DSA/ISE 5113 Advanced Analytics and Metaheuristics Homework #4

Instructor: Charles Nicholson

Due: See course website for due date

Requirement details

1. This homeworks is to be completed individually.
2. Your primary submission will be a single Word or PDF document and must be of professional quality: clean, clear, concise, yet thoroughly responsive.
3. Any code (e.g., AMPL or Python) must also be submitted separately. Your code MUST be well-documented. Failure to submit the files will result in a penalty.
4. You cannot use pre-existing Python packages for heuristics or metaheuristics. In fact, other than `numpy, copy, random`, and other basic utilities, you should seek specific permission from the instructor. That is, *you* are responsible for creating the logic yourself.

Question 1: Facility Location (46 points)

Consider the Firehouse Facility Location Problem formulated in class:

- Assume that the population is concentrated in I districts within the city and that district i contains p_i people.
- Preliminary analysis (land surveys, politics, and so forth) has limited the potential location of firehouses to J sites.
- Let $d_{ij} \geq 0$ be the distance from the center of district i to site j.
- We are to determine the "best" site selection and assignment of districts to firehouses given a limited budget B.
- Every district should be assigned to exactly one firehouse
- No district should be assigned to an "unused" site
- It is important that either at least sites 1 and 2 or sites 3 and 4 are selected.
- There is a fixed cost to build a firehouse on a site; and a variable cost for servicing the population in that district
- Need to minimize the "worst-case" distance

Remember, the complete mathematical formulation for this problem was provided in the lecture!

(a) (10 points) Create a valid AMPL model file for this problem (include the model in your submission document as well as a separate attachment.)

(b) (36 points) You will monitor the basic branch-and-bound algorithm used to solve a large binary programming problem. The problem instance is provided in the file `firehouse-d2.dat`. To solve such a large instance, you will need a full version of AMPL and CPLEX. You may use the NEOS server (https://neos-server.org) to solve this instance.

Please see instructions on how to work with NEOS below. Additionally, you will need to disable some of the advanced heuristics in CPLEX see the basic branch-and-bound. Instructions are provided below.

i. (12 points) Solve the provided instance of the facility location problem. What is the optimal objective value? How much of the budget was used? What was the total solution time? How many branch-and-bound nodes were explored? What was the root relaxation value? What and when (node number) was the the first incumbent value found?

ii. (18 points) Create a graph of upper bound and lower bound objective values by node number (if there are too many nodes, you may collect data and plot values for every k nodes, where $k > 1$ according to the size of the node tree.)

iii. (6 points) Turn the cutting plane options on (mipcuts and splitcuts) and re-solve. Describe how this affects the enumeration tree and solution time.

Information regarding NEOS

The NEOS Server is a free internet-based service for solving numerical optimization problems. Hosted by the Wisconsin Institute for Discovery at the University of Wisconsin in Madison, the NEOS Server provides access to more than 60 state-of-the-art solvers in more than a dozen optimization categories. By using NEOS you are able to have access to full commercial solvers (not limited versions) and solve large problems in with AMPL and CPLEX.

Submitting a job to NEOS https://neos-server.org/neos/

The basic steps are as follows:

1. select "Submit a job to NEOS"
2. scroll down to select your problem type, solver, and input interface (e.g., Linear Programming, CPLEX, AMPL Input)
3. read the instructions on the page that comes up: you will submit a model, data, commands file. Do not specify "CPLEX" as the solver in any options commands. An example of the correct formatting is available on Canvas.
4. enter your email address and submit
5. note the job number and password so you can access your results; if the job returns quickly, you will see the results in a matter of moments; otherwise, you can wait to be emailed by NEOS.

AMPL and CPLEX options

To demonstrate the basic branch-and-bound approach, please disable some of the more advanced solution techniques used in CPLEX. That is, turn off presolve, all cutting plane methods mip- cuts and splitcuts, and the built-in heuristics; finally, set the MIP search to the basic method in the CPLEX options. To do so, use these options: `presolve=0 cliques=-1 mipcuts=-1 splitcuts=-1 heuristicfreq=-1 mipsearch=1`

You will also need turn on some display options to evaluate the solution time and to access node-level results. Use these options: `timing=1 mipdisplay=5 mipinterval=1 mipsearch=1`

You can read about these options and others here:
`https://ampl.com/products/solvers/solvers-we-sell/cplex/options/`

Example files with appropriate options and NEOS submission requirements are available in the course website in the homework listing.

For the following problems, you will modify some provided Python code to implement algorithms to solve an instance of the knapsack problem. After implementing all of the code and solving the problem, you must provide a single table of all results similar to the following:

Table 1: Example of results summary (numbers are not realistic)

Algorithm	Iterations	# Items Selected	Weight	Objective
Local Search (Best Improvement)	3102	49	97	117
Local Search (Random Restarts: 100)	951	121	21	147
Local Beam Search	2102	87	32	184
etc.				

Knapsack Problem Definition Given n different items, where each item i has an assigned value (v_i) and weight (w_i), select a combination of the items to maximize the total value without exceeding the weight limitations, W, of the knapsack.

IMPORTANT!: When generating random problem instance set $n = 150$ and use a seed value (for the random number generator) of 5113. The max weight is 1500.

Question 2: Strategies for the problem (8 points)

(a) (2 points) Define a strategy for determining an initial solution to this knapsack problem for a neighborhood-based heuristic.

(b) (4 points) Recommend 3 neighborhood structure definitions that you think would work well for this problem. What is the size of each neighborhood you recommend?

(c) (2 points) During evaluation of a candidate solution, it may be discovered to be infeasible. In this case, a strategy for handling infeasibility.

Question 3: Local Search with Best Improvement (12 points)

Complete the original Python Local Search code provided to implement *Hill Climbing with Best Improvement.* Note you will need to implement a strategy for determining an initial solution, handling infeasibility, etc.

Apply the technique to the random problem instance and determine the best solution and objective value using your revised algorithm.

Question 4: Local Search with First Improvement (8 points)

Modify the completed Python Local Search code to implement *Hill Climbing with First Improvement.* Apply the technique to the random problem instance and determine the best solution and objective value using your revised algorithm.

Question 5: Local Search with Random Restarts (14 points)

Modify the completed Python Local Search code to implement *Hill Climbing with Random Restarts.* You may use Best Improvement or First Improvement (just clearly state your choice). Make sure to include the following:

- Make the number of random restarts an easily modifiable variable.
- Keep track of the best solution found across all of the restarts.

Apply the technique to the random problem instance and determine the best solution and objective value using your revised algorithm.

Question 6: Local Search with Random Walk (12 points)

Modify the completed Python Local Search code to implement *Hill Climbing with Random Walk.* You may use Best Improvement or First Improvement (just clearly state your choice). Make sure to include the following:

- Make the probability of random walk an easily modifiable variable.

Apply the technique to the random problem instance and determine the best solution and objective value using your revised algorithm.

Question 7: Extra credit: Local Beam Search (8 points)

Modify the completed Python Local Search code to implement *Local Beam Search.* You may use Best Improvement or First Improvement (just clearly state your choice). Make sure to include the following:

- Make the number of parallel searches an easily modifiable variable.

Apply the technique to the random problem instance and determine the best solution and objective value using your revised algorithm.