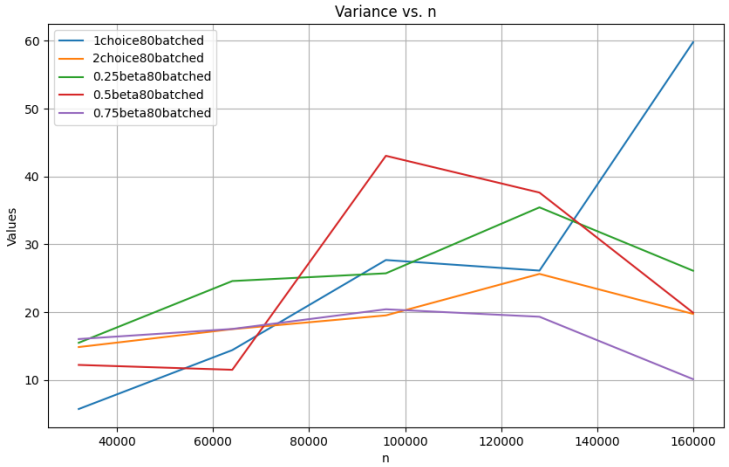
3.2.5 b=80*m

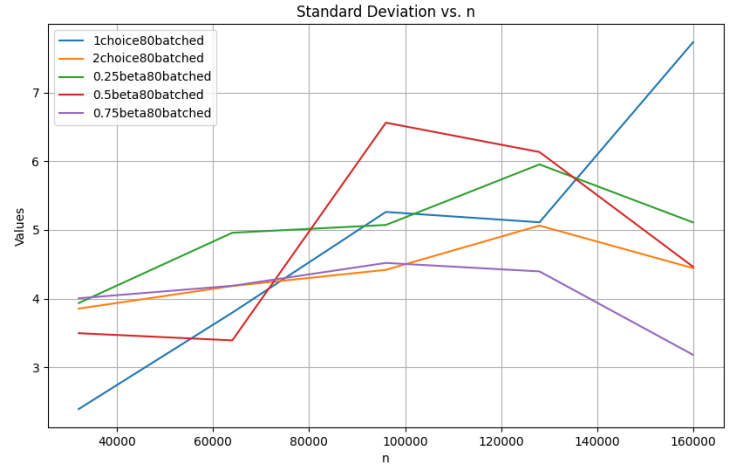


Gap	1 Choice	2 Choice	0.25 Beta	0.5 Beta	0.75 Beta
light-load	None	None	None	None	None
high-load	61.75	50.05	46.3	39.85	29.65

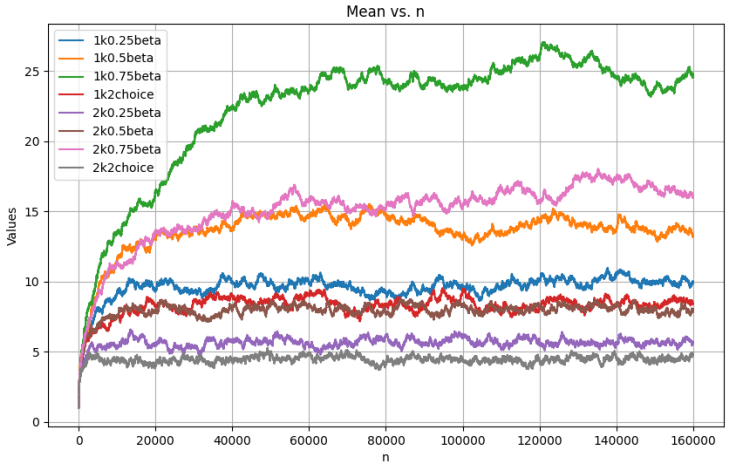
Table 6: Highlighted loads

- **Light-Load Scenario:** There is no light-load information due to the batches being bigger than m.
- **High-Load Scenario:** Here, the consistency of 0.75 Beta across all loads is evident. In contrast, the other schemes show a different result, with a clear correlation between instability and those schemes that rely more heavily on the two-choice.


Variance vs. n
1choice80batched
2choice80batched
0.25beta80batched
0.5beta80batched
0.75beta80batched
Values
n
60
50
40
30
20
10
40000
60000
80000
100000
120000
140000
160000


Standard Deviation vs. n
1choice80batched
2choice80batched
0.25beta80batched
0.5beta80batched
0.75beta80batched
Values
n
7
6
5
4
3
40000
60000
80000
100000
120000
140000
160000

3.3 K Based Allocations



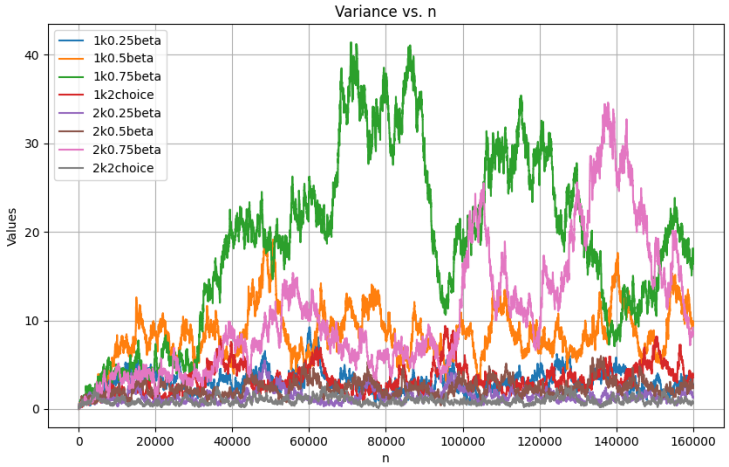
Gap	2 Choice	0.25 Beta	0.5 Beta	0.75 Beta
light-load	3.6	3.85	4.15	3.9
high-load	8.4	9.95	13.2	24.8

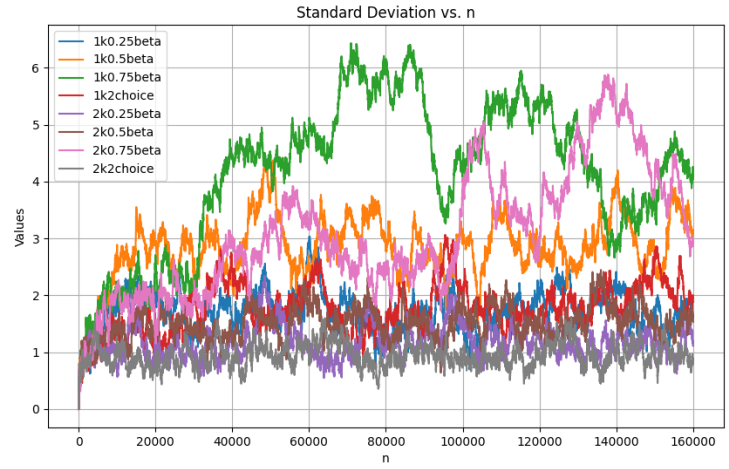
Table 7: k=1 Highlighted loads

Gap	2 Choice	0.25 Beta	0.5 Beta	0.75 Beta
light-load	3.5	3.3	3.65	3.8
high-load	4.8	5.6	8.95	15.95

Table 8: k=2 Highlighted loads

- **Light-Load Scenario:** All schemes seem to behave relatively the same with 2k and 0.25 Beta being the best with 3.3.
- **High-Load Scenario:** We can observe that the more heavily it relies on 1 choice, the worse the outcome may be. Also we can also notice the k = 2 gets a better result in every same scheme.





4 Conclusion

- **Standard Allocations:** The study confirms that allocation schemes with multiple bin choices (two-choice) significantly reduce the load gap compared to one-choice, as proven in the table 1. This is result was predictable, as the more we rely on two-choice allocation (lesser the beta) we can better balance the loads.
- **Batched Allocations:** When the batches are smaller, the two-choice scheme performs better. However, as the batch size increases, introducing a slight reliance on the one-choice strategy proves effective. This can be explained by the fact that

we don't update the bin status during the batch process. Randomized allocation helps maintain balance by preventing overloading the bin that initially had fewer balls at the beginning of the batch. In fact, this overfeeding behavior and its correction becomes apparent when the batch size reaches 50*m 3.2.4 and 80 * m 3.2.5. The one-choice allocation, as it doesn't depend on the data being updated, it remained the same between every batch value.

- **K Based Allocation:** For light loads, the chosen scheme doesn't seem to make much of a difference. However, as the load increases, it becomes evident that setting k = 2, relying on two choices, gets the best performance. This outcome was predictable, as the more comparisons we make between the load of the bins, the more evenly the allocation tends to be distributed, just like in the Standard Allocation.

5 Repository

The code for this assignment can be found at: https://github.com/ertem0/RA-balanced-alloc