Swetha Bhumireddy

Program 2 Submission

Data Structures

I used a numpy array to read in and store the data from the input file. I honestly primarily used this type over a regular array or something else because I got comfortable with using them in my CS 490 class for ICP assignments. It also seemed to be the best way to work with the data as a 2-dimensional array of the items' utility and weights. I ended up using a loop to iterate through the annealing. I made sure to have the loop break if an optimum option was reached that was within the weight limit. I then also used the general file i/o structure of python to be able to print the necessary information into an output file.

Running the Program

I had difficulty working through this problem, so I feel like there was a lot of additional work that I could add to it in future in order to make it a better program. I feel like I hadn't implemented the temperature aspect of the simulated annealing properly so ideally I would go back and do that better next time. Also, I think in terms of optimization, adding more functions (such as for the penalties, the temperature, etc.) to the program would have been better. Currently I only have one function that works to randomly select a portion of the items to include while running the simulated annealing. The content of this function I drew based off of some stackoverflow questions I was looking at in order to determine how best to randomly select a portion of a numpy array. Initially I was going to go down a route that suggested using the np.random.choice() function. However, I hit issues with this method when trying to incorporate it for use in a 2-dimensional array instead of a 1-dimensional array. This almost led to me giving up and considering just reading the inputs into a normal array and then running the randomization on that before throwing it into a numpy array. I then found a different approach which is what I ended up basing the function off of. This approach instead generated a random number of indices based on the shape of the numpy array and then grabbed those samples from the numpy array. Moving on, in future I would like to gain a better grasp of the ways through which some of the links I've listed below that I found while researching approached simulated annealing. I'd want to be able to implement some of their more detailed methods for this problem, including their visualizations of the simulated annealing by generating plots using the matplotlib library. I believe this would assist in providing more tools to gauge how well the simulated annealing is doing.

References

I used my notes from my CS 490 class. I also used the following links (listed below) when I was researching how to approach the simulated annealing problem. They provided me with a large range of

insight into how to work with this type of problem, however, some of them did appear more detailed than I needed. Apart from these links, there were a few references to code snippets I directly referenced in my submitted code, they were for the randomizing selected items function I mentioned previously, summing up the column of a numpy array (because I blanked and couldn't find that in my class notes), and for how best to output a numpy array to an output file. The links to these are listed directly where they are used in my code.

https://www.geeksforgeeks.org/simulated-annealing/

https://machinelearningmastery.com/simulated-annealing-from-scratch-in-python/

https://towardsdatascience.com/optimization-techniques-simulated-annealing-d6a4785a1de7

https://jamesmccaffrey.wordpress.com/2021/12/17/knapsack-problem-using-simulated-annealing-example/