TP 0 - Stochastic processes

Metaheuristics for optimization

September 16, 2019

Throughout the semester, you will have to implement algorithms that require efficient event generation with given probabilities. This series is an introduction to several possible methods. The problem is always the following : *let us consider N possible events occurring with probability P_i, $i \in \{0, ..., N-1\}$.* Generate a sequence of events such that each event occurs with the appropriate probability. The problem is not so trivial, as we will see through examples.

Simulation of a balanced dice

First, we simulate rolling a N-face balanced die. Each side of the die has a $1/N$ probability of being obtained.
For each roll of the die, generate a random number $r \in [0.1)$, and convince yourself that the event is $i = \lfloor rN \rfloor$. Events can be visualized as "drawers" in the space of possible random numbers. This case is very simple, because it is assumed that the operations involved can be performed in $O(1)$. Generate a long sequence of rolls of a six-sided die, and check that the frequency of appearance of each side is $\frac{1}{6}$.

Simulation of a biased coin toss

A coin has a probability p of falling on 'tail' and a probability $1-p$ of falling on 'head'. When $p \neq 0.5$, the coin is biased. To simulate the throw of a coin, generate a random number $r \in [0.1)$, and use the variable $\lambda = \lfloor r + p \rfloor$. What are the possible values of λ, and with what probabilities are they obtained ? Finally, use the following variable $x = \lambda a + (1-\lambda)b$ (a for 'tail' and b for 'head') to verify that the part you generated is biased with the p probability you defined.

Simulation of a double biased coin toss

This time, two coins have a probability p_1 and p_2 of falling on 'pile', and a probability $1-p_1$ and $1-p_2$ of falling on 'face'. Based on the previous exercise, explain how to simulate a double coin toss. Check that your reasoning is the right one by implementing it.

Roulette method

The last method considered in this TP is the roulette method. Given N events and their probabilities P_i, $i \in \{0, ..., N-1\}$, the initialization step consists in calculating the cumulative probabilities $P_i^{cumul} = \sum_{j=0,...,i} P_j$, $i \in \{0, .., N-1\}$. Generate a random number $r \in [0.1)$, and convince yourself that the event performed is the smallest i such as $r > P_i^{cumul}$. The search for i can be done using a linear path of $P_i^{cumulative}$ in $O(N)$ or a dichotomous search in $O(log(N))$. Implement the latter method and check that for a large number of events generated, the frequency of occurrence of each event corresponds to the probability associated with it.

dichotomous search
well known example is binary search.
selects between two distinct alternatives @ each step.
(Dichotomies)

Report

For this series, collect your code and a short report (2-5 pages) in an archive (.zip) called *name.first name.tp0.zip* that you will put on moodle in the directory *TP0*, by **Sunday, September 22, 2019.** This short report must include : (1) the description of each exercise, and (2) the methodology/pseudo code adopted to solve it, as well as (3) results obtained with your own code.

NB : For this series and all subsequent ones, only codes in C/C++, python, java and matlab will be accepted. Codes rendered in other languages will not be taken into account. In addition, the use of LaTeX (https ://openclassrooms.com/fr/courses/1617396-redigez-des-documents-de-qualite-avec-latex) is strongly encouraged. Finally, graphics that do not include a title or legends for the axes will not be taken into account.

* simply a bonus.

- all methods & Prob gen methods in Python
- latex formatting.
- general structure