project

October 19, 2023

0.1 GENERAL INFORMATION

This project is individual. Each project is different, even though they are all obviously related. Each student should work as independently as possible on their given project, without avoiding healthy discussions with other classmates. The evaluation of the project will consider various factors, namely:

- **Performance of the code**: By now you should know that some operations in Python are faster than others, so try to implement things most efficiently.
- **Explanation/order of the code**: Presenting messy and sloppy code is, first of all, a bad coding practice as it exponentially increases the chance of bugs. Also, it makes it hard for anyone trying to use your code to understand how to do so. Hence write clean code in which every important step is explained, as this will likely be rewarded from our side.
- **Explanation of the results you observe**: This is by far the most important factor for the grading. Once you have completed the coding architecture, we will ask some questions which will test your understanding of the problem and the results you are observing. Try answering those questions in a clear, careful, and detailed way. This will be a very important exercise also for you to truly gain some insights into some real-life optimization problems/situations.

Notice that in *no-point* above we claimed that finding the global minima in the optimization problem above will give extra points, as it is not the focus of this problem. So do not lose your mind on finding it, rather invest it into understanding why it is/isn't finding it.

HINT: I have thoroughly tested Chatbot GPT's ability to solve this problem, and I can ensure that you cannot count on it for anything. Actually, many statements are widely and wildly wrong as well as misleading, so I would highly suggest you to trust your intellect for this case.

0.2 PROBLEM DEFINITION

Select as *seed* value your ID(formatted as a string), and use it for the function **generate_data(n, seed)**, which needs as arguments a size for the problem n and, as mentioned, a seed. Looking at the code of this function will be of no use for doing well the project, so I would not waste time in back-engineering the data generation process: you might think this is smart, but it would be the exact opposite. The output will be a matrix C of dimension $[n \times n]$, where each entry $C[i, j]$ holds the value of your target function for inputs $x = i, y = j$. By this we mean

$$f(x = i, y = j) = C[i, j], \;\; i \in \{0, 1, 2, ...n - 1\}, \;\; j \in \{0, 1, 2, ..n - 1\}.$$

Hence the function f is defined on the discrete grid $\{0,1,2,..,n-1\} \times \{0,1,2...,n-1\}$, where here with $\times$ we imply the cartesian product of the set $\{0,1,2,..,n-1\}$ with himself. The goal is to try to optimize this function through *simulated annealing*. In particular, the goal is to analyze the performance/behavior of the algorithm with the size of the problem n.

For *simulated annealing* you can implement the following move proposal, i.e

$$x_{prop}|x^t = i = \text{random.choice}(\{[i-1]_n, [i+1]_n\}), \;\; y_{prop}|y^t = j = \text{random.choice}(\{[j-1]_n, [j+1]_n\}),$$

where with $[x]_n$ we indicate the *modulo-n* operation (and with x^t we indicate the value of x at the t-th iteration of the algorithm). Don't freak out, this just means that if $i = 0$, then have $i-1 = -1 = n-1$, while if $i = n-1$ then $i+1 = n = 0$. It is just a convenient way to deal with the borders of our grid domain. Obviously the same explanation above holds for the j indexes. With random.choice we indicate the function that selects randomly elements from a set with equal probability. In our case, it has to select randomly from two elements, i.e. $\{[i-1]_n, [i+1]_n\}$. There is a function in numpy that does precisely this operation, and to no surprise, it is called np.random.choice.

Once the above proposal is implemented, then we can just accept/reject a move by looking at the values of the function at the source and proposed point, i.e.

$$p\Big((x^t, y^t) \to (x_{prop}, y_{prop})\Big) = min\left(1, exp^{-(C[x_{prop}, y_{prop}] - C[x^t, y^t])}\right).$$

0.3 TASKS TO COMPLETE

Based on this information complete the following tasks.

- Implement a class for the problem(like the one done in class for the **TSP** problem) following the above instructions and the suggestions given in the general information area. Then use the class of simulated annealing used during the classes.
- The modulo operation defined above does seem a little too complex for its use(basically taking care of the borders), can you explain why it might be convenient both from a mathematical and implementation point of view to allow such border jumping? If I don't allow it, what part of the code should I change and why?
- Study the performance of the algorithm as the temperature changes for a fixed value of n, does the annealing procedure help in finding the minima? (RK: the minima should be around -100 for any n)
- Study the performance of the algorithm under random initialization for some reasonable increasing dimensions for the problem(like $n = [100, 200, 500, 1000, 5000]$) in terms of convergence speed and eventual convergence success. What can you say about it? NB: Notice that to find the minimum we can just np.min(f).
- Record at every step of the algorithm the acceptance probability of the moves, what can you say about them? Can this give you some insight into the landscape you are trying to optimize?

- Plot the optimization for a fixed value of n. What can you say about this landscape? Can you connect it to the behavior you observe in the acceptance probabilities and to the behavior of simulated annealing at different temperatures?
- After having observed all the above, do you think simulated annealing is an adequate algorithm for the problem considered? Is it better than, for example, a greedy strategy? Think of what could be the best optimisation algorithm for such a problem. Explain in detail your answer by directly referencing results/observations obtained in the previous points.

0.4 Submitting the project

The project has to be uploaded to the appropriate folder in BlackBoard. You should upload a zipped folder with the following format naming: **surname_id**. For example in my case this would be **silva_3005788**.

Inside this folder, you have to put **EVERY** code you use for your project (hence include for example the simulated annealing code) and the txt/pdf files with the answers to the questions. **For the coding part I should be able to open the zipped folder and easily run the code**.

The deadlines for submitting the projects are the ones indicated by Professor Saglietti in class. I will begin correcting the projects after the different available deadlines, hence if you submit them before one of them don't expect the correction to start immediately.