Université de Genève

Metaheuristics for optimization

TP 0: Stohastic processes

Author: Joao Filipe Costa da Quinta

E-mail: Joao.Costa@etu.unige.ch

September 24, 2021

UNIVERSITÉ DE GENÈVE

FACULTÉ DES SCIENCES

Département d'informatique

Simulation of a balanced dice

Intro

During this exercise we will be simulating rolls of a N-face balanced dice. This means that the dice has N possible outcomes at each throw, all faces are equally likely with a probability of 1/N.
For each roll we need to do the following three tasks:

(1) compute a new random value $r \in [0, 1)$, done with random() function from random package

(2) compute $i = N * r$

(3) compute $\lfloor i \rfloor$, done with floor() function from math package

After doing the three tasks, we get a value, that is the simulated roll.
The following graphic represents this phenomenon for N = 6, 15

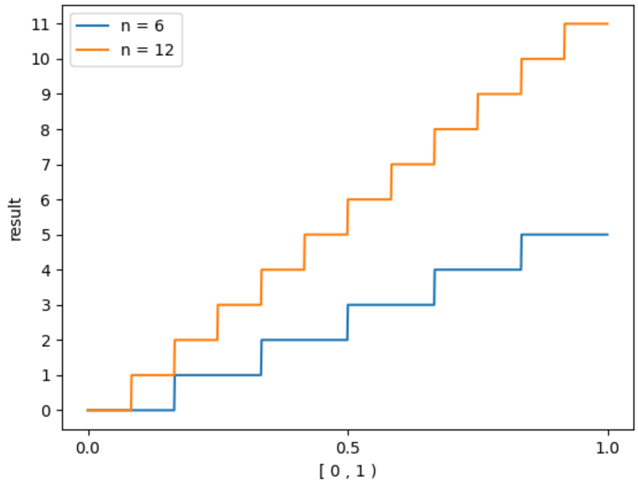

Figure 1: Values for N = $\{6, 12\}$

In the image above we see this kind of step function, this is due to the fact that we use the function .floor(). The function that does this computation can be found in the functions.py file, and its signature is the following:

balancedDice(N), where N corresponds to the number of faces in the balanced dice.

To check if the function works well we simply use it a large amount of times, or n times, if all faces are represented the same amount of times we know it is a good simulation. Obviously we need n to be large enough.

Results

For our results we will run the script with N = 6, and n = $\{100, 1000, 10000\}$.
To 'read' the results, we will be computing the frequency of each face. Let's say the face represented by the value 1 was seen 20 times out of 100 rolls, then its frequency is $20/100 = 0.2$.

Face	1	2	3	4	5	6
Frequency: 100 rolls	0.18	0.21	0.15	0.18	0.15	0.13
Frequency: 1000 rolls	0.184	0.17	0.148	0.165	0.167	0.166
Frequency: 10000 rolls	0.1742	0.167	0.1671	0.1663	0.1641	0.1613

Discrete Case