MSSC 6000 — Scientific Computing

Spring 2023 Take-Home Final Exam

Assigned: May 5, 2022
Due: May 12, 2022, 11:59pm, on D2L

You must work completely on your own. You may not discuss any part of this exam with any other person except Dr. Pantone, including classmates, other students, ChatGPT, etc.

At the end of this document, you will find guidelines for which resources are acceptable to use and which are not.

Your work must be submitted on D2L by 11:59pm on Friday, May 12. Please submit one written document for all non-code answers, and separate Python for each code answer requested below.

I intend to liberally distribute bonus points for algorithms that make clever design choices, effectively use some of the advanced topics we discussed, obtain especially good solutions, etc!

1. Some engineers are designing a cylindrical tank to store gas under pressure. The design of the tank can be boiled down to four variables, d_1, d_2, r, and L, where d_1 and d_2 are the thickness of the shell of different parts of the tank, r is the radius, and L is the length. The engineers want to design the tank to be as cheap as possible to produce, while still being strong enough to safely store the gas.

   They have determined that the cost of producing the tank, which depends on the values of the four variables, is:

   $$c(d_1, d_2, r, L) = 0.6224 d_1 r L + 1.7781 d_2 r^2 + 3.1661 d_1^2 L + 19.84 d_1^2 r.$$

   Based on the specifications given by the client, the dimensions of the tank must satisfy

   $$0.0625 \leq d_1 \leq 1, \qquad 0.0625 \leq d_2 \leq 1, \qquad r \geq 10, \qquad L \leq 200.$$

   In order to make sure the cylinder can safely store the gas, the dimensions must also satisfy the inequalities

   $$\begin{aligned} d_1 &\geq 0.0193 r, \\ d_2 &\geq 0.00954 r, \\ \pi r^2 L + \frac{4}{3}\pi r^3 &\geq 1296000, \\ L &\leq 240. \end{aligned}$$

   You job in this exercise is to help the engineers determine optimal dimensions that minimize cost while still satisfying the constraints above.

   (a) Describe the search space for this problem. State whether it is continuous or discrete.

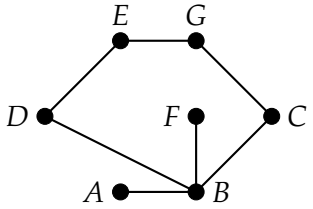
(b) In class we talked about hill-climbing and several variants of it. Select the one that you think would perform best on this problem, and justify your answer. Describe all details necessary to apply it to this problem (including, but not limited to: what is your tweak function, what are your stopping conditions, etc).

(c) Implement the hill-climbing variant that you chose in your previous answer.

Submit your code in a file called `[lastname]-cylinder.py`. It should have a function called `optimize` that, when called, performs the optimization in question and returns the optimal dimensions. It should run and return its answer in a few minutes, definitely no longer than five minutes.

(d) Describe what the Particle Swarm Optimization metaheuristic is and explain how you would apply it to this problem. What are some of the main differences between a PSO approach and a hill-climbing approach? You do not need to code anything for this part.

2. A graph is a set of vertices, connected in pairs by edges. An *independent set* in a graph is a subset of vertices S such that there are no edges connecting any of the vertices in S to each other. For example, in the graph below, the set $\{A, D, G\}$ is an independent set of size 3 because there are no edges connecting any of those three vertices to each other. The set $\{A, C, D, G\}$ is not an independent set because there is an edge between C and G.



Your goal in this question is to devise algorithms to determine the largest independent set in a given graph. In the graph above, the largest possible independent set size is 4, and there are several possibilities, e.g., $\{A, D, F, G\}$.

(a) Give an example of a real-world situation that can be modeled as a independent set maximization problem (it's okay if the real-world problem has other details / constraints involved).

(b) Describe the search space for this problem. If your input is a graph G with n vertices and k edges, what is the size of the search space?

(c) Invent and explain a greedy algorithm to try to find an independent set of maximal size. Then, state whether you think your greedy algorithm is optimal. If so, provide a brief justification; if not, provide a counterexample.

(d) Explain briefly how you could visualize the behavior of your algorithm and its solutions. What is the benefit of this?

(e) Describe how a branch-and-bound algorithm can be used to find the maximal independent set. Be sure to state clearly what your branching procedure is and what your bounding function is. (Is it an upper bound or a lower bound?)

(f) Choose one metaheuristic from the course, other than hill-climbing and its variants, and describe how you would use it to solve this problem. Justify why the metaheuristic you chose is appropriate, and be sure to explain each of the design decisions that have to be made for this particular metaheuristic.

(g) Implement your greedy algorithm and your metaheuristic algorithm.

The input format for a graph will be a list of two things: the first will be an integer representing the number of vertices in the graph, and the second will be a list of pairs of vertices representing the edges. For instance, the example graph in the in-person portion would be represented (using $A = 0$, $B = 1$, etc) as

```
input_graph = [7, [(0,1), (1,2), (1,3), (1,5), (2,6), (3,4), (4,6)]]
```

Submit a file called `[lastname]-ind-set.py` that has functions called `greedy` and `metaheuristic` that each take a graph as input (in the format above), and return a set of independent vertices as output that is as large as you can find. Compare the performance (the quality of solutions) of your two functions. I will run your two functions on several large graphs (up to 1000 vertices) to determine the quality of their solutions. Your greedy algorithm should run quickly (definitely under a minute), and your metaheuristic should run in no more than five minutes.

You may find it convenient to use an object-oriented approach to structure your code for this problem so that you can write several separate "helper functions" that can be reused, but it is not a requirement.

Take-Home Guidelines:
You may use:

- any lecture notes or lecture recordings from the course **this semester**.
- you may look at sample code from the course notes, but as explained below, you must write all of your own code
- the internet or other resources for specific questions about Python programming (for example, you may look for things like "How do I reverse a list?", but not "How do I implement backtracking?")

You may not use:

- any of the code that Dr. Pantone wrote in the course – all code for this exam must be written by you and only you
- any resources about the relevant class topics outside of lecture notes or lecture recordings
- any AI resources or coding assistants, including ChatGPT, Github Copilot, etc
- any other people – no discussion with classmates, other students, etc.

Please feel free to contact me with any questions about the exam or any of the guidelines above!