Universitat Autònoma de Barcelona

CENTRE DE RECERCA MATEMÀTICA

Genetic Algorithms:
a possible model for COVID-19

Optimisation

Pol Pastells Vilà and Narcís Font Massot
Master's degree in Modelling for Science and Engineering

January 2021

1 Introduction

The aim of this project is to use a genetic algorithm to solve a parameter optimization problem. A Genetic Algorithm (GA) is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms. [1]. We are going to use different selection and genetic operations to find the best individuals.

We have build a program that implements the GA using Python3 for the proposed model. With minor modifications, it can be used to solve parameter optimizations for similar models.

1.1 Motivation

We are going to find the optimal parameters for a possible model for COVID-19. The proposed model is based on a SEIR-like compartmental model, with 5 different compartments for the infectious, depending on the symptoms.[1]

The model has 11 different parameters and 3 initial conditions that have to be determined, based on the known data for 5 variables for 100 days and a population size $P = 1,000,000$.

[1]For more information check the assignment instructions attached with the program code.

2 Genetic Algorithm Implementation

In this section, we are going to expose all the basis we used on the implementation of the GA. An individual is defined as a Python class that contains the parameters and initial conditions of the model. It also solves the ODE system using Scipy's *odeint* and compares its result to the data computing a fitness function. Lastly, it plots the solution and comparison for the fittest individual found.

To get our results, we have used the following strategy in our program:

- For the zeroth generation, we generate 1000 random individuals, of which the 50 best are selected.
- We keep track of the top 10 individuals (elite).
- A first round of 50 generations is done, letting our program be exploratory. Here we recombine the 50 best individuals of the previous generation in random pairs. And create 3 extra mutations of every new individual, resulting in 200 individuals.
- Afterwords, we do a second round starting only with the elite individuals found at the end of the previous round. 50 more generations are computed with the addition of more mutations (7) to further explore the parameter space. This way, the second round is more exploitatory.

2.1 Fitness Function

To identify the individuals that could have generated the pandemic, we have to find the "fittest" individuals from the point of view of generating the observed data.

We are going to use the following function ("fitness function") to quantify the agreement between the data generated from an individual and the real data.

$$\max_{t=1,\dots,100}\{(A_d(t)-\overline{A_d(t)})^2+(I_2(t)-\overline{I_2(t)})^2+(Y(t)-\overline{Y(t)})^2+(R(t)-\overline{R(t)})^2+(D(t)-\overline{D(t)})^2\}/1e6$$

Where the variables with an overline denote the values of the *observed* data from the pandemic, so, to get a major agreement, we want this function to have the minimum possible value. We decided to divide the proposed function by a million to obtain more digestible numbers.

2.2 Recombination

Recombination consists of mixing and matching parts of two parents to form children. We have used the following Intermediate Recombination Algorithm. [2]

Note that we have also implemented the lineal recombination in our program, but we got better results using intermediate recombination.

Algorithm 29 *Intermediate Recombination*

$p \leftarrow$ positive value which determines how far long the line a child can be located ▷ Try 0.25
$\vec{v} \leftarrow$ first vector $\langle v_1, v_2, ...v_l \rangle$ to be crossed over
$\vec{w} \leftarrow$ second vector $\langle w_1, w_2, ...w_l \rangle$ to be crossed over

for i from 1 to l **do**
  repeat
    $\alpha \leftarrow$ random value from $-p$ to $1+p$ inclusive ▷ We just moved these two lines!
    $\beta \leftarrow$ random value from $-p$ to $1+p$ inclusive
    $t \leftarrow \alpha v_i + (1-\alpha) w_i$
    $s \leftarrow \beta w_i + (1-\beta) v_i$
  until t and s are within bounds
  $v_i \leftarrow t$
  $w_i \leftarrow s$
return $\vec{v}$ and $\vec{w}$

Figure 1: Intermediate Recombination Algorithm pseudocode.

2.3 Mutation

Mutation is used to maintain genetic diversity from one generation of the population to the next.

We have used non-uniform mutation [3], which is expressed as

$$x_{new} = x + \tau(x_{max} - x_{min})(1 - r^{(1-t/t_{max})^b})$$

Where:

- τ takes -1 or $+1$, each with a probability of 0.5.
- r is a random number in $[0, 1]$.
- t_{max} is the maximum number of generations.
- t is the current generation number.
- b is the design parameter (set to 0.5).

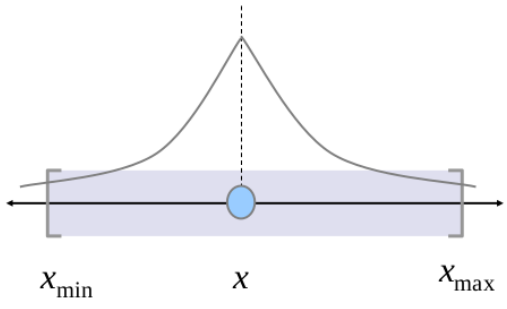

Figure 2: Non-uniform Mutation Scheme

With this method, the solution is more likely to be close to the original solution. Also, this tendency gets increased as the generation number increases.

2.4 Elitism

Elitism consists of directly injecting into the next population the fittest individuals from the previous population. These individuals are called the elite.

In our code, elitism is implemented in the recombine function, where before recombining, a copy of the elite individuals is made into the next generation. Even so, the original elite individuals are also used to recombine with the rest of individuals.

2.5 Allowed parameter ranges

In order to obtain a reasonable solution we restrict the parameter space according to some basic epidemiological knowledge as follows:

- $\beta \in [0.05, 0.5]$
- $\phi \in [0, 1]$
- $\varepsilon_i \in [0, 1]$
- $\varepsilon_y \in [0, 1]$
- $\sigma \in [0.05, 2]$
- $\gamma_1 \in [0.05, 2]$
- $\gamma_2 \in [0.05, 2]$
- $\kappa \in [0.005, 0.5]$
- $p \in [0.001, 1]$
- $\alpha \in [0.001, 0.5]$
- $\delta \in [0.001, 0.3]$
- $E(0) \in [1, 100]$
- $I_1(0) \in [1, 100]$
- $A(0) \in [1, 100]$

3 Results

In this section, we expose and analyze the results obtained with our implementation of the Genetic Algorithm. We are going to examine the fitness function values, the initial conditions and the parameter values obtained and compare the results with the observed data.

3.1 Fitness function

The fitness function values we get for 100 generations are exposed in Table 1.

We have obtained a major improvement in the first ten generations, and for the last generation, we have archived a fitness function value of 4.57, which is a very satisfactory result. Note that at the generation forty our program starts to struggle to get a better result on the fitness function, then we start with the exploitatory round to increase the program performance.

Generation	Value
1	1262.90
10	55.33
20	31.29
30	25.57
40	21.61
50	21.61
60	18.22
70	8.36
80	7.40
90	6.66
100	4.57

Table 1: Fitness function value for the different generations.

3.2 Initial conditions and parameters

The initial conditions and parameters obtained are the following.

Variable	Value
$E(0)$	41.1
$I_1(0)$	3.4
$A(0)$	34.9

Parameter	Value
β	0.35
ϕ	0.52
ϵ_I	0.43
ϵ_Y	0.07
σ	0.10
γ_1	0.08
γ_2	0.06
κ	0.11
p	0.52
α	0.05
δ	0.03

Table 2: Solution obtained with the GA.

3.3 Solution

We can plot the results obtained by solving the model using the previous parameters. We notice that the growth for all the compartments is still almost exponential, therefore the data is "easily" reproducible by a SEIR-like model.

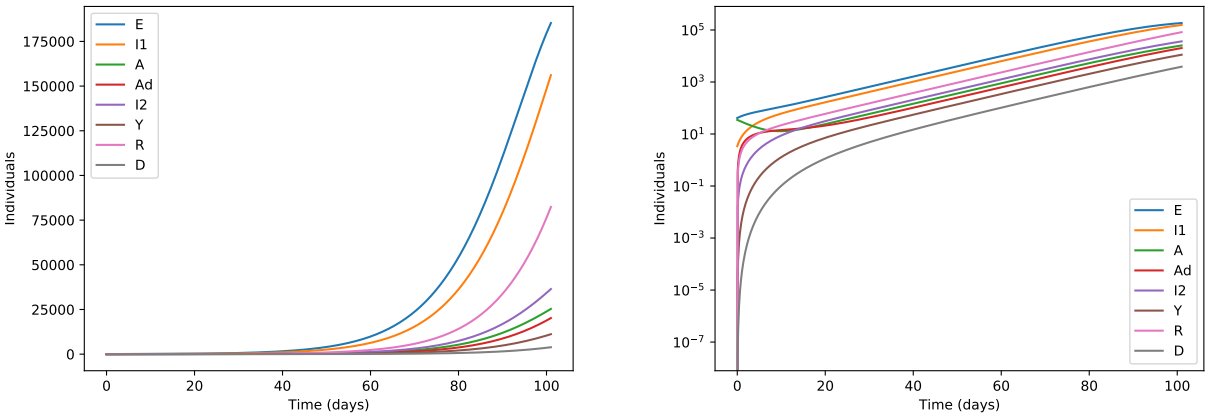

Figure 3: Model results in a linear (left) and logarithm scale (right). Susceptible individuals are not plotted.

Finally, we are going to compare the results obtained with the observed data.

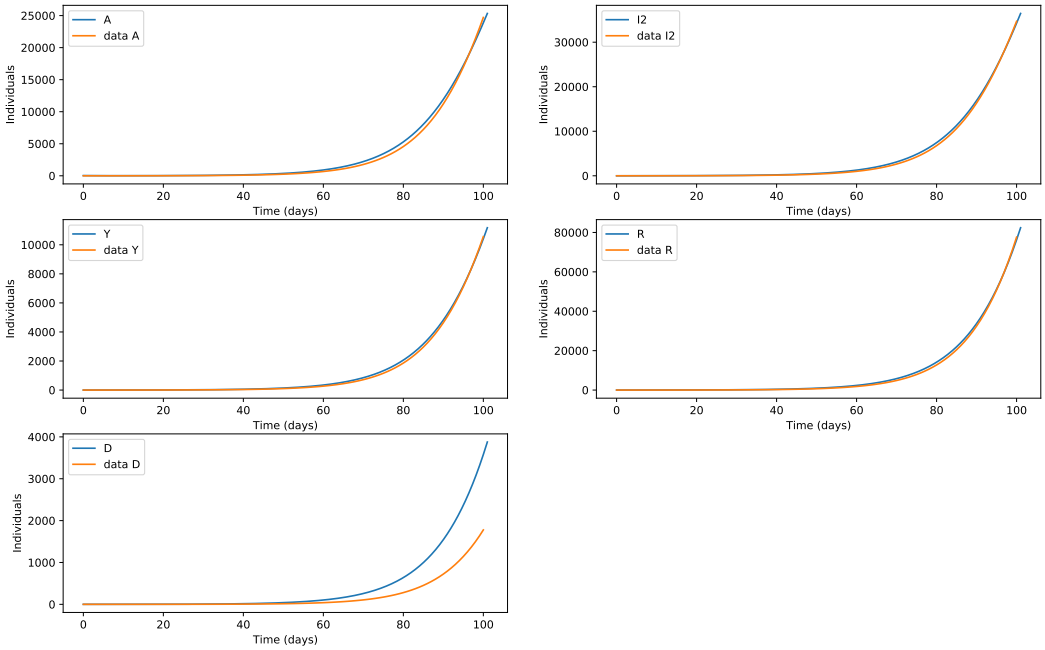

Figure 4: Results comparison with the observed data. x-axis: Time, y-axis: Individuals

Analyzing the previous figures, we can appreciate that the results obtained for the different variables match correctly with the observed data. The only variable where we can observe discrepancies with is the number of deaths, where the result obtained is bigger than in the observed data.

4 Conclusions

We have implemented a genetic algorithm using Python to solve a parameter optimization problem for a possible COVID-19 model. The obtained results match correctly with the data observed. Therefore we could use this model to predict the possible outcome some days into the future.

To further improve our work we could implement a steepest descent algorithm to improve every individual found by the genetic algorithm. We could also look into generating Gaussian mutations and changing the variance depending on the performance of the mutated respect to the original individual.

References

[1] Wikipedia, *Genetic algorithm* , https://en.wikipedia.org/wiki/Genetic_algorithm , January 2021.

[2] Lones, M. (2011). Sean Luke: Essentials of Metaheuristics. Zeroth Edition. 2009.

[3] Purdure Engineering, Lecture 4: Real-Coded Genetic Algorithm. https://engineering.purdue.edu/~sudhoff/ee630/Lecture04.pdf