

Bagus Hendrawan

Dr. Jesse Lowe

CS 461 Introduction to AI

6 November 2023

Genetic Algorithm Assignment

Github link :

https://github.com/Baguser7/CS-461_Genetic-Algorithm

1. Source Code (Python)

Link :

https://drive.google.com/drive/folders/1ss7W0C2LxmfTUV9pzlrC_QOCAYIUME2R?usp=drive_link

2. Output (Generated text (.txt))

Link :

https://drive.google.com/drive/folders/1g_05izJ0kbHFBIATo0NAxlS1UCn6rYuB?usp=drive_link

Population number : 500

Generation : 100

Mutation rate : 0.01

Crossover rate : 0.5

I'm using softmax method to decide the crossover process, some of the case that I try:

- Output with regular mutation (100 Generation)
- Output with mutation halved ($m/2$) (100 Generation)
- Output with mutation halved again ($m/4$) (100 Generation)
- Output with regular mutation but (1000 Generation)
- Output with NO mutation (100 Generation)

3. Short Report

Link :

https://drive.google.com/drive/folders/1yJuLhrE9k5hLId5Jg5u7UHUMiIce3D2U?usp=drive_link

a. What were the challenges in writing the program? Or did it seem to go smoothly from the beginning?

- The main challenge of this program is how you build the fitness function, which can be quite complicated depends on the criteria of the goals itself
- Another issue that I noticed is that the result is not quite of what I expected, basically the program find the best way to “cheat”. To get

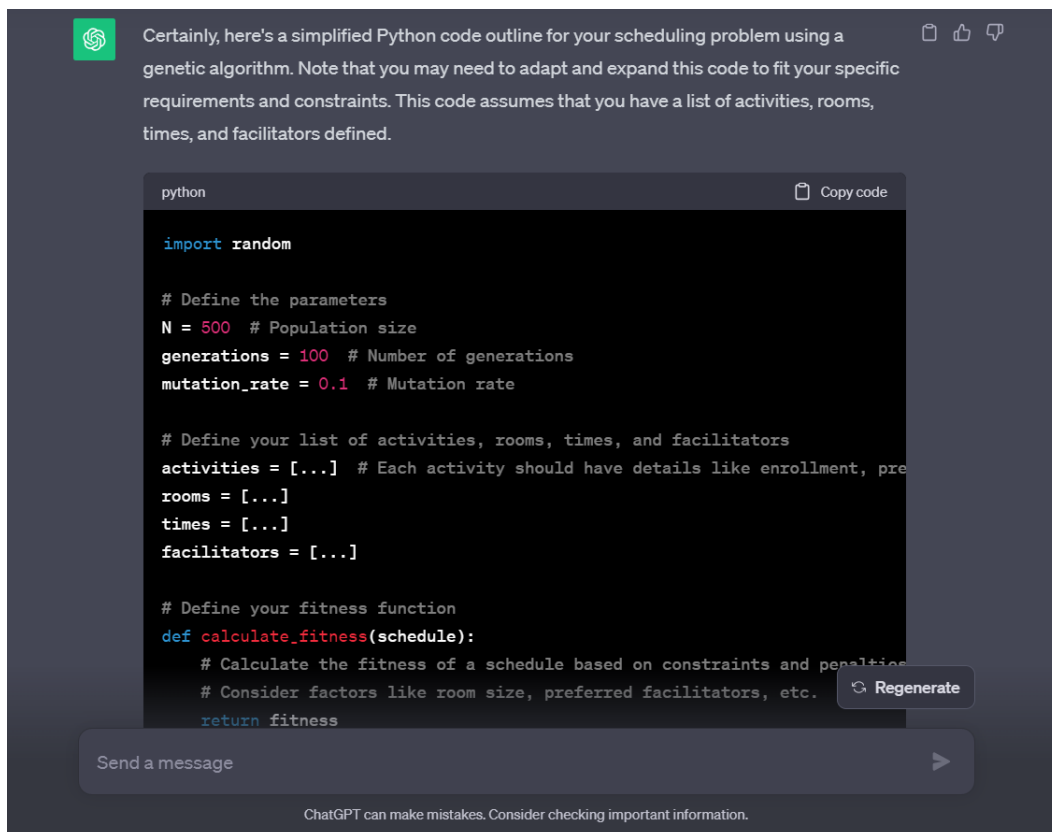
the best fitness number, which can lead to another problem if we not clearly defined the criteria.

- b. What do you think of the schedule your program produced? Does it have anything that still looks odd or out of place?
 - The result can be quite inconsistent (because of the stochastic approach), and as I say above, that the program try to maximize the fitness value by any means necessary (e.g find a loophole) which I noticed from the result.
 - The one thing that I notice is that they try to use the same room (and facilitator) quite frequently in different time schedule in order to maximize its value, which we must anticipated in much larger scale of program that this result can lead to another issue (underutilize, maintenance, fatigue etc) which we should refine in the fitness criteria.
 - Basically, they tend to use the “best” items or method over-and-over in order to maximize its fitness value, so the result given can be quite homogeneous.
- c. How would you improve the program, or change the fitness function?
 - I would say that I’d like to utilize the other room and facilitator that available, maybe give some penalty if using the same room or/and not using the facilitator available.
- d. Anything else you feel like discussing, asking about, bragging about, etc.
 - In my opinion, genetic algorithm really depends on how you define the fitness criteria, which we should be more careful for unexpected behaviour or result that we may create. which in more complex task it can be quite daunting, in regards how to define the fitness criteria and how you evaluate the result created.
 - Another things that I fascinated about, without the mutation (problems) we can be quite stuck on a plateau. Which is very interesting if we see it from philosophical perspective.

Reference :

- I have also use Chat-GPT to build this program, which is very impressive (and frightening) on how well the result it created. But sometime the result it provided is not goes the way we expected (especially in fitness function) which is they can misunderstand the prompt given and infer something that is not related to the prompt raised, it is quite understood because they can't really assume the situation and can only infer information from the prompt.

But it is impressive how they can improve with each prompt given.



The screenshot shows a ChatGPT chat window. At the top, the OpenAI logo is on the left, and icons for copy, like, and comment are on the right. The chat message reads: "Certainly, here's a simplified Python code outline for your scheduling problem using a genetic algorithm. Note that you may need to adapt and expand this code to fit your specific requirements and constraints. This code assumes that you have a list of activities, rooms, times, and facilitators defined." Below the text is a code block with a dark background and light-colored text. The code is a Python script for a genetic algorithm. It starts with an import statement for the 'random' module. Then, it defines several parameters: 'N' (Population size) set to 500, 'generations' set to 100, and 'mutation_rate' set to 0.1. Next, it defines lists for 'activities', 'rooms', 'times', and 'facilitators', each with a comment indicating they should contain details like enrollment, preferences, etc. Then, it defines a function 'calculate_fitness(schedule)' with a comment indicating it calculates fitness based on constraints and penalties, considering factors like room size and preferred facilitators. The function returns 'fitness'. To the right of the code block is a 'Copy code' button. Below the code block is a 'Regenerate' button. At the bottom of the chat window is a text input field with the placeholder 'Send a message' and a send button. A disclaimer at the very bottom states: 'ChatGPT can make mistakes. Consider checking important information.'

```
python Copy code
```

```
import random

# Define the parameters
N = 500 # Population size
generations = 100 # Number of generations
mutation_rate = 0.1 # Mutation rate

# Define your list of activities, rooms, times, and facilitators
activities = [...] # Each activity should have details like enrollment, pre
rooms = [...]
times = [...]
facilitators = [...]

# Define your fitness function
def calculate_fitness(schedule):
    # Calculate the fitness of a schedule based on constraints and penalties
    # Consider factors like room size, preferred facilitators, etc.
    return fitness
```

Regenerate

Send a message ➤

ChatGPT can make mistakes. Consider checking important information.



can you create the selected schedules function and new population function that apply softmax function to the fitness scores



Certainly, you can use a softmax function to select schedules and create a new population. The softmax function is often used to convert fitness scores into probabilities for selection. Here's how you can implement the selection and new population creation functions using the softmax function in Python:

```
python
import random
import math

# Assuming you have already defined the population and fitness_scores

def softmax_selection(population, fitness_scores, temperature=1.0):
    # Convert fitness scores to selection probabilities using the softmax function
    prob_weights = [math.exp(score / temperature) for score in fitness_scores]
    total_weight = sum(prob_weights)
    probabilities = [weight / total_weight for weight in prob_weights]

    # Select schedules based on the calculated probabilities
    selected_indices = random.choices(range(len(population)), weights=probabilities, k=len(population))
    selected_schedules = [population[i] for i in selected_indices]
```

Copy code

Regenerate

Send a message

ChatGPT can make mistakes. Consider checking important information.



can you create a crossover function from a given population for a genetic algorithm and also apply its mutation?



Certainly! To create a crossover function for a genetic algorithm and apply mutation, you typically perform these operations on pairs of individuals from the population. Here's a Python function that demonstrates both crossover and mutation on a given population:

```
python
import random

def crossover_population(population, crossover_rate):
    # Shuffle the population to ensure randomness
    random.shuffle(population)

    new_population = []

    for i in range(0, len(population), 2):
        if i + 1 < len(population):
            parent1 = population[i]
            parent2 = population[i + 1]

            if random.random() < crossover_rate:
```

Copy code

Regenerate

Send a message

ChatGPT can make mistakes. Consider checking important information.