EA PRACTICAL ASSIGNMENT

Evan Meltz
s3817911
s3817911@vuw.leidenuniv.nl

Giulia Rivetti
s4026543
s4026543@vuw.leidenuniv.nl

December 8, 2023

ABSTRACT

Genetic Algorithms (GA) and Evolution Strategies (ES) are evolutionary algorithms inspired by the process of biological evolution. They can be used to find approximate solutions to optimization problems, which is the focus of this assignment. In particular, our goal was to implement both a GA and ES solving two maximization problems from PBO: the *Ising Ring* and the *Low Autocorrelational Binary Sequences* problems. Then, we have evaluated the performance of our algorithms with the usage of the IOH Analyzer, which we used to have a visual assessment of our algorithms' performance.

1 Introduction

Evolutionary algorithms can be divided into two branches: Genetic Algorithms (GA) and Evolution Strategies (GA), both based on the concepts of natural selection and biological evolution. These two optimization algorithms have many applications, one of these being the focus of this assignment: solving optimization problems. In particular, we focused on two problems from the PBO library, the Ising Ring problem and the *Low Autocorrelational Binary Sequences (LABS)* problem. The former is concerned with finding binary sequences such that the reciprocal of the sequence's autocorrelation is maximized; the latter focuses on finding the spin configuration that minimizes the total energy, where the spins represent the magnetic moments of individual particles in a material. For this assignment we have used the one-dimensional ring variant of the Ising model. Our main goal was to implement a GA and a ES able to solve these problems by maximizing their corresponding fitness function. Then, we have assessed the performance of our algorithms through some data analysis and plots obtained with the IOH Analyzer framework. We have then compared the two different implementations that we have used to solve both F18 and F19 problems, namely GA and ES.
The report is organized as follows: section 2 explains the 4 different implementations that we have employed: GA solving F18 and F19 problems, and ES solving F18 and F19 problems; section 3 reports the data and plots that we have obtained using the IOH Analyzer as well as their analysis; finally, in section 4 we have reported our design choices and a conclusion about the results that we have obtained.

2 Algorithms

The following subsections describe in detail the implementations that we have used for both the Genetic Algorithm and the Evolution Strategy solving the Ising Ring and the LABS problems. The algorithms of the GA solving F18 and F19 problems are very similar in structure; their main difference lies in the usage of different genetic operators and different parameters for the two problems. In this way, by tweaking these parameters based on the problem at hand, we were able to maximize the fitness function, which is indeed the goal of both problems. The same holds for the ES that we have implemented; however one of the main differences between the GA and the ES, was the usage of an encoding technique for the ES. Indeed both the Ising Ring and the LABS problems require boolean values for the evaluation of each individual. The GA also expects individuals represented as bit-strings, whereas the ES works with real values, and therefore an encoding was needed to convert the real values representing each individuals to binary values. More specifically, the encoding that we have employed simply converts positive real values to 1 and negative real values to 0.

2.1 Genetic Algorithm solving F19

In this part we focus on analyzing the implementation of the GA solving the Ising Ring problem (F19). The pseudo-code of the algorithm is showed in Algorthm 1, which summarizes the different parameters and operators that we have used, as well as the general structure and functions of the algorithm.

Algorithm 1: Pseudo-code of GA solving solving problem F19

```
Input       : Population size = 7
              Crossover probability = 0.9
              Mutation rate = 0.5
              Number of tournaments = 20
Termination : The algorithm terminates when the number of evaluations has reached the budget limit (5000)
Initialization;
Evaluation;
while $num\_evaluations < budget$ do
    Tournament Selection;
    for $i = 1$ to $8$ do
        Uniform Crossover;
        Inversion Mutation;
    end
    Evaluation;
end
```

First, the algorithm initializes our population with 8 individuals, each one being a bit-string of length 50, and it also evaluates this initial population based on the fitness value of each individual.
Then, the GA enters into a loop, which terminates when the number of evaluations have reached a fixed budget, which in our case is 5000. After that, the algorithm performs the selection process, where, mimicking the survival-of-the-fittest principle, the individuals with higher fitness values are selected for reproduction. For solving the F19 problem, we have tried different kinds of selection (rank-based, tournament, roulette-wheel) and in the end we have chosen to use the tournament selection, since it was the one yielding to the highest average best fitness value for 20 independent runs. In particular, this kind of selection performs a certain number of tournament, 22 in our case. For each tournament a small number of individual is selected from the population and the one with the highest fitness value is then selected for reproduction.
At this point, the algorithm iteratively applies uniform crossover on two individuals taken from the offspring (the new population obtained after the selection mechanism). With this type of crossover, a bit string (the mask) with the same length as the two parents is created. Then, the function checks each bit of the mask: if it is 0, the child inherits the bit in that position from the first parent, otherwise it inherits it from the second parent.
In order to maintain genetic diversity within the population and thus to prevent the algorithm to converge too quickly to a non-optimal solution, mutation is then applied. In particular, the algorithm uses inverse mutation, which selects two points on an individual and then inverts the order of the bits of that individual between those two points, with a frequency determined by the mutation rate, which is 0.5 in our case.
Finally, the new population is evaluated again, meaning that the the fitness function is computed for each individual. At this point, a new evaluation starts and the same process is repeated again.

2.2 Genetic Algorithm for F18

This part focuses still on analyzing the implementation of the GA but now solving the Low Autocorrelation Binary Sequences problem(F19), also known as LABS for short. See below in Algorithm 2, the pseudo-code of the algorithm.

As you can see, the algorithm is very similar to that of the one used for the f18 problem in terms of the structure and parameter settings but uses different functions. This is due to the nature of the F18 problem which differs from that of the F19 problem. As a sidenote, this adaptability showcases the versatility of the genetic algorithm framework, allowing it to be applied to diverse problem domains by adjusting the underlying functions while maintaining a consistent structural framework. As previously mentioned, the algorithm acts the same for the initialization and evaluation. Furthermore, in the outer while loop, when the algorithm performs the selection process, tournament selection is again used, as it proved to be the highest yielding in terms of average best fitness across both problems.

Algorithm 2: Pseudo-code of GA solving problem F18

```
Input       : Population size = 7
              Crossover probability = 0.9
              Mutation rate = 0.5
              Number of tournaments = 20
              n=25
Termination : The algorithm terminates when the number of evaluations has reached the budget limit (5000)
1  Initialization;
2  Evaluation;
3  while $num\_evaluations < budget$ do
4      Tournament Selection;
5      for $i = 1$ to $8$ do
6          N-Point Crossover;
7          Random Resetting Mutation;
8      end
9      Evaluation;
10 end
```

At this point, within the inner for loop, which iterates over the size of the population, the algorithm now applies n-point crossover, a genetic recombination technique. In this function, 'n' (in our case 25) random crossover points are selected along the binary sequences of two parents, and the subsequences between these points are exchanged between the parents. Next mutation random resetting mutation is applied which involves randomly selecting a gene within the binary sequence, and then resetting the value to a new random binary value. The algorithm then ends the same way as before with the new population being evaluated again.

2.3 Evolution Strategy for F19

The pseudo-code for the Evolution Strategy solving the *Ising Ring* problem is implemented in algorithm 3.
The first step of the algorithm is the initialization of the population, which we have done by taking a number of individuals equal to $\mu = 22$; each individual, of length 50, was initialized with a float number between the specified lower and upper bounds. Then, we have performed an evaluation, in order to evaluate the fitness of each individual in the population by applying the objective function to each candidate solution. After these first two steps, the algorithm enters into a loop which terminates when the number of evaluations has reached the given budget of 5000. In here we have applied the so called discrete recombination, which randomly chooses two parents from the parent population and for each gene it chooses the value from one of the parents at random, so that the resulting combination of chosen values forms the new offspring.
The algorithm then proceeds by performing the individual σ-mutation on the offspring: each individual has an associated mutation step size σ, which is mutated using a random factor drawn from a normal distribution.
After this, we have again evaluated the fitness function for each individual and then applied a tournament selection, so the same type of selection used for the Genetic Algorithm solving problem F19.

2.4 Evolution strategy for F18

In this part, the Evolution Strategy was used to again solve the LABS problem. The pseudo-code for the Evolution Strategy is shown below in algorithm 4.

The initialization of the population, is done in the same way as with the f19, but the parameters have been changed to those which result in a higher best average accuracy. Furthermore the same evaluation method remains applicable for the labs problem.
Within the loop, the same function of discrete recombination is used as after experiments with multiple ones, this remained the best across both problems.
The algorithm differs with the mutation function as it now uses one σ-mutation which involves modifying the standard deviation (σ) of a Gaussian distribution which is then added to the current value of a selected gene.
The second evaluation remains present as well as the use of the tournament selection function as with the recombination it was found to still be the best after experimenting with several others.

Algorithm 3: Pseudo-code of ES solving problem F19

```
Input       : $\mu = 22$
              $\lambda = 100$
              $\tau = 0.316$
              $\tau' = 0.1$
              Lower bound = -3.5
              Upper bound = 3.5
              Number of tournaments = 5
Termination : The algorithm terminates when the number of evaluations has reached the budget limit (5000)
1 Initialization;
2 Evaluation;
3 while $num\_evaluations < budget$ do
4     Discrete recombination;
5     Individual $\sigma$-Mutation;
6     Evaluation;
7     Tournament Selection;
8 end
```

Algorithm 4: Pseudo-code of ES solving problem F18

```
Input       : $\mu = 20$
              $\lambda = 20$
              $\tau = 0.316$
              Lower bound = -5.0
              Upper bound = 5.0
              Number of tournaments = 5
Termination : The algorithm terminates when the number of evaluations has reached the budget limit (5000)
1 Initialization;
2 Evaluation;
3 while $num\_evaluations < budget$ do
4     Discrete recombination;
5     One $\sigma$-Mutation ;
6     Evaluation;
7     Tournament Selection;
8 end
```

3 Experimental Results

For the performance evaluation of our algorithms, we have used the *IOH Analyzer*, which enabled us to obtain useful plots and data from our algorithms. In this section, we report some indicators in order to assess the performance of both the Genetic Algorithm and the Evolution Strategy.
Table 1 reports two values for the Genetic Algorithm and Evolution Strategy solving both F18 and F19 problems: the average fitness refers to the mean best fitness value obtained for 20 independent runs of the algorithm, whereas the AUC represents the area under the ECDF curve.

Algorithm	Average fitness	AUC
GA for F19	48.8	0.98985
GA for F18	4.18	0.99670
ES for F19	44.8	0.94476
ES for F18	3.79	0.99675

Table 1: Average fitness value for 20 independent runs and AUC value for GA and ES solving both F18 and F19 problems

3.1 Results for solving F19

As already mentioned, we used the IOH Analyzer to obtain some plots in order to better understand and analyze the performance of our algorithms. Figure 2 shows the ECDF curve for the GA and ES solving the Ising Ring problem. This graph represents the empirical cumulative distribution function of the running time and in particular it shows the proportion of the runs that have found a solution of at least the required target value within the given budgets. By observing the plot we can see that for both algorithms' curves have an increasing behavior as the number of the function evaluations increases, which is an indicator of the fact that the two algorithms are working properly: as more evaluations are done, the proportion of runs that have found a solution continues to increasing, until it converges to 1, meaning that all of them have found a solution. By comparing the ECDF for the two algorithms, we can observe that even if the GA has a slower start, then it has a faster improvement and converges faster than the EA. This can be attributed to the fact that since the EA works with real-values but the problems that we have considered (F18 and F19) accepts binary strings, we had to encode the real values defining each individual into either a zero or a one. This however has the consequence of loosing precision in the representation of the individuals, impacting negatively the performance and causing a slower converge for the ECDF curve.

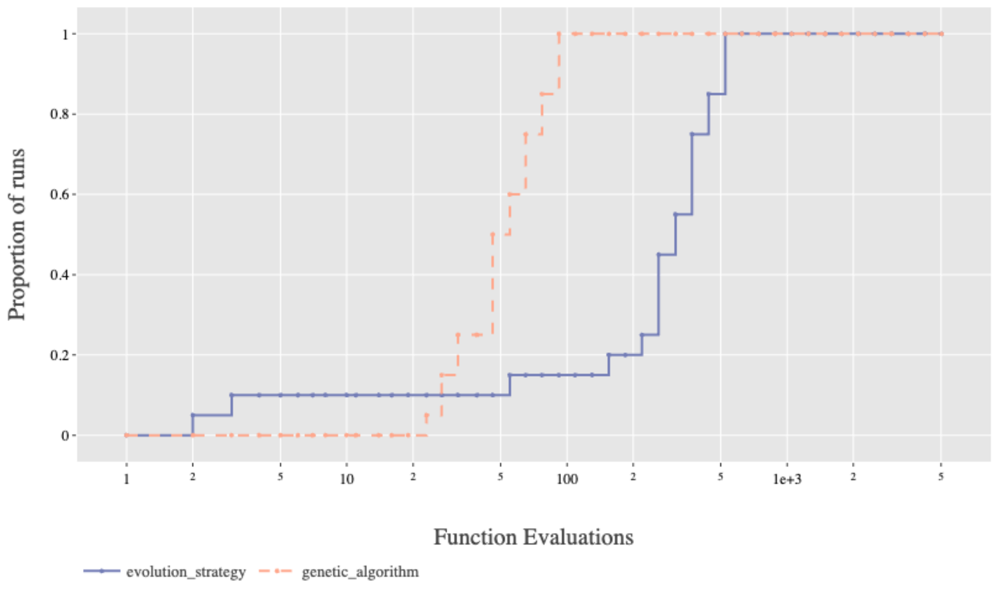

Figure 1: ECDF plot for the GA and ES solving the ISING Ring (F19) problem

Figure 2 shows the expected running time curve of GA and ES for solving the Ising Ring problem. In particular, this plot shows us the number of function evaluations required by our two algorithms algorithm to achieve the optimal fitness value for the F19 problem, which in our case is 50. Observing the two graphs, we can see that the GA needs less evaluations in order to reach the optimal fitness function value, more specifically it needs about 8600 evaluations out of 10000 (since our budget is 5000 and we have executed 20 independent runs). Instead, the EA needs more than 98400 evaluations to reach the optimum fitness value. In the graph the difference between the two curves may seem less significant than what numbers indicates, but this is only because we have scaled the y-axis to be in a log_{10} scale in order to have a better visualization of the behavior as a function of the best so far fitness value. At first it seems that the GA requires more evaluations to get to the same fitness value as the ES, but then after the best fitness value found so far reaches 32, the ES starts to need a significantly higher number of evaluations with respect to the GA. This can be again attributed to the fact that the GA is a more suitable choice for solving problem F19, since this kind of problem works with Boolean values and GA is characterized by individuals represented as binary strings. The usage of an ES in this case is not optimal because of the encoding step that needs to be done to evaluate the population, step that, as we already said before, can lead to a loss of precision. Moreover the fact that the ES requires more function evaluations can also be attributed to the larger populations size with respect to the GA, since the solution space in this case is larger and therefore the ES requires more evaluations for its exploration.

Figure 3 shows the empirical cumulative distribution function (ECDF) curve for the GA and ES now solving the (F18) LABS problem. This plot shows that, much like the F19 problem, both algorithms' curves increase with the function

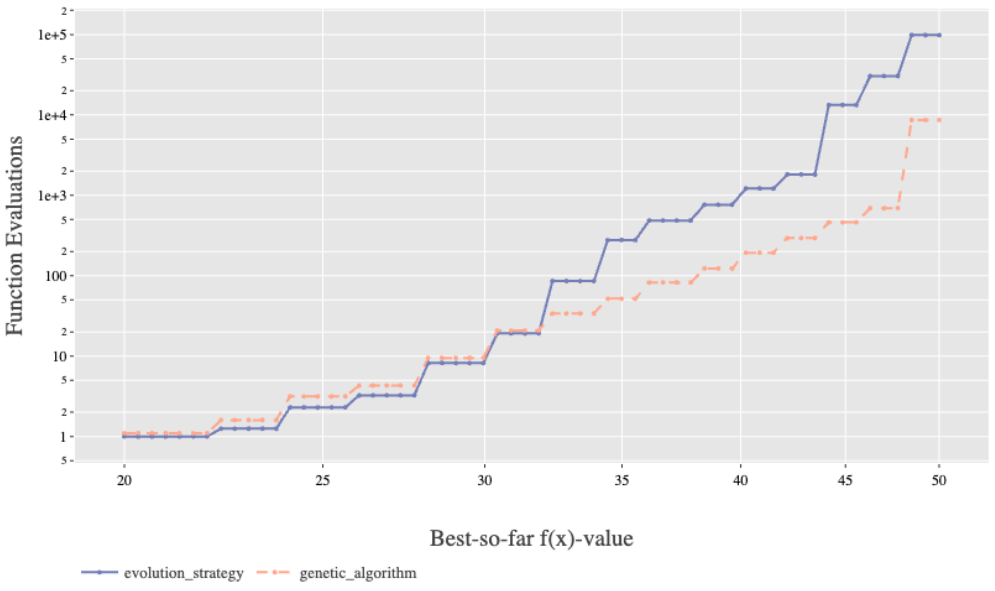

Figure 2: ERT plot for the GA and ES solving the ISING Ring (F19) problem

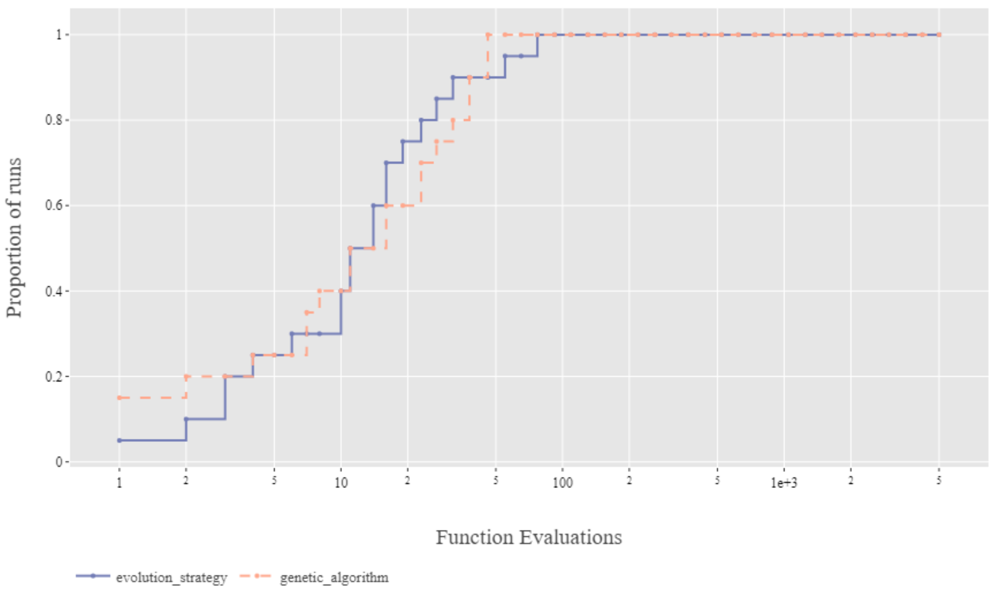

Figure 3: ECDF plot for the GA and ES solving the LABS (F18) problem

evaluations, showing that the algorithm is working and eventually the proportion of runs for the curves do converge at 1.

However, there is far less stagnation, compared to the curves for F19 which for both the GA and ES stagnate and then do a steep climb to reach 1, and instead with F18 the algorithms have a more consistent ascent before converging. Furthermore, for F18 the GA and ES curves are more similar compared to F19, which could be an indicator of their more comparable performances dealing with this problem. This may also further imply that the optimization landscape of LABS may be more amenable to both strategies, which could result in a more consistent and coordinated exploration of the solution space.

Lastly, the smoother ascent and closer alignment of the ECDF curves for F18 suggest that both algorithms converge more consistently and efficiently on solutions for the LABS problem. This behaviour compared to F19 highlights the problem-specific characteristics influencing the algorithms' convergence patterns.

Figure 4 shows the expected running time curve of GA and ES for solving the (F18) LABS problem. As with the F19 problem, we can see with two graphs for this problem perform in a similar matter to F19 where they perform about the same with the same number of evaluations until the reach a value of just under 2 at which point the ES starts to require more function evaluations than the GA.

However, unlike the F19, this does not imply that the GA is more suitable for solving the F18 problem as the ES reaches a higher best fitness value than the GA does, which is clearly displayed in the graph. We believe trade-off of the increased number of function evaluations, in this case 100000, that the ES needs is worth getting the higher accuracy that it can achieve.

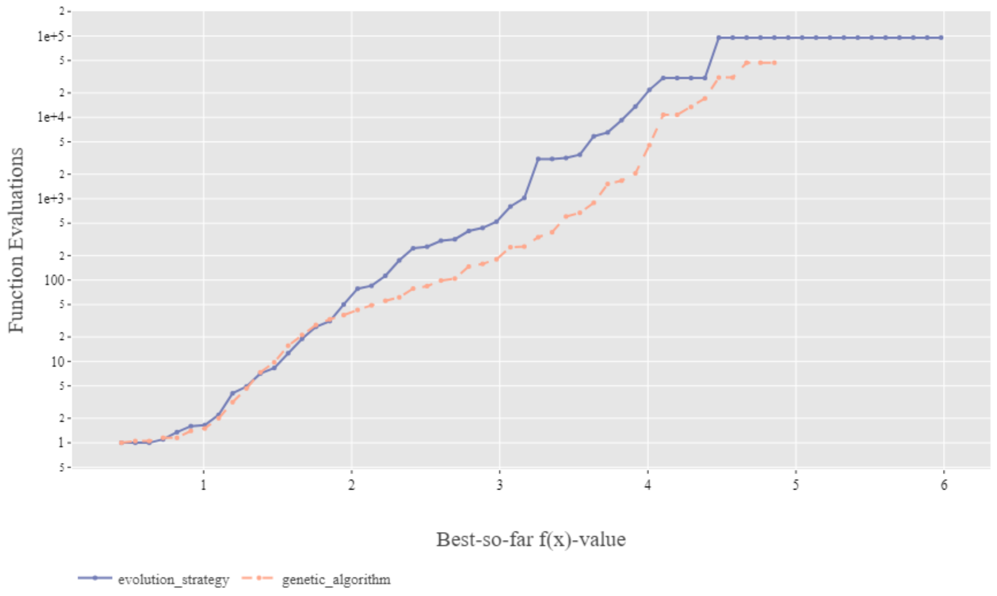

Figure 4: ERT plot for the GA and ES solving the LABS (F18) problem

4 Discussion and Conclusion

While implementing the different algorithms reported above, we have noticed that some changes in particular favoured an increase in the performance and we have reported them below.
For the Genetic Algorithm solving problem F19:

- We have decided in this case to set the population size to a moderate value (7) in order to favour the speed of convergence of the algorithm.
- We have observed that using a rather high mutation rate was beneficial when applying an inversion mutation, as it implied the introduction of random changes, helping the algorithm escape local optima and search for alternative and potentially better solutions [1].
- We have noticed that in this case the choice of a higher number of tournaments in the tournament selection phase was better in order to achieve higher performance.
- We have observed that for this case preserving information was important for maximizing the fitness functions and therefore we have decided to employ a uniform crossover, which indeed ensures that both parents contribute to the genetic makeup of the offspring.

- A high crossover probability has been observed to be beneficial because it helped accelerate the convergence of the GA by combining information from multiple parents, leading to the creation of potentially better solutions. [1]
- In general, the operators that we have used (uniform crossover, inversion mutation and tournament selection) for the GA, were adopted because they lead to higher performances in the specific case of solving the Ising Ring problem. Indeed we have tried different kinds of mutation (bit-flip, uniform, swap, ...) selection (proportional and tournament) and crossover (single-point, n-point, uniform, ...); however by comparing the performances in terms of higher average best found fitness values, we then decided to select the previously specified operators.

For the Genetic Algorithm solving problem F18:

- Using N-point crossover opposed to uniform crossover yielded a higher accuracy and The value of 'N' influences the resolution of the segments, being exchanged which can allow for various mixing of genetic material between parents during the crossover process.
- Using random resetting mutation also yields a higher accuracy for f18 as opposed to inversion mutation for f19 and with random resetting mutation the genetic diversity in the population is increased which allows the algorithm to be able to adapt to different areas within the solution space and attempt to discover improved solutions.

For the Evolution Strategy solving problem F19:

- We have observed that the choice of a tournament selection instead of a rank based selection greatly increased the average best fitness value for 20 independent runs.
- We have noticed that choosing a higher population size μ (compared to the populations size of the GA for F19) is beneficial for the performance of the algorithm since it favours the exploration of the solution space; however this has a negative impact on the cost, requiring more evaluations to find the optimal solution.
- We have chosen to have $\lambda >> \mu$, and therefore a larger number of offspring individuals generated at each iteration with respect to the number of parent individuals in the population. This choice not only enhanced the algorithm to explore a broader region of the solution space by generating a diverse set of offspring, but it also favoured a faster convergence in early evaluations.
- As also for the other algorithms, we have compared different operators types for mutation (one-sigma, individual-sigma, correlated, adaptive and non-uniform), selection (tournament and proportional) and recombination (discrete, intermediate and linear). The operators that we have selected in the end are the ones giving us the highest average best found fitness value for 20 independent runs.

For the Evolution Strategy solving problem F18:

- We noticed that expanding the upper and lower bound to 5 and -5 respectively, increases the search space and yields a higher overall best accuracy.
- Lowering λ to be the same size (20) as μ was beneficial to the accuracy for f18 directly in contrast to F19 which benefits instead from a significantly higher λ value compared to μ.
- The choice of having significantly larger than μ was beneficial, since it allowed the algorithm to emphasize the exploration of candidate solutions.
- Using one σ-mutation opposed to individual increases accuracy and introduced variability in step size as well as direction of the mutation, which increases the adaptability of the algorithm to be able to explore the solution space

In summation, in solving the Ising Ring (F19) and Low Autocorrelation Binary Sequences (F18) problems using Genetic Algorithms (GA) and Evolution Strategies (ES), we found valuable insights into the optimization process. For the Ising Ring problem (F19), the GA proved superior in performance, converging faster as well as achieving a higher average fitness value within the given budget of evaluations. This result can is likely due to the nature of the problem, which involves binary sequences and is well-suited for a GA with its binary representation. In addition, the ES, while adaptable, faced challenges due to the need for encoding real-values into binary strings. For the Low Autocorrelation Binary Sequences problem (F18), the ERT plot show that, while the GA needed fewer evaluations to reach certain fitness levels, the ES, due to obtaining a higher accuracy, justified the additional computational cost. Furthermore, the choice of recombination and mutation operators played a crucial role in achieving optimal solutions, with different problems benefiting from different functions instead of a one size fits all approach.

In future work, exploration could involve even more fine-tuning of the algorithmic parameters, experimenting with alternative functions, and looking into combined approaches that take the strengths of both the GA and ES. In addition, applying these algorithms to even more optimization problems would also help generalize the findings to uncover a wider context of insight into how effective they are.
Lastly, our study highlights the importance of aligning the choice of the algorithm used with the problem at hand, proving the need for adaptability and a deep understanding of the optimization landscape.

References

[1] Baeldung. Genetic algorithms: Crossover probability and mutation probability, 2023.