جامعة بيرزيت

BIRZEIT UNIVERSITY

Faculty of Engineering & Technology

Electrical & Computer Engineering Department

ARTIFICIAL INTELLIGENCE

ENCS3340

Project 1 Report

Prepared by:

Yousef Ghanem	1172333
Issam Al-ashhab	1172602

Instructor: Dr. Aziz Qaroush

Section: 1
Date: 12/12/2021

Abstract

In this report, we’re going to show how we formalized the problem of a Graduation projects distribution. Designed and implemented it in python code using AI’s genetics algorithm and got the best solution possible.

Table of Contents

Abstract 2
Problem Formalization 4
Design and Implementation 5
    Back-end implementation 5
    GUI and program testing 6
Conclusion 9
References 10

Problem Formalization

Our problem is (Graduation projects distribution). We got 36 groups of students and 36 projects to assign for them, every group got 3 selections of projects and every project has a supervisor, title and topic. Our goal is to assign the most of groups possible to their selected groups and the more the first selected project is the better (first selection is the most then second selection then the third).

As we using genetics algorithm to solve this problem the definition of the problem will be according to it.

The initial state of the problem is an empty population (the population contains a user-defined size of chromosomes, a chromosome consists of genes with length of number of groups, and values from 0 to the number of projects).

The operator is the next iteration (the iterations number is user-defined), it is done by generating a random population and making a generation on it by a number of times which is user-defined. In a generation we calculate the fitness in the population and sort it accordingly, then we take the best 2 chromosomes and save them to a new list, then take the best half of the population and do a crossover and mutation on it and repeat the operations on the next generation.

The goal test is by comparing the fitness of the current iteration to the best iteration yet and replacing it if it was better.

We can take the fitness as cost as we take the better fitness as a better solution.

Design and Implementation

Back-end implementation

We defined two objects (Chromosome and Population), the Chromosome parameters are (gene_length, genes, optimized and contents). genes are randomly filled of the length (gene_length) and the contents of range (contents). We also defined a method to calculate the fitness and optimized list (the gene is optimized ‘1’ if it equal one of the selections in the group of the same index or its topic is equal one of the selection’s topics, otherwise it’s not optimized ‘0’) in the current Chromosome. It returns a list with total fitness and fitness for each selection.

The Population parameters are (population_size and chromosomes). chromosomes is a list with (population_size) number of (Chromosome). We also defined a method which uses bubble sort to sort the current population’s chromosomes according to their fitness.

The main class consists of the start of the program, GUI, and the rest of the operations.

In the beginning, we start by defining some variables (best_iteration) that stores the best iteration yet, and (best_fitness) that has its fitness, (kill) to stop the program when needed.

We did a loop for the iterations which we create a new population with user-defined size and using it for generations. In a single generation we calculate the fitness of the current population to sort it accordingly, then take the best two chromosomes in the population (has the highest fitness) and store them in list called (best_fit).

After that we do a crossover which we take two chromosomes from the best half of the population (that has higher fitness) which we call them parents and assign a (crossover_point) which is taken randomly between 1 and the (gene_length) of a Chromosome. We then create 2 childs with no genes, then assign the genes of each parent to each of them from index: 0 to (crossover_point). After that we switch the parents for each child and apply the rest of the genes to it. If any of genes added after the (crossover_point) equals one that already in the child, we generate a different gene that doesn’t exist in the past genes and add it.

Before we add the childs to the new list (best_fit), we do a mutation on each child. It’s done by taking 2 random indexes of the child’s genes and switching them. And a way to improve the mutation we did condition where one of the genes at least should be not optimized.

After that we add both of the childs to (best_fit) and repeat the crossover until the (best_fit) has the same size as the population. And at last, we replace the population’s chromosomes by the ones stored in (best_fit), and repeat those operations as much as there are generations.

After the generations loop is done, we call it’s output an iteration. We use it to compare it to past iterations and find the best iteration (the one with best fitness). When the iterations loop is done, we will have the best solution that’s generated so far and it’s saved to a file named logs.

We added an option to optimize the solution, and it’s done by replacing the none-optimized genes by ones that have the same topic a selected projects.

GUI and program testing

In here, we’re talking about the program interface that’s shown to the user (look at Figure 1).

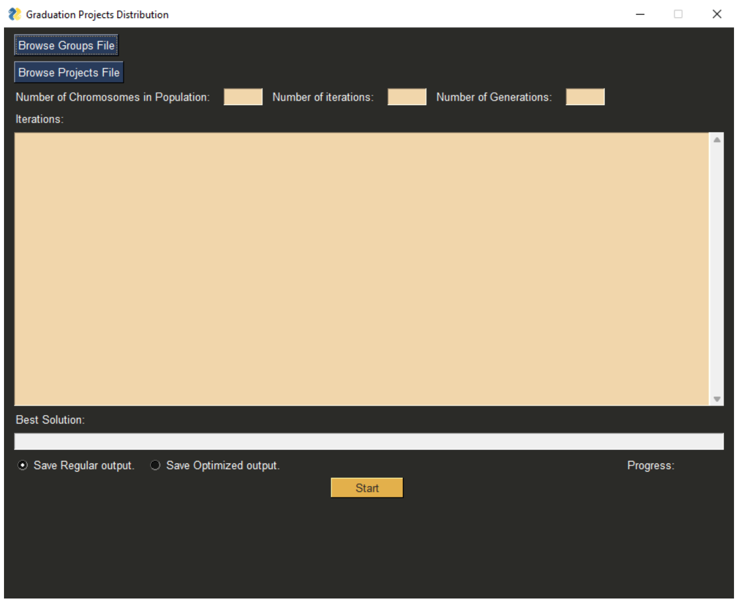
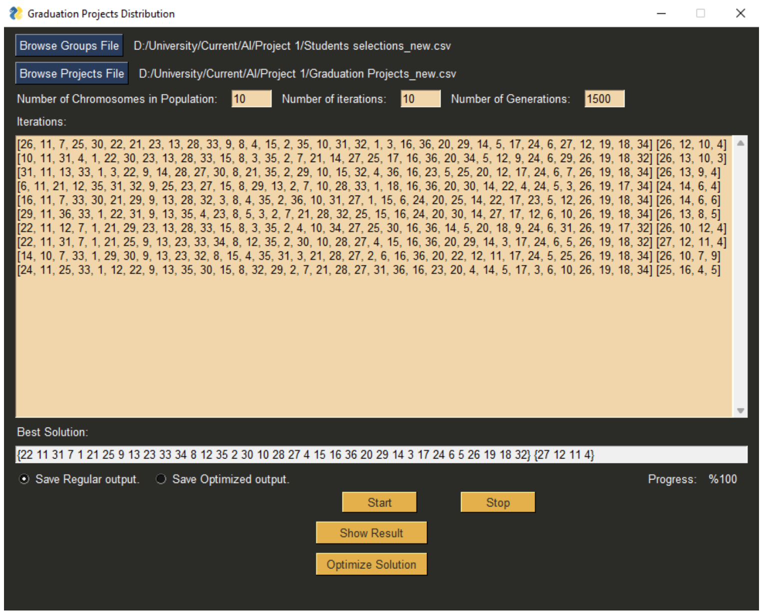


Figure 1

As shown in the figure above this the interface that is showed on the start of the program, we used (PySimpleGUI) python library to implement it. At first, we enter the groups and project csv files by browsing them, and then entering size of population, number of iterations and number of generations to be done.

As we click the start button all the operations we talked about in the previous section will start accordingly, and new buttons will appear (Stop, show result and optimize) look at Figure 2.

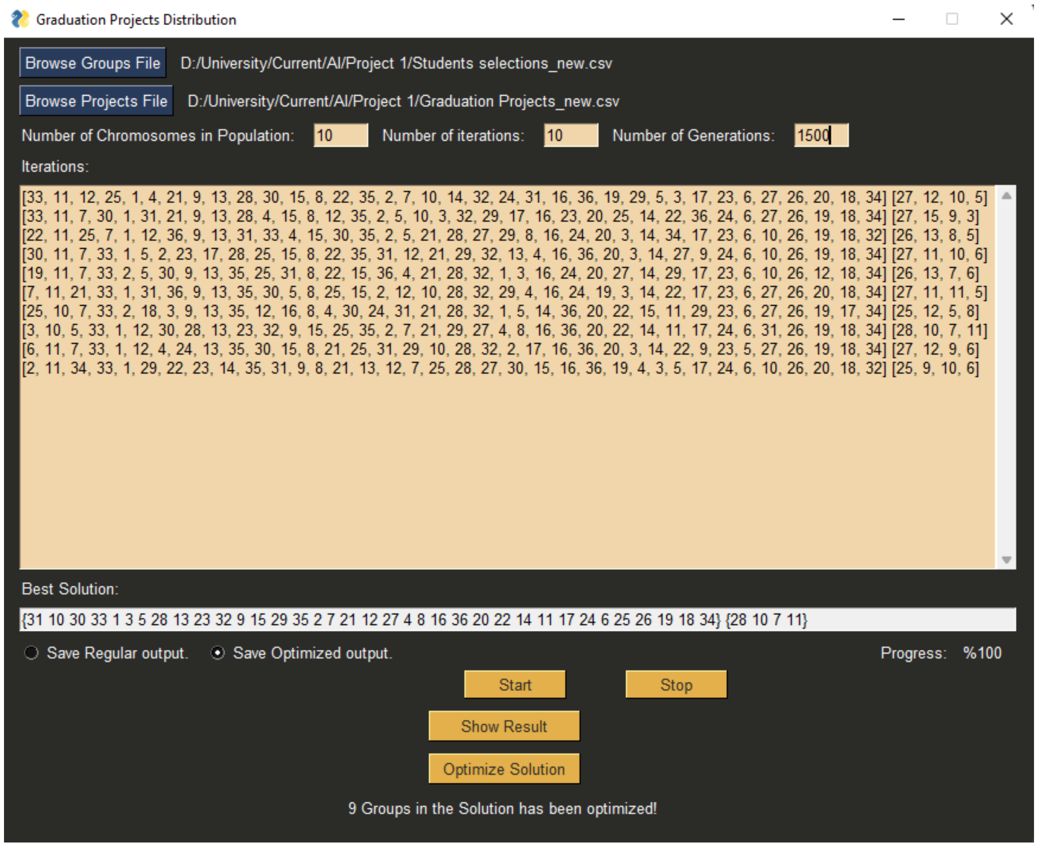


```
D:/University/Current/AI/Project 1/Students selections_new.csv
D:/University/Current/AI/Project 1/Graduation Projects_new.csv
Number of Chromosomes in Population: 10   Number of iterations: 10   Number of Generations: 1500
Iterations:
[26, 11, 7, 25, 30, 22, 21, 23, 13, 28, 33, 9, 8, 4, 15, 2, 35, 10, 31, 32, 1, 3, 16, 36, 20, 29, 14, 5, 17, 24, 6, 27, 12, 19, 18, 34] [26, 12, 10, 4]
[10, 11, 31, 4, 1, 22, 30, 23, 13, 28, 33, 15, 8, 3, 35, 2, 7, 21, 14, 27, 25, 17, 16, 36, 20, 34, 5, 12, 9, 24, 6, 29, 26, 19, 18, 32] [26, 13, 10, 3]
[31, 11, 13, 33, 1, 3, 22, 9, 14, 28, 27, 30, 8, 21, 35, 2, 29, 10, 15, 32, 4, 36, 16, 23, 5, 25, 20, 12, 17, 24, 6, 7, 26, 19, 18, 34] [26, 13, 9, 4]
[6, 11, 21, 12, 35, 31, 32, 9, 25, 23, 27, 15, 8, 29, 13, 2, 7, 10, 28, 33, 1, 18, 16, 36, 20, 30, 14, 22, 4, 24, 5, 3, 26, 19, 17, 34] [24, 14, 6, 4]
[16, 11, 7, 33, 30, 21, 29, 9, 13, 28, 32, 3, 8, 4, 35, 2, 36, 10, 31, 27, 1, 15, 6, 24, 20, 25, 14, 22, 17, 23, 5, 12, 26, 19, 18, 34] [26, 14, 6, 6]
[29, 11, 36, 33, 1, 22, 31, 9, 13, 35, 4, 23, 8, 5, 3, 2, 7, 21, 28, 32, 25, 15, 16, 24, 20, 30, 14, 27, 17, 12, 6, 10, 26, 19, 18, 34] [26, 13, 8, 5]
[22, 11, 12, 7, 1, 21, 29, 23, 13, 28, 33, 15, 8, 3, 35, 2, 4, 10, 34, 27, 25, 30, 16, 36, 14, 5, 20, 18, 9, 24, 6, 31, 26, 19, 17, 32] [26, 10, 12, 4]
[22, 11, 31, 7, 1, 21, 25, 9, 13, 23, 33, 34, 8, 12, 35, 2, 30, 10, 28, 27, 4, 15, 16, 36, 20, 29, 14, 3, 17, 24, 6, 5, 26, 19, 18, 32] [27, 12, 11, 4]
[14, 10, 7, 33, 1, 29, 30, 9, 13, 23, 32, 8, 15, 4, 35, 31, 3, 21, 28, 27, 2, 6, 16, 36, 20, 22, 12, 11, 17, 24, 5, 25, 26, 19, 18, 34] [26, 10, 7, 9]
[24, 11, 25, 33, 1, 12, 22, 9, 13, 35, 30, 15, 8, 32, 29, 2, 7, 21, 28, 27, 31, 36, 16, 23, 20, 4, 14, 5, 17, 3, 6, 10, 26, 19, 18, 34] [25, 16, 4, 5]
Best Solution:
{22 11 31 7 1 21 25 9 13 23 33 34 8 12 35 2 30 10 28 27 4 15 16 36 20 29 14 3 17 24 6 5 26 19 18 32} {27 12 11 4}
● Save Regular output.   ○ Save Optimized output.   Progress: %100
```

Figure 2

There are two radio buttons, which determines which solution get saved to the logs file. If first one is selected the regular output is saved and if the second one is selected the output after the optimization is saved.

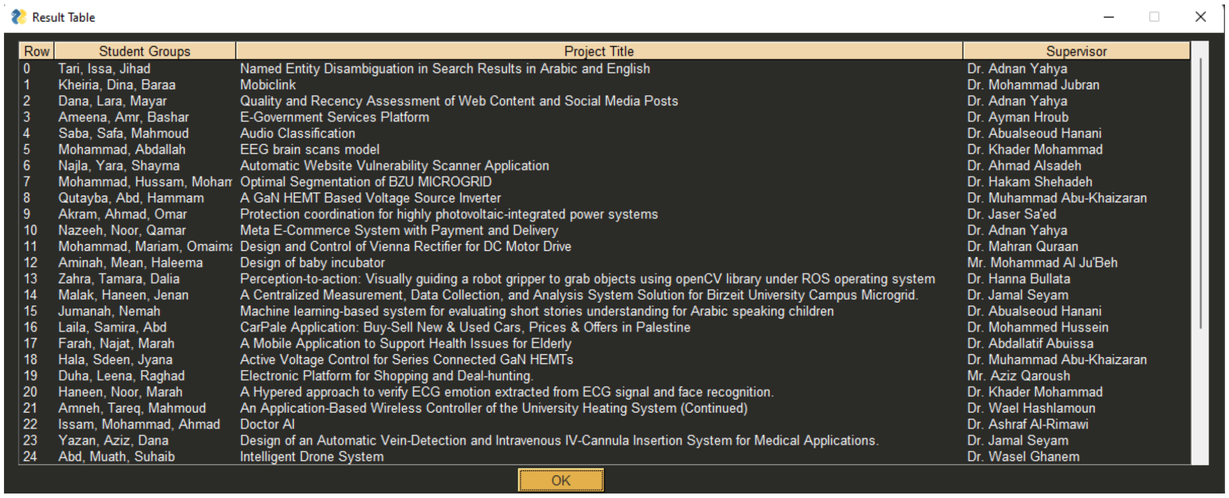
When optimizing solution, it prints how many groups got optimized, look at figure 3.



```
D:/University/Current/AI/Project 1/Students selections_new.csv
D:/University/Current/AI/Project 1/Graduation Projects_new.csv
Number of Chromosomes in Population: 10   Number of iterations: 10   Number of Generations: 1500
Iterations:
[33, 11, 12, 25, 1, 4, 21, 9, 13, 28, 30, 15, 8, 22, 35, 2, 7, 10, 14, 32, 24, 31, 16, 36, 19, 29, 5, 3, 17, 23, 6, 27, 26, 20, 18, 34] [27, 12, 10, 5]
[33, 11, 7, 30, 1, 31, 21, 9, 13, 28, 4, 15, 8, 12, 35, 2, 5, 10, 3, 32, 29, 17, 16, 23, 20, 25, 14, 22, 36, 24, 6, 27, 26, 19, 18, 34] [27, 15, 9, 3]
[22, 11, 25, 7, 1, 12, 36, 9, 13, 31, 33, 4, 15, 30, 35, 2, 5, 21, 28, 27, 29, 8, 16, 24, 20, 3, 14, 34, 17, 23, 6, 10, 26, 19, 18, 32] [26, 13, 8, 5]
[30, 11, 7, 33, 1, 5, 2, 23, 17, 28, 25, 15, 8, 22, 35, 31, 12, 21, 29, 32, 13, 4, 16, 36, 20, 3, 14, 27, 9, 24, 6, 10, 26, 19, 18, 34] [27, 11, 10, 6]
[19, 11, 7, 33, 2, 5, 30, 9, 13, 35, 25, 31, 8, 22, 15, 36, 4, 21, 28, 32, 1, 3, 16, 24, 20, 27, 14, 29, 17, 23, 6, 10, 26, 12, 18, 34] [26, 13, 7, 6]
[7, 11, 21, 33, 1, 31, 36, 9, 13, 35, 30, 5, 8, 25, 15, 2, 12, 10, 28, 32, 29, 4, 16, 24, 19, 3, 14, 22, 17, 23, 6, 27, 26, 20, 18, 34] [27, 11, 11, 5]
[25, 10, 7, 33, 2, 18, 3, 9, 13, 35, 12, 16, 8, 4, 30, 24, 31, 21, 28, 32, 1, 5, 14, 36, 20, 22, 15, 11, 29, 23, 6, 27, 26, 19, 17, 34] [25, 12, 5, 8]
[3, 10, 5, 33, 1, 12, 30, 28, 13, 23, 32, 9, 15, 25, 35, 2, 7, 21, 29, 27, 4, 8, 16, 36, 20, 22, 14, 11, 17, 24, 6, 31, 26, 19, 18, 34] [28, 10, 7, 11]
[6, 11, 7, 33, 1, 12, 4, 24, 13, 35, 30, 15, 8, 21, 25, 31, 29, 10, 28, 32, 2, 17, 16, 36, 20, 3, 14, 22, 9, 23, 5, 27, 26, 19, 18, 34] [27, 12, 9, 6]
[2, 11, 34, 33, 1, 29, 22, 23, 14, 35, 31, 9, 8, 21, 13, 12, 7, 25, 28, 27, 30, 15, 16, 36, 19, 4, 3, 5, 17, 24, 6, 10, 26, 20, 18, 32] [25, 9, 10, 6]
Best Solution:
{31 10 30 33 1 3 5 28 13 23 32 9 15 29 35 2 7 21 12 27 4 8 16 36 20 22 14 11 17 24 6 25 26 19 18 34} {28 10 7 11}
Save Regular output. [✓ Save Optimized output.]   Progress: %100
9 Groups in the Solution has been optimized!
```

Figure 3

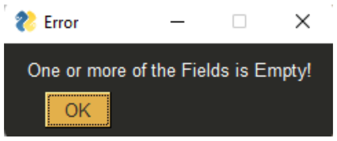
In the figure 4 below we can see a window generated to show the solution as a table.



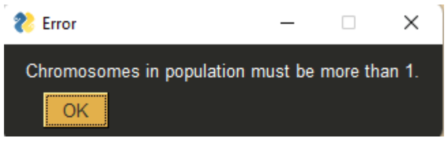
Row	Student Groups	Project Title	Supervisor
0	Tari, Issa, Jihad	Named Entity Disambiguation in Search Results in Arabic and English	Dr. Adnan Yahya
1	Kheiria, Dina, Baraa	Mobiclink	Dr. Mohammad Jubran
2	Dana, Lara, Mayar	Quality and Recency Assessment of Web Content and Social Media Posts	Dr. Adnan Yahya
3	Ameena, Amr, Bashar	E-Government Services Platform	Dr. Ayman Hroub
4	Saba, Safa, Mahmoud	Audio Classification	Dr. Abualseoud Hanani
5	Mohammad, Abdallah	EEG brain scans model	Dr. Khader Mohammad
6	Najla, Yara, Shayma	Automatic Website Vulnerability Scanner Application	Dr. Ahmad Alsadeh
7	Mohammad, Hussam, Mohan	Optimal Segmentation of BZU MICROGRID	Dr. Hakam Shehadeh
8	Qutayba, Abd, Hammam	A GaN HEMT Based Voltage Source Inverter	Dr. Muhammad Abu-Khaizaran
9	Akram, Ahmad, Omar	Protection coordination for highly photovoltaic-integrated power systems	Dr. Jaser Sa'ed
10	Nazeeh, Noor, Qamar	Meta E-Commerce System with Payment and Delivery	Dr. Adnan Yahya
11	Mohammad, Mariam, Omaima	Design and Control of Vienna Rectifier for DC Motor Drive	Dr. Mahran Quraan
12	Aminah, Mean, Haleema	Design of baby incubator	Mr. Mohammad Al Ju'Beh
13	Zahra, Tamara, Dalia	Perception-to-action: Visually guiding a robot gripper to grab objects using openCV library under ROS operating system	Dr. Hanna Bullata
14	Malak, Haneen, Jenan	A Centralized Measurement, Data Collection, and Analysis System Solution for Birzeit University Campus Microgrid.	Dr. Jamal Seyam
15	Jumanah, Nemah	Machine learning-based system for evaluating short stories understanding for Arabic speaking children	Dr. Abualseoud Hanani
16	Laila, Samira, Abd	CarPale Application: Buy-Sell New & Used Cars, Prices & Offers in Palestine	Dr. Mohammed Hussein
17	Farah, Najat, Marah	A Mobile Application to Support Health Issues for Elderly	Dr. Abdallatif Abuissa
18	Hala, Sdeen, Jyana	Active Voltage Control for Series Connected GaN HEMTs	Dr. Muhammad Abu-Khaizaran
19	Duha, Leena, Raghad	Electronic Platform for Shopping and Deal-hunting.	Mr. Aziz Qaroush
20	Haneen, Noor, Marah	A Hypered approach to verify ECG emotion extracted from ECG signal and face recognition.	Dr. Khader Mohammad
21	Amneh, Tareq, Mahmoud	An Application-Based Wireless Controller of the University Heating System (Continued)	Dr. Wael Hashlamoun
22	Issam, Mohammad, Ahmad	Doctor AI	Dr. Ashraf Al-Rimawi
23	Yazan, Aziz, Dana	Design of an Automatic Vein-Detection and Intravenous IV-Cannula Insertion System for Medical Applications.	Dr. Jamal Seyam
24	Abd, Muath, Suhaib	Intelligent Drone System	Dr. Wasel Ghanem

Figure 4

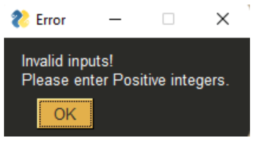
In the figures below are some popups you may encounter during to missing or invalid inputs.


Error

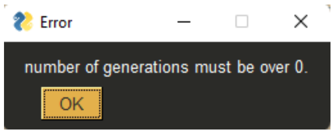
One or more of the Fields is Empty!


Error

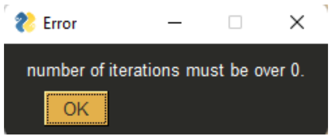
Chromosomes in population must be more than 1.


Error

Invalid inputs!
Please enter Positive integers.

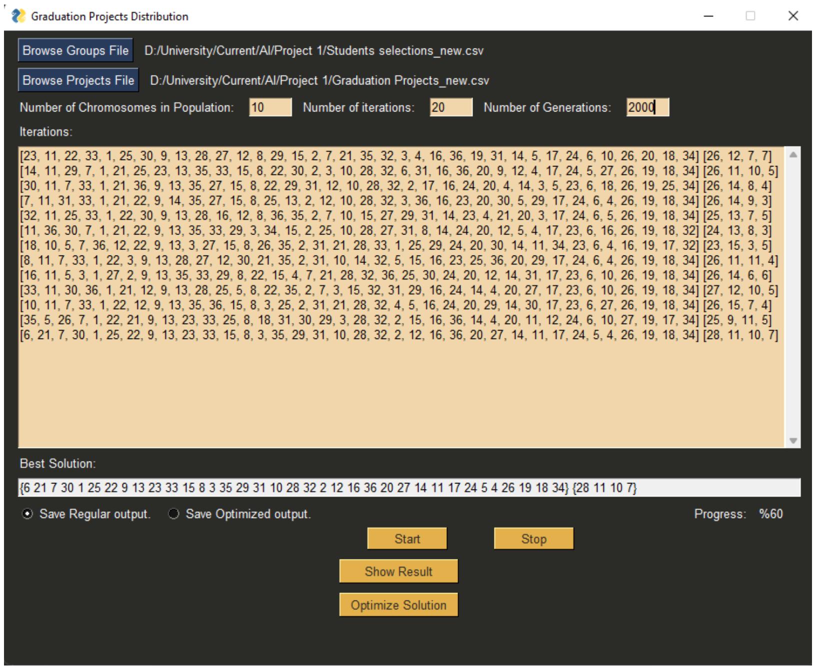

Error

number of generations must be over 0.


Error

number of iterations must be over 0.

In figure 5 below, the program is stopped at 60% of its progress.


Graduation Projects Distribution

D:/University/Current/AI/Project 1/Students selections_new.csv

D:/University/Current/AI/Project 1/Graduation Projects_new.csv

Number of Chromosomes in Population: 10 Number of iterations: 20 Number of Generations: 2000

Iterations:

```
[23, 11, 22, 33, 1, 25, 30, 9, 13, 28, 27, 12, 8, 29, 15, 2, 7, 21, 35, 32, 3, 4, 16, 36, 19, 31, 14, 5, 17, 24, 6, 10, 26, 20, 18, 34] [26, 12, 7, 7]
[14, 11, 29, 7, 1, 21, 25, 23, 13, 35, 33, 15, 8, 22, 30, 2, 3, 10, 28, 32, 6, 31, 16, 36, 20, 9, 12, 4, 17, 24, 5, 27, 26, 19, 18, 34] [26, 11, 10, 5]
[30, 11, 7, 33, 1, 21, 36, 9, 13, 35, 27, 15, 8, 22, 29, 31, 12, 10, 28, 32, 2, 17, 16, 24, 20, 4, 14, 3, 5, 23, 6, 18, 26, 19, 25, 34] [26, 14, 8, 4]
[7, 11, 31, 33, 1, 21, 22, 9, 14, 35, 27, 15, 8, 25, 13, 2, 12, 10, 28, 32, 3, 36, 16, 23, 20, 30, 5, 29, 17, 24, 6, 4, 26, 19, 18, 34] [26, 14, 9, 3]
[32, 11, 25, 33, 1, 22, 30, 9, 13, 28, 16, 12, 8, 36, 35, 2, 7, 10, 15, 27, 29, 31, 14, 23, 4, 21, 20, 3, 17, 24, 6, 5, 26, 19, 18, 34] [25, 13, 7, 5]
[11, 36, 30, 7, 1, 21, 22, 9, 13, 35, 33, 29, 3, 34, 15, 2, 25, 10, 28, 27, 31, 8, 14, 24, 20, 12, 5, 4, 17, 23, 6, 16, 26, 19, 18, 32] [24, 13, 8, 3]
[18, 10, 5, 7, 36, 12, 22, 9, 13, 3, 27, 15, 8, 26, 35, 2, 31, 21, 28, 33, 1, 25, 29, 24, 20, 30, 14, 11, 34, 23, 6, 4, 16, 19, 17, 32] [23, 15, 3, 5]
[8, 11, 7, 33, 1, 22, 3, 9, 13, 28, 27, 12, 30, 21, 35, 2, 31, 10, 14, 32, 5, 15, 16, 23, 25, 36, 20, 29, 17, 24, 6, 4, 26, 19, 18, 34] [26, 11, 11, 4]
[16, 11, 5, 3, 1, 27, 2, 9, 13, 35, 33, 29, 8, 22, 15, 4, 7, 21, 28, 32, 36, 25, 30, 24, 20, 12, 14, 31, 17, 23, 6, 10, 26, 19, 18, 34] [26, 14, 6, 6]
[33, 11, 30, 36, 1, 21, 12, 9, 13, 28, 25, 5, 8, 22, 35, 2, 7, 3, 15, 32, 31, 29, 16, 24, 14, 4, 20, 27, 17, 23, 6, 10, 26, 19, 18, 34] [27, 12, 10, 5]
[10, 11, 7, 33, 1, 22, 12, 9, 13, 35, 36, 15, 8, 3, 25, 2, 31, 21, 28, 32, 4, 5, 16, 24, 20, 29, 14, 30, 17, 23, 6, 27, 26, 19, 18, 34] [26, 15, 7, 4]
[35, 5, 26, 7, 1, 22, 21, 9, 13, 23, 33, 25, 8, 18, 31, 30, 29, 3, 28, 32, 2, 15, 16, 36, 14, 4, 20, 11, 12, 24, 6, 10, 27, 19, 17, 34] [25, 9, 11, 5]
[6, 21, 7, 30, 1, 25, 22, 9, 13, 23, 33, 15, 8, 3, 35, 29, 31, 10, 28, 32, 2, 12, 16, 36, 20, 27, 14, 11, 17, 24, 5, 4, 26, 19, 18, 34] [28, 11, 10, 7]
```

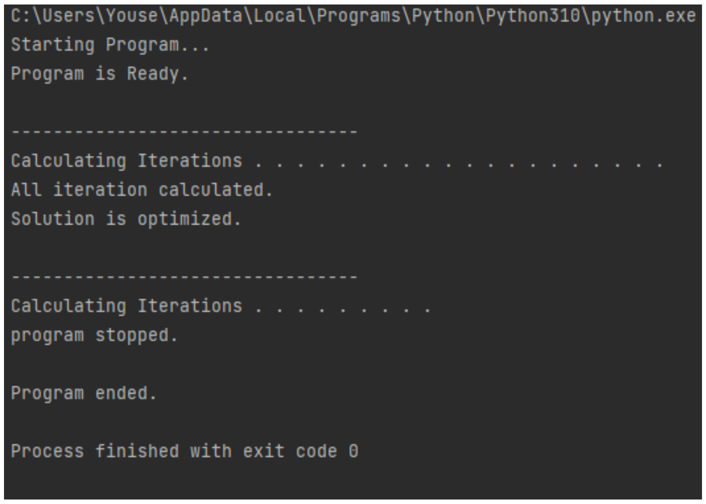
Best Solution:

{6 21 7 30 1 25 22 9 13 23 33 15 8 3 35 29 31 10 28 32 2 12 16 36 20 27 14 11 17 24 5 4 26 19 18 34} {28 11 10 7}

[✓ Save Regular output.] Save Optimized output. Progress: %60

Figure 5

In the console we print the current state of the program as shown in figure 6 below.


```
C:\Users\Youse\AppData\Local\Programs\Python\Python310\python.exe
Starting Program...
Program is Ready.

--------------------------------
Calculating Iterations . . . . . . . . . . . . . . . . . . . .
All iteration calculated.
Solution is optimized.

--------------------------------
Calculating Iterations . . . . . . . . .
program stopped.

Program ended.

Process finished with exit code 0
```

Figure 6

Conclusion

In this project, we worked on a real-life problem, used our knowledge in formalizing it and to try to get a solution by using genetics AI algorithm.

We learned how to implement the algorithm from our vision to benefit the most from its parameters and found new ways to get the best solution possible and optimizing it to be more satisfying.

We also tried new features that can be used in python such as threading and (PySimpleGUI).

References

1. https://www.youtube.com/watch?v=uQj5UNhCPuo
2. https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3
3. AI slides.