Global minimum/maximum using Enforced Hill Climbing Algorithm

Buțco George

1 Introduction

Hill Climbing is an **heuristic algorithm** used in numerical analysis. This algorithm is used to find the global minimum or maximum of a function. It uses incremental changes and if a change is better then the original value, it becomes the new solution.

This relative simple algorithm can be used for problems such as **Travelling Salesman Problem**.

Enforced Hill Climbing is a Hill Climbing Algorithm Variant witch uses **Breadth-first search** to not get stuck in a local optimum.

2 Implementation

2.1 Pseudo Code

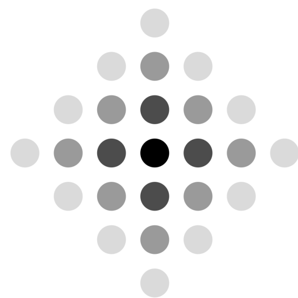
Enforced Hill Climbing

```
open-queue ← [initial-vector]
best ← f(initial-vector)
while (list is not empty) do
    current-vector ← pop vector from open-queue
    successors ← list of vectors visible from the current-vector
    for each next-vector ∈ successors do
        next-vector-value ← f(next-vector)
        if (next-vector-value is better than best) then
            clear successors
            clear open-queue
            best ← next-vector-value
        end if
        push next-vector in open-queue
    end for
end while
```

2.2 Successors Optimization

An approach can be to create the successors list from incrementing in every direction from the current-vector.

This naive approach for finding the "neighbours" will check the same vector many times being hard to check vectors that are far from the current solution.



A good approach can be:
Defining an axis vector as a tuple (value, index, direction)

$$\text{Axis Vector} = \begin{cases} \texttt{value} = [x_0, x_1, \ldots x_k \ldots x_{n-1}] & k, n \in N, k < n, n \geq 1 \\ \texttt{index} = k & k \in N, k < n \\ \texttt{direction} = -1 \texttt{ or } 1 \texttt{ (positive or negative)} \end{cases}$$

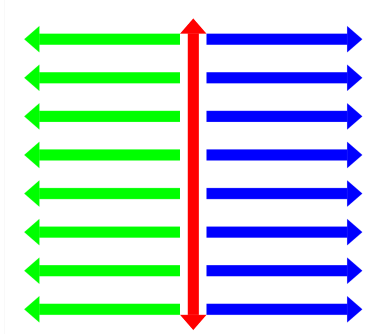
The successors of an Axis Vector are:

$$\begin{cases} [v_0, v_1, v_2], k < n-1 \\ [v_0], k = n-1 \end{cases}$$

$$v_0 = \begin{cases} \texttt{value} = [x_0, x_1, \ldots (x_k + step \cdot direction) \ldots x_{n-1}] \\ \texttt{index} = k \\ \texttt{direction} = \texttt{direction of the current vector} \end{cases}$$

$$v_1 = \begin{cases} \texttt{value} = [x_0, x_1, \ldots x_k, (x_{k+1} + step) \ldots x_{n-1}] \\ \texttt{index} = k+1 \\ \texttt{direction} = 1 \end{cases}$$

$$v_2 = \begin{cases} \texttt{value} = [x_0, x_1, \ldots x_k, (x_{k+1} - step) \ldots x_{n-1}] \\ \texttt{index} = k+1 \\ \texttt{direction} = -1 \end{cases}$$



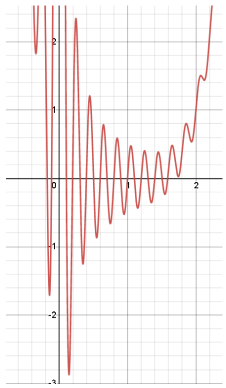
2.3 C++ Implementation

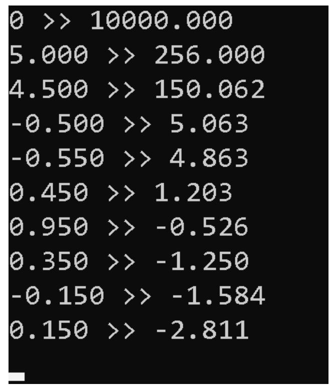
[Link to GitHub folder]

2.4 Testing

- Gramacy Lee Function:

$$\frac{sin(10\pi x)}{2x} + (x-1)^4$$





- Heart Function:

$$x^2 + \left(\frac{3y}{2} - \frac{x^2 + abs(x) - 6}{x^2 + abs(x) + 2}\right)^2 - 36$$