(EXTRACT FROM) ASSIGNMENT 1 - EVOLUTIONARY ALGORITHMS

Antonio Mone
a.mone@umail.leidenuniv.nl

Riccardo Majellaro
r.majellaro@umail.leidenuniv.nl

February 5, 2022

This is an extract from the full report of the first assignment of the course Evolutionary Algorithms. Only "Implementation" and "Experimental Results" sections are included.

Implementation

A genetic algorithm's main loop is composed of three main methods, namely mating selection, crossover and mutation. Two different types of selection were implemented for this assignment. The first, called fitness proportionate selection, works by assigning a probability to each of the candidate solutions $x \in \mathbf{X}$ following the formula $p_i = \frac{f_i}{\sum_{j=1}^{\mu} f_j}$, $i = 1, \dots, \mu$, and using them to attach the elements of $\mathbf{X}$ to a portion of the interval $[0, 1)$. Then, given a probability $p \in [0, 1)$ drawn from a uniform distribution, the solution related to the portion of interval in which p falls gets selected. In the case where not all the candidates were equals and $\sum_i f_i \neq c \times \mu$, with $c = \min_i f_i$, we applied $p_i = \frac{f_i - c}{\sum_j f_j - c}$ to calculate the probabilities. This scaling increases the probabilities of the solutions with fitness above the mean, while decreases the ones below. The second selection method that has been implemented is the tournament selection. Given k random individuals from $\mathbf{X}$, a fixed probability p_t and a p drawn from a uniform distribution, we still assign a portion of the interval $[0, 1)$ to each of the k candidates, but this time with $p_i = p_t \cdot (1 - p_t)^{i-1}$. Notice that with this technique there is a chance that p falls in a range related to no solutions; in this case the fittest individual is selected. In both proportionate and tournament selections the size of the resulting population is determined by the number of times the processes are repeated. Moreover, in the tournament selection, instead of performing k evaluations every tournament, all the population is evaluated just once. In this way, although is possible to also evaluate individuals not picked in the tournaments, we end up saving iterations because the vast majority of time $|\mathbf{X}| < k \cdot N$, where N is the number of tournaments.
For the crossover we tried both uniform and n-points methods. The former, given a pair of candidate solutions, swaps each bit of the two with probability 0.5. The latter splits the pair in n random points, then alternately swaps their split parts. In order to choose the pairs, we permute the individuals and get the couples following the obtained order.
The mutation consists in flipping each bit of the elements of $\mathbf{X}$ with probability p_m. This probability can be either fixed or proportional. In the last case, specific for the one max problem, $p_{mi} = \frac{1}{2(f_i+1)-l}$ for $i = 1, \dots, \mu$, where l is the length of the bit streams. The result is that the mutation rate decreases as the fit of a solution increases.
These three methods are included in the class $GeneticAlgorithm$. With four parameters is possible to choose the problem to solve, the available budget, the size of the initial population and whether to use (μ, λ) or $(\mu + \lambda)$. When a instance of this class is called, the genetic algorithm starts: it executes 10 independent runs for the chosen problem and keeps track of the best solutions. For reproducibility purposes we set a different seed for each run.
Note that the starting population is randomly initialized drawing every bit of each candidate from a discrete uniform distribution in $[0, 1]$ using $numpy.random.randint$.
An additional detail about the implementation is in the evaluation of the population obtained at the end of an iteration: instead of evaluating after the mutation and before starting the next cycle, we decided to exploit the assessments used in the mating selection to evaluate the previously obtained candidates. In this way we can drastically reduce the total evaluations while preserving the original behaviour of the genetic algorithm.

Experimental Results

As anticipated we tested our methods over three problems: One Max, Leading Ones and Low Autocorrelation Binary Sequences. For each experiment, the fixed budget, or maximum number of function evaluations, was equal to $50 \cdot l^2$, where l is the length of the bit streams. In the first two problems we considered a l of 50, hence a budget of 125000 evaluations, while in the LABS $l = 32$, therefore a maximum of 51200 evaluations.

One Max

The One Max problem is the simplest of the three. The goal is to reach the optimum score of 50. To solve it, we tried numerous configurations of our $GeneticAlgorithm$ class, but we selected only the most interesting of them. Their results obtained by fixing the seeds for all the experiments at 55 are showed in Figure 1. The first one (named ga_om_exp1) starts with a random population of 10 individuals and at each iteration selects 10 offsprings through the tournament selection, with $k = 10$ (therefore full population tournaments) and a probability $p_t = 0.6$. For the crossover it utilizes a 7-points split, while the mutation probability is fixed at $\frac{1}{50}$. Moreover, it doesn't unify the offsprings with the parents after the mutation (indicated with the parameter $plus = False$ in $GeneticAlgorithm$). As we can see, this set up achieved the best results with a mean used budget of 582 and standard deviation of 93. The second experiment (ga_om_exp2) is identical to the first one, but the initial and selected population is equal to 50. In this way we consumed more budget at each iteration hoping to reduce them, but the results proved that this was not the case. The next one (ga_om_exp3) differs from the first because of the proportional mutation rate and the union of parents and offsprings after the mutation. The reason why it demonstrated to be slower is the additional number of evaluations done while computing the proportional mutation probabilities. Then, we changed the crossover technique in the ga_om_exp1 configuration with a uniform one (ga_om_exp4) and, despite the great performance, we could not reach a lower number of iterations. In the last experiment (ga_om_exp5) we set the proportionate selection as mating method, while we left the rest like in the first one. As shown in the figure, this configuration is the one achieving the worst performance. Looking at the standard deviations, it is clear that the first solution is more reliable in its performance compared to the other: in fact the results over the 10 runs were strictly close to their mean.

Leading Ones

In order to solve the Leading Ones problem (once again the optimum is 50), we firstly experimented with an initial population of 30 individuals, tournament selections of 10 rounds with groups of size 10 and $p_t = 0.6$. The crossover was a 4-points split, the mutation probability equal to $\frac{1}{50}$ and $plus$ was set to be $True$ (ga_lo_exp1). Setting a seed of 80, the mean of the evaluations needed in the 10 runs to find the optimum resulted to be almost 4000, hence only about $\frac{1}{13}$ of the total budget (Figure 2). We therefore tried to simplify the algorithm and further cut the evaluations by ceasing to consider the parents along with their offsprings after the mutation (ga_lo_exp2). The result was positive, with approximately 1000 of mean budget saved and a very similar standard deviation. Then, we decided to change the n-points crossover to a uniform one (ga_lo_exp3). In this case we achieved slightly better mean and standard deviation over the runs. Finally, we tried to increase k and the number of tournaments to 30, but since this did not change the performance, we also doubled the mutation rate to $\frac{1}{25}$: in this way the standard deviation dropped by almost half while the mean lightly descends (ga_lo_exp4). Note that we also tried different configurations using the proportionate selection, but the obtained results were significantly worst than the ones shown.

Low Autocorrelation Binary Sequences

The LABS problem is the undoubtedly the hardest of the three. The optimum target is 8, but unfortunately we were never able to reach it within the given budget. Our first attempt was made using a proportionate selection with 10 spins while starting from a random population of 10 individuals. The chosen crossover was the 4-points, while the mutation probability was fixed at $\frac{1}{10}$ and $plus$ set to $False$ (ga_labs_exp1). In Figure 3 is possible to see how this setup could already achieve a mean bigger than 4 and a maximum target over 5, precisely 5.12. As next step, we set $plus$ to be $True$ and found that, unlike with the first two problems, we were able to achieve better results (ga_labs_exp2). In the third experiment we tried the tournament selection with 10 rounds, $k = 4$ and $p_t = 0.6$ (ga_labs_exp3). As expected, we managed to reach higher values, specifically a maximum target of 7.53 and a mean around 5. Starting from this setup, we changed the crossover to a uniform one: unfortunately this configuration could not perform better than the last one (ga_labs_exp4). Note that we also experimented with lower mutation rates, but without significant results.

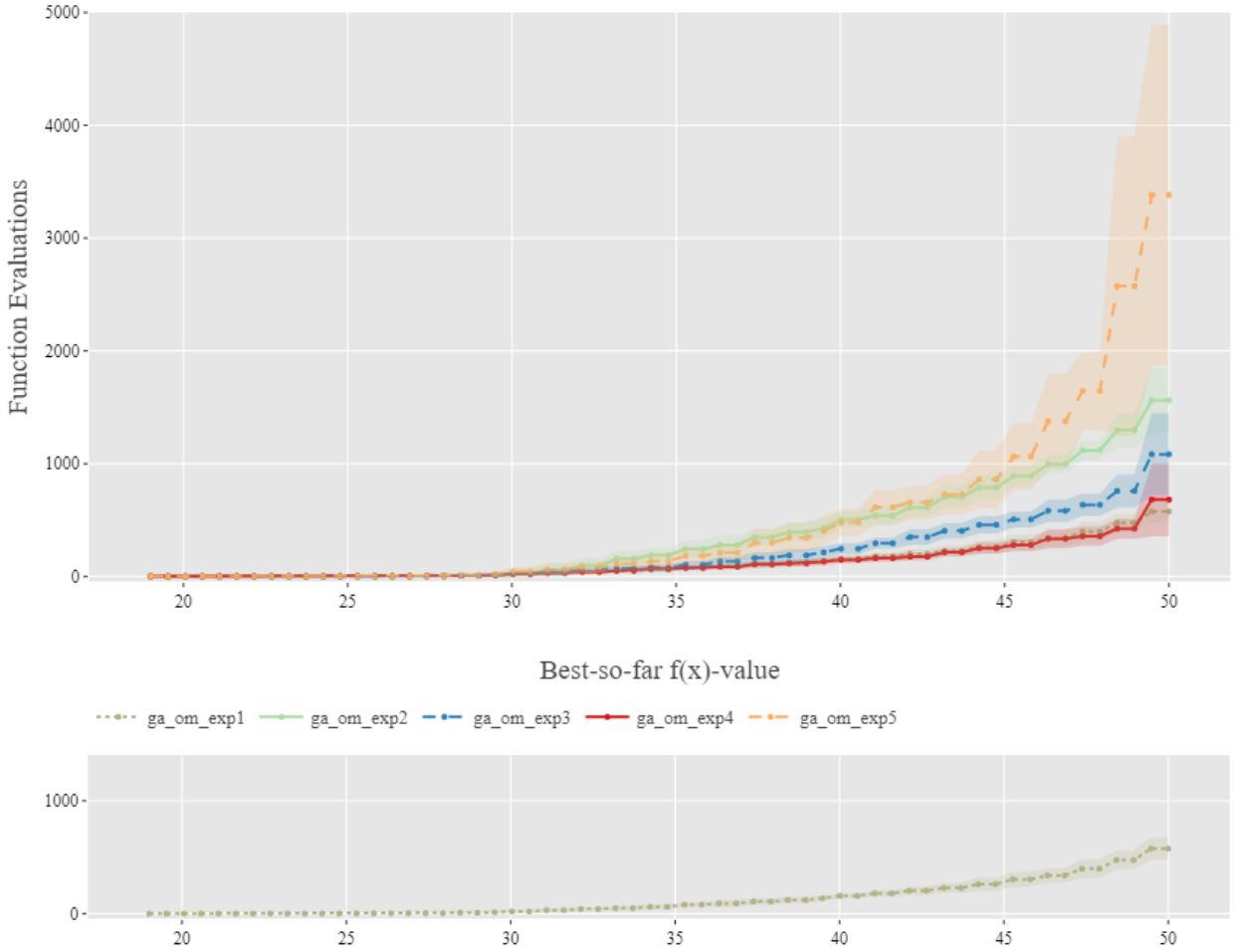

Figure 1: Used budget in function of the best-so-far target for One Max problem. The image on the bottom is meant to show the best solution more clearly, mostly covered in the upper figure.

Table 1: Experiments summary

Problem	*ID*	*initial_pop*	*plus*	*mating*	*mating_params*	*crossover*	*p_m*	*seed*
OM	**ga_om_exp1**	10	False	Tournament	[10,10,0.6]	7-points	$\frac{1}{50}$	55
OM	ga_om_exp2	50	False	Tournament	[50,10,0.6]	7-points	$\frac{1}{50}$	55
OM	ga_om_exp3	10	True	Tournament	[10, 10, 0.6]	7-points	Proportional	55
OM	ga_om_exp4	10	False	Tournament	[10, 10, 0.6]	uniform	$\frac{1}{50}$	55
OM	ga_om_exp5	10	True	Proportionate	[10]	7-points	Proportional	55
LO	ga_lo_exp1	30	True	Tournament	[10, 10, 0.6]	4-points	$\frac{1}{50}$	80
LO	ga_lo_exp2	30	False	Tournament	[10, 10, 0.6]	4-points	$\frac{1}{50}$	80
LO	ga_lo_exp3	30	False	Tournament	[10, 10, 0.6]	uniform	$\frac{1}{50}$	80
LO	**ga_lo_exp4**	30	False	Tournament	[30, 30, 0.6]	uniform	$\frac{1}{25}$	80
LABS	ga_labs_exp1	10	False	Proportionate	[10]	4-points	$\frac{1}{10}$	55
LABS	ga_labs_exp2	10	True	Proportionate	[10]	4-points	$\frac{1}{10}$	55
LABS	**ga_labs_exp3**	10	True	Tournament	[10,4,0.6]	4-points	$\frac{1}{10}$	55
LABS	ga_labs_exp4	10	True	Tournament	[10,4,0.6]	uniform	$\frac{1}{10}$	55

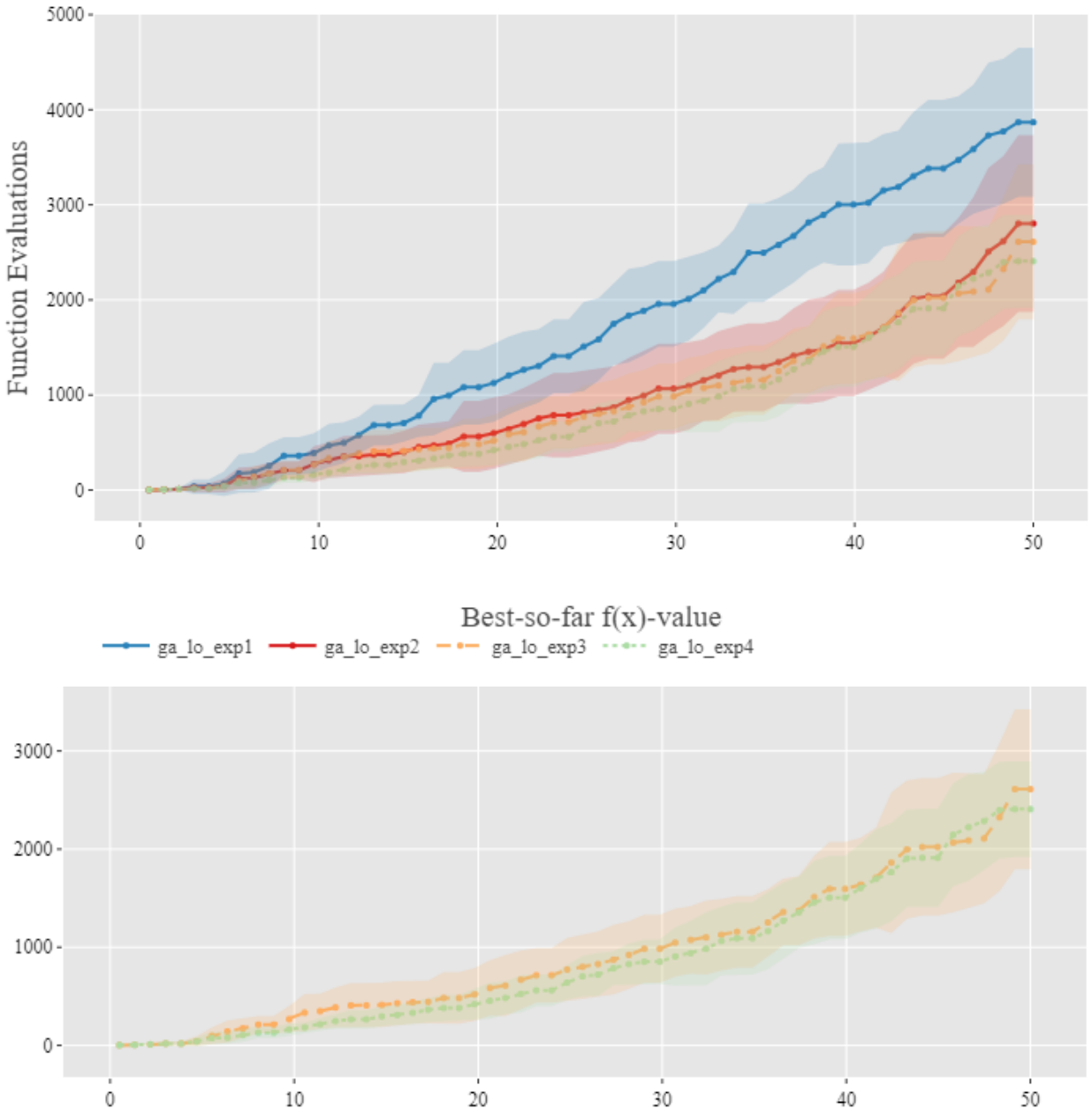

Figure 2: Used budget in function of the best-so-far target for Leading Ones problem. The image on the bottom is meant to show the standard deviation difference in between the best two solutions more clearly.

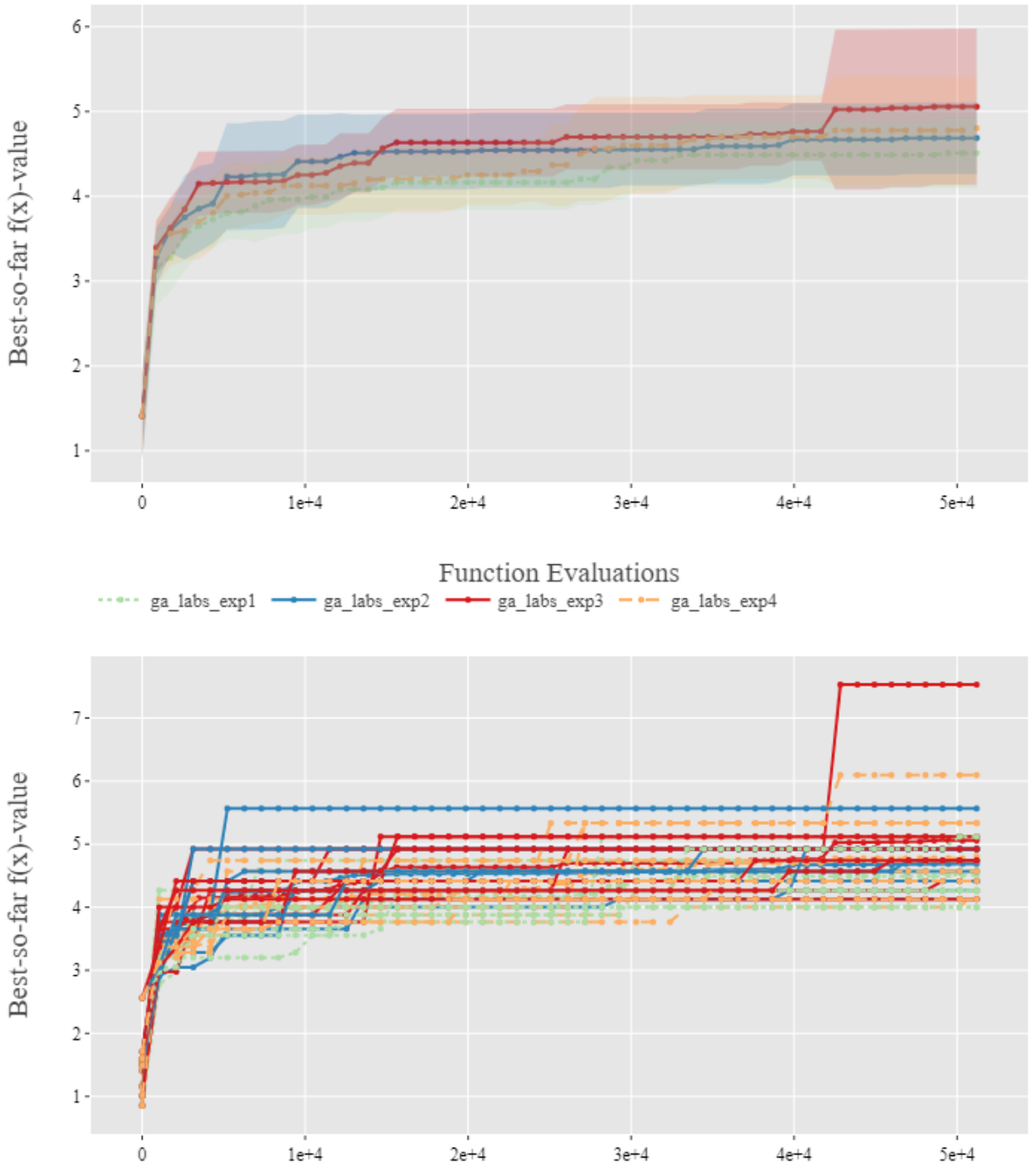

Figure 3: Best-so-far target in function of the used budget for LABS problem. The image on the bottom is meant to show the maximum target reached in the different runs.