ISE 5113 Advanced Analytics and Metaheuristics
Homework #6

Instructor: Charles Nicholson

Due: See course website for due date

Requirement details

1. Homeworks are to be completed in teams of 1, 2, or 3.
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. Note: For multiple problems, you are requested to include *excerpts* of relevant, commented Python code in your write-up file. Please do not copy and paste the entire code, but choose as little code as necessary to answer the question. Typically, this means no than a few lines of code each time.
4. Full code Python code files must also be submitted separately. Your code MUST be well-documented. Failure to submit the files will result in a penalty. For this problem you may submit separate code for each problem or submit a single file with multiple functions associated with each coding problem.
5. You cannot use pre-existing Python packages for heuristics or metaheuristics. In fact, other than `numpy, copy, random`, or other basic utilities, you should seek specific permission from the instructor if you have any doubt. That is, *you* are responsible for creating the logic yourself.

In this assignment for several problems you will modify some provided Python code to implement algorithms to solve the same 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 with Random Restarts ($k = 100$)	9510	121	21	147
Local Search with Random walk ($p = 0.25$)	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 you must use the code provided and values: $n = 150$; max weight of 2500; and, use a seed value (for the random number generator) of 51132023.

Question 1: Strategies for the problem (15 points)

(a) (5 points) Define and defend two strategies for determining an initial solution to this knapsack problem for a neighborhood-based heuristic. Copy and paste relevant excerpts of commented Python code to implement your preferred choice.

(b) (5 points) Describe two neighborhood structure definitions that you think would work well for this problem. Compute the size of each neighborhood.

(c) (5 points) During evaluation of a candidate solution, it may be discovered to be infeasible. In this case, provide two strategies for handling in feasibility. Copy and paste relevant excerpts of commented Python code for the handling infeasibility.

Question 2: Local Search with Best Improvement (10 points)

Complete the original Python Local Search code provided to implement *Hill Climbing with Best Improvement*. You will need to implement your strategy for determining an initial solution and handling infeasibility, etc., and complete the code as needed.

(a) (10 points) Apply the technique to the random problem instance and determine the best solution and objective value using your implemented algorithm.

Question 3: Local Search with First Improvement (14 points)

Modify the Python Local Search code to implement *Hill Climbing with First Improvement.*

(a) (10 points) Apply the technique to the random problem instance and determine the best solution and objective value using your implemented algorithm.

(b) (4 points) Using Python code excerpts, identify the logic differences between the code for Question 2 and Question 3.

Question 4: Local Search with Random Restarts (16 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, k, an easily modifiable parameter.
- Keep track of the best solution found across all of the restarts.

(a) (8 points) Apply the technique to the random problem instance and determine the best solution and objective value using your algorithm.

(b) (4 points) Report results for at least two different values of k.

(c) (4 points) Include relevant excerpts of commented Python code to implement the solution.

Question 5: Local Search with Random Walk (20 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, p, of random walk an easily modifiable parameter.

(a) (12 points) Apply the technique to the random problem instance and determine the best solution and objective value using your implemented algorithm.

(b) (4 points) Report results for at least two different values of p.

(c) (4 points) Include relevant excerpts of commented Python code to implement the solution.

Question 6: Stochastic Hill Climbing (25 points)

Modify the completed Python Local Search code to implement *Stochastic Hill Climbing*. You may use Best Improvement or First Improvement (just clearly state your choice).

(a) (15 points) Apply the technique to the random problem instance and determine the best solution and objective value using your implemented algorithm.

(b) (6 points) xplanation of how you determine probabilities for the stochastic solution choices.

(c) (4 points) Include relevant excerpts of commented Python code to implement the solution.