
Metaheuristics for Optimization

SERIES 1 : NK-LANDSCAPE MODELS

Return no later than September 29th, 2019

1 NK-landscape models

Our goal in this exercise is to illustrate a class of models known as *NK-landscape models*. These have been introduced in order to generate fitness landscapes, which roughness can be tuned by a single *degree* parameter K . Hence, these models are well suited to evaluate the efficiency of metaheuristics.

Let $x = (x_1, x_2, \dots, x_N)$ be a binary sequence of length N , i.e. $x \in \{0, 1\}^N$. The global fitness F is then given by the sum of local fitness contributions f_K :

$$F(x) = \sum_{i=1}^{N-K} f_K(x_i, \dots, x_{i+K})$$

This F is the quantity we would like to maximize.

If $K = 0$ then $F(x) = \sum_{i=1}^N f_0(x_i)$. In such a case, F has only one maximum reached by a sequence x where the values of all variables x_i are identical.

An example

The values of f_K must be specified for all binary sequences of length $K + 1$. Assume we have $K = 1$ and $N = 10$. Defining F requires specifying the values of f_1 for all chains of length 2. A possible choice would be

| | | | | |
|-------|----|----|----|----|
| | 00 | 01 | 10 | 11 |
| f_1 | -1 | 1 | 1 | -1 |

The two chains maximizing the fitness are $x = 0101010101$ and $x = 1010101010$.

2 Work to do

You are asked to investigate how the parameter K impacts on the landscape's roughness for chains of length $N = 21$. Consider the three landscapes defined by the following local fitness functions f_K for $K = 0, 1, 2$:

| | | |
|-------|---|---|
| | 0 | 1 |
| f_0 | 2 | 1 |

| | | | | |
|-------|----|----|----|----|
| | 00 | 01 | 10 | 11 |
| f_1 | 2 | 3 | 2 | 0 |

| | | | | | | | | |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| f_2 | 0 | 1 | 1 | 0 | 2 | 0 | 0 | 0 |

For every K , run 50 times each of the two optimization methods explained in section 2.1 and investigate the “stability” of obtained solutions. Intuitively, the rougher the landscape the less stable the solutions should be. Though different ways for assessing stability are possible, we propose to compute the pairwise Hamming distances between the 50 obtained solutions and then plot the histograms of these distances.

Describe and discuss the behaviour and the performance (in terms of the solutions quality) of each exploration method.

2.1 Methods

You are requested to use the following two optimization methods :

- *Deterministic Hill-Climbing* : Generate randomly an initial sequence of N bits. Then, among its neighbors, choose the one with the highest fitness. A “neighbor” of a given sequence x is intended to denote a sequence differing by only one bit from x . The algorithm ends when the fitness cannot be increased anymore.
- *Probabilistic Hill-Climbing* : The principle is similar to the deterministic Hill-Climbing method, except that the selection among the neighbors is performed in a stochastic manner. In particular it means that neighbors with higher fitness are more likely to be selected than neighbors with lower fitness ; their probability of selection is proportional to the value of their fitness, $P(x') = \frac{F(x')}{\sum_{y \in V(x)} F(y)}$, for $x' \in V(x)$ where $V(x)$ is the set of all the neighbors of x . Implement an *aspiration* process, such that a neighbor that has a better fitness than the current best explored solution, is always selected. Here the algorithm ends after some pre-specified number of steps (we suggest to fix this number as ten times the mean result obtained previously for deterministic climbing). The result to be returned is the *best* result reached during the exploration.

2.2 Code

The user must be able to input the parameter K and the optimization method to use, and the algorithm must return the sequence x maximizing the fitness, as well as the corresponding value of the fitness, in the form $x : F(x)$.

The choice of programming language is left to you appreciation (Python ; C(++) ; Java ; Matlab). No other languages are accepted. Using external libraries (including pieces from fellow’s code) is prohibited.

3 Work to return

Each student is required to give back a *personal* work consisting of a code and a concise but precise report in .pdf format (4-5 pages) displaying an introduction to the problem to be solved, a description of employed metaheuristic, experiments carried through with corresponding results, and a discussion. Both report and code have to be compacted in a archive file (.zip) entitled ‘*Surname_Name_TP number*’ and put on Moodle (TP1).