User Manual for Non-dominated Sorting Genetic Algorithm II (NSGA-II) using Binary Sequencing

Sonal Shibu

October 2022

1 Introduction

Genetic Algorithms were developed in the 1960s by John Holland and his peers. Non-dominated Sorting Genetic Alogorithm II was developed by Kalyanmoy Deb in the early 2000s. It has been used widely in the computer science and engineering sectors to find the optimum solutions by *probing* the design space, when the exact objective/ mathematical functions are not known. Genetic Algorithm are based on Darwin's theory of evolution. Where the strongest parents will spread it genes to their offspring. Parents is a collection of elements within the code that have information about the parameter of the problem the user is trying to solve. Within the code, some functions such as crossover and mutations are shown. These functions are there to explore all sections design space [1, 2]

1.1 Crossover and Mutation

Crossover function is when an element is swapped with another element. This is a completely random process. To remove any bias from the final solution. Mutation is when the element switches to a reciprocal counterpart. Meaning if the element in chosen is 0 it will switch to 1 and vice versa. This is also random to remove any bias. It is recommend to change crossover and mutation rates within the script as it is a function of population size[3, 4, 5, 6]

2 How to use the code

2.1 Minimum System Requirements

1. Modern Operating System such as
   - Windows 7 or 10
   - Mac OS X 10.11 or higher, 64-bit
   - Linux: RHEL 6/7, 64-bit (almost all libraries also work in Ubuntu)
2. x86 64-bit CPU (Intel / AMD architecture)
3. 4 GB RAM
4. 5 GB free disk space
5. The following module(s) must be downloaded on the local PC:
   - Numpy
   - Random
   - Sys

It just be mentioned as more data points are asked from the user, the higher RAM is needed to store the data. Also the code was written in Spyder Python IDE, therefore it recommend by author to use the same platform to avoid any complications. However the script should work in other IDE.

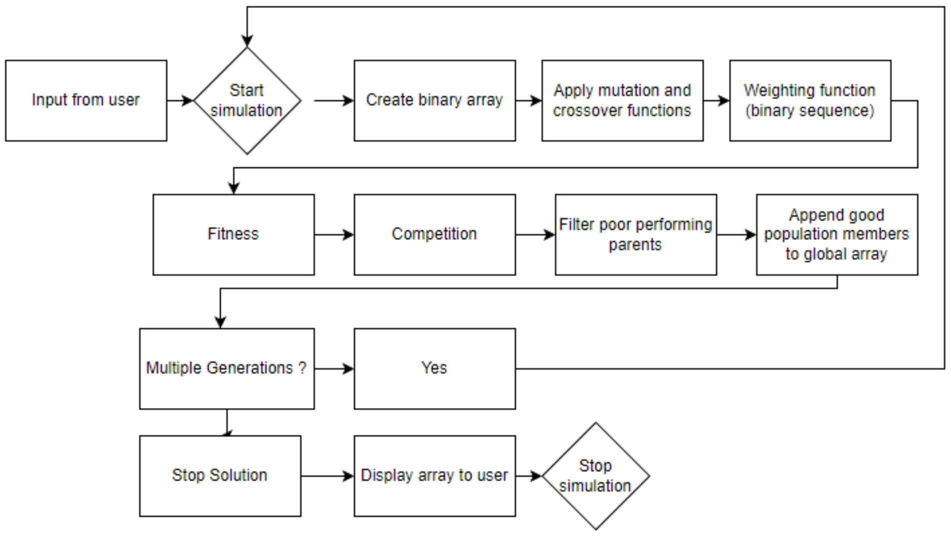
Input from user
Start simulation
Create binary array
Apply mutation and crossover functions
Weighting function (binary sequence)
Fitness
Competition
Filter poor performing parents
Append good population members to global array
Multiple Generations ?
Yes
Stop Solution
Display array to user
Stop simulation

Figure 1: Flow Chart

2.2 How it works

Figure 1 shows the flow chart of the code. The user only needs to change four key parameters. The number of *generations*, *population size*, *population maximise* and *Population maximise*. Firstly these four parameter must be integers and population size has a condition as a condition where the integer must be even.

When choosing variables for the *population maximise* and *Population maximise*. The user needs to use integers for how many bytes it has in a binary matrix. Take integer 13, in binary, this is 1101 therefore it has four byte. Thus, the user would input the number four into the variable. Now if the user wants a floating point number in the final solution, the user would have to divide that number at the end of simulations.

Example: the user wants to maximise a length of a metallic nail. The optimisation critera is the nail cannot be equal or greater than 25mm, therefore the maximum length of this problem is 24.99mm. This is not possible to convert to binary, but simply multiplying this number by 100 shall give an integer therefore making it acceptable for this code. 2499 in binary is 100111000011 and this has 12 elements. Therefore the user should input 12 in the maximum variable. However at the end of simulations the user should divide the global matrix by 100 to return to float numbers. The same principle is applied to the minimise variable.

It must be mentioned adding more byte to the program will cause the script to run longer, so caution is advised. Moreover, the use of floating points in the previous example is the only way to have decimal number but this creates another problem, there is high chance that number will increase greater than 2499, as the binary array might show all the binary elements being true i.e (111111111111 or 4095 in decimal number) this is great as it not useful information to the user. So caution is advised when choosing the accuracy the floating points.

It is up to the user to know how to interrupt data from the arrays. As this code is for general case. If the arrays are very large exceeding 200 data points it is advised to keep the values with python and using the graphing modules within python rather than exporting the data to Microsoft Excel or Google Excel.

3 Legal

The author assumes no responsibility for any damages that might occur while using the script. Use the script at your own risk.

References

[1] Charles Darwin. *The Origin of Species by Means of Natural Selection, Or, The Preservation of Favoured Races in the Struggle for Life*. Books, Incorporated, Pub., 1913.

[2] Kalyanmoy Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE transactions on evolutionary computation* 6.2 (2002), pp. 182–197.

[3] Kenneth Alan De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* University of Michigan, 1975.

[4] John J Grefenstette. "Optimization of control parameters for genetic algorithms". In: *IEEE Transactions on systems, man, and cybernetics* 16.1 (1986), pp. 122–128.

[5] Duc Pham and Dervis Karaboga. *Intelligent optimisation techniques: genetic algorithms, tabu search, simulated annealing and neural networks*. Springer Science & Business Media, 2012.

[6] J David Schaffer et al. "A study of control parameters affecting online performance of genetic algorithms for function optimization". In: *Proceedings of the 3rd international conference on genetic algorithms.* 1989, pp. 51–60.