Optimisation TD 3
Genetic algorithm

C. Frindel

18 Novembre 2016

Theory

A genetic algorithm is a bio-inspired optimisation algorithm. It is inspired from the natural evolution of a set of genes (a genome) to converge to an optimal mutation. Like all methods studied before, the goal is to find a set of parameters minimizing the cost function. However, the approach is different because if it not based on differential tools but instead is heuristic. The genetic algorithm is adapted in particular when the evaluation of the cost function is indirect, and/or when the function is not differentiable, and/or when the set of parameters is very large.
It is a simple algorithm to understand. We consider each parameter as a gene, and the set of gene is called the genome of a given individual. Optimisation is with the following steps (see Fig. 1).

1. Build a initial population of genomes chosen randomly.
2. Population is evaluated : estimate the value of the cost function for each genome.
3. Genomes are sorted wrt their cost. Those with highest costs are selected. For every selected genome, a mutation and/or a "crossing over" with another genome is performed.
4. The new population is in turn evaluated and muted until convergence. The genome of the population which minimizes the cost function is considered as a solution to the problem of optimization.

1 Exercise : random walk

We'll apply the genetic algorithm to a rather simple problem. We'll modell a random march of dimension 1 on T time steps, in direction d :

$$x(t+1) = x(t) + d$$

with d a random number drawn uniformly from $\{-1, 1\}$, and $t = \{0, \dots, T-1\}$.
We wish to optimise the trajectory such that it stays in a given range $[-r_0; r_0]$. The corst of a trajectory will be the number of steps outside of range $[-r_0; r_0]$.

1. Write a function which generates a random genome. It takes T (march length) in entry and returns a vector of length T whose values are drawn randomly from $\{-1, 1\}$.
2. Write the function which computes the cost of a genome. It takes in entry : a genome and r_0, the range. It returns a scalar.
3. Initialisation : Write a function which creates an initial population of N genomes of length T.
4. Write a function which sorts the genomes of a population wrt their cost. Hint : use `numpy.argsort()`.
5. Write a function which selects the N_s genomes having the cost values the highest (select by rank). Take $N_s = N/2$ for example. Output should be the population to mute/cross

6. Genome mutation. Genome mutation is defined by a rate T_m corresponding to the probability that a gene has to mute. So here, browse over all genes of a genome and mute them with probability T_m. Write the function that does this to the selected population.

7. Genome crossing-over. Crossing-over is defined by a rate T_c indicating the probability of a crossing-over happening for a given genome. Position of the crossing over and other genome with which the crossing over is done will be drawn randomly.

8. Update current population with muted population and increment the generation counter.

9. Compute the mean cost of the new population, and the minimum cost.

10. Which stopping criteria would you use ? Run the algorithm with $T = 500$, $r_0 = 4$, $N = 100$, $T_m = 0.05$ and $T_c = 0.1$. Draw the evolution of the mean cost as generations go by.

11. Make the parameters vary (N, r_0, T_m et T_c) to study their influence.

12. Describe the algorithm's behavoir (evolution of mean and minimum cost) and interpret the convergence given of the used parameters.

13. What differences can you see with differential methods (gradient descent, Newton) ?

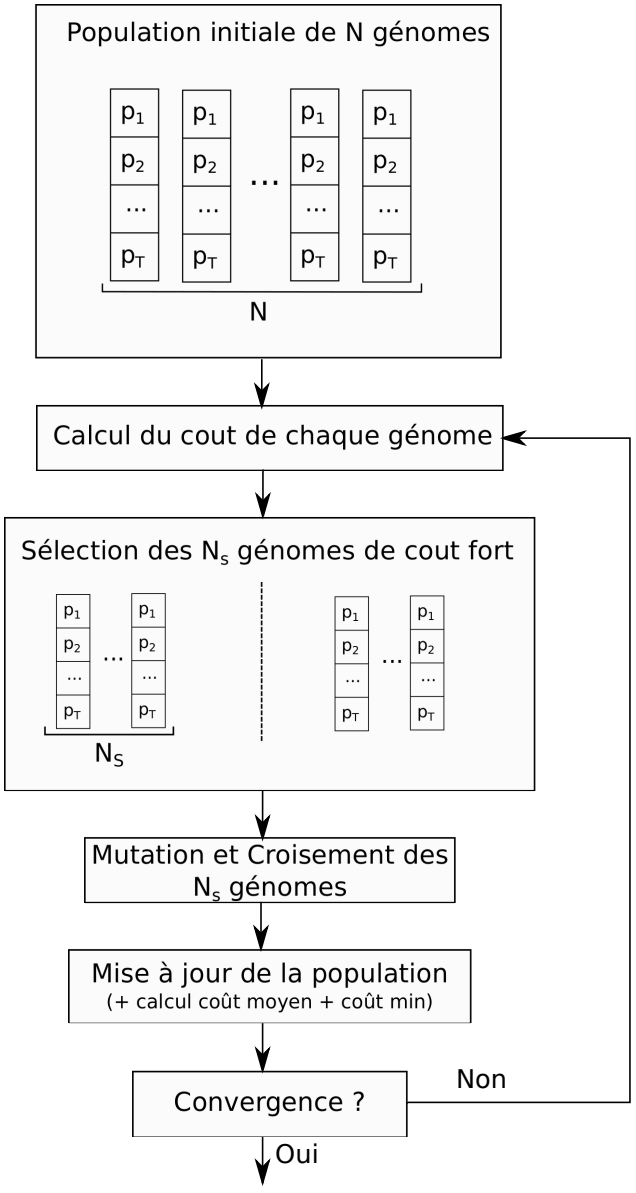

Figure 1: Schéma de principe de l'algorithme génétique