|  |
| --- |
| [**Genetic Algorithms for Multi-knapsack balance problem**](https://www.cs.us.es/docencia/aulavirtual/mod/page/view.php?id=2608) |
| Main practical assignment |
| Guillermo Diz Gil  Carmen Mª Muñoz Pérez |
|  |

Tabla de contenido

[**Genetic algorithm** 2](#_Toc74962964)

[**Combining individuals** 2](#_Toc74962965)

[**Single point crossover** 2](#_Toc74962966)

[**Triple point crossover** 2](#_Toc74962967)

[**Uniform crossover** 2](#_Toc74962968)

[**Mutations in individuals** 2](#_Toc74962969)

[**Selection mechanisms** 3](#_Toc74962970)

[**Roulette wheel selection** 3](#_Toc74962971)

[**Tournament selection** 3](#_Toc74962972)

[**Multi-knapsack balance problem analysis** 3](#_Toc74962973)

[**Alternative 1** 4](#_Toc74962974)

[**Alternative 2** 4](#_Toc74962975)

[**Alternative 3** 4](#_Toc74962976)

[**Codification of the problem** 4](#_Toc74962977)

[**Genes** 4](#_Toc74962978)

[**Individual length** 4](#_Toc74962979)

[**Decode** 4](#_Toc74962980)

[**Fitness** 4](#_Toc74962981)

[**Type** 5](#_Toc74962982)

[**Bibliography** 5](#_Toc74962983)

### **Genetic algorithm**

### **Combining individuals**

There are different ways of obtaining chromosomes by combining other individuals. In all of them there will be two randomly chosen parents from which a new chromosome will arise.

It should be noted that normally when such combinations of two individuals are made, two offspring chromosomes are usually the result. In our case, it has been decided to obtain one offspring for each of the two parents. This is due to reasons such as less complexity in the implementation and the fact that it is easier to obtain a generation with an odd number of chromosomes.

The different types of crossovers that have been implemented will be detailed below:

### **Single point crossover**

A random crossover point will be chosen and applied to both parents. Thus, each parent will be divided into two parts. The new chromosome will be formed in the following way: all the genes before the crossover point are inherited from first parent, while all the genes after the crossover point are inherited from the second parent.

It should be noted that the chromosome length must be at least two genes long.

### **Triple point crossover**

Three random crossover points will be chosen and applied to the two parents. Thus, each parent will be split into four parts. The new chromosome will be formed in the following way: the even segments of genes will be inherited from the first parent, while the odd segments of genes will be inherited from the second parent.

It should be noted that the chromosome length must be at least six genes long.

### **Uniform crossover**

This time there is no crossover point. Thus, for each position in the offspring, it will be randomly chosen which parent inherits from, i.e, each gene is chosen randomly either from the first parent or from the second one.

It should be noted that the chromosome length can be whatever.

### **Mutations in individuals**

Mutation is an operator used to maintain some diversity between the chromosomes of one generation and the next. Thus, there will be a probability of mutation, i.e., the probability that the chromosomes in the population may change. When this mutation occurs, the new chromosome that is generated is randomly selected.

### **Selection mechanisms**

When performing such algorithms, there will be occasions when certain individuals must be chosen from a population.

For this purpose, there are several selection mechanisms that normally favour those chromosomes with better fitness, although there is always a random component.

### **Roulette wheel selection**

This selection method is part of fitness-proportional selection, which consists of random selection but giving a higher probability to chromosomes with higher fitness. So, each chromosome has a probability of being selected proportional to its fitness value, which is calculated using the given fitness function. Moreover, this selection method can only be used when we have a maximization problem.

In the roulette wheel selection, first we calculate for everyone in the population, its associated cumulative sum of the values of the fitness. After that, a list of random numbers between 1 and the total sum of values is generated. Finally, for each random number the first chromosome whose cumulative sum is greater than or equal to it is returned.

Regarding our implementation, it will not be allowed to use this selection method if it is a minimisation problem. It is still possible that there is some negative fitness or 0 in maximisation so that so it is important to say that non-positive fitness value will be assumed as 1.

It should be noted that a chromosome can be selected multiple times.

### **Tournament selection**

To select an individual with this method, k chromosomes will be chosen from the population and the best one (the one with the best fitness) will be selected. This process will be carried out until the desired number of chromosomes is achieved.

This method can be used in both maximisation and minimisation problems. Moreover, the higher the k, the higher the selection pressure.

It should be noted that a chromosome can be selected multiple times.

### **Multi-knapsack balance problem analysis**

**Statement** - *Given a list of items, where each item has a weight associated with it, and given a number n, the problem is to find a balanced distribution of the objects into n subsets, in such a way that they all have the same weight (or as similar as possible).*

Once the library for solving genetic algorithms has been built, we focus on modelling the multi-knapsack balance problem.

To do so, we first analyse the statement. Once analysed, several interpretations of the problem were considered. These interpretations will be presented in general terms below and we will comment on which one has been chosen.

### **Alternative 1**

There are n subsets, and each subset has a different maximum capacity. In addition, there is a list of items and not all of them must be chosen. Therefore, the aim is to maximise the number of items that are distributed among all subsets, while minimising the difference in the total weight of each subset.

### **Alternative 2**

There are n subsets that have a maximum capacity, and it is the same for all. In addition, there is a list of items and not all of them must be chosen. Therefore, the aim is to maximise the number of items that are distributed among all subsets, while minimising the difference in the total weight of each subset.

### **Alternative 3**

There are n subsets that do not have a maximum capacity. In addition, there is a list of items, and they all must be distributed among the subsets. Therefore, it is intended to minimise the difference in the total weight of each subset.

Initially, when analysing the problem, it was thought that not all items should be taken and that subsets had a maximum weight. But, first, we realised that all the items in the statement must be taken, since it tells you that they are distributed among the subsets and does not specify that any item can be left untaken. On the other hand, it does not specify any restriction about a maximum capacity of the subsets. Furthermore, if there were a maximum capacity, you would have to consider whether all the items fit in the capacity of all the subsets. Therefore, it was determined that the problem would be modelled according to alternative 3.

### **Codification of the problem**

### **Genes**

1,2, 3, … X (with X being the number of knapsacks available)

### **Individual length**

Y (with Y being the number of objects available)

### **Decode**

Each gene at the position i of the chromosome represents the knapsack where the object i has been put into, so every object is put in only and just only one knapsack.

### **Fitness**

We calculate the distance between the total weight of each knapsack individually to the perfect average weight that a knapsack should have if all the objects were perfectly distributed along all the knapsacks, allowing us to minimize the difference between them.

### **Type**

MIN

### **Bibliography**

First, before starting the assignment, we review unit 5 and practice 4, both of which deal with genetic algorithms:

* <https://www.cs.us.es/docencia/aulavirtual/pluginfile.php/10022/mod_resource/content/1/unit-05-2020-21.pdf> (Slides from unit 5)
* Code use as practice 4

Once we started to implement both the library for the realisation of genetic algorithm problems and for the modelling of the problem itself, the official Python documentation has been consulted on several occasions as well as other sources:

* <https://docs.python.org/3/library/random.html>
* <https://github.com/microsoft/pylance-release/blob/main/DIAGNOSTIC_SEVERITY_RULES.md#diagnostic-severity-rules>
* <https://rico-schmidt.name/pymotw-3/collections/namedtuple.html>
* <https://stackoverflow.com/questions/18296755/python-max-function-using-key-and-lambda-expression>
* <https://stackoverflow.com/questions/1260792/import-a-file-from-a-subdirectory>
* <https://stackoverflow.com/questions/4142151/how-to-import-the-class-within-the-same-directory-or-sub-directory>
* <https://stackoverflow.com/questions/22842289/generate-n-unique-random-numbers-within-a-range>
* <https://stackoverflow.com/questions/14748910/generate-a-set-of-sorted-random-numbers-from-a-specific-range>
* <https://stackoverflow.com/questions/58792963/time-complexity-for-adding-elements-to-list-vs-set-in-python>
* <https://stackoverflow.com/questions/53422887/python-how-do-i-randomly-mix-two-lists/53423019>
* <https://stackoverflow.com/questions/7529376/pythonic-way-to-mix-two-lists>
* <https://stackoverflow.com/questions/47406741/disable-auto-wrap-long-line-in-visual-studio-code>
* <https://stackoverflow.com/questions/63314452/python-autopep8-formatting-not-working-with-max-line-length-parameter>
* <https://softwareengineering.stackexchange.com/questions/308972/python-file-naming-convention>
* <https://stackoverflow.com/questions/9195455/how-to-document-a-method-with-parameters>
* <https://stackoverflow.com/questions/58622/how-to-document-python-code-using-doxygen>