Genetic Algorithms
The Difference Between Using a Genetic Algorithm versus Using Hill Climbing and Simulated Annealing Algorithms in Finding The Minimum of Some Numeric Functions

Melinte Daria-Elena - Group 2E3, Vamanu Petru-Gabriel - Group 2A6

December 2022

1 Abstract

This paper presents how the minimum of some numeric functions can be generated using a Genetic Algorithm.

It also presents the difference between using a Genetic Algorithm and using Hill Climbing (with the following implementations: First Improvement, Best Improvement and Worst Improvement), along with Simulated Annealing (which is hybridising the Hill Climbing First Improvement method).

The Genetic Algorithm, using proper probabilities and optimizations can lead to better results than the other algorithms, results obtained in a way shorter time. However, HC and SA have a smaller standard deviation, meaning that the values are more constant, but those algorithms have a limitation: they reach a certain point where getting a good result requires a very large time of computation. Genetic Algorithms have a larger deviation, but the core result is closer to the global minimum. They also require less computational time, and if they were to match the time of HC and GA (by increasing the number of generations and the size of the population) they would give results close to the real minimum.

2 Introduction

Following the previous reports [[Pet22]] [[Dar22]], it was concluded that each function behaves differently, and that there is no better method for generating the minimum of any function, between Hill Climbing and Simulated Annealing, each of them having some advantages and disadvantages.

Knowing this, the report is going to analyse how a genetic algorithm is behaving when trying to generate the minimum of those functions and it'll compare the results with the best ones from the following reports.

The previous methods used are Hill Climbing [referred from now on as 'HC'], with the folling implementation methods:

- First Improvement [referred from now on as 'FI']
- Best Improvement [referred from now on as 'BI']
- Worst Improvement [referred from now on as 'WI']

along with Simulated Annealing [referred from now on as 'SA'], which is hybridising the HCFI method.

Other adnotated terms:

- Evolutionary Algorithm [referred from now on as 'EA']
- Genetic Algorithm [referred from now on as 'GA']
- Wheel of Fortune Genetic Algorithm [referred from now on as 'WF-GA'][[Eug]]

- Optimised Wheel of Fortune Genetic Algorithm [referred from now on as 'WF-GA(o)'][[Eug]]

The 4 studied functions are:

- De Jong's 1 function
- Schwefel's function
- Rastrigin's function
- Michalewicz's function

3 Methods

3.1 Evolutionary Algorithm

An evolutionary algorithm ['EA'] is a generic population-based metaheuristic optimization algorithm. An EA uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the quality of the solutions. Evolution of the population then takes place after the repeated application of the above operators. [[Wika]]

3.2 Genetic Algorithm

In computer science and operations research, a genetic algorithm ['GA'] is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms. 'GA' are commonly used to generate high-quality solutions to optimization and search problems by relying on biologically inspired operators such as mutation, crossover and selection. [[Wikb]][[Gee]].

This is the most popular type of EA. One seeks the solution of a problem in the form of strings of numbers (traditionally binary, although the best representations are usually those that reflect something about the problem being solved), by applying operators such as recombination and mutation (sometimes one, sometimes both).

An individual (or chromosome) is defined as a bitstring, and a population of individuals (chromosomes) is defined as a vector of bitstrings.

WF-GA implements the following steps:

1. Generate the initial population of individuals randomly (First generation) and evaluate the fitness of each individual in the population by sufficient fitness achieved.
2. Repeat the following regenerational steps until termination:
    - Select the fittest individuals for reproduction (Parents) using the following process:
        - Compute the total sum of each fitness score;
        - Each individual receives a probability which is equal to $fitness\_score/total\_finess\_sum$;
        - Each individual i receives an unique interval which is lower bounded by the previous individual's upper bound and has length equal to it's probability; For the case where i = 0, the lower bound is 0;
        - Next, a random probability is generated, and the individual whose interval contains the probability is added to the new population. This step is repeated for $maximum\_population\_size$ times.
    - Breed new individuals through crossover and mutation operations to give birth to offspring.
    - Evaluate the newly generated population

3.3 The Functions Used [[GEA]]

3.3.1 De Jong's function

De Jong's function is also known as sphere model. It is continuos, convex and unimodal.

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

The global minimum:

$$f(x) = 0, x_i = 0, \forall i \in [1, n], \forall n \in \mathbb{N}$$

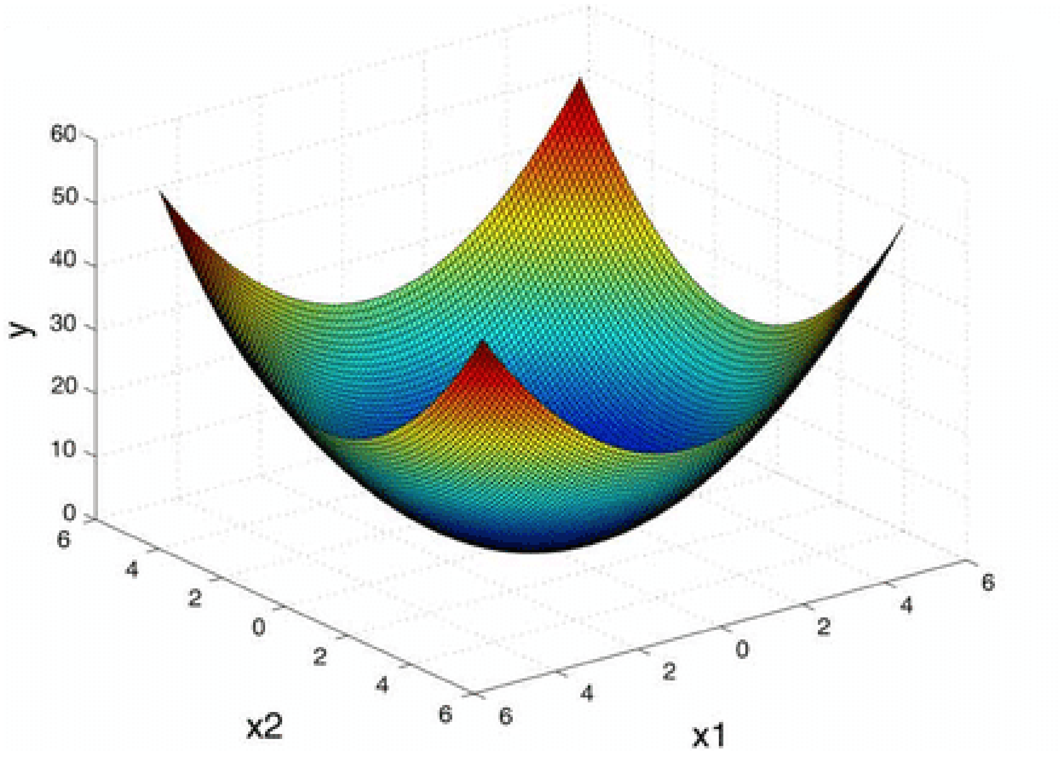

Fig. 1. De Jong's function for n = 2 [[Jon]]

3.3.2 Schwefel's function

Schwefel's function 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.

$$f(x) = 418.9829 \cdot n + \sum_{i=1}^{n}(-x_i \sin \sqrt{|x_i|}), x_i \in [-500, 500]$$

The global minimum:

$$f(x) = 0, x_i = 0, \forall i \in [1, n], \forall n \in \mathbb{N}$$

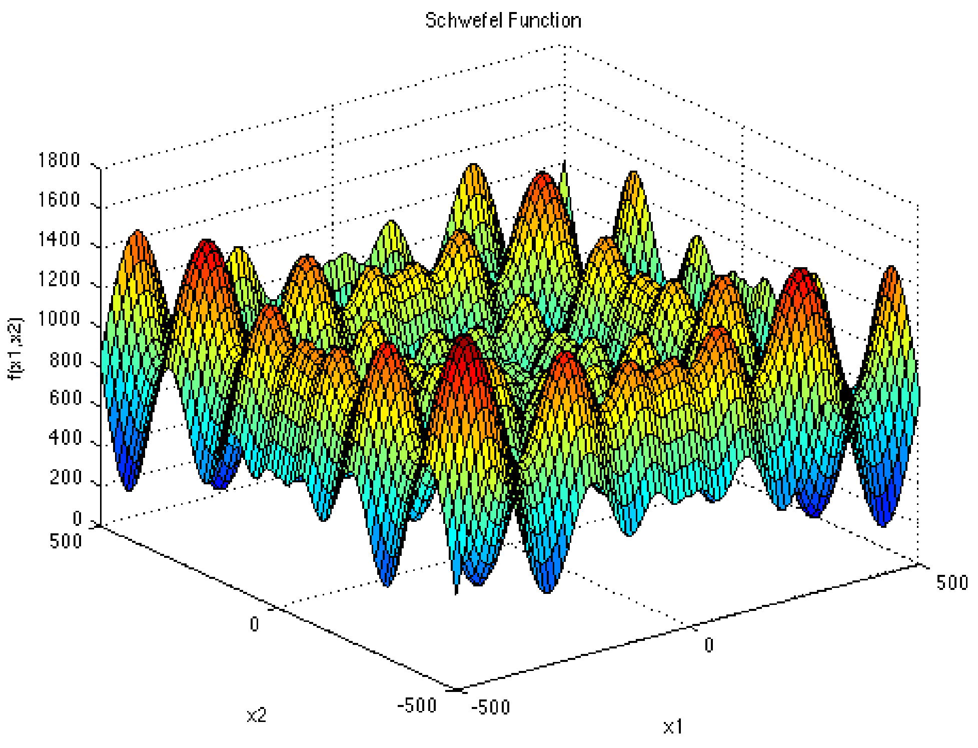

Fig. 2. Schwefel's function for n = 2 [[Sch]]

3.3.3 Rastrigin's function

Rastrigin's function is based on De Jong's first function 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.

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

The global minimum:

$$f(x) = 0, x_i = 0, \forall i \in [1, n], \forall n \in \mathbb{N}$$

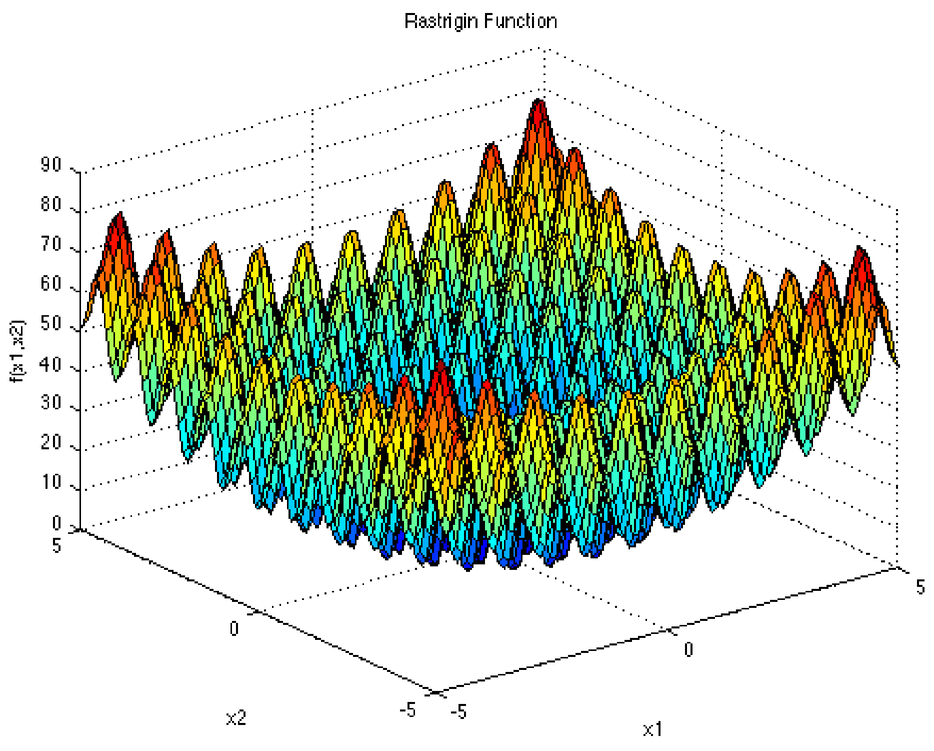

Fig. 3. Rastrigin's function for n = 2 [[Ras]]

3.3.4 Michalewicz's function

The Michalewicz function is a multi-modal 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).

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

The global minimum:

$$n = 5 : f(x) = -4.687658$$
$$n = 10 : f(x) = -9.66015$$
$$n = 30 : f(x) = -29.63088[[\text{Cha}]]$$

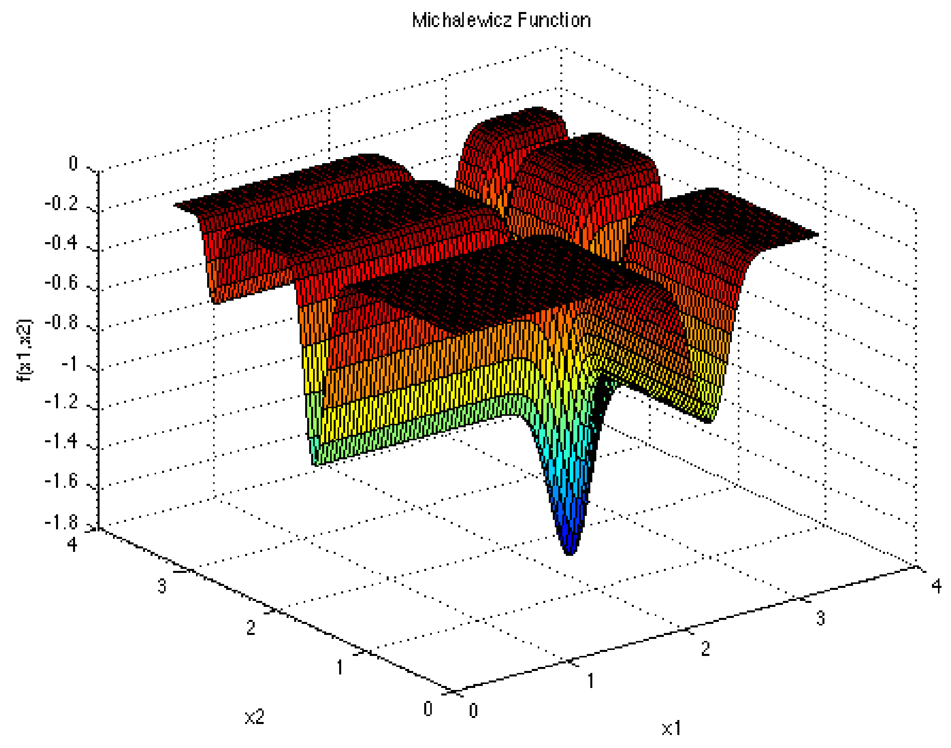

Fig. 4. Michalewicz's function for n = 2 [[Mic]]

4 Experimental set-up

For HC and SA, each function has been tested on 5, 10 and 30 dimensions, with 30 tests (samples). WF-GA and WF-GA(o) were also tested on 5, 10 and 30 dimensions, but with 50 tests (samples) each. For the observed GA's, the population size was 200 and the number of generations was 2000.

4.1 Fitness function

The fitness function has a great impact in the selection process of the population. It is used to measure better the quality of the chromosomes and it is based on the numerical function that needs to be optimised. It needs to be positive function and it is created in order to maximize the selection's quality (the best adapted individuals get big results on the fitness function).

Considering g one of the studied functions, a constant big enough is added in order to create a strictly positive result for the fitness function to work correctly

$$f'(x) = g(x) + C,$$

where:

- C = 30 for Michalewicz's function (as the global minimum for 30 dimensions is -29.63088)
- C = 0.0001 for De Jong's, Rastrigin's and Schwefel's functions (as the global minimum is 0)

Then, the fitness function f is obtained by inverting the resulted function f':

$$f(x) = 1/f'(x)$$

4.2 Mutation and crossover

For each function there were used different probabilities for the following genetic operators:

- Mutation
- Cross-over

For WF-GA and WF-GA(o), on 5 and 10 dimensions, the chosen mutation probability is 0.002 and for cross-over the chosen probability is 0.8.

One notice is that on smaller dimensions, the mutation and crossover probabilities are important, but their variation does not impact the results greatly.

On the other hand, on 30 dimensions the mutation and crossover probabilities impact the quality of the results greatly, each function having different behaviours under those probabilities. For 30 dimensions, the following probabilities had the best results for WF-GA.

	Mutation Probability	Cross-over Probability
De Jong's function	0.0001	0.9
Rastrigin's function	0.002	0.95
Schwefel's function	0.0008	0.89
Michalewicz's function	0.00035	0.98

The mutation process is done after each selection step and it is done as follows:

- each individual of the population is selected;
- for each locus there is generated a random probability;
- the gene is mutated if the generated probability is smaller than the chosen mutation probability.

The cross-over is single-pointed, and the parents' breaking point is chosen randomly. For the cross-over process, the population is randomly shuffled for the parents to be chosen randomly, and after that the following steps are repeated for each group of parents:

- for each cross-over trial a random probability is generated;
- a descendant is created from the two parents if the generated probability is smaller than the chosen cross-over probability.

4.3 Optimizations

For the optimized algorithm there were used 4 optimizations:

- Elitism: the best 5% individuals of the population are selected by default and skip the Wheel of Fortune selection.
- Usage of Gray Codes: Instead of decoding the bitstrings as binary, they are decoded as Gray Codes [[Wikc]]

- The worst 5% individuals of the population are discarded and replaced with new randomly generated ones (method inspired by Harmony Search and Reverse Elitism)

Idem to WF-GA, each function has different behaviours under different mutation and cross-over probabilities. For 30 dimensions, the following probabilities had the best results for WF-GA(o).

	Mutation Probability	Cross-over Probability
De Jong's function	0.0001	0.9
Rastrigin's function	0.002	0.95
Schwefel's function	0.0008	0.89
Michalewicz's function	0.00035	0.98

5 Results and Analysis

5.1 De Jong's function

5.1.1 Analysis

One observation is that De Jong's function behaves worse under WF-GA's than any form of HC or SA. WF-GA succeeds in finding the global minimum on smaller dimensions, but on 30 dimensions it fails to find the actual global minimum, even though its results aren't far from the desired value. WF-GA(o) succeeds on finding the global minimum on 30 dimensions as well, but compared to HC and SA, where the worst generated value is still the global minimum, WF-GA and WF-GA(o) don't behave the same.

Despite its slightly inaccuricy, it is fair to say that the WF-GA's have a way smaller running time, having a median time of 13.3136 on 30 dimensions, which is 81 times faster than HCBI, and around 11 times faster than HCWI and SA.

5.1.2 Results Table

De Jong's 1 Function	**5 dimensions**					
	WF-GA	WF-GA(o)	HCFI	HCBI	HBWI	SA[FI]
global value	0.00000					
best	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
worst	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
median	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
standard deviation	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
IQR	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
median time	3.53402	2.51667	4.9875	78.536	19.171	145.88
	10 dimensions					
	WF-GA	WF-GA(o)	HCFI	HCBI	HBWI	SA[FI]
global value	0.00000					
best	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
worst	0.04054	0.00001	0.00000	0.00000	0.00000	0.00000
median	0.00003	0.00000	0.00000	0.00000	0.00000	0.00000
standard deviation	0.00572	0.00000	0.00000	0.00000	0.00000	0.00000
IQR	0.00004	0.000003	0.00000	0.00000	0.00000	0.00000
median time	6.80852	4.71057	9.1957	290.64	58.836	92.071
	30 dimensions					
	WF-GA	WF-GA(o)	HCFI	HCBI	HBWI	SA[FI]
global value	0.00000					
best	0.03774	0.00000	0.00000	0.00000	0.00000	0.00000
worst	0.93934	0.00023	0.00000	0.00000	0.00000	0.00000
median	0.20998	0.00003	0.00000	0.00000	0.00000	0.00000
standard deviation	0.20987	0.00004	0.00000	0.00000	0.00000	0.00000
IQR	0.23059	0.0000375	0.00000	0.00000	0.00000	0.00000
median time	20.681	13.3136	17.0856	1086.1	148.2	144.08

5.2 Rastrigin's function

5.2.1 Analysis

Over all dimensions, WF-GA gives worse results than HC and SA, but WF-GA(o) gives way better results than any other algorithm, as the number of dimensions increases. On 30 dimensions, WF-GA(o) gives the smallest median, and reaches a closer value to the global minimum (2.55204 being the smallest value found, which is at least 10 times smaller than the other ones).

The biggest improvement for Rastrigin's function comes from using the Gray Codes. Without it, WF-GA(o)'s results were similar to SA's results, but the usage of Gray Codes helped the algorithm to overcome some local minimas and find other local ones such as 2.55204.

5.2.2 Results Table

Rastrigin's Function	**5 dimensions**					
	WF-GA	WF-GA(o)	HCFI	HCBI	HBWI	SA[FI]
global value	0.00000					
best	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
worst	6.08081	0.00001	0.99496	0.99496	2.00002	1.00001
median	1.23296	0.00000	0.00000	0.00000	1.00001	0.00000
standard deviation	1.30777	0.000001	0.46374	0.30359	0.43226	0.5054
IQR	1.53112	0.00000	0.99496	0.00000	0.17685	0.99496
median time	2.6652	2.57214	206.1306	74.43	1237.099	683.9208
	10 dimensions					
	WF-GA	WF-GA(o)	HCFI	HCBI	HBWI	SA[FI]
global value	0.00000					
best	0.00043	0.00004	2.23078	2.23078	5.23587	4.22575
worst	12.29853	4.94419	5.93825	4.70748	9.99933	7.44644
median	4.47363	0.00146	4.46156	3.4666	7.353795	5.58201
standard deviation	3.20065	0.99856	1.02384	0.65894	1.15128	0.86044
IQR	4.97465	1.18442	1.98486	0.99496	1.412685	0.92307
median time	6.79221	4.71057	226.0727	114.08	515.9031	1021.959
	30 dimensions					
	WF-GA	WF-GA(o)	HCFI	HCBI	HBWI	SA[FI]
global value	0.00000					
best	85.79215	2.55204	36.32933	21.83114	39.39459	33.24138
worst	140.90518	51.85954	42.91354	35.12975	50.62801	44.76401
median	109.49524	23.12909	40.0004	32.006165	47.51276	39.46713
standard deviation	14.29135	12.40574	2.04623	3.85514	2.90894	3.20104
IQR	14.1727	14.80710	2.14308	3.85159	1.35426	3.67694
median time	19.99931	13.87165	467.7573	825.7466	5144.311	758.2673

5.3 Schwefel's function

5.3.1 Analysis

For Schwefel's function, WF-GA and WF-GA(o) generate significant better results than HC and SA, with the minimum of time complexity. The most noticed result is that on 30 dimensions, using WF-GA(o), which is 0.82779. This means that GA's succeed in overcoming the function's deceptiveness, which is caused by the global minimum, which is geometrically distant from the next best local minima.

Even though WF-GA has alone better results than any HC and SA, it can be said that the significant difference between WF-GA and WF-GA(o) is the usage of Gray Codes for the second one, which makes possible the escape from the local minimas.

5.3.2 Results Table

Schwefel's Function	**5 dimensions**					
	WF-GA	WF-GA(o)	HCFI	HCBI	HBWI	SA[FI]
global value	0.00000					
best	0.00131	0.00006	0.00068	0.00007	26.88441	0.00105
worst	6.64515	0.31109	0.20864	0.00194	80.64972	27.09039
median	0.3118	0.10373	0.10436	0.00069	53.81873	0.20772
standard deviation	1.09768	0.09099	0.07216	0.00053	14.02533	6.79638
IQR	0.10605	0.20609	0.10304	0.00061	0.20665	0.20732
median time	3.07557	3.39502	21.3469	44.28178	300.1138	47.13488
	10 dimensions					
	WF-GA	WF-GA(o)	HCFI	HCBI	HBWI	SA[FI]
global value	0.00000					
best	0.2537	0.00085	61.43091	0.10692	188.18211	54.07839
worst	242.76589	0.79979	272.02261	119.34472	268.83008	322.26153
median	33.707895	0.22388	153.57800	30.97500	226.09357	258.35777
standard deviation	64.12902	0.17488	47.16723	28.45835	20.67468	70.60492
IQR	95.32042	0.13932	60.28615	34.02928	26.88265	75.71593
median time	5.93304	6.330292	39.6005	133.9764	982.1233	35.03067
	30 dimensions					
	WF-GA	WF-GA(o)	HCFI	HCBI	HBWI	SA[FI]
global value	0.00000					
best	619.47038	0.82779	1526.49669	868.69176	1661.64966	1484.99991
worst	2553.85749	1119.61843	1974.92907	1572.99045	1990.33731	1996.80887
median	1584.71965	435.88024	1810.30834	1324.04481	1883.11072	1883.17991
standard deviation	424.22334	295.96936	116.05713	165.88311	83.45027	136.47694
IQR	552.68525	435.19501	154.74617	220.92253	124.60610	173.88256
median time	17.48498	17.86499	17.18828	173.577	1152.675	68.33316

5.4 Michalewicz's function

5.4.1 Analysis

WF-GA(o) does not use Gray Codes because the function behaves as a needle in the haystack and most of its values are far from the minimum. The Gray Codes make the search be a more dynamic one by having a flipped bit give a different number, not having a number mathematically related by a stable conversion between numbers.

Even though WF-GA has a general behaviour worse than HC and SA, WF-GA(o) manages to also obtain exact results for 5 dimensions, and the best results for 30 dimensions.

The results on 10 dimensions don't stand out, but the execution time is better for WF-GA and WF-GA(o) than for HC and SA on any dimension.

5.4.2 Results Table

Michalewicz's Function	**5 dimensions**					
	WF-GA	WF-GA(o)	HCFI	HCBI	HBWI	SA[FI]
global value	-4.687658					
best	-4.66901	-4.68765	-4.68765	-4.68765	-4.68713	-4.68765
worst	-4.31789	-4.52267	-4.68593	-4.68532	-4.66646	-4.68591
median	-4.5457	-4.68291	-4.68759	-4.68764	-4.67047	-4.68751
standard deviation	0.09380	0.0419	0.00043	0.00063	0.00646	0.00051
IQR	0.1501	0.04172	0.00013	0.00013	0.01123	0.00070
median time	2.29284	2.5713	105.846	62.03143	216.8206	602.69
	10 dimensions					
	WF-GA	WF-GA(o)	HCFI	HCBI	HBWI	SA[FI]
actual value	-9.66015					
best	-9.1915	-9.5687	-9.60625	-9.58665	-9.3071	-9.51136
worst	-7.62261	-8.59043	-9.31353	-9.27662	-8.56723	-9.14475
median	-8.71632	-9.19345	-9.408685	-9.39694	-8.77951	-9.310765
standard deviation	0.34971	0.24213	0.08714	0.08580	0.17144	0.09172
IQR	0.44174	0.34732	0.12531	0.12856	0.11405	0.14261
median time	4.31461	4.84138	412.259	141.4643	517.9206	607.9623
	30 dimensions					
	WF-GA	WF-GA(o)	HCFI	HCBI	HBWI	SA[FI]
actual value	-29.63088					
best	-26.98699	-28.04323	-26.60306	-27.36084	-24.09767	-26.82681
worst	-22.32601	-24.79845	-25.73246	-26.55666	-22.61183	-25.67059
median	-24.61899	-26.76367	-26.033265	-26.978745	-23.13036	-26.09066
standard deviation	1.11612	0.73773	0.27988	0.26775	0.48315	0.33037
IQR	1.62637	1.14417	0.35630	0.401607	0.68375	0.49607
median time	12.71202	13.51155	763.7577	1426.031	4210.193	1276.721

6 Conclusion

Genetic Algorithms can be a very good solution for finding the minimum of a numeric function, especially if optimised properly: each function requires different improvements such as different probabilities for the genetic operators, applying different meta-algorithms or heuristics, depending on the analyzed function (Gray Codes work really well on Rastrigin's, but are not good on Michalewicz's).

It is fair to say, however, that an increase in the number of generations used would further improve the results, but it is fair to say that a too big increase would get the WF-GA capped. The increase of population, however, would not bring much accuracy improvement, but rather an increase in the running time, as the results obtained using a population of 200 individuals are slightly different from those obtained using a population of 150 individuals.

Another aspect that would further improve the results would be to find more exact probabilities for mutation and cross-over than the ones found using exhaustive trials.

In comparison with HC and SA, GA's give results differently:

- HC and SA have a smaller interval of values for the minimum, while WF-GA and WF-GA(o) have a wider one.
- GA's intervals are closer to the global minimum than HC and SA, meaning that by being wider, they include smaller values, and on bigger dimensions having even the median way smaller than the median for HC and SA.

GA's runtime is smaller than HC's and SA's. If the number of generations were increased enough, GA would reach the time of HC and SA, but giving way better results because the values are better after each generations (and the worst ones being replaced).

Overall, Genetic Algorithms are a great way to approach the problem of finding the global minimum of a numeric function. It is a better alternative than heuristic algorithms only on multi-modal functions.

They also represent a trade in time: more time for finding a good approach in a genetic way for a smaller run time of the algorithm, or less time spent for finding a heuristic algorithm for a bigger run time.

Bibliography

[Dar22] Melinte Daria-Elena. "Genetic Algorithms The Use of Hill Climbing and Simulated Annealing Algorithms in Finding The Minimum of Some Mathematical Functions". In: (2022).

[Pet22] Vamanu Petru-Gabriel. "[GA] The difference between two heuristic algorithms for determining the minimum of a function: Hill Climbing & Simulated Annealing". In: (2022).

[Cha] Nicolas Durand Charlie Vanaret Jean-Baptiste Gotteland. *The report which holds the global minimum for Michalewicz's function, for n = 30 (Page 5, Table 3).* URL: https://arxiv.org/pdf/2003.09867.pdf.

[Eug] Croitoru Eugen. *Notiuni: Algoritmi Genetici.* URL: https://profs.info.uaic.ro/~eugennc/teaching/ga/#Notions04.

[GEA] GEATbx. *Informations about the numerical functions.* URL: http://www.geatbx.com/docu/fcnindex-01.html.

[Gee] with contribution from Atul Kumar Geeks for Geeks. *Genetic Algorithms.* URL: https://www.geeksforgeeks.org/genetic-algorithms/.

[Jon] De Jong. *De Jong's 1 Function.* URL: https://www.researchgate.net/figure/llustrates-the-2D-Dejong-Dejong-2D-and-2D-Ackley-Ackley-2D-functions-The-selected_fig4_285729352.

[Mic] Michalewicz. *Michalewicz's Function.* URL: https://www.sfu.ca/~ssurjano/michal.png.

[Ras] Rastrigin. *Rastrigin's Function.* URL: https://www.sfu.ca/~ssurjano/rastr.png.

[Sch] Schwefel. *Schwefel's Function.* URL: https://www.sfu.ca/~ssurjano/schwef.png.

[Wika] Wikipedia. *Evolutionary algorithm.* URL: https://en.wikipedia.org/wiki/Evolutionary_algorithm.

[Wikb] Wikipedia. *Genetic algorithm.* URL: https://en.wikipedia.org/wiki/Genetic_algorithm.

[Wikc] Wikipedia. *Gray Code.* URL: https://en.wikipedia.org/wiki/Gray_code.