Optization of functions using hillclimbers, genetic algorithms and hybrids

Mihai Bojescu
Master in Artificial Intelligence and optimization
Faculty of Computer Science of University "Alexandru Ioan Cuza" of Iași
Iași, Romania
bojescu.mihai@gmail.com

***Abstract*—This document contains the results of the experiments of running two hillclimbers, a genetic algorithm and a hybrid algorithm on 4 functions with the scope of finding the minimum of each of the functions. This document additionally details what parameters were used in order to achieve the specified results. The functions that the algorithms were applied on were the Rastrigin's function, Reosenbrock's function, Michalewicz's function and Griewangk's function. Each algorithm was run 5 times in order to collect better metrics.**

***Index Terms*—hillclimber, genetic algorithm, hybrid, hybridization, Rastrigin, Rosenbrock, Michalewicz, Griewangk, optimization**

I. Introduction

This document details the results of experiments on optimizing 4 well-known functions using 4 different algorithms: two hillclimber, a genetic algorithm and a hybrid algorithm. The aim of the experiments was to observe the inner-workings of each algorithm, how they behave given different functions to optimise, their downsides and their upsides.

The functions that are detailed in this document are the Rastrigin's, Rosenbrock's, Michalewicz's and Griewangk's functions, each being run with the scope of being minimised.

The hybrid algorithm is built using a genetic algorithm as base and a binary hillclimber as a helper, with the aim to combine the benefits of both the specified algorithms.

II. Hillclimbers

A. Description

Hillclimbing is a mathematical optimization technique belonging to the family of local search. The algorithm is iterative and starts with an arbitrary solution to a given problem, which attempts to perfect by performing incremental changes.

The algorithm may regrettably become ensnared within local optima, precluding its ability to escape from such configurations. Nevertheless, it continues to find utility across diverse problem domains, owing to its computational efficiency and swift derivation of an acceptably effective solution.

B. Time complexity

The algorithm has a low time complexity of $\mathcal{O}(nlogn)$, thus it is usually implemented on systems that would need an aproximate solution, them often being realtime systems.

C. Implementations

Hillclimbers can be implemented in multiple ways: binary, floating-point, integer, tree representation etc.. This paper details a binary and a floating-point variant. The following subsections detail how the two compare by comparing the results of the experiments on the 4 described functions.

III. Floating-point hillclimber

A. Description

Floating-point hillclimbers try to emulate how humans would solve an optimization issue, by using decimal numbers. This representation is easy for users to read and thus can be debugged easier.

B. Parameters

The floating-point hillclimber was run using the following parameters:

1) An initial random floating point solution
2) A step size of 0.1
3) An acceleration parameter of 0.1
4) A total of 100 steps

C. Results of the experiments

The results of the experiments were the following:

TABLE I
Results

Alg.	Dims.	Minimum	Mean	Median	Maximum
Rastrigin	2	6.85	15.9	16.0	24.8
	30	4.04e+02	4.9e+02	5.13e+02	5.28e+02
	100	1.6e+03	1.68e+03	1.67e+03	1.77e+03
Rosenbrock	2	5.21	54.1	43.2	1.25e+02
	30	5.51e+03	7.19e+03	6.78e+03	9.69e+03
	100	2.05e+04	2.45e+04	2.53e+04	2.69e+04
Michalewicz	2	-0.801	-0.801	-0.801	-0.801
	30	-5.39	-4.59	-4.59	-3.8
	100	-16.0	-14.2	-13.7	-13.5
Griewangk	2	DnF	DnF	DnF	DnF
	30	DnF	DnF	DnF	DnF
	100	DnF	DnF	DnF	DnF

TABLE II
TIMING IN SECONDS

Alg.	Dims.	Minimum	Mean	Median	Maximum
Rastrigin	2	7.19	8.3	7.58	10.8
	30	73.9	1.11e+02	1.11e+02	1.47e+02
	100	1.88e+02	2.73e+02	2.26e+02	4.53e+02
Rosenbrock	2	18.2	18.5	18.6	18.7
	30	1.3e+02	1.34e+02	1.35e+02	1.38e+02
	100	1.97e+02	3.61e+02	4.15e+02	4.17e+02
Michalewicz	2	12.4	16.9	18.2	18.8
	30	96.7	1.26e+02	1.35e+02	1.4e+02
	100	1.81e+02	2.68e+02	2.3e+02	4.3e+02
Griewangk	2	13.1	20.3	20.2	27.7
	30	66.2	1.08e+02	1.1e+02	1.47e+02
	100	4.21e+02	4.28e+02	4.29e+02	4.32e+02

IV. BINARY HILLCLIMBER

A. Description

While floating-point hillclimbers work using decimal representation, binary hillclimbers work with a representation familiar to computers: the binary representation. Binary hillclimber can provide better performance than floating-point hillclimbers, but they require higher degrees of fine-tuning.

B. Parameters

The binary hillclimber was run using the following parameters:

1) An encode function which encoded the given representation to an array of bytes
2) A decode function which performed the reverse operation of the encode function
3) An initial random floating point solution that is later encoded to binary
4) A total of 100 steps

C. Results of the experiments

The results of the experiments were the following:

TABLE III
RESULTS

Alg.	Dims.	Minimum	Mean	Median	Maximum
Rastrigin	2	0.0	0.0	0.0	0.0
	30	0.0	0.0	0.0	0.0
	100	0.0	0.0	0.0	0.0
Rosenbrock	2	0.99	1.5	1.5	2.0
	30	26.0	27.2	27.4	27.9
	100	82.8	86.1	85.8	89.9
Michalewicz	2	-0.871	-0.702	-0.783	-0.37
	30	-23.1	-22.2	-22.0	-21.5
	100	-78.3	-75.0	-75.5	-70.8
Griewangk	2	0.127	0.127	0.127	0.127
	30	DnF	DnF	DnF	DnF
	100	DnF	DnF	DnF	DnF

TABLE IV
TIMING IN SECONDS

Alg.	Dims.	Minimum	Mean	Median	Maximum
Rastrigin	2	1.58e+02	1.87e+02	1.68e+02	2.55e+02
	30	1.58e+04	2.21e+04	2.13e+04	2.99e+04
	100	1.64e+05	1.72e+05	1.69e+05	1.83e+05
Rosenbrock	2	1.58e+02	3.14e+02	3.44e+02	4.12e+02
	30	1.57e+04	1.89e+04	1.59e+04	2.82e+04
	100	1.7e+05	1.79e+05	1.8e+05	1.86e+05
Michalewicz	2	1.62e+02	3.14e+02	3.35e+02	4.23e+02
	30	1.52e+04	2.51e+04	2.64e+04	3.25e+04
	100	1.59e+05	1.63e+05	1.64e+05	1.67e+05
Griewangk	2	1.95e+02	3.88e+02	4.1e+02	5.38e+02
	30	1.77e+04	2.22e+04	2.01e+04	3.08e+04
	100	1.69e+05	1.72e+05	1.71e+05	1.77e+05

V. GENETIC ALGORITHMS

A. Description

A genetic algorithm represents a way to solve problems from the combinatorial, numerical, optimization, timetabling and scheduling, robotics etc. spaces, in general being very weakly coupled to problems, thus can be used in multiple problem spaces with rather little modifications.

B. Selection function

Selection is the first primary function of a genetic algorithm. The selection function picks the next peers which will have their genes passed to the next generation. There are multiple ways to implement the selection and the code provides two of them: tournament selection and roulette-wheel selection. In this paper, only the tournament selection method was used.

C. Crossover function

Crossover is the second primary function of genetic algorithms. It receives two parents' genes and returns two children that inherit parts of the genes of the two parents. In this paper, the crossover function supports multi-point crossovers.

D. Mutation function

Mutation is the third primary function of genetic algorithms. Given a child, it changes its genes based on a given mutation chance parameter. The mutation is performed on each bit of the child's genes. The purpose of the function is to decrease the chance the algorithm gets stuck in a local optima. In case the individual performs lower due to the changes made, the individual might get dropped the next time the selection function is run.

E. Fitness function

The fitness function represents a function external to the genetic algorithm which it will have to optimise. In this paper, the fitness function is represented by the 4 functions to optimise: Rastrigin's, Rosenbrock's, Michalewicz's and Griewangk's.

F. Time complexity

The time complexity of genetic algorithms can be difficult to determine as it depends on many factors such as the size of the population, the number of generations and the specifics of the algorithm. In general, this would be considered $\mathcal{O}(n^2)$ or $\mathcal{O}(n^3)$.

G. Parameters

The genetic algorithm was run using the following parameters:

1) An encode function which encoded the given representation to an array of bytes
2) A decode function which performed the reverse operation of the encode function
3) An initial population of 100 individuals
4) A total of 100 generations
5) Tournament selection using samples of 20 individuals
6) Multi-point crossover in 2 points: bit 4 and bit 9
7) A mutation chance of 0.0001

H. Results of the experiments

The results of the experiments were the following:

TABLE V
RESULTS

Alg.	Dims.	Minimum	Mean	Median	Maximum
Rastrigin	2	0.0	0.00406	0.0	0.0162
	30	62.4	7.25e+19	1.76e+10	2.9e+20
	100	7.95e+19	6.31e+73	3.46e+39	2.53e+74
Rosenbrock	2	5.7e-05	0.602	0.204	2.0
	30	30.9	1.99e+02	32.4	6.99e+02
	100	1.38e+03	4.18e+150	1.49e+117	1.67e+151
Michalewicz	2	-0.801	-0.6	-0.799	1.78e-19
	30	-14.5	-13.4	-13.4	-12.3
	100	-40.8	-39.6	-39.3	-38.6
Griewangk	2	0.0	0.0382	0.0239	0.105
	30	DnF	DnF	DnF	DnF
	100	DnF	DnF	DnF	DnF

TABLE VI
RUNTIME IN SECONDS

Alg.	Dims.	Minimum	Mean	Median	Maximum
Rastrigin	2	2.48e+03	2.55e+03	2.57e+03	2.59e+03
	30	2.11e+04	2.57e+04	2.6e+04	2.96e+04
	100	7.19e+04	8.9e+04	7.77e+04	1.29e+05
Rosenbrock	2	4.93e+03	5.24e+03	5.33e+03	5.35e+03
	30	2.14e+04	3.39e+04	3.14e+04	5.12e+04
	100	7.35e+04	8.06e+04	7.78e+04	9.35e+04
Michalewicz	2	3.62e+03	4.92e+03	5.34e+03	5.38e+03
	30	2.1e+04	3.63e+04	3.62e+04	5.16e+04
	100	6.59e+04	7.59e+04	7.37e+04	9.03e+04
Griewangk	2	3.85e+03	4.54e+03	4.3e+03	5.71e+03
	30	DnF	DnF	DnF	DnF
	100	DnF	DnF	DnF	DnF

VI. HYBRIDIZATION

A. Motivation

While the 2 algorithms can be used independently of each other and can in turn produce good results, they can also be merged into 1 single algorithm that performs both functions at the same time in the hopes that the solutions are found faster.

In general the Hybrid algorithm inherits from the binary genetic algorithm, while using the hillclimbing algorithm n times.

The algrorithm *might* deceive the user into thinking it is simple in complexity, but having to rely on both the binary genetic algorithm *and* the hillclimber algorithm in the same function calls brings higher complexity than the two algorithms taken without other dependencies.

B. Time complexity

While during a high-level view of the algorithm it appears to have a low complexity, this would have the complexity $\mathcal{O}(n^2+n)$, as both the genetic and the hill climbing algorithms are run in tandem.

C. Parameters

The hybrid algorithm was run using the following parameters:

1) An encode function which encoded the given representation to an array of bytes
2) A decode function which performed the reverse operation of the encode function
3) An initial population of 100 individuals
4) A total of 100 generations
5) Tournament selection using samples of 20 individuals
6) Multi-point crossover in 2 points: bit 4 and bit 9
7) A mutation chance of 0.0001
8) The hillclimber is run once 10 generations
9) The hillclimber is the floating-point hillclimber due to its performance

D. Results of the experiments

The results of the experiments were the following:

TABLE VII
RESULTS

Alg.	Dims.	Minimum	Mean	Median	Maximum
Rastrigin	2	0.0	0.0	0.0	0.0
	30	0.0	0.0	0.0	0.0
	100	0.0	1.62e+36	6.47	6.49e+36
Rosenbrock	2	3.59e-13	0.249	0.00313	0.99
	30	12.2	3.35e+155	20.9	1.34e+156
	100	1.16e+138	1.01e+156	1.34e+156	1.34e+156
Michalewicz	2	-0.985	-0.725	-0.958	5.5e-15
	30	-24.3	-23.0	-22.8	-22.2
	100	-82.5	-79.9	-79.9	-77.3
Griewangk	2	0.0	0.0	0.0	0.0
	30	DnF	DnF	DnF	DnF
	100	DnF	DnF	DnF	DnF

TABLE VIII
RUNTIME IN SECONDS

Alg.	Dims.	Minimum	Mean	Median	Maximum
Rastrigin	2	4e+03	5.77e+03	4.75e+03	9.59e+03
	30	1.95e+05	2.18e+05	1.97e+05	2.85e+05
	100	1.69e+06	1.73e+06	1.73e+06	1.75e+06
Rosenbrock	2	4e+03	5.78e+03	5.12e+03	8.87e+03
	30	1.74e+05	2.09e+05	1.91e+05	2.79e+05
	100	1.66e+06	1.7e+06	1.7e+06	1.71e+06
Michalewicz	2	4e+03	7.98e+03	9.02e+03	9.9e+03
	30	1.99e+05	2.36e+05	2.35e+05	2.73e+05
	100	1.62e+06	1.64e+06	1.64e+06	1.64e+06
Griewangk	2	7.98e+03	9.16e+03	8.94e+03	1.08e+04
	30	DnF	DnF	DnF	DnF
	100	DnF	DnF	DnF	DnF

VII. CONCLUSIONS

Each presented algorithm has its use-cases. While some *can be* used in real-time systems due to their speed, they are prone to getting stuck in the local optima, and while others are really powerful, they are not fast enough to provide solutions in sensible time. A hybridization of the two algorithms would partially mitigate the downsides, perhaps with added complexity.

REFERENCES

[1] Luchian Henri; Eugen Croitoru. “Genetic algorithms (2022)”, Faculty of Computer Science of University “Alexandru Ioan Cuza” of Iași.

[2] Russell, Stuart J.; Norvig, Peter (2003). “Artificial Intelligence: A Modern Approach (2nd ed.)”, Upper Saddle River, New Jersey: Prentice Hall, pp. 111-114, ISBN 0-13-790395-2

[3] Banzhaf, Wolfgang; Nordin, Peter; Keller, Robert; Francone, Frank (1998). “Genetic Programming - An Introduction”. San Francisco, CA: Morgan Kaufmann. ISBN 978-1558605107.