HW0 Report

Roșu Cristian-Mihai

October 20, 2019

1 Introduction

Implement a deterministic and a heuristic search method. Test and compare them using 4 benchmark functions. One of the functions should be **Rastrigin**'s Function:

$$f(x) = A \cdot n + \sum_{i=1}^{n} \left[x_i^2 - A \cdot cos(2\pi x_i)\right], A = 10, x_i \in [-5.12, 5.15]$$

The functions should be tested using the following numbers of dimensions: 2, 5, 20. In the case of heuristic search methods, a proper sample size (minimally 30) should be used.

Write a report detailing and explaining your motivation, method, results, and draw conclusions.

1.1 Motivation

For the three other functions, apart from Rastrigin's, I chose **Michalewicz**'s, **Rosenbrock**'s and the **Sphere** function.

$$f(x) = -\sum_{i=1}^{n} \left[\sin(x_i^2) \cdot \sin(\frac{i{x_i}^2}{\pi})^{2m}\right], m = 10, x_i \in [0, \pi]$$

$$f(x) = \sum_{i=1}^{n-1} \left[100({x_i}^2 - x_{i+1})^2 + (x_i - 1)^2\right], x_i \in [-5, 10]$$

$$f(x) = \sum_{i=1}^{n} {x_i}^2, x_i \in [-5.12, 5.12]$$

The reason why I chose those three specifically is because they are among the ones who can work with any number of variables, but more importantly because their codomain is of relatively small size, compared with the rest of the options. Among them, Michalewicz's function was chosen especially for its unique global minima.

2 Method

The methods I used are as follows:

For the heuristic method, I decided to use a simple random search algorithm. It randomly generates a real number and compares the value obtained after passing it through the function to the variable holding the current minimum. This process is repeated through a million iterations in order to cover a large portion of the function's domain. In order to cut down on the time needed to perform these operations, this task is split among two threads, using two asynchronous functions, each calculating its own minimum over just 500,000 iterations. Two threads is an optimal choice, since more than that resulted in an improvement of just one second(i.e. 2 and 3 threads).

For the deterministic method, I decided on using a Brute-force search algorithm. It iterates through the entirety of the function's domain and similarly makes comparisons to find the minimum. Due to its nature, the algorithm works best in the 2-dimensional scenario, finding the solution in a huge amount of time in the other cases.

Because of this, I opted for a second, "lazier" approach, which also generates solutions by passing through the function's whole domain, but it does so by making all of the solution's components the same(i.e. for n=2 and x=[1,1], when n=3 then x=[1,1,1]). This brings satisfactory results only because of the four functions that are being tested, whose global minima are obtained from values in that form.

3 Experiment

The experiment consists in simply running the algorithms once, in the case of the deterministic ones, and running the heuristic one 30 times in order to obtain consistent results which can be safely compared to our expectation.

The results sought to obtain are the minima found by the algorithms an the times needed to reach those minima. Regarding the heuristic method, we are instead searching for the average of these we get from the 30 iterations.

The code for this can be found on Github.[1]

4 Results

Below are 4 tables corresponding to each of the 4 functions that hold the results to the experiment: Rastrigin, Michalewicz, Rosenbrock and the Sphere function, in that order.

Method	Dimension n	2	5	20
Lazy	Min	0	0	0
	Time	32ms	56ms	112ms
Deterministic	Min	0	-	-
	Time	1039ms	>30min	>>30min
Nedeterministic	Min	0.000304872	3.54069	145.893
	Max	0.0171115	9.37915	174.082
	Average	0.00534376	6.93064	162.607
	Min time	5030ms	9226ms	22925ms
	Max time	5319ms	9486ms	23629ms
	Average time	5128ms	9304ms	23087ms

Figure 1: Rastrigin's function results

Method	Dimension n	2	5	20
Lazy	Min	-1.00096	-2.38927	-7.94934
	Time	9ms	17ms	35ms
Deterministic	Min	-1.79779	-	-
	Time	106ms	>30min	>>30min
Nedeterministic	Min	-1.80163	-4.52292	-9.53519
	Max	-1.80137	-3.79666	-7.96045
	Average	-1.80156	-4.10483	-8.58571
	Min time	5263ms	9477ms	22805ms
	Max time	5402ms	9814ms	23314ms
	Average time	5307ms	9554ms	22984ms

Figure 2: Michalewicz's function results

Method	Dimension n	2	5	20
Lazy	Min	$2.33877 \cdot 10^{-25}$	$5.84692 \cdot 10^{-25}$	$2.33877 \cdot 10^{-24}$
	Time	31ms	52ms	99ms
Deterministic	Min	$8.56874 \cdot 10^{-28}$	-	-
	Time	1033ms	>30min	>>30min
Nedeterministic	Min	$2.02137 \cdot 10^{-6}$	0.0329426	17.4257
	Max	$7.68284 \cdot 10^{-5}$	0.327834	45.3134
	Average	$2.43108 \cdot 10^{-5}$	0.179344	35.5202
	Min time	4933ms	8960ms	21356ms
	Max time	5304ms	9267ms	21670ms
	Average time	5040ms	9058ms	21477ms

Figure 3: Sphere function results

Method	Dimension n	2	5	20
Lazy	Min	0	0	0
	Time	42ms	75ms	155ms
Deterministic	Min	0	-	-
	Time	2186ms	>30min	>>30min
Nedeterministic	Min	$2.78091 \cdot 10^{-5}$	13.0901	33526
	Max	0.0026726	55.7838	87850
	Average	0.000686644	29.1294	62896
	Min time	4723ms	9202ms	22150ms
	Max time	5125ms	9711ms	22462ms
	Average time	4825ms	9307ms	22299ms

Figure 4: Rosenbrock's function results

5 Conclusions

The deterministic method is the most efficient out of the two, however that holds true only while we are working with a low number of dimensions. As we scale the dimension upwards, the Brute-force approach quickly proves to be way too inefficient time wise. On the other hand, although the heuristic method drifts away from the correct solution, its times are consistent across the different dimensions, and the different function in fact.

References

[1] Github repository for the project
`https://github.com/Nenma/ga-hw0`

[2] Wikipedia page for Brute-force algorithms
`https://en.wikipedia.org/wiki/Brute-force_search`

[3] Simple Latex tutorial I used
`http://www.docs.is.ed.ac.uk/skills/documents/3722/3722-2014.pdf`