Tien-Tso Ning (Kense)

29 September 2019

Metaheuristics TP1

Introduction

The goal of the exercise is to define a NK-landscape model, in order to learn more about and obtain practice working with metaheuristics. The overview of the problem is simple: Let x be a bit sequence $(x_0, x_1, \dots, x_{n-1})$ of length N. Then a fitness function F is defined, which is the sum of the local fitness functions f. We wish to maximize F according to local fitness functions which are dependent on K (the number of nearby bits), using two different methods: *deterministic hill-climbing*, and *probabilistic hill-climbing*.

Methods

In order to begin designing the two methodologies, several functions had to be created that would be utilized in the hill-climbing methods, mainly: $start\_location$, $evaulate\_fitness$, and $calculate\_neighbors$. These three functions serve the following purpose:

$start\_location$ generates a sequence of N bits, randomly distributed (using the $random.uniform$ function). In our exercise, we chose to define our sequence of N bits as a list of integers.

$evaluate\_fitness$ takes a sequence of N bits (as a list of integers) and provides the fitness value according to the provided K value (and associated look-up table as a dictionary). The K value acts as a "window-size" to the evaluation of bits and is reflected in the part of code $seq[\, i : i + kVal + 1]$. Which has the K value defining the slice of indexes sent to the look-up table.

$calculate\_neighbors$ generates all the neighbor sequences, which are just sequences of bits within a Hamming-Distance of 1 to the provided sequence of bits. This function is implemented by flipping each bit in the provided sequence, and storing each resulting sequence in a list as a neighbor.

$Deterministic\ Hill\ Climbing$: The process for deterministic hill-climbing is straight-forward: starting at a given sequence, find the neighbor with the best fitness value, and repeat until the fitness is maximized. The process stops once the current sequence's fitness value is greater-than or equal-to the available neighbors' fitness values. The implementation for our code is recursive in nature to iterate the process, and terminates once the condition of hitting a maximum (local or otherwise) fitness value.

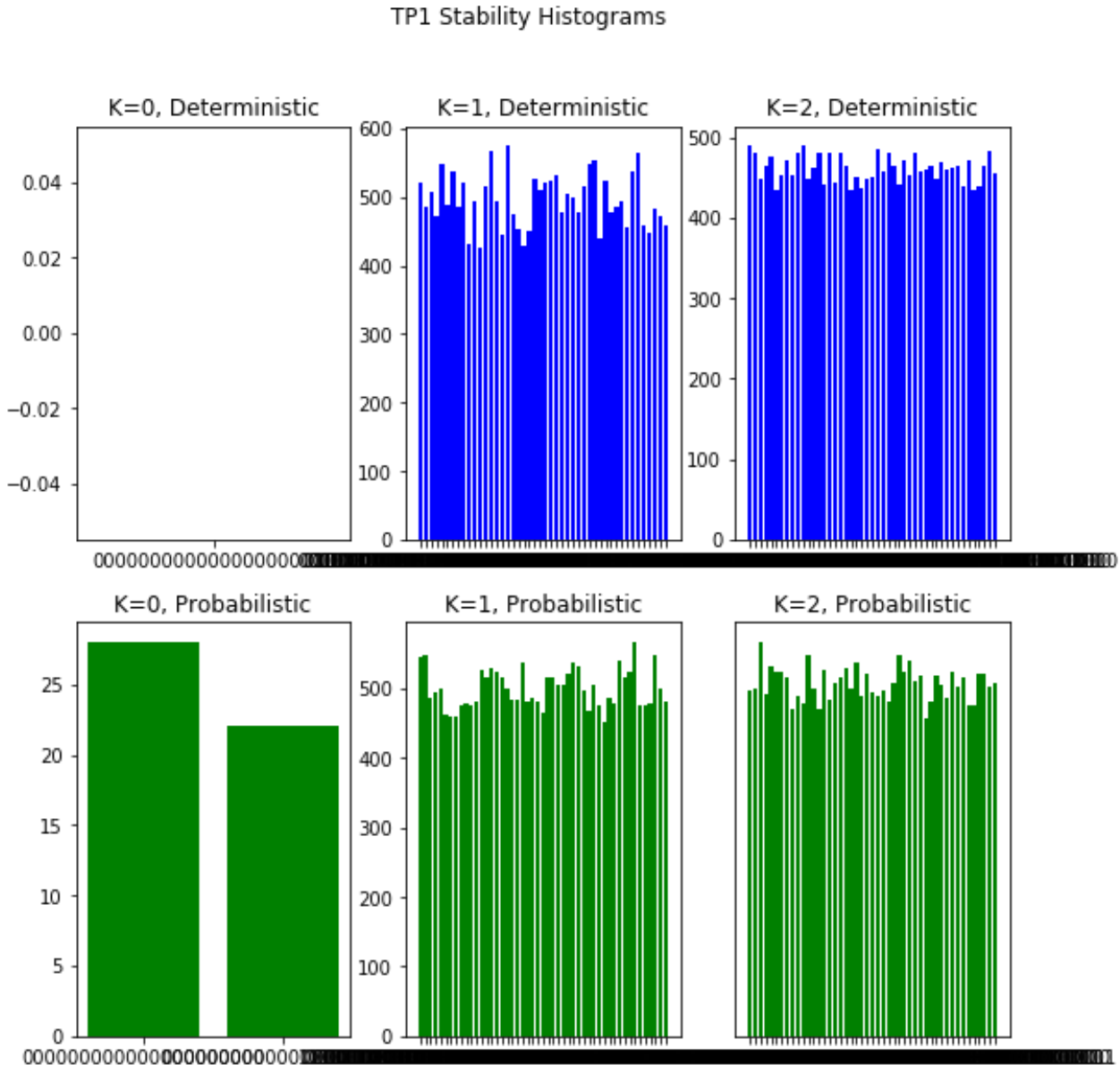
$Probabilistic\ Hill\ Climbing$: The process for probabilistic hill-climbing is two part: starting at a given sequence, assign each neighbor a probability of being chosen, defined by their fitness value:

$$P(x') = \frac{F(x')}{\sum_{y \in V(x)} F(y)}, where\ V(x) is\ the\ set\ of\ all\ neighbors\ to\ x.$$

The implementation of choosing using the assigned probability is the same as "roulette" from TP0. Constructing a cumulative probability of the neighbors, and generating a random value in order to determine which neighbor is chosen. The second part is an addendum to the selection process of the first. If a neighbor has the best fitness value recorded so far in the process, choose it every time (in other words, grant it 100% probability of being chosen). This is called an aspiration process.

Results

The process for both methods were ran 50 times each, with each K value ($K = 0, 1, 2$). The resulting sequence of bits (determined by the method as the "optimized" sequence) from all trials are stored, and the Hamming-Distance is calculated in order to determine whether or not the method is "stable." Stability is determined by evaluating how different the solution we obtained is from all the other solutions we have obtained from the other trials. $Figure\ 1$ displays the stability of each of the 50 trials.



$Figure\ 1$

Discussion

The results of the histogram show each method (deterministic and probabilistic hill-climbing) and for each K value, the stability, which is measured by taking the pair-wise Hamming distance of all solution outputs from the run_trial functions. A pair-wise Hamming distance calculation is done by taking each solution sequence from the fifty trials, and comparing it to every other solution sequence from the same fifty trials. As the graphs indicate, stability for $K = 0$ trials are either perfectly stable (deterministic) or relatively stable (probabilistic). This makes sense since $K = 0$ doesn't vary very much. For $K = 1$, and $K = 2$ in the deterministic method, the stability seems to be better in $K = 2$. In the probabilistic method, stability is rather the same between the trials.