ISE 5113 Advanced Analytics and Metaheuristics
Homework #4

Instructor: Charles Nicholson

Due: See course website for due date

Requirement details

1. Homeworks are to be completed in teams of one or two. The team assignments are available on Canvas.
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., Python) 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.
4. You cannot use preexisting 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 (Random Restarts: 100)	9510	121	21	147
Local Beam Search (20 parallel searches)	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 51132021.

Question 1: Strategies for the problem (12 points)

(a) (3 points) Define and defend a strategy for determining an initial solution to this knapsack problem for a neighborhood-based heuristic.

(b) (5 points) Describe 3 neighborhood structure definitions that you think would work well for this problem. Compute the size of each neighborhood.

(c) (4 points) During evaluation of a candidate solution, it may be discovered to be infeasible. In this case, provide 2 strategies for handling in feasibility.

Question 2: Local Search with Best Improvement (12 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.

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 (10 points)

Modify the 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 algorithm.

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 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 algorithm. Report results for at least two different values of k.

Question 5: Local Search with Random Walk (16 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 variable.

Apply the technique to the random problem instance and determine the best solution and objective value using your algorithm. Report results for at least two different values of p.

Question 6: Local Beam Search (18 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 k an easily modifiable variable.

Apply the technique to the random problem instance and determine the best solution and objective value using your algorithm. Report results for at least two different values of k.

Question 7: Stochastic Hill Climbing *or* Stochastic Beam Search (16 points)

Your choice – either implement Stochastic Hill Climbing *or* Stochastic Beam Search. Make sure to include:

- Explanation of how you determine probabilities for the stochastic solution choices.

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

Note: You may implement and test *both* algorithms for 10 points of extra-credit.