ISE 5113 Advanced Analytics and Metaheuristics
Homework #4

Instructor: Charles Nicholson

See course website for due date

Requirement details

1. Submit all of your well-documented (e.g. commented) Python code.
2. Provide appropriate output of your code. Please no more than 1 page of output per problem.
3. You may work in teams of 2 for this problem. Teams will not be assigned, you can ask around. You may also work solo.
4. You cannot use available Python packages that do all of the work for you – you must code the logic to receive a grade!

In this assignment for several problems you will modify some provided Python code to implement heuristic 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:

Algorithm	Iterations	Objective
Local Search (Best Improvement)	3102	117
Local Search (First Improvement)	951	112
Local Search (Random Restarts)	9,681	147
Local Search (Random Restarts) + allowed infeasible solutions	14,412	194
Simulated Annealing	2102	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 $n = 100$ and use a seed value (for the random number generator) of 5113.

Question 1: Strategies for the problem (14 points)

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

(b) (3 points) Recommend 3 neighborhood structure definitions that you think would work well with the example knapsack problem in this assignment.

(c) (3 points) What is the size of each of the neighborhoods you recommended?

(d) (4 points) Identify 2 neighborhood structure definitions that you think would NOT work well with the example knapsack problem in this assignment. Explain why.

(e) (2 points) In the evaluation of a given solution, an infeasible may be discovered. In this case, provide 2 strategies for 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.* Note you will need to implement your strategy for determining an initial solution, handling infeasibility, and possible your neighborhood structures.

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

Question 3: Local Search with First Improvement (5 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 4: Local Search with Random Restarts (8 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 5: Local Search with Random Walk (8 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 6: Simulated Annealing (20 points)

Using the completed Python code as a base, implement *Simulated Annealing.* Make sure to include the following:

- Explanation of how you determined the initial temperature.
- Well-defined the temperature schedule (the temperature update procedure, the number of iterations performed at a given temperature, etc.)
- Explanation of the stopping criterion.

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

Question 7: Tabu Search or Variable Neighborhood Search (30 points)

Using the completed Python code as a base, implement **either** a *Tabu Search* or *Basic VNS* to solve the knapsack problem.

For TS, make sure to include the following:

- Explain the tabu criterion, tabu tenure, aspiration criterion, etc. and any other parameters you include.
- Long-term memory is optional.

For Basic VNS, make sure to include the following:

- You must define and use at least 3 different neighborhood structures.
- Define the local search component.

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

Question 8: Summary (5 points)

What are your thoughts regarding the performance of the neighborhood-based heuristics that you implemented? Which technique would you recommend for this problem? Did you try any variations (e.g., allowing infeasible moves, changing the initial solution strategy, different cooling processes, different stopping criteria, etc.?) If so, what seemed to be most effective?