POLITECNICO DI TORINO

Computer aided simulations and performance evaluation

Academic year 2020/21

VASSALLO Maurizio, s276961

Contents

Bins & Balls

1.1 Introduction

This experiment involves a number N of bins and balls: for each ball, some random bins are chosen, one or more than one, depending on the dropping policy; given those bins a ball is put in one of them. The goal is to evaluate the maximum bins occupancy and compare the results with the theoretical ones.
There are 2 dropping policy:

- Random Dropping: for each ball a single random bin is chosen and the ball is put in it;
- Random Load Balancing: for each ball d random bins are chosen and the ball is put only in the one with the lowest occupancy. In this simulation the values of d used are 2 and 4.

1.2 Tasks

1.2.1 Input Parameters

The input parameters of the simulation are:

- Number N of bins and balls;
- The seed value used to initialize a pseudorandom number generator;
- The confidence level used to calculate the confidence interval;
- The number of runs: the number of times that we run our simulation. This is done in order to have more accurate results.

1.2.2 Output Metrics

The output metrics of the simulation are:

- Number N of bins and balls;
- The lower confidence interval value;
- The average max occupancy value;
- The upper confidence interval value;
- The relative error.

All these value are stored in a file and each field is tab separated.

1.2.3 Main data structure

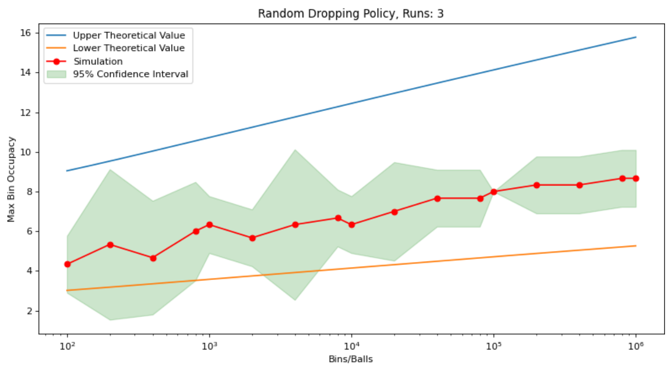
The data structure used is a numpy array of integers. This array has length *num_bins* and it stores the bins occupancy. This data structure allows to have a constant access time ($\mathcal{O}(1)$) and also a worst-case constant access time ($\mathcal{O}(1)$).

1.2.4 Simulator Inputs & Outputs

The whole simulation runs inside a script where the simulator runs multiple times for multiple values of bins and balls.
The output is a .dat file that contains the output metrics, therefore this file will contain a number of lines equal to the number of different values of bins and balls used. This .dat file is then elaborated by scripts in order to create some plots.
There are 3 scripts for plotting:

- One plots the results of the simulation in order to have a comparison with the theoretical occupancy values;
- One plots the different performances of the dropping policy;
- One plots the relative errors for different values of the number of runs.

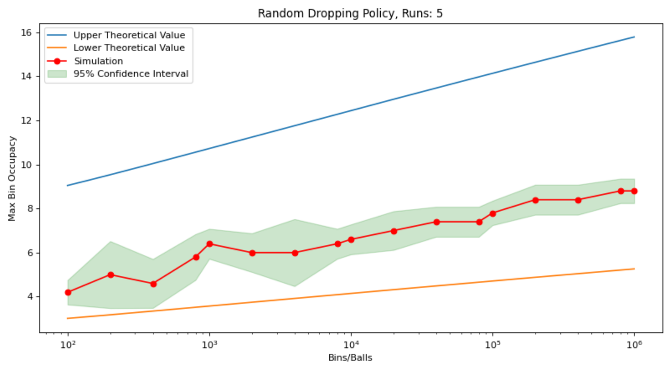
1.2.5 Coherence Simulation & Theoretical Formula



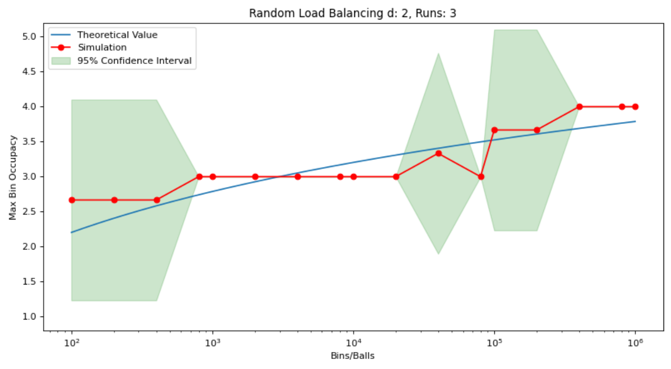
It is possible to see that 3 runs are not enough since the confidence interval is not inside the theoretical values.
The lower bound is calculated using the formula:

$$\text{expected\_max\_occupancy} = \frac{\log n}{\log \log n} \tag{1.1}$$

where n is the number of bins and balls,
while the upper bound is just 3 times this formula.



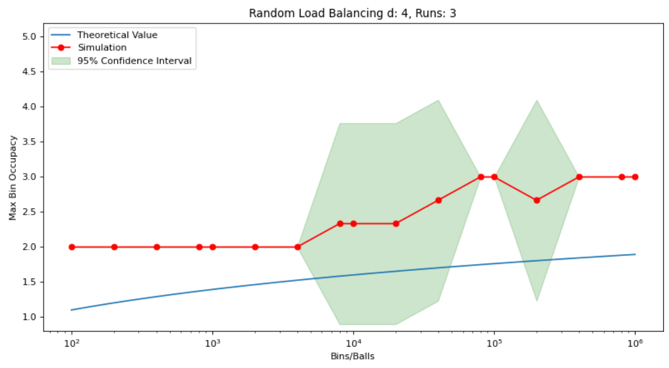
It is possible to see that 5 runs are enough since the confidence interval is inside the theoretical values.



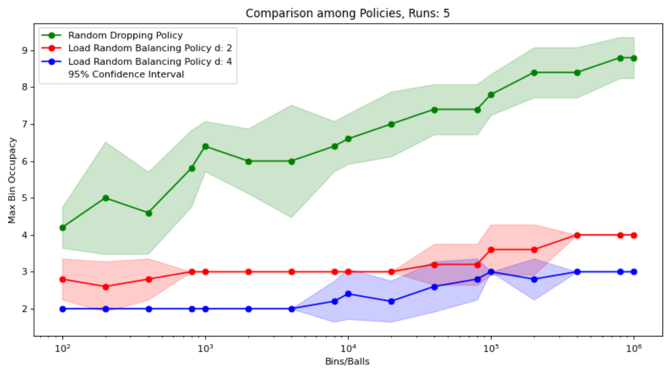
Even with 3 runs the Load Balancing with *d=2* is near to the theoretical value.

$$\text{expected\_max\_occupancy} = \frac{\log \log n}{\log d} \tag{1.2}$$

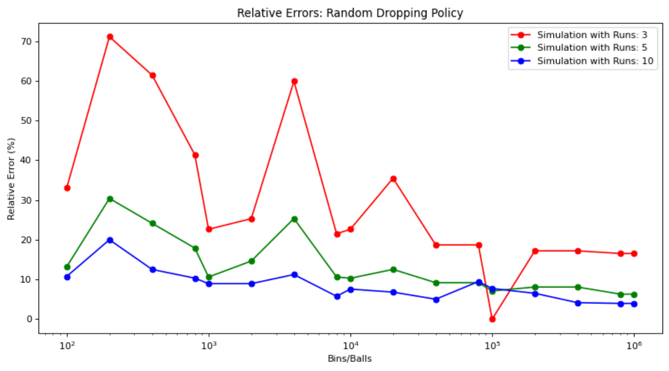
Where n is the number of bins and balls and d the number of random bins chosen.
It is possible to see that this formula is much smaller than formula 1.1, so we expect a lower occupancy value.



Even with 3 runs the Load Balancing with *d=4* is quite near to the theoretical value. It is possible to see that the average occupancy is lower with respect to the Load Balancing with *d=2* (The 2 plots have the same y-axis scale).



With this graph it is possible to see the differences between the policies and in particular: the Balancing policy works better than the Random one and that with an increasing number of bins selected the maximum occupancy decreases.
There must be a trade-off between the number of bins selected and the script execution time because with large values of d the occupancy decreases but the execution time increases: in the extreme case with *d=#bins* we would put the ball in the least occupied bins (similar to execute *np.argmin(bins)*) and have an average maximum occupancy of 1 (one ball in each bin) but that would require some time, especially for large number of bins.



It is possible to see that the relative errors decrease with increasing the number of runs.

$$\text{rel\_error} = 2\frac{\Delta}{x} \tag{1.3}$$

where Δ is the half the *CI* length.

Similar results are obtained with the Loading Balancing policy; graphs are omitted.

Birthday Paradox

2.1 Introduction

This experiment involves a number m people: for each person a random number, the birthday in the case of Birthday Paradox, is chosen among n possible values.
The goal is to evaluate:

- The probability of conflict. A conflict is experienced when two people have the same random number (same birthday);
- The minimum number of people required for a conflict.

2.2 Tasks

2.2.1 Input parameters

The input parameters of the simulation are:

- Number of possible "days": in this simulation, this value can be: [365, 10^5, 10^6];
- The seed value used to initialize the pseudorandom number generator;
- The confidence level used to calculate the confidence interval;
- The number of runs: the number of times that we run our simulation. This is done in order to have more accurate results;
- A flag depending if we want to calculate the conflict probability or the minimum number of people needed to experience a conflict.

2.2.2 Output metrics

The output metrics of the simulation are:

1. Conflict Probability:
    - Number N of persons;
    - The lower confidence interval value;
    - The average max occupancy value;
    - The upper confidence interval value;
    - The relative error.
2. Minimum number of people:
    - The lower confidence interval value;
    - The average number of people;
    - The upper confidence interval value;
    - The relative error;
    - The theoretical value.

All these value are stored in a file and each field is tab separated.

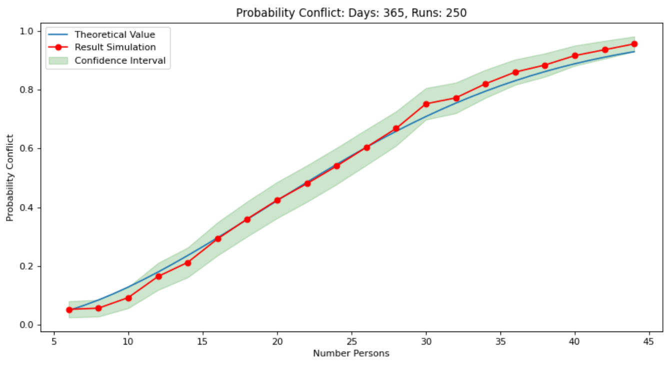
2.2.3 Simulator main data structure

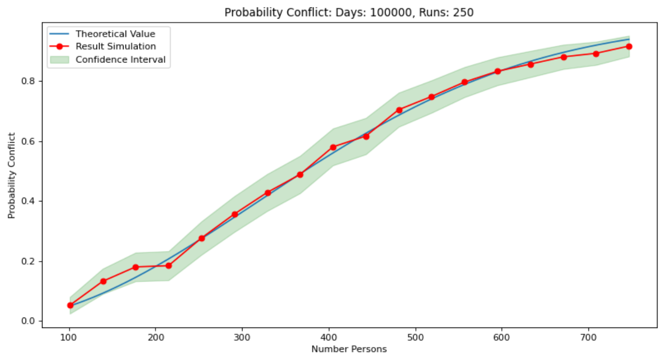
The data structure used is a numpy binary array to store whether the element (day) at position i is occupied or not (in the case of Birthday Paradox, there is already, at least, one person who was born in that day).
This data structure allows to have a constant access time ($\mathcal{O}(1)$) and also a worst-case constant access time ($\mathcal{O}(1)$).
For the calculation of the probability there is a counter that keeps how many conflicts happened for a given number of people. This is used to calculate the probability as $prob(conf) = \frac{\#conflicts}{\#people}$ at each run.

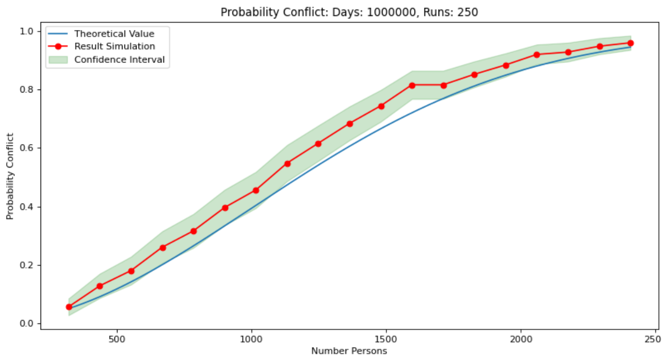
2.2.4 Simulator inputs and outputs

The whole simulation runs inside a script. The simulator runs each time for different values of people or just one in the case of finding the minimum number of people for a conflict.
At each iteration the output is a .dat file that contains the output metrics, therefore this file will contain a number of lines equal to the number of different values of people or just one line in the case of minimum case. For the first case, the .dat file is then elaborated by a script to create the plots.

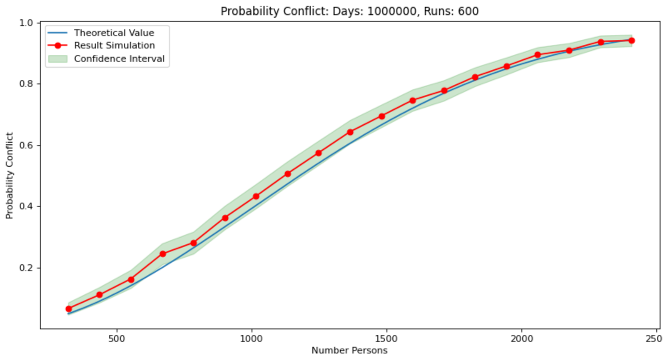
2.2.5 Coherence simulation & theoretical formula







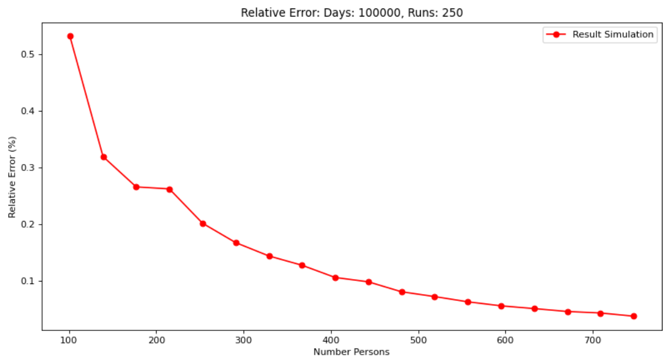
For 10^6 case, even with 250 runs the simulation is not very close to the theoretical result.



Better results can be obtained increasing the number of runs but such a huge number of runs requires more time to be executed.

2.2.6 Is the theoretical formula for the conflict probability accurate?

Yes, how it can be seen from the previous graphs, the formula is accurate but for large number of days it is less accurate and it requires an higher number of runs.



It is possible to see the accuracy of the theoretical formula through the relative errors plot that the they tend to 0.

2.2.7 Required elements given a priori probability

It is possible to calculate the number of elements using the following formula:

$$num\_elements = \sqrt{2 \times number\_days \times \log\left(\frac{1}{1-p}\right)}$$

where p is the wanted probability, $p \in [0, 1)$.

With this formula it is possible to find all values of m given a value of p, the only value non possible to find is p=*1* since this is not allowed in the formula (division by 0) but we have 100% probability of conflict if $number\_days$+1 elements (people) are chosen.

2.2.8 Summary results

n	**num runs**	**Avg num people for a conflict**	**95% CI**	**Theo. value**	**MAE**
365	250	22.73	1.44	23.94	1.21
10^5	250	401.72	27.13	396.33	5.39
10^6	250	1154.26	79.76	1253.31	99.05
10^6	600	1230.41	51.70	1253.31	22.90

The number of runs are chosen such that the relative errors (1.3) were under 0.15 but in case of 10^6 days the relative error is a bit smaller than the others (0.11 vs 0.14) since with 250 runs the MAE value was greater than the CI value.