Genetic Algorithm Report

The Genetic Algorithms assignment was a difficult but interesting assignment. I chose to work in Python, and utilized pandas due to previous experience and ease of use for dataframe objects. Initially, I kept track of each schedule as an instance of a “schedule” class, however I quickly gave up on this approach due to massive overhead and unnecessary complexity. By using pandas dataframes, I could keep track of a population of schedules by appending lists with randomly generated schedules, and then passing the lists iteratively through several other functions such as my fitness function. Another challenge I faced was developing the fitness function. Using pandas dataframes was a nice and clean option for the project in all aspects other than in the implementation of the fitness function. With this function being around 150 lines of code, it is a really ugly chunk of code that is also very time inefficient. Developing this wasn’t really conceptually difficult as all of the constraints were written in the assignment document, it was more so just time consuming. Eventually, I had deepseek refine the function which helped execution time by just a little. Lastly, the biggest challenge in writing this program was trying to avoid converging scores after ~30 generations. To fix this, I focused on tweaking my mutate() function and how I was utilizing my breeding function in my gen\_alg() function. In my mutate function, I added mutation\_rate as a variable taken by the function and have the rate increased each time the best fitness remained the same for 10 generations. For the breed() function as it was used in gen\_alg(), I initially performed breeding after mutating which did not logically make any sense as I should only be mutating the pool of children. After fixing this bug, I noticed immediate improvement.

From my evaluation, The final schedule produced appears to earn points on all possible point-earning fields. Something interesting is that it failed to realise that Tyler did not need to be assigned any activities, and consistently gave him assignments. Nothing really looks odd or out of place.

If I had created the fitness function myself, I think I would have made teachers being underloaded a larger point deduction, I am interested to see whether or not this would lead to more schedules that don’t include Tyler at all. In general, I would like to try other selection methods. I tried elitism and tournament style selection, and opted to use tournament style, however I think it would've been interesting to try roulette method, or something more random. I say this primarily because I consistently had the issue of converging top scores, and the only way I could work around it was after 15 generations of the same top score, completely randomizing the bottom 80 percent of the population. This actually worked really well, however it is far from elegant and somewhat defeats the purpose of maintaining a gene line generation to generation. I think the bulk of the issue here was rooted in not mutating aggressively enough early on, and being too strict on selection, leading to early tunneling toward local maxima.