Genetic Algorithms

Homework 2

Ciobotaru Mihai , 2B3

December 7, 2021

Abstract

The purpose of this study is to prove the efficiency of genetic algorithms in solving multi-modal functions. In order to conduct this research, I compared the results I gained with the results of other heuristic search algorithms. Doing that, I have found genetic algorithms are a very useful tool in solving multi-modal functions in cases where the search space is too large for deterministic algorithms to solve.

1 Introduction

This paper‘s purpose is to present the behaviour and efficiency of a basic genetic algorithm(GA) and compare it with other heuristic search algorithms: Steepest-Ascent Hill Climbing(SAHC), Nearest-Ascent Hill Climbing(NAHC) and Simulated Annealing(SA). In the following pages I will explain what a genetic algorithm is in detail and prove it‘s power through an experiment . This experiment involves search for the global minima of four different multi-modal functions with the algorithms mentioned earlier. With the help of the results, we will make a comparison between them and draw conclusions.

2 Genetic Algorithm

2.1 Definition and explanation

Genetic algorithms are a family of algorithms inspired by evolution. These algorithms encode a potential solution to a specific problem on a simple chromosome-like data structure and apply recombination operators to these structures in order to reach the best possible solution it can find.

An implementation of a genetic algorithm begins with a population of chromosomes and after that, for multiple iterations called generations, there are used reproductive methods (genetic operators) in such a way that the chromosomes which represent a better solution for the target problem are given more chances to ’reproduce’ than those chromosomes which represent poorer solutions. The ’goodness’ of a solution is typically defined with respect to the current population.

It is a good idea to use a genetic algorithm when the search space is so large that using an exhaustive search would be impossible. More than that, genetic algorithms do not provide an exact global maxima/minima but they offer a good enough solution in a good amount of time. Other heuristic search algorithms such as Hill Climbing may be better to use if the function to be tested is uni-modal.

In nature, the individual that has better survival traits will survive for a longer.This in turn provides it a better chance to produce offspring with its genetic material. In time, the population will consist with lot of genes from the superior individuals and less from the inferior individuals. The existence of competition among individuals of a population determine them to get fitter.

The concept described above is also present in genetic algorithms.As a consequence, mostly only the fittest solutions get passesd to the next generation. By random chance, a small number of unfit solutions are also picked in order to prevent getting stuck in a local accumulation point.A very small number of unfit solutions are closer to the global maxima/minima that a fit solution. There are many selection functions to choose from, but the fitness function(used to evaluate a solution) is dependent on the problem.

2.2 Genetic operators

Crossover is a genetic operator in which the parents exchange information(genetic material) at a random position in the chromosome. In some genetic algorithms the offspring are added to the new population and in other algorithms they replace their parents.

Mutation is an asexual recombination operator(genetic operator) in which the information of a chromosome is slightly modified in order to maintain the diversity of the population. The mutation operator ensures that the algorithms doesn‘t converge in a local maxima/minima and gives the random chance to find a better result.

3 Methods used

3.1 Algorithm

Algorithm 1 Genetic Algorithm

```
Require: $generationsNum > 0$ and $i = 0$
1: population ← $randomPopulation()$
2: while $i < generationsNum$ do
3:     $mutatePopulation(population)$
4:     $crossoverPopulation(population)$
5:     $mutatePopulation(population)$
6:     $evaluatePopulation(population)$
7:     $selectPopulation(population)$
8: end while
9: return $population.bestSol()$
```

3.2 Mutation

The mutation function iterates through every bit of every solution in the current population and has a random probability of 1% to invert that bit. In case a chromosome‘s information is changed, the parent remains in the population and offspring too. Applying this algorithm once before the crossover operator and once after it, allows us to have chromosomes untainted, chromosomes with only 1% of information changed due to mutation and chromosomes with 2% changed due to mutation.

3.3 Crossover

The crossover operator picks randomly three pairs of positions in chromosomes. For each pair we will swap the bits from the starting position to the end position. The offspring resulted in the process replace the parents in the population.Also we will randomly pick only 60% of the population to crossover.

3.4 Fitness function

$fitness(candidate) = 1 - |\frac{candidate.evalValue() - min}{(max - min) + \epsilon}|$

Max variable is the maximum evaluation value from the population and min is the minimum.

The result value will be in the interval [0,1].

3.5 Selection

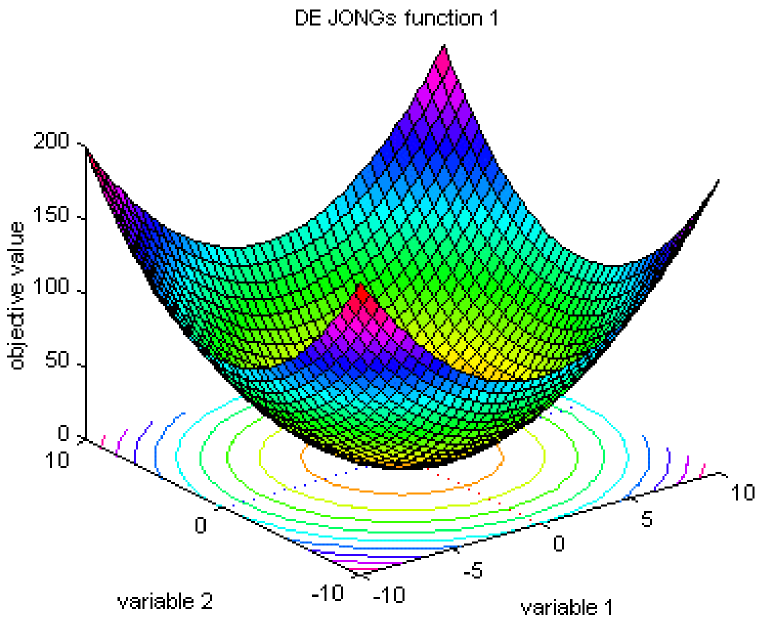
The selection function sorts all the solutions from the best to the worst and then iterates n times through a loop (n being the population size). For each iteration we pick a candidate from the population and add it it to the new one. In order to pick a solution we generate a random number from 0 to 98 and compare it to a certain value and ,if it is lower than that value, the solution is picked. That value is equal with the fitness of the candidate (a number from 0 to 1) multiplied by the $accept_rate$ variable that is equal with 0,70. Therefore, every solution has a maximum probability of 70% to be picked.

4 Functions to be tested

4.1 De Jong‘s function .

The simplest test function is De Jong’s function 1. It is also known as sphere model. It is continuos, convex and unimodal.

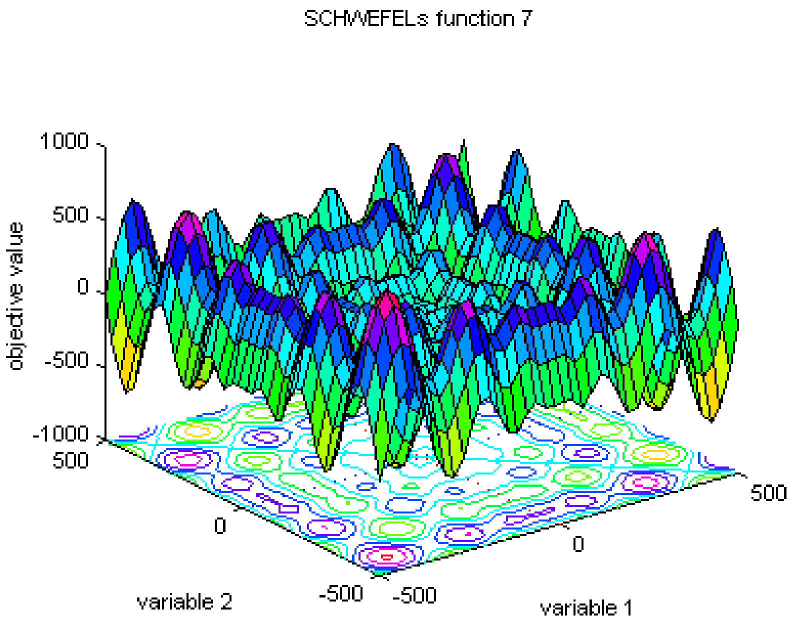
f : $[-5,12,5,12]^n \Rightarrow R, f(x) = \sum_{i=1}^{n} x_i^2$



4.2 Schwefel’s function .

Schwefel’s function [Sch81] is deceptive in that the global minimum is geometrically distant, over the parameter space, from the next best local minima. Therefore, the search algorithms are potentially prone to convergence in the wrong direction.

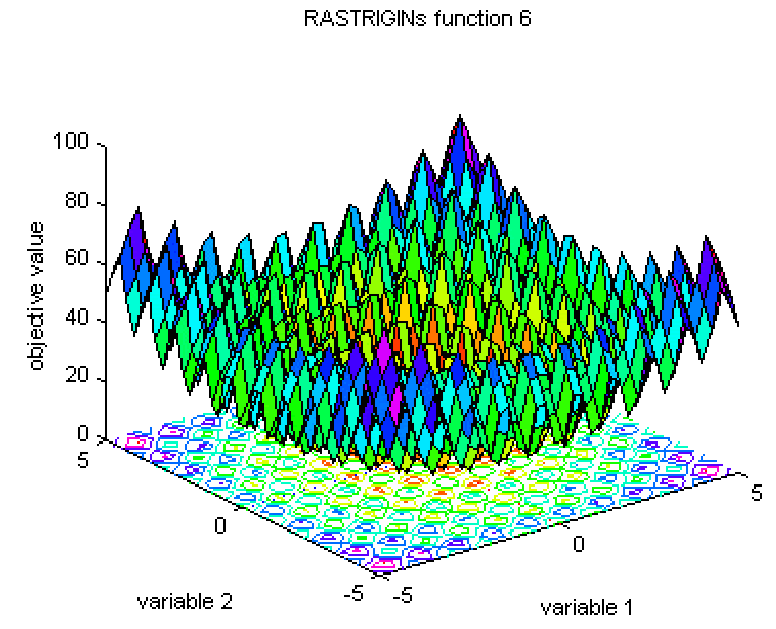
f : $[-500,500]^n \Rightarrow R, f(x) = \sum_{i=1}^{n} -x_i \cdot sin(\sqrt{|x_i|})$



4.3 Rastrigin's function .

Rastrigin's function is based on function 1 with the addition of cosine modulation to produce many local minima. Thus, the test function is highly multimodal. However, the location of the minima are regularly distributed.

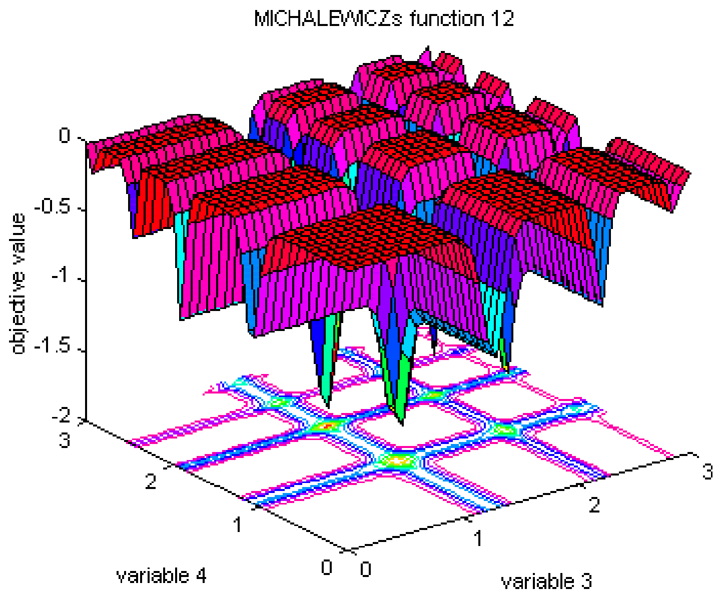
f : $[-5,12,5,12]^n \Rightarrow R, f(x) = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot (2 \cdot \pi \cdot x_i))$



4.4 Michalewicz's function .

The Michalewicz function [Mic92] is a multimodal test function (n! local optima). The parameter m defines the "steepness" of the valleys or edges. Larger m leads to more difficult search. For very large m the function behaves like a needle in the haystack (the function values for points in the space outside the narrow peaks give very little information on the location of the global optimum).

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



5 Experiment

For this experiment we will be comparing the time and value result's of a Genetic Algorithm, Hill Climbing algorithms and Simulated Annealing in relation with the functions presented above.Each algorithm will target to find the global minimum of the function and test it for 5th, 10th and 30th dimension each with a precision of 5 decimals.We are interested in finding the best,worst and average time and values and inserting them in tables to have a good visual representation of the results . The algorithms depicted will be implemented in C++ ,compiled with g++ and ran on Ubuntu 18.04.6 LTS OS with Intel® Core™ i5-8265U CPU @ 1.60GHz × 8 processor .

Hill Climbing and Simulated Annealing algorithms go through 500 iterations and we will have a sample of 400 results. For the genetic algorithm we will have a population of 450 solution before the genetic operators with 450 generations and a sample of 50 results.

5.1 De Jong's 1 function .

Dimensions	Algorithm	Average	Worst	Best	Avg. Time(s)
5	GA	0.00000	0.00000	0.00000	22.58351
	NAHC	0.00000	0.00001	0.00000	0.50772
	SAHC	0.00000	0.00001	0.00000	0.52688
	SA	0.00001	0.00003	0.00000	0.53893
10	GA	0.00000	0.00000	0.00000	28.79274
	NAHC	0.00001	0.00001	0.00000	1.76225
	SAHC	0.00001	0.00001	0.00000	1.85105
	SA	0.00001	0.00001	0.00000	1.88168
30	GA	0.00010	0.00006	0.00003	63.60260
	NAHC	0.00002	0.00002	0.00001	12.41868
	SAHC	0.00002	0.00002	0.00001	15.14539
	SA	0.00002	0.00002	0.00001	15.23265

5.2 Schwefel's function .

Dimensions	Algorithm	Average	Worst	Best	Avg. Time(s)
5	GA	-2094.61625	-2094.39409	-2094.91367	26.23100
	NAHC	-2090.77954	-1701.52869	-2094.91444	1.04017
	SAHC	-2091.86359	1701.61822	-2094.91444	1.08996
	SA	-2084.94706	-1701.49871	-2094.91443	1.10650
10	GA	-4189.27586	-4188.89152	-4189.61848	38.08962
	NAHC	-4178.62142	-3796.01101	-4189.82880	3.61197
	SAHC	-4182.73306	-3796.08427	-4189.82887	4.00180
	SA	-4167.90092	-3796.08427	-4189.82633	4.02178
30	GA	-12394.98571	-12118.25510	-12567.87570	61.84964
	NAHC	-12242.16245	-10432.01064	-12568.95746	8.72211
	SAHC	-12551.84651	-12175.45655	-12569.48148	17.91231
	SA	-11957.89946	-12491.76075	-12569.37348	16.52397

5.3 Rastrigin's function .

Dimensions	Algorithm	Average	Worst	Best	Avg. Time(s)
5	GA	0.04967	1.23934	0.00010	20.64697
	NAHC	6.49286	21.01186	0.00025	0.52538
	SAHC	1.43719	11.13935	0.00010	0.53120
	SA	3.04881	16.08704	0.00010	0.54977
10	GA	1.78470	4.95716	0.00019	28.75531
	NAHC	12.45414	27.20517	1.24021	1.82583
	SAHC	2.79471	22.23488	0.00035	1.89475
	SA	6.47576	29.62160	0.00081	1.95420
30	GA	20.67289	51.11093	4.95925	59.64764
	NAHC	70.49311	21.01064	1.214796	9.10564
	SAHC	31.18246	9.90016	0.00244	9.19241
	SA	28.43003	10.28954	0.00306	9.18831

5.4 Michalewicz's function .

Dimensions	Algorithm	Average	Worst	Best	Avg. Time(s)
5	GA	-4.64707	-4.51579	-4.68765	21.11427
	NAHC	-4.14842	-3.00788	-4.68752	0.53119
	SAHC	-4.12068	-2.53429	-4.68764	0.54422
	SA	-4.11096	-2.03949	-4.68757	0.54804
10	GA	-9.38963	-9.05056	-9.62041	29.61517
	NAHC	-8.17759	-6.38718	-9.51387	1.88050
	SAHC	-8.29387	-6.01194	-9.59792	1.98624
	SA	-8.29976	-5.66804	-9.55632	1.99319
30	GA	-27.35190	-25.75154	-28.37507	61.84964
	NAHC	-24.20723	-20.87499	-26.85518	7.67801
	SAHC	-25.01129	-22.67337	-27.47162	9.19241
	SA	-24.93673	-22.32254	-27.20926	9.18831

6 Comparison

The results shown above depict the efficiency of each algorithm in finding the global minima. Based on the results we can conclude how efficient genetic algorithms are. As shown above the results of the genetic algorithms are more concentrated, meaning that the interval in which the results are is smaller compared to the ones from the other heuristic search algorithms. We can observe that, in most cases, the genetic algorithm proved to return better results but in more time. More than that, we can see that for bigger dimensions, the efficiency of the algorithm may give poorer results that being a consequence of not enough optimisation of the algorithm.

7 Conclusion

This study proved the efficiency of genetic algorithms in relation with finding the global minima. The report succeeded to show that these algorithms are slightly better than other heuristic search algorithms for this problem in particular. As a final remark, with the help of this research, we observed genetic algorithms's importance for solving difficult computing problems.

References

[1] Mathew, Tom V. "Genetic algorithm." Report submitted at IIT Bombay (2012). |
http://datajobstest.com/data-science-repo/Genetic-Algorithm-Guide-[Tom-Mathew].pdf

[2] Wang SC. (2003) Genetic Algorithm. In: Interdisciplinary Computing in Java Programming. The Springer International Series in Engineering and Computer Science, vol 743. Springer, Boston, MA |
https://link.springer.com/chapter/10.1007/978-1-4615-0377-4_6f

[3] Wikipedia - Hill Climbing |
https://en.wikipedia.org/wiki/Hill_climbing

[4] Wikipedia - Simulated annealing |
https://en.wikipedia.org/wiki/Simulated_annealing

[5] Wikipedia - Genetic algorithm |
https://en.wikipedia.org/wiki/Genetic_algorithm

[6] Eugen Croitoru: Teaching: Genetic Algorithms |
https://profs.info.uaic.ro/∼eugennc/teaching/ga/

[7] GEATbx - Example Functions |
http://www.geatbx.com/docu/fcnindex-01.html