AG Tema 1

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. Our solution would be N **bitstrings**, where N will be the number of dimensions we will compute on each function. A Hamming Neighbor is a bitstring that differs by only one bit from the current solution.

The Iterated Hill-Climbing with Best Improvement Selection Algorithm passes through all of the Hamming-1 neighbours of a bitstring (named initial candidate) and if it finds a better candidate it will continue to run with this candidate. The algorithm stops when there is no other candidate that is better then our current one.

The Iterated Hill-Climbing with First Improvement Selection has a similar behaviour to the Best version, except that when in finds a better candidate it immediately takes him, without considering the other neighbors.

The Simulated Annealing Algorithm picks a random neighbor, tests if he is better than its current solution and if that is valid, it replaces the current candidate. What is most important though is the fact that this algorithm permits the choosing of a weaker, non-optimal candidate with the purpose that this weaker candidate might in turn lead to the global optimum.

2.3 Implementation

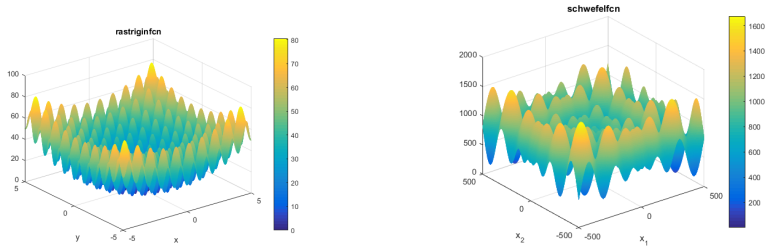
In this method we will choose the **bitstring** representation of real floating numbers. Float type values have 4 bytes, consisting of a sign bit, an 8-bit excess-127 binary exponent, and a 23-bit mantissa. By modifying one bit of the value we can change the sign of the number, the mantissa or the exponent. Float has a 7-digit precision, and by using a formula for binding this representation to the domain interval we make sure we can check as many valid candidates as possible.

$$X_{real} = a + decimal(x_{biti}) * (b - a)/(2^n - 1)$$

The data structure we are going to use is a standard library std::bitset. This structure is useful because it places each bit in an actual physical bit and also that it is allocated on stack, meaning it will be faster to access than other options like std::vector.

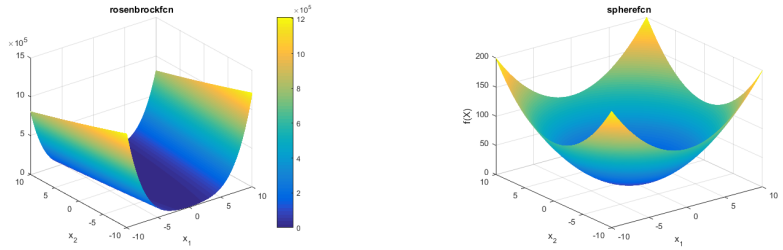
3 Experiment

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. For each function, the 3 algorithms ran 30 times (without the inner loops) on 2,5 and 30 dimensions each. For a function an analysis took about 5 minutes. If we had stopped at 4 minutes we could noticed some of the values(especially



(a) Rastrigin Function

(b) Schwefel Function



(c) Rosenbrock Function

(d) Sphere Function

Figure 1: The 4 functions

for the Schwefel function) would be 200 function units worse, but for the rest it wouldn't change much.

3.1 Rastrigin

Hillclimbing-best on Rastrigin has produced the following results:

Dim.	Mean	Max	Min	Median	MRT(sec.)	RT(min.)	StdDev
2	0.00000	0.00000	0.00000	0.00000	0.00037	0.00019	0.00000
5	0.00000	0.00000	0.00000	0.00000	0.00275	0.00137	0.00000
30	0.00000	0.00000	0.00000	0.00000	0.42538	0.21269	0.00000

Hillclimbing-first on Rastrigin has produced the following results:

Dim.	Mean	Max	Min	Median	MRT(sec.)	RT(sec.)	StdDev
2	32.79528	48.31204	12.89403	28.94584	0.30109	0.15054	10.34233
5	89.89455	120.2332	61.13435	88.61257	0.30113	0.15056	17.24461
30	565.88526	694.0710	440.6955	572.75143	0.30146	0.15073	55.83071

Simulated Annealing on Rastrigin has produced the following results:

Dim.	Mean	Max	Min	Median	MRT(sec.)	RT(sec.)	StdDev
2	1.05696	1.66620	0.16476	1.11902	3.00601	1.50300	0.41327
5	3.66980	7.62761	1.31342	3.61088	3.00601	1.50300	1.11049
30	31.46571	43.14999	17.94916	31.79051	3.00602	1.50301	6.33809

3.2 Schwefel

Hillclimbing-best on Schwefel has produced the following results:

Dim.	Mean	Max	Min	Median	MRT(sec.)	RT(sec.)	StdDev
2	0.00122	0.00122	0.00122	0.00122	0.00123	0.00062	0.00000
5	0.00317	0.00317	0.00317	0.00317	0.01268	0.00634	0.00000
30	0.01660	0.01660	0.01660	0.01660	2.41241	1.20620	0.00000

Hillclimbing-first on Schwefel has produced the following results:

Dim.	Mean	Max	Min	Median	MRT(sec.)	RT(sec.)	StdDev
2	604.45936	856.69659	172.89160	633.80743	0.30109	0.15055	178.29133
5	1946.29082	2660.72168	1215.65112	1974.07056	0.30113	0.15057	427.47421
30	12294.58330	14812.13867	9339.17969	12382.93994	0.30148	0.15074	1208.66385

Simulated Annealing on Schwefel has produced the following results:

Dim.	Mean	Max	Min	Median	MRT(sec.)	RT(sec.)	StdDev
2	56.25339	131.42291	0.13794	35.87091	3.00601	1.50300	53.59844
5	203.26813	587.84790	2.15259	187.91071	3.00601	1.50301	125.55201
30	2282.76950	3094.87305	1772.73828	2239.87109	3.00602	1.50301	333.35353

3.3 Sphere

Hillclimbing-best on Sphere has produced the following results:

Dim.	Mean	Max	Min	Median	MRT(sec.)	RT(sec.)	StdDev
2	0.00000	0.00000	0.00000	0.00000	0.00009	0.00004	0.00000
5	0.00000	0.00000	0.00000	0.00000	0.00053	0.00026	0.00000
30	0.00000	0.00000	0.00000	0.00000	0.07298	0.03649	0.00000

Hillclimbing-first on Sphere has produced the following results:

Dim.	Mean	Max	Min	Median	MRT(sec.)	RT(sec.)	StdDev
2	12.96183	32.76800	0.32572	9.17156	0.30109	0.15055	10.32909
5	41.14571	68.45519	11.06628	42.82627	0.30112	0.15056	14.07320
30	266.07762	392.56210	137.92204	267.10228	0.30141	0.15071	49.44672

Simulated Annealing on Sphere has produced the following results:

Dim.	Mean	Max	Min	Median	MRT(sec.)	RT(sec.)	StdDev
2	0.09440	0.34697	0.00567	0.07151	3.00601	1.50300	0.08863
5	0.52180	1.62801	0.07646	0.46727	3.00603	1.50301	0.35979
30	5.23466	8.42584	2.48623	5.01753	3.00601	1.50301	1.53626

3.4 Rosenbrock

Hillclimbing-best on Rosenbrock has produced the following results:

Dim.	Mean	Max	Min	Median	MRT(sec.)	RT(sec.)	StdDev
2	0.61137	0.61137	0.61137	0.61137	0.00148	0.00074	0.00000
5	3.32159	3.32159	3.32159	3.32159	0.00438	0.00219	0.00000
30	28.06901	28.06901	28.06901	28.06901	0.45371	0.22685	0.00000

Hillclimbing-first on Rosenbrock has produced the following results:

Dim.	Mean	Max	Min	Median	MRT(sec.)	RT(sec.)	StdDev
2	122450.46	695256.31	339.85	18353.16	0.30	0.150	176665.46
5	654464.53	1591305.62	92365.70	638715.37	0.30	0.150	401149.81
30	6071569.85	9449637.00	3750092.25	6035927.25	0.30	0.150	1463527.42

Simulated Annealing on Rosenbrock has produced the following results:

Dim.	Mean	Max	Min	Median	MRT(sec.)	RT(sec.)	StdDev
2	1.24227	7.21554	0.03192	0.76147	3.00601	1.50300	1.51518
5	36.62623	640.08514	2.04132	5.33220	3.00602	1.50301	114.71004
30	3359.21576	8925.05859	36.19930	3727.94263	3.00601	1.50301	2616.54935

From these results we can observe that in most cases, a larger number of dimensions leads to a more erroneous solution, and a larger number of iterations (and the observation before about time) leads to a better solution. For more dimensions we might need to do more iterations to get an equally better result.

4 Conclusions

We can see that Hill Climbing and Simulated Annealing produce similar results. Simulated Annealing reaches solution faster, while HillClimbing reaches them more precisely. This is because HC iterates between its neighbors, while SA will sample randomly.

References

BenchMarking Functions
http://benchmarkfcns.xyz

Wikipedia
Global optimization
https://en.wikipedia.org/wiki/Global_optimization#Deterministic_methods

Hill Climbing
Optimisation algorithm
https://en.wikipedia.org/wiki/Hill_climbing

Simulated Annealing
Optimisation algorithm
https://en.wikipedia.org/wiki/Simulated_annealing

Float
C Data Type
https://docs.microsoft.com/en-us/cpp/c-language/type-float?view=vs-2019