AG Tema 0

Stefan Petrovici

October 2019

1 Introduction

This report takes a closer look at what are the particularities of the heuristic algorithms. We will compare results obtained from four functions: Rastrigin, Schwefel, Rosenbock and Sphere. These results will then allow us to obtain a conclusion about heuristic algorithms.

1.1 Problem

Most problems we try to solve have a precise answer. But sometimes computing the solution requires many resources, and in some cases we might be forced to rethink our strategy. For these situations we need special methods that give us approximate answers with resources we are capable of consuming. The most important of these resources are time and storing space. Well, storing space as a consumable resource has in recent times been less worrying, because of the combinatorial power of the memory arrangement: With each memory unit we add, we gain a the number of combinations between it and the previous units. The more important aspect is time, and in the modern world we depend on it very much. In this aspect the algorithms we analyze in this report can be particularly useful.

The problem we want to solve in this paper is finding the global minimum of a give function by using these algorithms. The problem is not a trivial one, since a function can have multiple local minima, but some of those might not also be global.

2 Method

The three algorithms we will analyze are:

1. Iterated Hill-Climbing with Best Improvement Selection
2. Iterated Hill-Climbing with First Improvement Selection
3. Simulated Annealing

2.1 Pseudocode

The pseudocode for Hillclimber best ascent is shown below:

```
begin
t := 0
initialize best
 repeat
  local := FALSE
  select a candidate solution (bitstring) vc at random
  evaluate vc
  repeat
   vn := Improve(Neghborhood(vc))
   if eval(vn) is better than eval(vc)
    then vc := vn
   else local := TRUE
  until local
  t := t + 1
  if vc is better than best
   then best := vc
  until t = MAX
end
```

The pseudocode for Simulated Annealing best ascent is shown below:

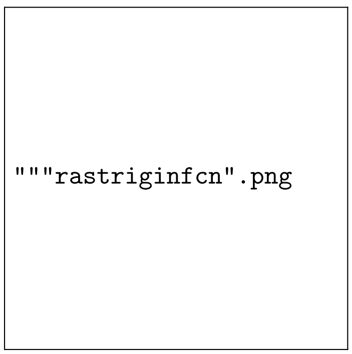
```
begin
 t := 0
 initialize the temperature T
 select a current candidate solution (bitstring) vc at random
 evaluate vc
 repeat
  repeat
   select at random vn - a neighbor of vc
   if eval(vn) is better than eval(vc)
    then vc := vn
    else if random[0,1) < exp(-|eval(vn)-eval(vc)|/T)
     then vc := vn
  until (termination-condition)
  T := g(T; t)
  t := t + 1
 until (halting-criterion)
end
```

2.2 Description

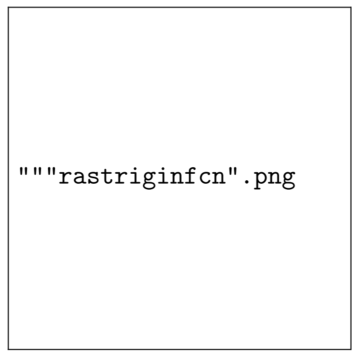
Firstly, we have to talk about the representation of our solution. In this method we will choose the **bitstring** representation of real floating numbers. Float type values have 4 bytes, consisting of a sign bit, an 8-bit excess-127 binary exponent, and a 23-bit mantissa.

By modifying one bit of the value we can change the sign of the number, the

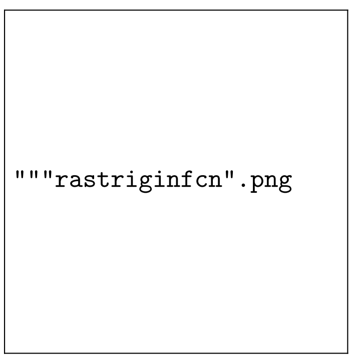
3 Experiment



(a) Initial condition



(b) Rupture



(c) DFT, Initial condition

(d) DFT, rupture

Figure 1: The 4 functions

Both Hill climbing and Simulated Annealing are anytime algorithms: they can return a valid solution even if they are interrupted at any time before it ends.

Name	AlgorithmType	Dimension	Result	RunTime
rastrigin	deterministic	2	0.009918	0.00040
rastrigin	deterministic	5	0.024796	0.00043
rastrigin	deterministic	20	0.099182	0.00169
schwefel	deterministic	2	0.000041	0.01949
schwefel	deterministic	5	0.000193	0.03109
schwefel	deterministic	20	0.000773	0.10881
rosenbrock	deterministic	2	0.003508	0.00031
rosenbrock	deterministic	5	0.014032	0.00020
rosenbrock	deterministic	20	0.066650	0.00085
sphere	deterministic	2	0.000050	0.00027
sphere	deterministic	5	0.000125	0.00028
sphere	deterministic	20	0.000500	0.00085

Figure 2: The Deterministic results from the algorithm

Name	AlgorithmType	Dimension	Mean	Max	Min	Median	MeanRunTime	StdDistrib
rastrigin	heuristic	2	17.68021	24.87877	8.95900	17.91134	0.41819	6.15868
rastrigin	heuristic	5	44.34899	76.61167	12.94215	46.76892	1.30510	14.84982
rastrigin	heuristic	20	171.43078	239.82538	97.53253	171.66728	20.46675	31.35698
schwefel	heuristic	2	375.06847	572.48767	217.13969	335.57802	0.50770	147.73321
schwefel	heuristic	5	929.81045	1441.21899	454.01654	917.96906	1.87225	226.78800
schwefel	heuristic	20	3398.46616	4522.69238	2369.05884	3376.19214	29.17817	571.28253
rosenbrock	heuristic	2	3.76382	12.29180	0.32211	0.36406	0.27174	5.22217
rosenbrock	heuristic	5	11.33697	151.63109	0.07082	1.52969	0.43053	37.52215
rosenbrock	heuristic	20	15.52329	19.42897	0.14097	16.32754	2.02218	3.39839
sphere	heuristic	2	0.00001	0.00003	0.00000	0.00001	0.26923	0.00001
sphere	heuristic	5	0.00005	0.00011	0.00001	0.00004	0.40770	0.00002
sphere	heuristic	20	0.00017	0.00022	0.00011	0.00017	1.42898	0.00003

Figure 3: The Heuristic results from the algorithm

4 Conlusions

It is obvious from these results that done correctly, the algorithms for guessing the real value of a minimum can offer great advantages to the programmer.

References

BenchMarking Functions
`http://benchmarkfcns.xyz`

Wikipedia
Global optimization

https://en.wikipedia.org/wiki/Global_optimization#Deterministic_methods

Branch and Bound source
https://en.wikipedia.org/wiki/Branch_and_bound#cite_ref-8

Global minimum Search Algorithm MIT
https://www.youtube.com/watch?v=0Ro5IU9a55s&list=PL3940DD956CDF0622&index=38&t=2696s