**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG - HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**---o0o---**



REPORT PROJECT1

SEARCHING

**Subject : INTRODUCTION TO ARTIFICIAL INTELLIGENCE**

**Teacher : Nguyễn Ngọc Thảo**

**Class : 21CLC05**

**Name : Đoàn Gia Phú – 21127133**

**Huỳnh Minh Quang – 21127149**

**Phú Trường – 21127xxx**

**Nguyễn Xuân Lộc – 2112716xx**

*Thành phố Hồ Chí Minh, ngày 01 tháng 04 năm 2023*

**MỤC LỤC**

[MỤC LỤC 2](#_Toc131118248)

[COMPLETION ASSESSMENT TABLE 4](#_Toc131118249)

[ASSESSMENT PLAN TABLE 5](#_Toc131118250)

[I. Brute force searching 7](#_Toc131118251)

[II. Branch and bound 7](#_Toc131118252)

[III. Local beam search 7](#_Toc131118253)

[IV. Genetic algorithms 7](#_Toc131118254)

[1. Algorithm description 7](#_Toc131118255)

[2. Overall 10](#_Toc131118256)

[V. Tài liệu tham khảo 11](#_Toc131118257)

# COMPLETION ASSESSMENT TABLE

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Algorithms** | **Complete** | **Unfinished** |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 | Genetic | 100% complete |  |

# ASSESSMENT PLAN TABLE

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Name** | **Mission** | **Complete** |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 | Huỳnh Minh Quang | Code Genetic algorithms | 100% |

# Brute force searching

# Branch and bound

# Local beam search

# Genetic algorithms

## Algorithm description

* The genetic algorithm is implemented in the function Genetic\_Algorithms(). The function first defines several helper functions for selection, crossover, and mutation of the population of chromosomes, as well as a fitness function that calculates the fitness of each chromosome in the population.
* The genetic algorithm then proceeds as follows:
* **generate\_random(length)**
* A population of chromosomes is generated randomly.
* This function takes the length of the chromosome as input and generates a random binary array with the given length.

Text

Description automatically generated

* **generate\_population(population\_size, length)**
* This function takes the population size and the length of the chromosome as inputs. It generates a population of random chromosomes with the given size and length.

Text

Description automatically generated

* **fitness\_function(chromosome, w, w, v)**
* This function takes a chromosome (i.e., a binary array representing the items selected), the Knapsack's storage capacity, the weights of the classes, and the values of the classes as inputs. It returns the fitness value of the chromosome, which is the total value of the selected items if their total weight does not exceed the Knapsack's storage capacity, or zero otherwise.

Text

Description automatically generated

* **selection(population, fitness\_values, num\_parents)**
* The best parents are selected from the population using the selection operator.
* This function takes the population (i.e., a list of chromosomes), their corresponding fitness values, and the number of parents to select as inputs. It returns the parents with the best fitness values, which are selected using the tournament selection method.

A screenshot of a computer

Description automatically generated with medium confidence

* **crossover(parents, num\_offsprings)**
* Offsprings are generated by applying the crossover operator to the parents.
* This function takes the parents (i.e., a list of chromosomes) and the number of offsprings to generate as inputs. It returns the offsprings generated using the one-point crossover method.

Text

Description automatically generated

* **mutation(offsprings, mutation\_rate)**
* The offsprings are mutated using the mutation operator.
* This function takes the offsprings (i.e., a list of chromosomes) and the mutation rate as inputs. It returns the offsprings after performing the bit-flip mutation with the given mutation rate.

Text

Description automatically generated

* **genetic(W, m, w, v, c, num\_generations, population\_size, num\_parents, num\_offsprings, mutation\_rate)**
* This function takes the Knapsack's storage capacity, the number of classes, the weights of the classes, the values of the classes, the maximum number of items of each class that can be selected, the number of generations, the population size, the number of parents to select, the number of offsprings to generate, and the mutation rate as inputs. It performs the genetic algorithm with the given parameters and returns the maximum value of the selected items and the binary array representing the selected items.

A screenshot of a computer

Description automatically generated with medium confidence

* **Genetic\_Algorithms()**
* This function is the main function that calls the other functions to perform the genetic algorithm for the Knapsack problem. It sets the parameters for the genetic algorithm, reads the input file, performs the genetic algorithm, and writes the output file.
* Text

  Description automatically generated
* The genetic algorithm is run with the following parameters:
  + **num\_generations:** the number of generations to run the algorithm for.
  + **population\_size:** the size of the population of chromosomes.
  + **num\_parents:** the number of parents to select from the population in each generation.
  + **num\_offsprings:** the number of offsprings to generate by applying the crossover operator to the parents.
  + **mutation\_rate:** the probability of mutating each gene in an offspring.

## Overall

* The genetic algorithm is a powerful optimization technique that can be applied to a wide variety of problems, including the knapsack problem. However, it is important to carefully tune the algorithm parameters and evaluate its performance to ensure that it produces high-quality solutions in a reasonable amount of time.

# Tài liệu tham khảo

1. <https://viblo.asia/p/thuat-toan-di-truyen-ung-dung-giai-mot-so-bai-toan-kinh-dien-phan-1-RQqKLxJzK7z>
2. <https://websitehcm.com/genetic-algorithm-thuat-toan-di-truyen-trong-python/>

\_\_\_Hết\_\_\_