Research and testing of optimization algorithms

Radiela Yorgova 19017297

1 INTRODUCTION

This report consists of two parts – research and experimentation. The research part requires a choice of optimization algorithm and its comparison to the Genetic Algorithm(GA). All literature sources are peer-reviewed materials, and the chosen optimization technique is the Firefly Algorithm(FA). The goal of the experimentation is to find the optimal parameters and operators for an implemented Genetic Algorithm. These observations got made from the data collected from a thorough parameter sweep. The parameters' testing illustrates how the change of the parameters and operators affects GA's performance.

2 BACKGROUND RESEARCH

Nature is the best inventor around and the inspiration for many science and engineering accomplishments. In the field of Computer Science, nature-inspired algorithms prevail in solving complex optimization problems (Fang et al., 2015). The Genetic Algorithm(GA) is a nature-inspired evolutionary metaheuristic. Inspired by the Theory of Evolution and the concepts of natural selection, inheritance, and mutation. (Leardi, 2009). Another nature-inspired metaheuristic is the Firefly Algorithm(FA). Inspired by the bioluminescent way of communication between fireflies, it falls under the class of Swarm Intelligence (Wang et al., 2017). *Source code for these algorithms is provided in the appendix*.

FA and GA are both population-based stochastic algorithms –they start with a randomly generated population, where every individual is a candidate solution. All solutions get explored to find the best one and exploit it. Every solution gets evaluated by a fitness function, determining its utility. The FA fitness function is the brightness of the fireflies. They are unisex insects, and their attraction gets based on their brightness.
Fireflies with dimmer lights move towards brighter individuals or randomly if none are found (Ritthipakdee, 2014). An optimal solution is reached when the population swarms around one or more local maximums. The population can automatically subdivide into subgroups, finding the local and global optima simultaneously (Attia, 2017). Whereas in GA, finding the global optima isn't a simultaneous manner. The initial population doesn't find it, only its offsprings do.

Both methods avoid getting trapped in local maxima by introducing random solutions. GA uses crossover and mutation operators to keep the population diverse. FA uses a randomization parameter (α) to determine the individuals' movement when there is no brighter firefly around. The other parameters of FA are the absorption coefficient (γ) –which influences the speed of finding an optimal solution (Mo, 2013), the attractiveness based on the light intensity (I) of the fireflies, and the size of the population (N). FA has fewer parameters than GA and their tunning is easier, which makes FA less complex and faster to compute.

Recent advances of FA demonstrate that the algorithm can be optimized even further. One example is the Chaotic Improved Firefly Algorithm(CFA), where the parameters of the standard FA are replaced with chaotic systems. A study of the application of twelve chaotic maps (Gandomi et al., 2013) shows that deterministic chaotic signals improve the global search of the algorithm. The study concludes that the application of the Gaussian map outperforms the standard FA and the other chaotic maps utilized. Recent applications of FA include a modified FA used for tracking expanding oil spills and the area (Banerjee et al., 2018), FA for the optimization of a computer-aided processing planning system (Zubair and Mansor, 2019), and the applications of FA in image processing (Dey et al., 2020) – *see appendix for sources.*

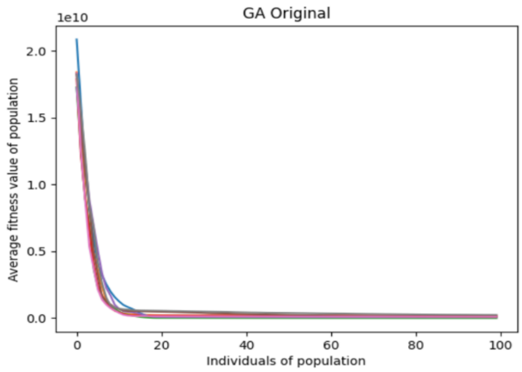
3 EXPERIMENTATION

The genetic algorithm can be very efficient for some problems and quite poor for others. The algorithm needs tunning of its parameters to get the best performance. The experimentation part consists of a parameter sweep – finding the optimal parameters for minimization with real numbers and testing the performance of the algorithm with different operators.

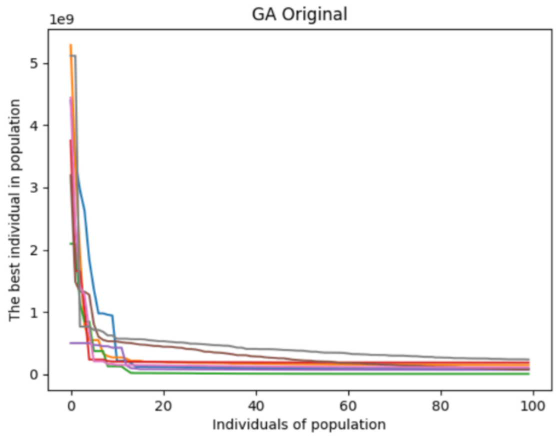
The GA for testing utilizes elitism –the best solution from the initial population replaces the worst solution in the offspring population. The operators are tournament selection between two individuals, one-point crossover, random resetting mutation, and Rosenbrock fitness function.

The parameters for optimization are population size (P), number of genes (N), iterations, mutation rate, and mutation step. The range for generating random genes in initialization is (-100; 100). All values concluded are the average of 10 runs and over.

Test 1: Parameter sweep
The values of the parameters for the first test are : N = 10; P = 50; iterations = 100; mutation rate = 0.06; mutation step = 1. The graphs below visualize the results for the average fitness and the fittest individuals of the population.



Graph 1: Average fitness of the population



Graph 2 : The fittest individuals

The fittest individual value in this test is not an optimal solution but overall, the algorithm shows minimization.

Case 1:
The first parameter to test is the number of genes (N). *Figure 1* shows that even the slightest change in the number of genes can drastically change the solution.

N = 10	N = 20
98 372 007.70099247	828 840 339.2293178

Figure 1. Comparison of N = 5; 10; 20

The fewer genes there are per chromosome, the smaller its fitness value would be. That is simply because the algorithm works with real numbers and the fitness of each chromosome is based on the summation of its genes.

Case 2:

The next parameters for testing are the number of iterations and the size of the population. Better solutions are observed when increasing the population size. The more individuals (candidate solutions) there are, the higher is the chance of finding more optimal or near-optimal solutions. Better solutions are also observed when increasing the number of iterations. In every iteration, all solutions get optimized by the GA operators, and it's beneficial to increase the iterations. Although, too many iterations would make the execution time slow. The results from *Figure 2* indicate that the optimal number of iterations is within the range(200 – 300).

Iterations/Population	50	100	200	300	500	1000
25	104 365 257, 49932	24 053 648, 91306	20 782 238, 93552	4 683 288, 79556	2 383 431, 40633	507 272, 06928
50	155 136 568, 29309	40 271 583, 83721	4 912 715, 62817	1 313 067, 3392	300 049, 39797	30 693, 73479
100	133 925 139, 98957	21 724 251, 60166	4 068 480, 92019	939 808, 04618	162 589, 18985	24 655, 02718
200	135 389 019, 63157	46 397 596, 98533	3 031 450, 69698	1 120 580, 47995	152 779, 34412	18 532, 60341
300	75 392 952, 40943	31 405 154, 28756	3 882 874, 4044	1 217 233, 48544	99 473, 60328	10 709, 39968
500	63 985 830, 91325	20 404 966, 46789	3 525 592, 69609	823 357, 62588	65 393, 0192	13 699, 16692

Figure 2: Observing iteration and population size parameters

The population size of 1 000 provides the best solutions so far and the testing will continue with this value for the parameter (P).

Case 3:

Testing the influence of the mutation rate parameter over a population of 1 000 and mutation step of [1.0]. There is a preferred range for mutations rates (0.001 – 0.9). Low mutation rates limit the adaptation of organisms to changing environments. However, too high mutation rates can compromise the fitness of a population when it's in a stable environment.

In both cases from *Figure 3*, the mutation rate of [0.05] gives the best results. This test case also concludes that working with 300 iterations produces significantly better solutions.

Mutation Rate	Iterations = 200	Iterations = 300
0.02	21 662, 989934	16 825, 028768
0.03	11 744, 429093	7 492, 057109
0.04	9 358, 769241	6 105, 320656
0.05	7 762, 466589	3 214, 316088
0.06	18 532, 603415	10 709, 399687
0.09	5 144, 4099	19 521, 669714

Figure 3: Observing the mutation rate parameter

Case 4: Testing of the mutation step parameter

In this GA, mutations occur randomly on separate genes. A random probability parameter value gets generated for the selected gene. If that value falls under the mutation rate range, the gene gets altered by the mutation step. The value of the gene will be increased or decreased, depending on its proximity to the optimization goal. *Figure 4* shows the results from testing the mutation step parameter – its optimal value is [6].

Mutation Step	Mutation Rate = 0.05
1	3 214, 315088
4	923, 311728
5	76, 966712
6	5, 087194
7	6, 412067
8	7, 580055
10	14, 785006

Figure 4: Testing of the mutation step parameter

Further testing was done on the population size. *Figure 5* shows that the best solution of the original GA is [3, 5073] with a population of 5 000.

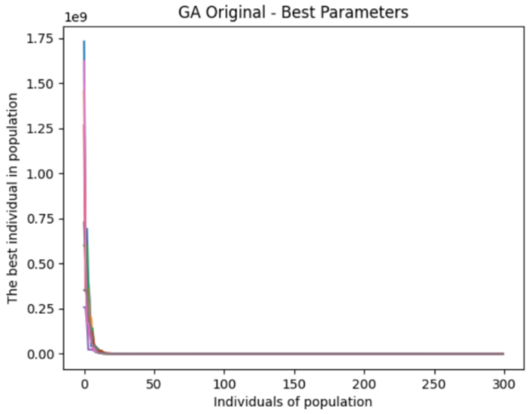
Population	Solutions
2 000	5, 55216
2 500	4, 79479
3 000	4, 09488
3 500	3, 61366
5 000	3, 50733
5 500	3, 52988

Figure 5: Additional testing of the population size

Test 2: Comparison of different types of operators

Case 1: Tournament vs Roulette wheel selection

From the previous testing, the optimal value found with tournament selection is [3, 5073] – *Graph 3.*



Graph 3

In GAs, the selection process is usually accomplished by tournament – where the fitter out of two or more individuals gets selected for mating, or a Roulette Wheel system –where the fitter individuals have a better chance to be picked. The selection process forms a mating pool from copies of the fittest individuals and narrows down the search space.
Testing of the Roulette wheel selection concludes the optimal parameters for the operator, highlighted in green in *Figures 6, 7, 8, and 9.*

Population size	Solution
1 000	55 596 614, 0538
500	147 240, 03356
300	18 593, 68071
200	10 644, 82338
100	2 487 212, 93449
50	13 633 414, 661538

Figure 6: Testing of population size parameter

Iterations	Solution
300	55 596 614, 0538
200	1 208 170, 3228
150	2 449 126, 3115
100	17 182 497, 9875

Figure 7: Testing of iterations parameter

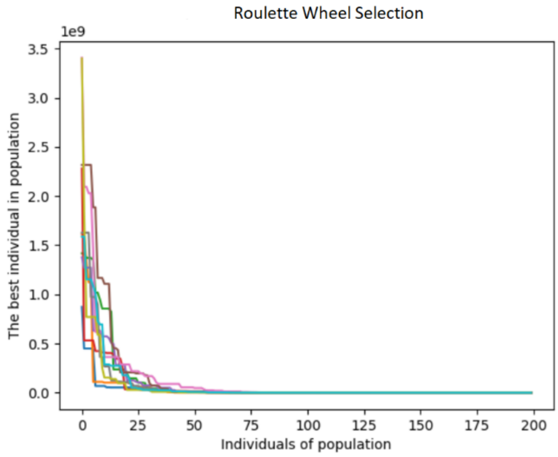
Mutation rate	Solution
0.01	1 273 035.9573
0.02	64 525.1649
0.03	150 538, 4164
0.04	554 059.8772
0.05	698 664, 5366
0.07	1 245 024.9409

Figure 8: Testing of mutation rate parameter

Mutation step	Solution
7	756 601.0711
6	40 968.7676
5	126 977, 4437
4	343 366, 5826

Figure 9: Testing of mutation rate parameter

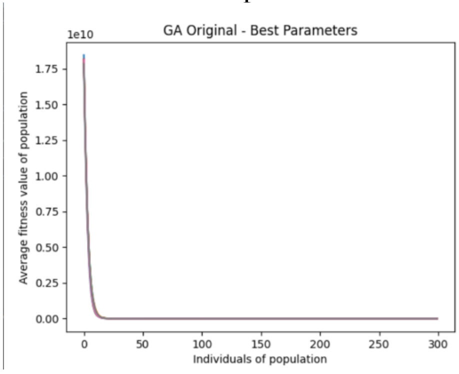
The optimal parameters for GA with Roulette wheel selection are P = 200, Iterations = 200, Mutation rate = 0.02, Mutation step = 6. The best solution found over multiple runs is [9 968, 3212]. Compared with Tournament selection, the roulette wheel produces worse solutions and takes longer to compute. The worst disadvantage of the roulette wheel system is that it clusters the population with nearly identical solutions. It shows better results on smaller populations because larger ones produce a greater number of optimal solutions, and the population gets trapped in local optima earlier.



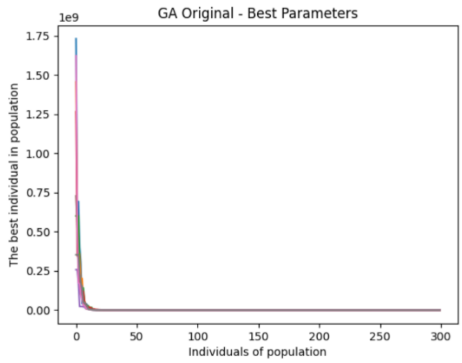
Graph 4: The best solutions with Roulette wheel

Case 2: One-point vs Multipoint vs Arithmetic Crossover

The crossover operator explores the search space and is used to create genetic variations. Just like organisms gain traits and features from their parents, offsprings of GA are a mix of their parents' chromosomes. One-point crossover selects a random point in two chromosomes and switches their genes after that point. *Graphs 5 and 6* visualize the one-point crossover results.



Graph 5: One-point crossover average fitness of the population



Graph 6: One-point crossover fittest individuals

Multipoint crossover selects two random values, used to determine a block of genes to be swapped between the chromosomes. It performs better with smaller populations because it disrupts the parents' genes more and creates more new genotypes (*Figure 10*). Decreasement of the population size reduces the computational time.

Population size	Solution
500	360, 9434
1 000	5, 37787
1 500	4, 3502
2 000	3, 6654
3 000	3.7321
5 000	24.8398

Figure 10: Testing of population size with multipoint crossover

Iterations	Solution
200	5, 3651
300	3, 6654
400	115.9462

Figure 11: Testing of the number of iterations with multipoint crossover

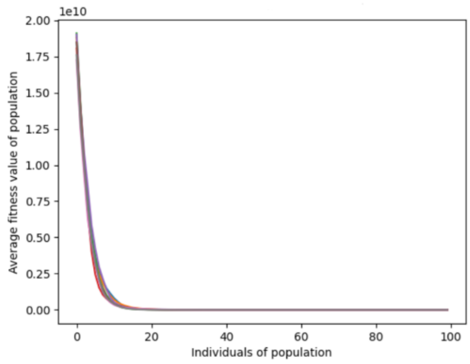
Mutation rate	Solution
0.03	5.0337
0.05	3, 6654
0.07	11.5112

Figure 12: Testing mutation rate with multipoint crossover

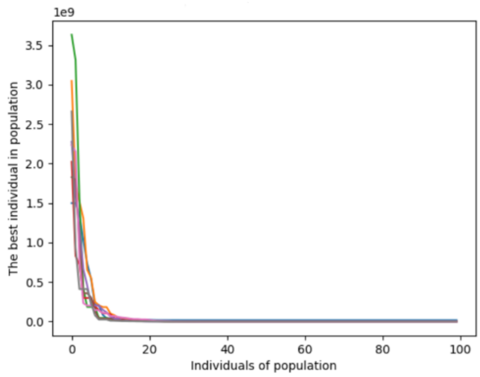
Mutation step	Solution
5	99.3442
6	3, 6654
7	304.3114

Figure 13: Mutation step with multipoint crossover

Figures 11, 12, and 13 show that the other optimal parameters of GA with multipoint crossover are the same as with one-point crossover. The only difference between them is the smaller optimal population size needed for multipoint. *Graphs 6 and 7* visualize the best results with multipoint crossover.



Graph 7: Multipoint crossover – average fitness of the population



Graph 8: Multipoint crossover – fittest individuals

The arithmetic crossover is suitable for real number GAs, as it linearly recombines the parent chromosomes or some percentage of them.

```
Child 1 = a.Parent1 + (1-a).Parent2
Child 2 = a.Parent2 + (1-a).Parent1
```

It introduces a new parameter to the algorithm – arithmetic pointer (a), which is usually within the range (0.0-1.0). *Figure 14* visualizes the best results from testing the arithmetic pointer.

Arithmetic pointer	Solution
0.4	303.0208
0.6	253, 8327
0.5	322.4089

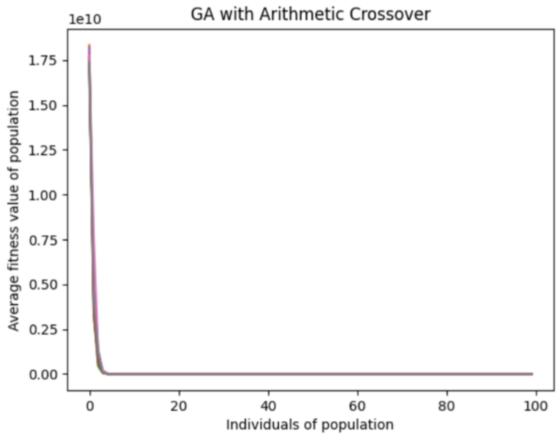
Figure 14: Optimal values for the arithmetic pointer parameter

Although the solutions are further away from the global optima, it uses smaller populations for producing optimal solutions *Figure 15*.

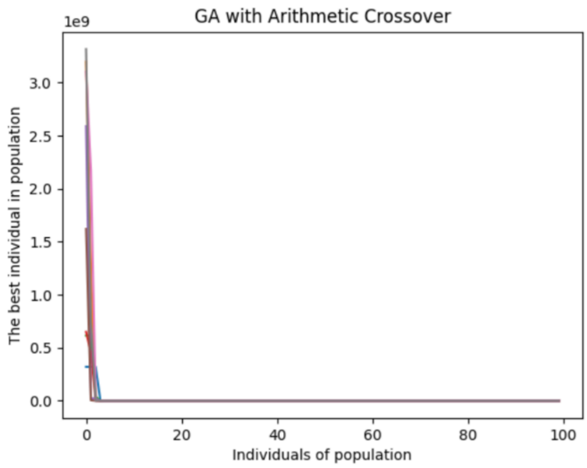
Population size	Solution
1 000	287, 0401
2 000	253, 8327
5 000	272, 3298

Figure 15: Testing population size with arithmetic crossover

The best performance of GA with arithmetic crossover is presented in *Graphs 9 and 10*.



Graph 9: Arithmetic crossover: Average fitness of the population



Graph 10: Arithmetic crossover: Fittest individuals

The Graphs (5 – 10) show a visual comparison between the three types of crossover operators observed. Overall, multipoint crossover prevails because of its low computational time and its high efficiency.

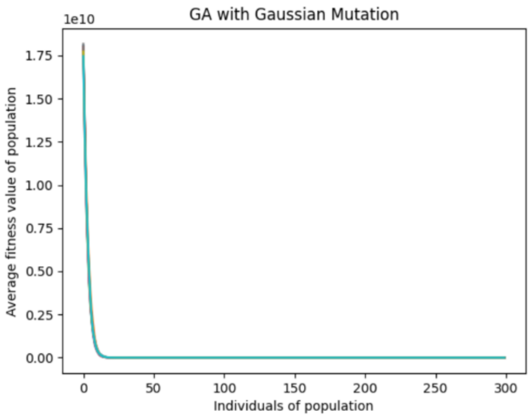
Case 3: Random resetting vs Gaussian mutation

Graphs 5 and 6 visualize the random resetting mutation solutions. The Gaussian mutation operator creates new offspring by applying a random value from a gaussian distribution to every gene. It is more suitable for real number GAs because it glides the generated gene values.

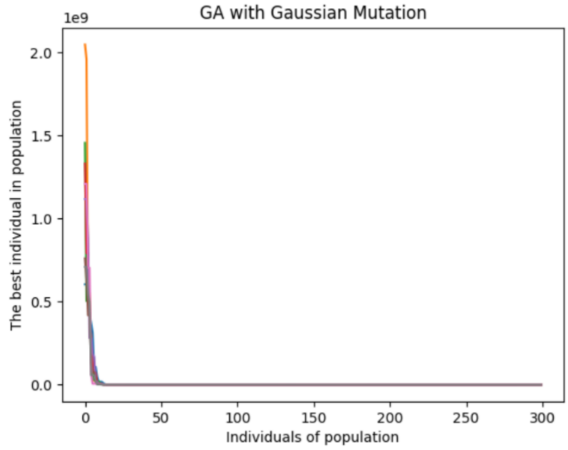
Population size	Solution
500	3.6237
1 000	3.4692
2 000	3.4452
5 000	3.4346

Figure 16: Testing the population size influence on Gaussian mutation

Figure 16 shows that better solutions are produced with the growth of the population size because there are more mutated individuals occurring. *Graphs 11 and 12* observe the results of the Gaussian mutation operator.



Graph 11: Average fitness of the population with Gaussian mutation



Graph 12: Fittest individuals with Gaussian mutation

The optimal mutation rate is shown in *Figure 17.* It produces an overall better solution [3.48]

Mutation rate	Solution
0.06	3, 9094
0.0335	3, 4839
0.03	3, 5088
0.02	3, 7181

Figure 17: Gaussian mutation: mutation rate testing

Gaussian mutation uses a mutation step parameter to determine its distribution range.

Mutation step	Solution
0.5	15, 7701
1	3, 4839
3	3, 5465

Figure 18: Testing of the mutation step parameter in Gaussian mutation

Overall, the gaussian mutation proves more successful and time-efficient, than the random resetting method – it glides the solutions towards the global optima.

Case 4: Rosenbrock vs Ackley fitness functions

All the previous graphs represent solutions of the Rosenbrock objective function.

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

Rosenbrock Fitness function

$$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos 2\pi x_i\right)$$

Ackley Fitness function

The global optima for Ackley function is around [-20].

Population size	Solutions
300	-11.7486
500	-12.1368
1 000	-12.4868
2 000	-12.5421

Figure 19: Ackley function: population size

Iterations	Solutions
50	-11.6878
100	-12.1937
300	-12.2017
500	-12.2941
1 000	-12.1893

Figure 20: Ackley function: iterations

Mutation rate	Solution
0.03	-12.0399
0.05	-12.2941
0.07	-12.3139
0.075	-12.3383
0.08	-12.3184
0.085	-12.0714
0.09	-12.1292

Figure 21: Ackley function: Mutation rate

Another popular function is the Ackley fitness function. For the test of Ackley, the initial population will be within the range (-32; 32). For fair testing, the same operators will be used, as in the testing of the Rosenbrock function (optimal parameter values).

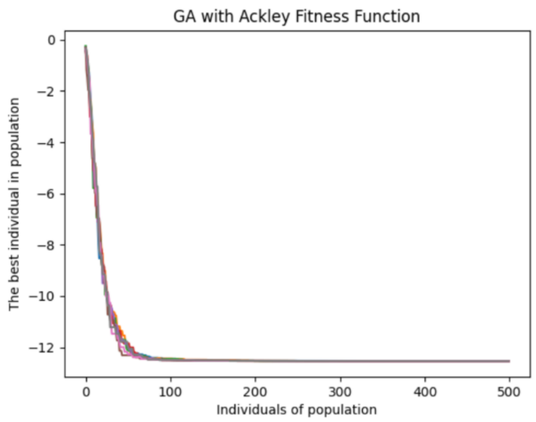
Mutation step	Solution
1	-12.0549
3	-12.2621
6	-12.3383
9	-12.1536

Figure 22: Ackley function: Mutation step

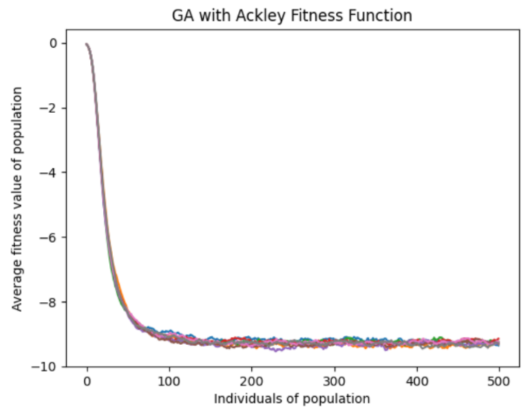
Best solution	-12.5389
Population size	5 000
iterations	500
Mutation rate	0.075
Mutation step	6

Figure 23: Ackley: optimal parameters

The testing from *Figure 19* concludes that Ackley function thrives on larger populations. *Figures 20, 21*, 22, and 23 show the optimal values for the other parameters. Overall, the Ackley function is more difficult to implement and has a longer runtime, as it prefers larger populations and number of iterations. *Graphs 13 and 14* visualize the best solutions brought.



Graph 13: Fittest individuals with Ackley fitness function



Graph 14: Average fitness of the population with Ackley function

4 CONCLUSIONS

The research part concludes that the Firefly optimization algorithm outperforms the GA. Newer methods, such as adding chaos or clustering have improved the algorithm even further. Even so, the Genetic algorithm still holds its importance in real-world applications.

The experimentation part concludes that operators and parameters have different influences on GAs. Larger populations tend to produce better solutions, as they have a higher chance of containing more optimal solutions – more mutated individuals. The Tournament selection thrives over Roulette wheel systems with its fast computational time and efficiency. Multipoint crossover significantly reduces the population size necessary and therefore, the computational time, which gives it the advantage over One-point and Arithmetic crossover. Gaussian mutation works better than random resetting, as it glides the solutions in initialization which enhances the performance of the algorithm. The Rosenbrock fitness function is easier to implement and has a faster run time than the Ackley function.

For further improvement of this report, recent advances of GAs could also be compared with FA. There is literature available on GAs with chaos and clustering. Hybrid algorithms with GA and FA properties are also worth looking at.

REFERENCES

- Fang, W., Li, X., Zhang, M. and Hu, M. (2015) Nature-Inspired Algorithms for Real-World Optimization Problems. *Journal of Applied Mathematics*. pp. 1-2. doi:10.1155/2015/359203

- Leardi, R. (2009) Genetic Algorithms. *Comprehensive Chemometrics* [online]. pp. 631–653. doi:10.1016/b978-044452701-1.00039-9

- Wang, H., Wang, W., Zhou, X., Sun, H., Zhao, J, Yu, X., and Cui, Z., (2017) Firefly algorithm with neighborhood attraction. *Information Sciences* [online]. 382-383 pp. 374–387.

- Ritthipakdee, A., Thammano, A., Premasathian, N., and Uyyanonvara, B., (2014). An improved firefly algorithm for optimization problems. *In The 5th International Symposium on Advanced Control of Industrial Processes* pp. 159-164.

- Johari, N.F., Zain, A.M., Noorfa, M.H. and Udin, A. (2013) Firefly Algorithm for Optimization Problem. *Applied Mechanics and Materials*. 421 pp. 512–517. doi:10.4028/www.scientific.net/amm.421.512

- Attia, K.A.M., Nassar, M.W.I., El-Zeiny, M.B. and Serag, A. (2017) Firefly algorithm versus genetic algorithm as powerful variable selection tools and their effect on different multivariate calibration models in spectroscopy: A comparative study. *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy* [online]. 170 pp. 117–123.

- Mo, Y., Ma, Y. and Zheng, Q. (2013) *Optimal Choice of Parameters for Firefly Algorithm*.

- Gandomi, A.H., Yang, X.-S. ., Talatahari, S. and Alavi, A.H. (2013) Firefly algorithm with chaos. *Communications in Nonlinear Science and Numerical Simulation*. 18 (1), pp. 89–98. doi:10.1016/j.cnsns.2012.06.009

Appendix (118 words)

```
                Firefly Algorithm


Generate initial population of fireflies Pi(i=1,2,...,N)
Fitness function f(P), P = (x1,...,xn)
Light intensity I at Pi is determined by f(Pi)
Define light absorption coefficent (y)

While (t < MaxGeneration)
  for i = 1:n all n fireflies
    for j = 1:n all n fireflies
      if (Ij > Ii), Move firefly i towards j in d-dimension; end if
        Attractiveness varies with distance r via exp[-yr]
        Evaluate new solutions and update light intensity
    end for j
  end for i

  Rank the fireflies and find the current best
  end while

Process results and visualization
```

*Source Code for Firefly Algorithm (*Gandomi, 2013)

```
Genetic Algorithm

Procedure: Tournament selection

  While population size < max pop_size, do:

    Generate pop_size random number r
    Calculate individual fitness, total fitness, and average total fitness
    Select 2 individuals P-times (P = pop_size)

    if ind1 > ind2:
      Select the first chromosome, otherwise the second one
    End If
  End While
  Return chromosomes with better higher value

End Procedure
```

Source Code for Genetic Algorithm with Tournament Selection

```
Genetic Algorithm

Procedure: Roulette wheel selection

  While population size < max pop_size, do:

    Generate pop_size random number r
    Calculate individual fitness, total fitness, and proportional fitness (Sum)

    Spin the wheel pop_size times
    If Sum < r, then:
      Select the first chromosome, otherwise select the jth one

    End If
  End While
  Return chromosomes with better higher value

End Procedure
```

Source Code for Genetic Algorithm with Roulette wheel selection

Materials on recent applications of FA:

- Banerjee, A., Ghosh, D. and Das, S., (2018). Modified firefly algorithm for area estimation and tracking of fast expanding oil spills. *Applied Soft Computing*, *73*, pp.829-847.

- Zubair, A.F. and Mansor, M.S.A., (2019). Embedding firefly algorithm in optimization of CAPP turning machining parameters for cutting tool selections. *Computers & Industrial Engineering*, *135*, pp.317-325.

- Dey, N., Chaki, J., Moraru, L., Fong, S. and Yang, X.S., (2020). Firefly algorithm and its variants in digital image processing: A comprehensive review. *Applications of firefly algorithm and its variants*, pp.1-28.