TP6: Genetic Algorithms and Function Minimization

BY MARTINO FERRARI

1 Function Minimization

In this tp we will try to minimize the fitness function

$$f(x,y) = -\left|\frac{1}{2}\cdot x\cdot \sin\left(\sqrt{|x|}\right)\right| - \left|y\cdot \sin\left(30\cdot\sqrt{\left|\frac{x}{y}\right|}\right)\right|$$

with $x, y \in [10\colon 1000] \cap \mathbb{N}$.

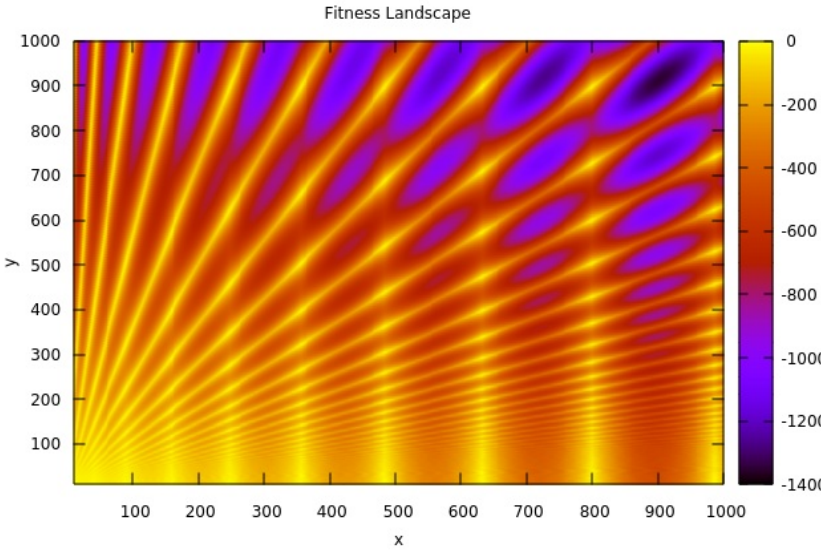

Figure 1. Fitness landscape

In figure 1 the *fitness landscape* of the above function is shown. The *global minimum* of the function can be found in position $(x = 903, y = 917)$ with value -1356.5.

To represent the research space S we will use two chains of 10 bits, each able to represent numbers from 0 to 1023.

$$x, y \in [0, 1023] \cap \mathbb{N}$$

For this reason I chose to map the function domain using the following relations:

$$x' = \frac{x\cdot 990}{1023} + 10$$

$$y' = \frac{y\cdot 990}{1023} + 10$$

In this way x' and y' are in the range $x', y' \in [10, 1000] \cap \mathbb{R}$, and by using the round function it is possible to return in the $\mathbb{N}$ set:

$$x'' = \text{round}(x')$$

$$y'' = \text{round}(y')$$

and the function will be:

$$f(x'', y'') = -\left|\frac{1}{2} \cdot x'' \cdot \sin\left(\sqrt{x''}\right)\right| - \left|y'' \cdot \sin\left(30 \cdot \sqrt{\frac{x''}{y''}}\right)\right|$$

I removed the internal absolute values as $\mathbb{N}$ is positive defined ($\mathbb{N} = [0, \infty]$).

Finally the research space has 2^{20} virtual solutions and only $\sim 2^{19.9}$ real solutions (due to the round and upper and lower bounds).

2 Genetic Algorithms

The Genetic Algorithms (**GA**), introduced in the 70th by Holland and John, are part of bigger group of metaheuristics called the *Evolutionary algorithms*. This family of metaheurstics is based on some of the principles of evolution and natural selection (some presented of them by C. Darwin in *"On the Origin of Species"*).

In particular the **GA** family is a population-based metaheuristic, where each individuals of the population, $x_i \in S$, represents a possible solution of the problem.

It's also important to define some terminology, the coding of each individual is seen as his *DNA*, the position or the value of x_i is called *genotype*, while its fitness is called *phenotype*. Finally the iterations are called *generations*.

The population of the **GA** will evolve at each generation by applying the following actions:

1. Selection: selection of the individuals to evolve from
2. Crossover: crossover of couples of parents
3. Mutation: random mutations of the individuals

As it's possible to imagine, the names are linked to the evolutionary and genetic homonymous processes.

We can schematically express this process of evolution of the population $P(t)$ at the generation t as:

$$P(t) \underset{\text{selection}}{\longrightarrow} P^1(t) \underset{\text{crossover}}{\longrightarrow} P^2(t) \underset{\text{mutation}}{\longrightarrow} P^3(t) = P(t+1)$$

It's also possible to exchange the worst element of $P(t+1)$ with the best of the previous generation.

It's also important to see that the size of the population is constant (it's true also for the intermediary populations P^1, P^2, P^3). The result of this operation represents the research operator U.

Often the $P(0)$ is randomly generated.

In the next subsections, I will explain in details selection, crossover and mutation.

2.1 Selection

The selection action represents somehow the process of *Natural Selection*, that C. Darwin expressed as *"Survival of the fittest."*, and that selects the bests individuals of the population from which the next generation will evolve. However simply selecting the best individuals will be counterproductive as it will probably focus the evolution in a sub optimal region of S (due an excessive intensification). There are many different selection strategies to avoid this problem and to add some randomness, in particular:

- Fitness proportional selection: each individual x_i has probability p_i to be selected:

$$p_i = \frac{f(x_i)}{\sum_{j=1}^{n} f(x_j)}$$

- Selection by tournament: where k random individuals are competing and the one that will win the tournament will be selected.

- Selection by rank: after ordering the individuals by the fitness, the probability p_i of one individual i to be selected will be:

$$p_i = \frac{1}{n}\left[\beta - 2(\beta - 1)\frac{i-1}{n-1}\right]$$

  with $1 \leqslant \beta \leqslant 2$.

The main problem of the selection action is the fast convergence of the individuals of the population to the best individual, that will reduce the probability of success in rough landscapes. For this reason, choosing the correct selection method for each problem is critical.

2.2 Crossover

The crossover operation reflects the chromosomal crossover where two chromosoms of the parents exchange parts of their genetic information. In the same way, with a given p_c crossover probability, pairs of individuals exchange parts of their coding to generate the next generation.

To do so, the population $P^1(t)$ will be divided in $\frac{n}{2}$ pairs and the crossover operation will be applied with probability p_c. In particular, we saw 3 possible crossover methods:

- Single-point crossover: the couple code will be splitted in two parts and the parts will be exchanged.

- Multiple-point crossover: a variation of the previous method such that the code will be splitted in two or more parts.

- Uniform crossover: each bit of a parent code has a given probability to be exchanged with one from the other parent.

2.3 Mutation

Each individual of $P^2(t)$ will have a given p_m probability of mutation. In particular, with a binary coding of the solution, each bit of a solution will have p_m probability to mutate (to switch).

In this way it's possible to maintain diversity in the population and better explore the landscape.

2.4 Guiding Parameters

To summarize a **GA** has 3 guiding parameters:

1. n: the size of the population
2. p_c: the crossover probability
3. p_m: the mutation probability

The influence of each parameter will be studied in section 5.

3 GA and Function Minimization

We need to define for **GA**, as for all the other metaheuristics, few elements:

- the solution space S
- the initial solution x_0, or in this case the initial population $P(0)$
- the fitness function $f(x)$
- the research operator U
- the stop condition

The space solution S and the fitness function $f(x)$ have been introduced in section 1.

The initial population $P(0)$ will be randomly chosen in S.

The research operator U, as specified in section 2, will be the ensemble of the selection, crossover and mutation operations.

Finally, the stop condition can be either a number of generations to reach $t_{\max}$ (or similarly a number of fitness evaluations) or a given fitness f_{target} to reach.

4 Implementation

I implemented the algorithm in C++ extensively using the object-oriented features of the language, and reusing all the metaheurstic framework developed till now. The individuals represent a possible solution as an array of n bit (20 in this case) that will be splitted in 2 to represent x and y.

The selection is done using tournament method with $k = 5$ individuals per tournament.

The crossover is done using the one-point crossover method with a mid-break policy (meaning that I'm exchanging x_1 with y_2 and y_1 with x_2).

Finally, the worst individual of the next generation is replaced by the best of the previous one.

5 Results

This time the landscape of S is defined mathematically and there is no need to study it more. it is shown in figure 1 and figure 2.

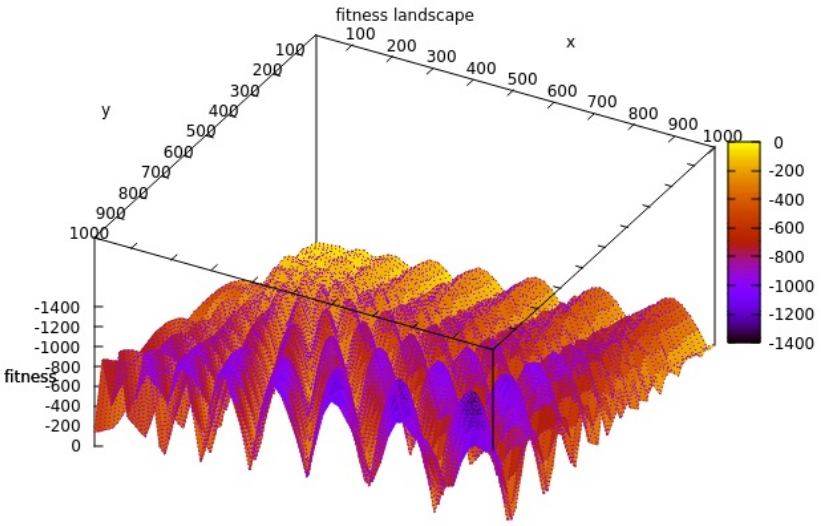

Figure 2. 3D fitness landscape

As it possible to see in both figure 1 and 2 the landscape is rough and has multitude of local minima. The global optimum has value of -1356.5 at position ($x = 903, y = 917$).

Before analyzing the performance of my implementation of the **GA**, I analyzed the impact of the parameters p_c, p_m and n.

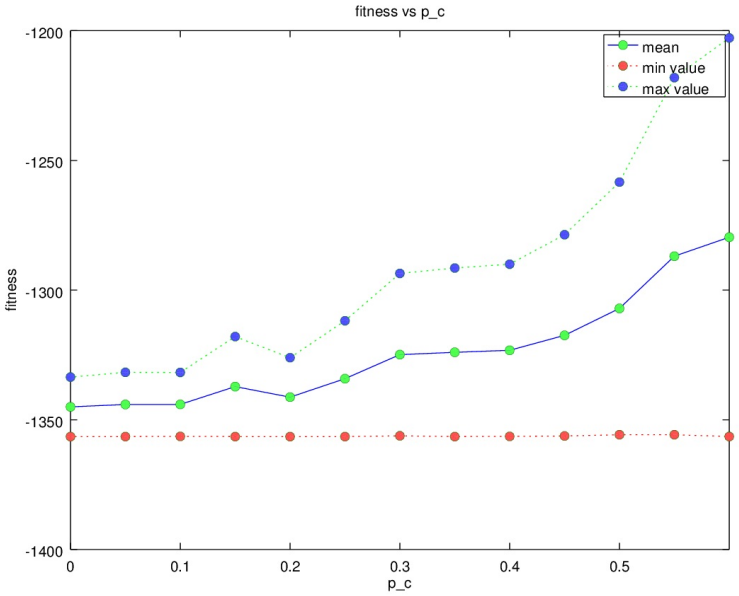

Figure 3. Fitness vs p_c (50 executions per point, $n = 100$, $p_m = 0.1$)

In figure 3 it is possible to see the evolution of the fitness vs p_c. It's easy to see that while increasing p_c the diversification increases as well and the fitness has a larger spread.

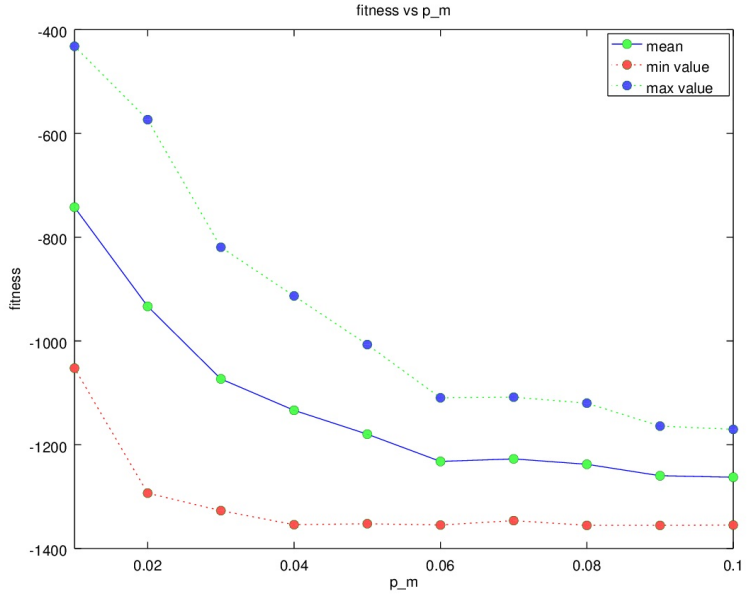

Figure 4. Fitness vs p_m (50 execution per point, $n = 100$, $p_c = 0.6$)

In figure 4 it is possible to see how the probability of mutation p_m is important to diversificate and correctly explore the landscape. Between $p_c = 0.01$ and $p_c = 0.1$ the average fitness falls from ~ -800 to ~ -1350.

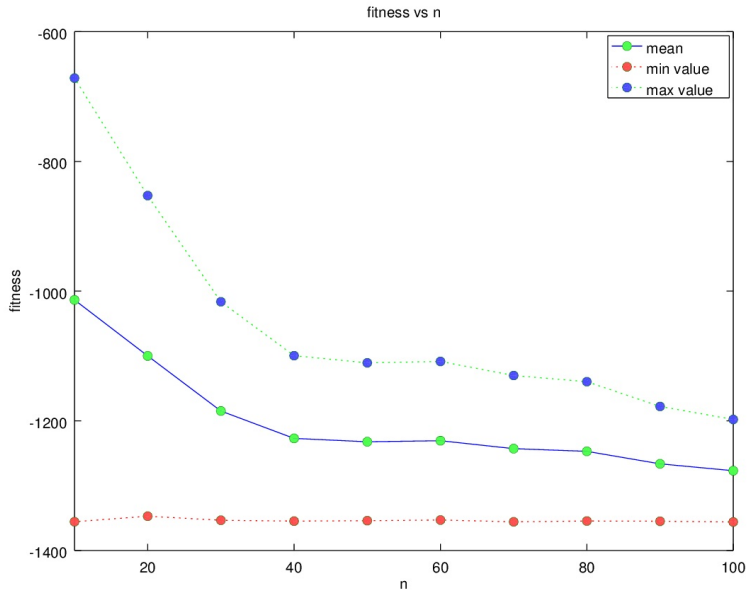

Figure 5. Fitness vs. n (50 execution pr point, $p_m = 0.1$, $p_c = 0.6$)

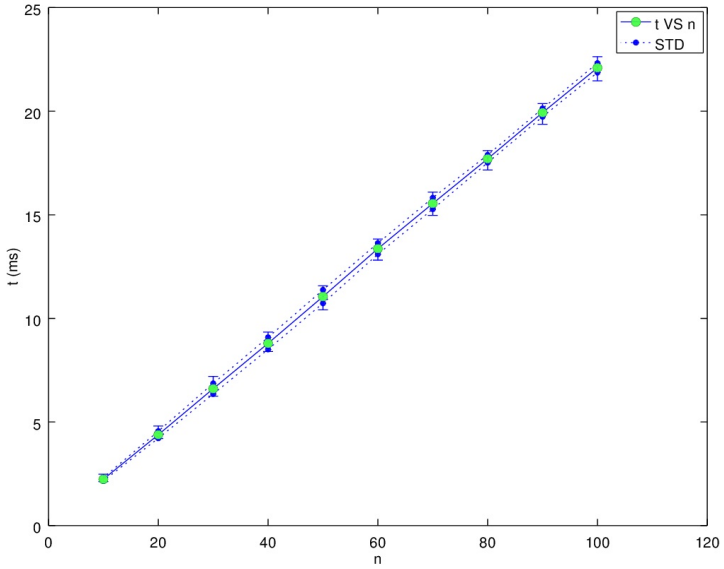

Figure 6. t_{exec} vs n (50 execution per points, $p_m = 0.1$, $p_c = 0.6$)

In figure 5 it is possible to see that also the size n of the population is very important and with a too small population the convergence of the individuals to the best one is too quick and freezes the research in sub optimal regions. Moreover the execution time grows only linearly with the size of the population, as shown in figure 6.

Numerical results of the previous experience are shown in table 1.

	f_{best}	f_{mean}	STD	n_{evals}
$p_m = 0.01, p_c = 0.6$	-1347.8	-1045.6	169.2	5000
$p_m = 0.1, p_c = 0.6$	-1356.5	-1330.4	32.6	5000
$p_m = 0.1, p_c = 0$	-1356.5	-1348.2	16.7	5000
$p_m = 0.1, p_c = 0.6$	-1356.5	-1339.5	26.3	5000

Table 1. Table of results

After this considerations I chose to use $n = 100$, $p_m = 0.1$ and $p_c = 0.6$ as it seems to be the best combination of parameters.

The last parameter we can change is the $t_{\max}$, the maximum number of generations.

In particular n_{eval} the number of fitness evaluations (a very useful, machine-indipendent, number to compare metaheurstics performance) can be computed as:

$$n_{\text{eval}} = n \cdot t_{\max}$$

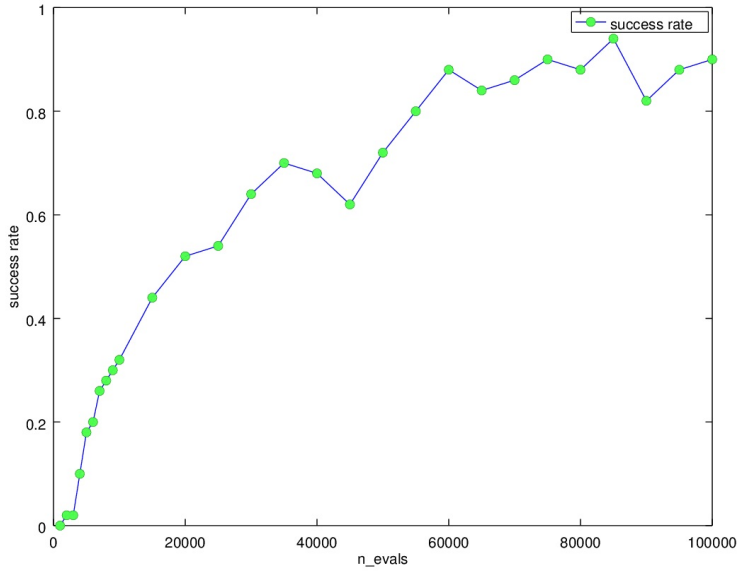

Figure 7. Success rate vs n_{eval}

As it's possible to see in figure 7 the success rate (number of time the global best is found over number of trials) has a sort of logarithmic trend, and it reaches very good performance already at 20000 fitness evaluations (as shown in figure 8). After 40000 evaluations the success rate is greater than 50%.

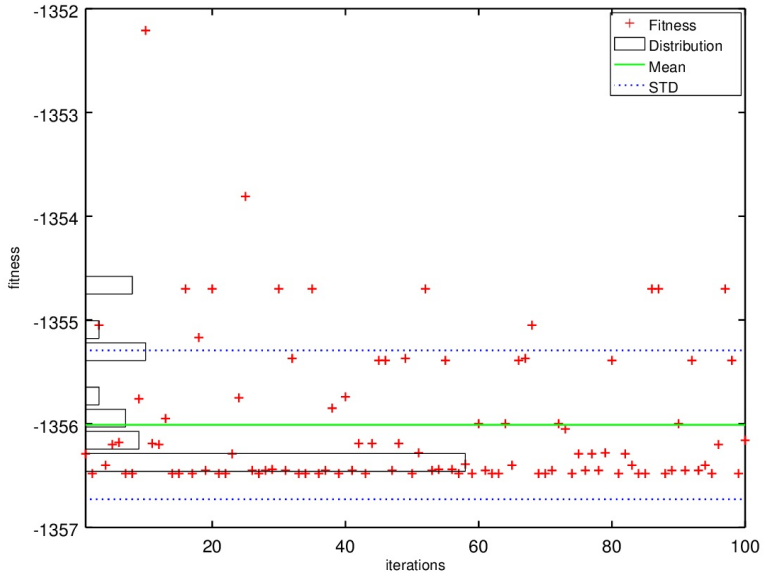

Figure 8. 100 run with $n_{\text{eval}} = 20000$

However if we take in to account all the results with a relative $(d = (f - f_{\text{best}})/f_{\text{best}})$ distance of 0.1% and 0.2% the results are even better, as shown in figure 9.

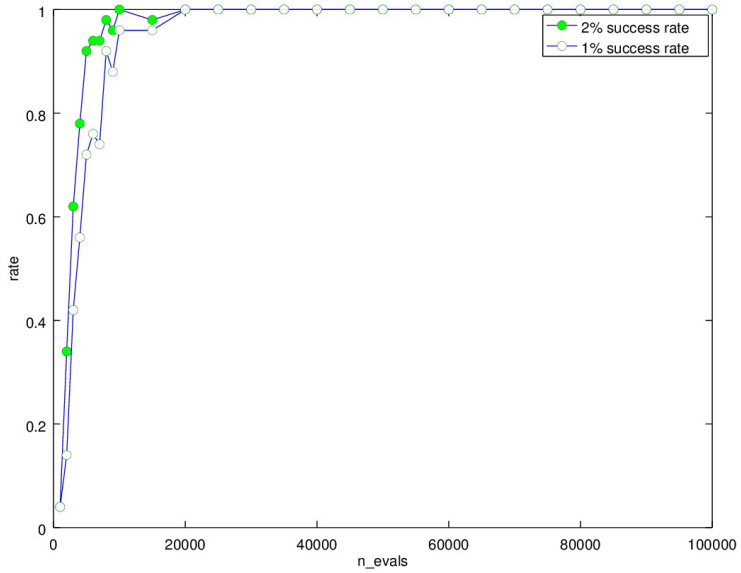

Figure 9. 0.2% and 0.1% sucess rate v.s. n_{eval}

In this case already with 10000 evaluations the success rate is close to 1.

6 Project structure

compile.sh	is the compilation script
src/	contains all the source code (above the core components)
→/geneticalg.*	is the code that defines both the solution representation and the **GA**
→/main.cpp	main file
bin/	contains all the binary
→/ga	runs the **GA**
→/ga_runner.sh	runs the **GA** 100 times
bin/results	contains the generated csv and other output files

The code and the reports can also be found on my github repository: Metaheuristic.