A comparative study of numeric function optimisation algorithms

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

***Abstract*—This document contains a study of the runtimes differences of a binary hillclimber, a floating-point hillclimber, a genetic algorithm, a hybrid genetic algorithm and the particle swarm optimisation algorithm. The study was conducted on 4 numeric functions which are used for benchmarking: Rosenbrock’s, Michalewicz’s, Griewangk’s and Rastrigin’s. The algorithms were executed on an Intel Core i7 1260p processor at a constant 2100MHz. Each algorithm is analysed for its timings and results. The use-cases for each algorithm are also presented.**

***Index Terms*—Hillclimbing, Genetic algorithm, Hybridisation, Particle swarm optimisation, Rastrigin, Rosenbrock, Michalewicz, Griewangk, optimisation, numeric function optimisation**

I. Introduction

This document will perform a comparative study of the 4 already studied optimisation algorithms: hillclimbing, genetic algorithms, a hybrid algorithm - a genetic algorithm + a hillclimber - and the particle swarm optimisation.

The algorithms were run in the following variants:

1) Hillclimber: Binary, Floating-point
2) Genetic algorithm: Binary
3) Hybrid algorithm: Binary Genetic algorithm + Binary hillclimber
4) Particle swarm optimisation algorithm: Floating-point

In this document, the following results were used:

1) For Hillclimbers, Genetic algorithms, Hybrid algorithms: new runs
2) For Particle swarm optimisation: runs from the previous algorithm study

II. Motivation

The studied algorithms work in very different ways and have its intended use-cases. Due to this, they can perform really differently from each other given the same function to optimise. This behaviour will be studied in this document in order to find the best performing algorithm, the fastest algorithm and the best overall algorithm.

III. Benchmark functions

The algorithms were run on the following 4 benchmark functions:

- Rastrigin’s
- Griewangk’s
- Rosenbrock’s
- Michalewicz’s

A. Rastrigin’s function

Rosenbrock’s function, often referred to as the ”banana function” or ”valley function,” is a non-convex mathematical problem commonly used to test optimization algorithms. It features a long, narrow, parabolic shaped valley, and the global minimum is inside this curved, narrow valley. The function poses difficulties for optimization algorithms due to the flatness of the valley, requiring methods capable of efficiently navigating along the elongated, curved path to find the optimal solution.

The function has the following definition:

$$f(x) = \sum_{i=1}^{n-1} *100 * (x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \tag{1}$$

With $-2.048 \leq x_i \leq 2.048$

B. Griewangk’s function

Griewangk’s function is another classical benchmark in optimization, known for its multiple global minima separated by flat regions. The function contains oscillations, and the challenge lies in finding the global minimum in the presence of these oscillations and flat regions. Griewangk’s function is frequently employed to assess an algorithm’s ability to balance exploration and exploitation in complex search spaces, making it a relevant test case for optimization algorithms.

The function has the following definition:

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} *cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{2}$$

With $-600 \leq x_i \leq 600$

C. Rosenbrock’s function

Rosenbrock’s function, often referred to as the ”banana function” or ”valley function,” is a non-convex mathematical problem commonly used to test optimization algorithms. It features a long, narrow, parabolic shaped valley, and the global minimum is inside this curved, narrow valley. The function poses difficulties for optimization algorithms due to the flatness of the valley, requiring methods capable of efficiently navigating along the elongated, curved path to find the optimal solution.

The function has the following definition:

$$f(x) = \sum_{i=1}^{n-1} *100 * (x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \qquad (3)$$

With $-2.048 \leq x_i \leq 2.048$

D. *Michalewicz's function*

Michalewicz's function is a multimodal optimization problem proposed to evaluate the performance of optimization algorithms in handling functions with multiple peaks. It is characterized by its sharp, narrow peaks, and the location of the global optimum depends on the problem's dimensionality. The function's multimodal nature challenges algorithms to explore and exploit different regions of the search space efficiently, making it a valuable test case for optimization studies.

The function has the following definition:

$$f(x) = \sum_{i=1}^{n} *sin(x_i) * (sin(\frac{i * x_i^2}{\pi}))^2 * m \qquad (4)$$

With $i = 1 : n, m = 10, 0 \leq x_i \leq \pi$

IV. Performance metrics

This document will present the following metrics:

1) Solution quality
2) Convergence speed (in seconds)

V. Setup

A. *Environment setup*

The algorithm was executed on a laptop with an Intel i7 1260p processor with 12 cores running at 2100MHz and 32GB of DDR4 RAM. As for the software setup, the algorithm was run using Python 3.11.6 and numpy 1.26.0 on a system running Linux 6.1.66 LTS.

B. *Algorithm configurations*

For the binary hillclimber, the following parameters were used:

1) iterations = 100
2) bit shifts per iteration = 1
3) criteria function = either iterations is higher than 100, or the result of the current iteration is within ± 0.05 of the best results of the current function

For the floating-point hillclimber, the following parameters were used:

1) iterations = 100
2) initial value = uniform random values within the limits for the functions above
3) step size = 0.01
4) acceleration = 0.01
5) criteria function = either iterations is higher than 100, or the result of the current iteration is within ± 0.05 of the best results of the current function

For the binary genetic algorithm, the following parameters were used:

1) generations = 100
2) population size = 100
3) population initial values = uniform random values within the limits for the functions above
4) selection function = roulette wheel selection
5) criteria function = either generation is higher than 100, or the best result within the current generation is within ± 0.05 of the best results of the current function

For the hybrid algorithm, the following parameters were used:

1) for the binary genetic algorithm:
    a) generations = 100
    b) population size = 100
    c) population initial values = uniform random values within the limits for the functions above
    d) selection function = roulette wheel selection
    e) criteria function = either generation is higher than 100, or the best result within the current generation is within ± 0.05 of the best results of the current function
2) for the binary hillclimber:
    a) run interval = 10 generations of the genetic algorithm
    b) iterations = 100
    c) bit shifts per iteration = 1

For the particle swarm optimisation algorith, the following parameters were used:

1) $w \in [0.0, 1.0]$
2) $\phi_p \in [0.0, 1.0]$
3) $\phi_g \in [0.0, 1.0]$
4) $\phi_r \in [0.0, 5.0]$
5) $iterations \in \mathbb{N}, iterations \in [0, 500]$

VI. Results

A. *Rastrigin's function*

The results and timings were the following:

Dims	Algorithm	Best	Mean	Median	Worst
2	Binary hillclimber	0.0000	14.9101	9.0019	38.6621
	Floating-point hillclimber	16.9851	20.5340	16.9851	46.3439
	Binary genetic algorithm	0.0	0.0040	0.0	0.0162
	Hybrid algorithm	0.0	0.0040	0.0	0.0162
	PSO algorithm	3.1526	289299.3240	51.0222	24180855.8853
30	Binary hillclimber	339.5148	454.9727	456.3475	539.9070
	Floating-point hillclimber	527.7436	532.3054	527.7441	553.3553
	Binary genetic algorithm	2.11e+04	2.57e+04	2.6e+04	2.96e+04
	Hybrid algorithm	0.0	0.0	0.0	0.0
	PSO algorithm	64.6967	37317.4387	525.9135	2935912.0319
100	Binary hillclimber	1739.7672	1806.6790	1822.5269	1853.5566
	Floating-point hillclimber	1731.1418	1734.3988	1731.1426	1757.0236
	Binary genetic algorithm	7.19e+04	8.9e+04	7.77e+04	1.29e+05
	Hybrid algorithm	0.0	1.62e+36	6.47	6.49e+36
	PSO algorithm	187.4847	1517.4707	1726.1001	3642.6197

Fig. 1: Results for the Rastrigin's function

Dims	Algorithm	Best	Mean	Median	Worst
	Binary hillclimber	0.0005s	0.0009s	0.0010s	0.0013s
	Floating-point hillclimber	0.0004s	0.0024s	0.0025s	0.0046s
2	Binary genetic algorithm	0.0164s	0.5570s	0.5691s	1.0929s
	Hybrid algorithm	0.0456s	1.2920s	1.3241s	2.5673s
	PSO algorithm	0.2834s	1.1082s	1.0721s	2.4293s
	Binary hillclimber	0.0008s	0.0115s	0.0122s	0.0218s
	Floating-point hillclimber	0.0005s	0.0060s	0.0062s	0.0114s
30	Binary genetic algorithm	0.0586s	2.9419s	2.9874s	5.8580s
	Hybrid algorithm	0.0984s	3.1343s	3.2080s	6.2076s
	PSO algorithm	2.2084s	10.5571s	10.0289s	25.3332s
	Binary hillclimber	0.0014s	0.0481s	0.0492s	0.0940s
	Floating-point hillclimber	0.0005s	0.0061s	0.0062s	0.0117s
100	Binary genetic algorithm	0.3342s	10.1814s	9.3397s	21.7908s
	Hybrid algorithm	0.5890s	19.0899s	20.2209s	33.9739s
	PSO algorithm	0.9628s	3.9553s	3.8626s	9.0846s

Fig. 2: Timing results for the Rastrigin's function

B. *Griewangk's function*

The results and timings were the following:

Dims	Algorithm	Best	Mean	Median	Worst
	Binary hillclimber	0.0000	4.9814	0.4961	43.2721
	Floating-point hillclimber	11.6667	11.9533	11.9204	12.3932
2	Binary genetic algorithm	0.0	0.0382	0.0239	0.105
	Hybrid algorithm	0.0000	1e+24	2e+73	3e+73
	PSO algorithm	0.3774	1896448667.6981	39.3992	189644194583.7015
	Binary hillclimber	582.9004	644.2461	629.8168	710.2131
	Floating-point hillclimber	860.6576	860.6872	860.6872	860.7193
30	Binary genetic algorithm	901.0866	nan	3e+73	4e+73
	Hybrid algorithm	1214.0724	nan	7324.9636	7e+73
	PSO algorithm	4.7487	279511.3896	289.7916	27807028.3559
	Binary hillclimber	3123.8240	3248.9170	3250.0484	3341.4136
	Floating-point hillclimber	2742.5421	2743.1359	2743.1457	2743.7381
100	Binary genetic algorithm	5537932.3788	nan	2e+73	7e+73
	Hybrid algorithm	696766.9824	nan	8e+72	6e+73
	PSO algorithm	30.3857	4184.1950	479.8178	244163.0585

Fig. 3: Results for the Griewangk's function

Dims	Algorithm	Best	Mean	Median	Worst
	Binary hillclimber	0.0007s	0.0050s	0.0052s	0.0091s
	Floating-point hillclimber	0.0005s	0.0099s	0.0100s	0.0190s
2	Binary genetic algorithm	0.0318s	1.5684s	1.6009s	3.0972s
	Hybrid algorithm	0.0576s	1.6682s	1.7091s	3.3035s
	PSO algorithm	0.3287s	1.4607s	1.3627s	3.0342s
	Binary hillclimber	0.0008s	0.0148s	0.0151s	0.0287s
	Floating-point hillclimber	0.0009s	0.0217s	0.0223s	0.0423s
30	Binary genetic algorithm	0.2013s	10.5891s	10.8153s	20.9261s
	Hybrid algorithm	0.3762s	11.5120s	11.7995s	22.7946s
	PSO algorithm	1.6164s	7.4485s	7.4466s	15.6951s
	Binary hillclimber	0.0022s	0.0961s	0.0980s	0.1904s
	Floating-point hillclimber	0.0015s	0.0547s	0.0554s	0.1079s
100	Binary genetic algorithm	0.6192s	28.8511s	33.4532s	48.4885s
	Hybrid algorithm	0.5305s	16.4300s	16.8095s	32.5512s
	PSO algorithm	7.2799s	21.5996s	20.5704s	48.4895s

Fig. 4: Timing results for the Griewangk's function

C. *Rosenbrock's function*

The results and timings were the following:

Dims	Algorithm	Best	Mean	Median	Worst
	Binary hillclimber	697403.0000	4978467.7368	5181341.0000	9056637.0000
	Floating-point hillclimber	543139.0000	9907950.2079	10046630.0000	18969279.0000
2	Binary genetic algorithm	31833633.0000	1568375441.6040	1600930586.0000	3097171277.0000
	Hybrid algorithm	57641551.0000	1668182656.3564	1709052497.0000	3303516854.0000
	PSO algorithm	1.4622	1.6474	302.7872	1.6474531444333964e+23
	Binary hillclimber	811582.0000	14789101.0891	15114468.0000	28681202.0000
	Floating-point hillclimber	871513.0000	21690034.6733	22342907.0000	42333879.0000
30	Binary genetic algorithm	201338970.0000	10589126369.6139	10815318528.0000	20926146046.0000
	Hybrid algorithm	376194608.0000	11511991836.3564	11799458194.0000	22794584309.0000
	PSO algorithm	17.1632	17351534044.6623	1249.6179	1734833610964.9421
	Binary hillclimber	2217651.0000	96100663.4554	97976613.0000	190385361.0000
	Floating-point hillclimber	1518261.0000	54712906.7129	55411947.0000	107929377.0000
100	Binary genetic algorithm	619190135.0000	28851090713.1980	33453187623.0000	48488516055.0000
	Hybrid algorithm	530543138.0000	16429975044.9901	16809543140.0000	32551173886.0000
	PSO algorithm	94.8038	3906404.6049	9587.0894	313951466.4197

Fig. 5: Results for the Rosenbrock's function

Dims	Algorithm	Best	Mean	Median	Worst
	Binary hillclimber	0.0007s	0.0050s	0.0052s	0.0091s
	Floating-point hillclimber	0.0005s	0.0099s	0.0100s	0.0190s
2	Binary genetic algorithm	0.0318s	1.5684s	1.6009s	3.0972s
	Hybrid algorithm	0.0576s	1.6682s	1.7091s	3.3035s
	PSO algorithm	0.2898s	1.0112s	0.9163s	2.6347s
	Binary hillclimber	0.0008s	0.0148s	0.0151s	0.0287s
	Floating-point hillclimber	0.0009s	0.0217s	0.0223s	0.0423s
30	Binary genetic algorithm	0.2013s	10.5891s	10.8153s	20.9261s
	Hybrid algorithm	0.3762s	11.5120s	11.7995s	22.7946s
	PSO algorithm	1.2425s	4.9661s	5.0002s	10.7783s
	Binary hillclimber	0.0022s	0.0961s	0.0980s	0.1904s
	Floating-point hillclimber	0.0015s	0.0547s	0.0554s	0.1079s
100	Binary genetic algorithm	0.6192s	28.8511s	33.4532s	48.4885s
	Hybrid algorithm	0.5305s	16.4300s	16.8095s	32.5512s
	PSO algorithm	3.6067s	12.7077s	12.4873s	29.4881s

Fig. 6: Timing results for the Rosenbrock's function

D. *Michalewicz's function*

The results and timings were the following:

Dims	Algorithm	Best	Mean	Median	Worst
	Binary hillclimber	-0.871	-0.702	-0.7832	-0.3741
	Floating-point hillclimber	-0.8013	-0.8012	-0.8013	-0.7977
2	Binary genetic algorithm	-0.9998	nan	-0.0000	0.9956
	Hybrid algorithm	-0.9945	nan	-0.0001	0.9870
	PSO algorithm	3.1526	289299.3240	51.0222	24180855.8853
	Binary hillclimber	-7.2930	-5.7429	-5.5782	-4.3115
	Floating-point hillclimber	-1.4753	-1.4628	-1.4753	-1.3643
30	Binary genetic algorithm	-3.5158	nan	-0.5152	3.4891
	Hybrid algorithm	-3.5358	nan	-0.0950	3.2950
	PSO algorithm	64.6967	37317.4387	525.9135	2935912.0319
	Binary hillclimber	-14.4502	-13.5119	-13.6185	-12.1896
	Floating-point hillclimber	-13.6247	-13.6233	-13.6247	-13.5821
100	Binary genetic algorithm	-9.9868	nan	0.1035	5.2646
	Hybrid algorithm	-82.5231	-79.9512	-79.9654	-77.3173
	PSO algorithm	187.4847	1517.4707	1726.1001	3642.6197

Fig. 7: Results for the Michalewicz's function

Dims	Algorithm	Best	Mean	Median	Worst
	Binary hillclimber	0.0002s	0.0003s	0.0003s	0.0004s
	Floating-point hillclimber	0.0005s	0.0047s	0.0050s	0.0076s
2	Binary genetic algorithm	0.0158s	0.5924s	0.6046s	1.1609s
	Hybrid algorithm	0.0511s	1.4647s	1.4954s	2.8858s
	PSO algorithm	0.3663s	1.2831s	1.0812s	3.6374s
	Binary hillclimber	0.0011s	0.0329s	0.0337s	0.0642s
	Floating-point hillclimber	0.0007s	0.0166s	0.0170s	0.0326s
30	Binary genetic algorithm	0.0931s	4.8493s	4.9421s	9.6158s
	Hybrid algorithm	0.1663s	6.8405s	7.1329s	12.2268s
	PSO algorithm	1.9223s	7.0661s	6.5579s	16.1011s
	Binary hillclimber	0.0024s	0.0949s	0.1044s	0.1611s
	Floating-point hillclimber	0.0027s	0.1152s	0.1174s	0.2275s
100	Binary genetic algorithm	0.6308s	31.9273s	32.4360s	62.7921s
	Hybrid algorithm	1.2074s	20.0007s	20.5068s	36.6042s
	PSO algorithm	5.1177s	21.6275s	19.5296s	48.1029s

Fig. 8: Timing results for the Michalewicz's function

VII. Observations

From the gathered data, the following observations can be made:

A. *Binary hillclimber*

The algorithm provides the best overall timings. There are very few instances where the algorithm is beaten by the floating-point hillclimber and this can be attributed to random chance or wrong parameters being used. Overall, timings were below 9ms.

As for the results, the algorithm performed poorly one of the functions. On the other functions, the results were better, but still not as good as the other algorithms.

The algorithm works best in the following conditions:

- The system has very very few resources (for example on embedded devices with CPU power in few MHz and memory in Kilobytes)
- The system needs to perform optimisations in real-time (for example in games, where each nanosecond matters)
- The function to optimise is ideally unimodal, with few local optimas (for example, Michalewicz's function)

B. *Floating-point hillclimber*

The algorithm tends to be slower than the binary hillclimber and also depends on floating-point hardware acceleration, which may not be present on all systems. When floating-point hardware acceleration is present, the algorithm is really fast, in general finishing finding solutions in matter of 19ms.

As for the results, the algorithm performed performed better on the Rosenbrock's function, and overall the worst results for each function were better than the binary hillclimber.

The algorithm works best in the following conditions:

- The system has a floating-point hardware accelerator (for example on embedded devices such as Raspberry PIs, ESP32s)
- The system needs to perform optimisations in real-time (for example in games, where each nanosecond matters)
- The function to optimise is ideally unimodal, with few local optimas (for example, Michalewicz's function)

C. *Binary genetic algorithm*

Since the algorithm is overall on the large size and has to perform many steps per generation, the timings of this algorithm were on the worse side. A saving point for the algorithm is the fact that it uses binary operations, thus improving performance a bit.

The timings were:

- under 3.02 seconds, for 2 dimensions of each function
- under 20 seconds, for 30 dimensions of each function
- under 48 seconds, for 100 dimensions of each function

The results were better than the previous two algorithms on both the best and worst cases, excepting Rosenbrock's function.

The algorithm works best in the following conditions:

- The system has lots of compute power available (for example on a modern laptop, a desktop, a server)
- The system does not need to perform optimisations in real-time (for example when performing genetic research on proteins)
- The function to optimise is multimodal (for example, Rastrigin's)

D. *Hybrid algorithm*

The algorithm is rather complex. This is generally reflected in its timings, usually being few miliseconds or seconds over the timings of the plain binary genetic algorithm with few exceptions.

The timings were:

- under 3.04 seconds, for 2 dimensions of each function
- under 22 seconds, for 30 dimensions of each function
- under 32 seconds, for 100 dimensions of each function

As for the results, they were rarely on the better side.

The algorithm works best in the following conditions:

- The system has moderate compute power available (for example on a modern laptop, a desktop, or higher)
- The system does not need to perform optimisations in real-time (for example when performing genetic research on proteins)
- The function to optimise is multimodal (for example, Rastrigin's)

E. *Particle swarm optimisation*

The timings for the algorithm depends a lot on the given function. Overall, they were higher than the ones of the binary genetic algorithm, and even the ones for the hybrid algorithm.

The timings were:

- under 3.6 seconds, for 2 dimensions of each function
- under 25 seconds, for 30 dimensions of each function
- under 48 seconds, for 100 dimensions of each function

Regardless of the rather poor timings, the algorithm provided explored the most of the solution space, sometimes finding better solutions than the binary genetic algorithm and the hybrid algorithm.

The algorithm works best in the following conditions:

- The system has lots of compute power available (for example on a desktop, a server, or higher)
- The system does not need to perform optimisations in real-time (for example when performing genetic research on proteins)
- The function to optimise is multimodal (for example, Rastrigin's)

VIII. Conclusions

Each of the presented algorithm has one or more tradeoffs. Some work best on systems with few resources and/or realtime systems, some provide great results but with a huge computational cost. The choice of what algorithm should be used remains to the latitude of the programmer, given the constraints of each system and the desired outcome. There exists no best algorithm for a given problem.

References

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

[2] https://profs.info.uaic.ro/~eugennc/teaching/nim/ (2023)

[3] 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

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

[5] Hartmut Pohlheim, The Genetic and Evolutionary Algorithm Toolbox (2006), http://www.geatbx.com/docu/fcnindex-01.html#P129_5426

[6] Hartmut Pohlheim, The Genetic and Evolutionary Algorithm Toolbox (2006), http://www.geatbx.com/docu/fcnindex-01.html#P140_6155
[7] Hartmut Pohlheim, The Genetic and Evolutionary Algorithm Toolbox (2006), http://www.geatbx.com/docu/fcnindex-01.html#P160_7291
[8] Hartmut Pohlheim, The Genetic and Evolutionary Algorithm Toolbox (2006), http://www.geatbx.com/docu/fcnindex-01.html#P204_10395
[9] https://en.wikipedia.org/wiki/Hill_climbing (2023)
[10] https://en.wikipedia.org/wiki/Genetic_algorithm (2023)
[11] https://en.wikipedia.org/wiki/Particle_swarm_optimization (2023)
[12] https://www.marksmannet.com/RobertMarks/Classes/ENGR5358/Papers/pso_bySHI.pdf (2004)
[13] https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization/ (2023)
[14] https://www.youtube.com/watch?v=JhgDMAm-imI (2018)