**AI - Report**

**Genetic algorithyms –**

**Knapsack Problem**

Author: Alejandro Manuel Gestoso Torres

Professor: Agustín Riscos Núñez

Subject: Artificial Inteligence

Index

[Introduction 2](#_Toc106143454)

[Implementation of the algorithm 3](#_Toc106143455)

[Representations 3](#_Toc106143456)

[Modifications for the multi-knapsack 5](#_Toc106143457)

[Instances considered 5](#_Toc106143458)

[Statistics 5](#_Toc106143459)

[Bibliography 5](#_Toc106143460)

# Introduction

The main goal of this assignment consists of develop a genetic algorithm, using python, to solve the knapsack problem and its variant, the multi-size knapsack problem.

The multi-size knapsack states that:

*Given a list of items L, where each item has a weight associated with it, the problem is to find a partition of the items into several subsets associated with multiple knapsacks, in such a way that the free space in the knapsacks is minimized.*

*We will assume that we can use a finite number of sizes for the knapsacks (the list of allowed sizes/capacities should be provided as input).*

*We will assume that we can use an arbitrary number of knapsacks of the same size.*

In this report I will write about how the standard basic algorithm was made and how was modified to fulfill the multi-size algorithm. After that, I will show the graphics about how effective the algorithm was, what was the best solution found and the statistics.

# Implementation of the algorithms

## Representations

To represent this problem, we must define the individuals of the population:

* Genes: Whether the item is on the knapsack or not, it is a value between [0,1]. In case of being in the knapsack, it will have a value of a 1, and in case of not being in it, it will have a 0.
* Positions: A list of weights of every item. For example, given a list of weights like [50,60,70], it means that the item with index 0 will have weight 50, the item with index 1 will have weight 60 and the item with index 2 will have weight 70.
* Population: Every possible solution, that must be evaluated with the fitness function to whether discard (or not) it, depending on how much value has.

And we must define the functions:

* Decode: It must be defined a function that receives a chromosome and interprets its data returning the possible solution, so the fitness function can evaluate it.
* Fitness: This function is used to evaluate how good is the answer proposed by the chromosome and giving a value which will be used afterwards in the tournament selection to pick the best chromosome. The function used is this one:

Texto, Carta

Descripción generada automáticamenteAs we can see, in this function, for every chromosome we obtain its weight and if it is bigger than the maximum weight allowed, we impose a penalty by multiplying its weight by 8000, this make that the answers that are not correct has a value of fitness so bigger that in the future tournament selection, they will always lose.

* Texto

  Descripción generada automáticamenteCrossover: This function receives an instance of the problem and a population of parents and breeds them to obtain new population to continue the experimentation. The function used is this one:

In this case, this function is given us by the teacher and ensures that given a list of parents, they breed a new population of children.

* Mutation: This function receives a population and a probability of mutation and whether changes or not the population according to that given probability. The used function is this one:

This function is given us by the teacher too.

* Tournament selection: This function receives a population, two numbers and condition and picks the number of better possible solutions considering its fitness and the condition, that can be maximum (max) or minimum (min) fitness. The function used is:

Texto

Descripción generada automáticamenteIn this case, the function was given by the teacher and about the two number I talked about upwards, the first one (n) is the number of winners of the tournament and the second one (k) is the number of best picks chosen between the population. It also has an auxiliary function named choose that picks the best possible solution that after that, will be selected as winners in the main function.

* New generation: This function uses all the functions described upwards combined to obtain a new complete generation. The function used is:

Una captura de pantalla de un celular con texto e imagen

Descripción generada automáticamente con confianza mediaThis function was given by the teacher and as we can see, is a combination of the previous ones, because, it makes a tournament selection, a crossover and a mutation and returns a brand-new generation.

* Genetic algorithm: This function uses all the previous functions to provide a solution to the problem.

Texto

Descripción generada automáticamenteThe function uses the populator to generate an initial population, calculates a threshold and ensures that is a number divisible by two, that will make that the number of parents always will be an even number, and if not, by subtracting one to the number, will be even. Then we run the new generation algorithm and return a possible solution and its fitness.

## Modifications for the multi-knapsack

To face the challenge to adapt the genetic algorithm to solve the multi-knapsack problem, many adjustments were considered, for example, creating two types of genes, one to say if the item were selected or not and other to locate to which knapsack was assigned. But the final idea was simply to use the genes to locate where goes every item, having 0 to the items that are not in any knapsack, and use the numbers 1 to x to express the knapsack where the item goes.

The modifications made to adapt the knapsack algorithm to the multi-knapsack are these:

* The decode function: The new decode function has to be more complex than 0,1 values if the item is in or not in the knapsack, so, in order to decode the chromosomes of the answer, we apply this function, having as a result a dictionary where the key is the knapsack that we are talking about and the values are a list of 1s and 0s if the number is in the answer or not.

Texto

Descripción generada automáticamente

* The decimal to binary function: To obtain a list of 1s and 0s to represent the number of the chromosomes that we will evaluate in the fitness function.

Texto

Descripción generada automáticamente

* The fitness function: The new fitness function needs an auxiliary one, that evaluates the fitness of every knapsack of the multi-knapsack. After that, the main function sums the fitness of every knapsack to an only number that summarizes the total fitness of the possible solution of the multi-knapsack.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* The main genetic algorithm function: The main algorithm is more or less the same, the only difference is that this function needs the number of knapsacks that we are considering in the problem.

Texto, Carta

Descripción generada automáticamente

## Instances considered

# Statistics

# Bibliography