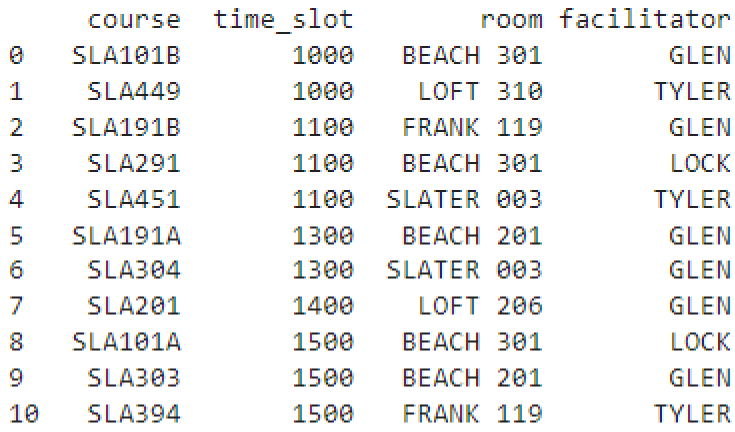
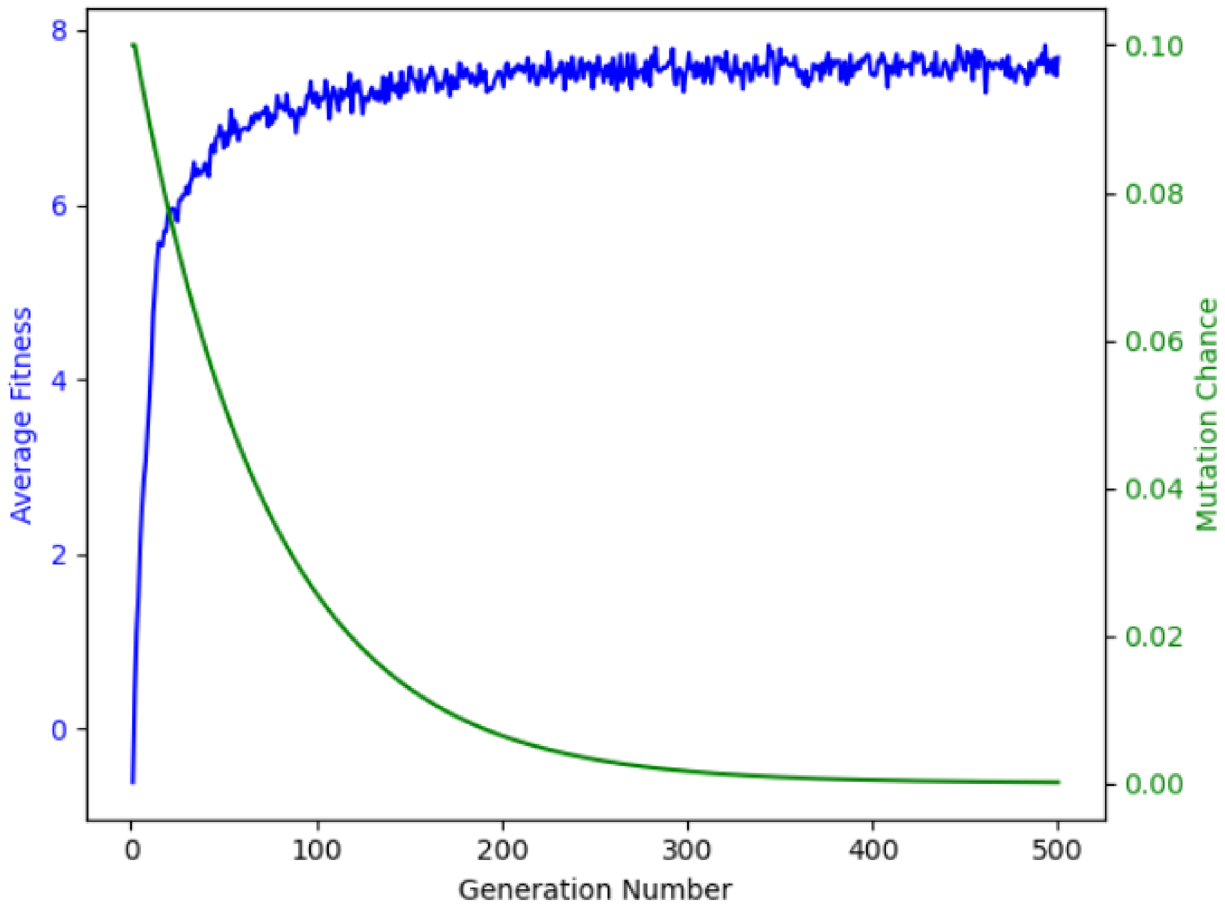
Final Schedule



```
      course  time_slot        room facilitator
0    SLA101B       1000   BEACH 301        GLEN
1     SLA449       1000    LOFT 310       TYLER
2    SLA191B       1100   FRANK 119        GLEN
3     SLA291       1100   BEACH 301        LOCK
4     SLA451       1100  SLATER 003       TYLER
5    SLA191A       1300   BEACH 201        GLEN
6     SLA304       1300  SLATER 003        GLEN
7     SLA201       1400    LOFT 206        GLEN
8    SLA101A       1500   BEACH 301        LOCK
9     SLA303       1500   BEACH 201        GLEN
10    SLA394       1500   FRANK 119       TYLER
```

Fitness over the generations



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

Creating the program to create chromosomes, the initial generation and subsequent generations went very easily. However, I tried to implement parallelism with a new library, asyncio, and it didn't work with jupyter notebooks. I've used a parallel backend for others, but I did not want to take the time to refactor the program considering 500 generations took about 20 minutes. However, I left the asyncio in place for use in local environments. Once I had the code running, I thought things were going great, but when I printed the schedule, it had assigned one instructor to every class. I was certain something was wrong with my function that evaluated fitness, but I eventually figured out that my interpretation of,

> *"If any facilitator is scheduled for consecutive time slots,*
> *same rules for SLA101 and SLA191 below."*

resulted in a reward of .5 for every consecutive time slot that outweighed the penalty for simultaneous classes and over 4 classes. I elected to only award the consecutive time slots for classes SLA101 and SLA191.

ii) What do you think of the schedule your program produced? Does it have anything that still looks odd or out of place?

I find the schedule my program produced acceptable. However, it would be better if it placed the 101 and 191 classes farther apart, but the reward for subsequent facilitators and penalty for over 4 classes makes this harder to evolve. I tried playing around with generation size, mutation rate, the mutation decay rate, and how many survivors there were between generations, but it never quite created a schedule with 101/191 in the first two hours and last two hours.

iii) How would you improve the program, or change the fitness function?

A blind genetic algorithm has uses, but I couldn't help but think that adding constraints would greatly improve efficiency. Simply not allowing the program to double book rooms or facilitators would allow the reward to focus on optimality. I considered just killing population members that violated this, but they are bred out relatively quickly. However, if children in each generation weren't allowed to break constraints, it would vastly improve the gene pool. I also felt the room constraints were too lenient.

iv) Anything else you feel like discussing, asking about, bragging about, etc.

I was satisfied with the intuitiveness of creating a genetic algorithm. My partner, a biology major, isn't normally that intrigued by computer science, but she thought this was really cool. It was interesting to see how conflicting rewards and punishments interplayed, even if it took me hours to figure out the algorithm was exploiting the heuristic against my intent. It was good to play with asyncio, but, unless I see a clear need, I'll probably stick with the parallelism libraries I'm familiar with.