Introduction to Analytics and Optimization Assignment 3 Part 2

In this assignment, you will have to implement a heuristic solution based on tabu search to the basic version of the knapsack problem. You can solve this task in pairs, or individually, the deadline is the 22th of June.

In the knapsack problem, there are n items, with associated values $(v_1, v_2, \ldots, v_n)$ and weights $(w_1, w_2, \ldots, w_n)$, and the value W specifying the capacity of the knapsack. The task is to choose the subset of the items with maximal total value such that the total weight is not more than W. A feasible solution can be represented in the form of $(x_1, x_2, \ldots, x_n)$, with $x_i = 0$ indicating that item i is not included in the knapsack and $x_i = 1$ indicating that item i is included in the knapsack. For example, if we have 4 items with values $(4, 5, 2, 7)$ and weights $(2, 3, 4, 6)$ and total capacity 9, than the solution $(0, 1, 0, 1)$ indicates that the second and fourth items are included in the knapsack, resulting in total value 12 and total weight 9; this is actually the optimal solution in this case.

In the assignment, you have to use the main ideas of the tabu search algorithm to find a solution to this problem. While you can make your own choices regarding different features of the implementation (neighbourhood, tabu list, penalty, stopping criterion, initialization), I recommend a set of choices:

- The algorithm should stop when a predefined number of iterations, T, is reached.
- The trivial feasible solution $(0, 0, \ldots, 0)$ can be used as the starting value of the algorithm. This will be the initial optimal solution, which is updated in each iteration when the identified solution is better than the present best solution.
- A possible move in finding a new solution in an iteration can be to change the assignment of a single item; either take it out or put it in the knapsack. This means that every solution has n number of neighbours (at least in the beginning, later the tabu list will limit this).
- The tabu list contains the items that have been moved (taken out or put in) in the previous specified number of iterations. This number, m (memory of the tabu list), can be any value from 1 to $n - 1$ (if it is at least n, you run out of possible moves after a while). For example if $m = 5$, and you take out (put in) item i, then for the next 5 iterations you cannot put it back (take it out).
- In every iteration, every neighbour that can be reached with a move that is not included in the tabu list should be evaluated by calculating first: (i) the total value of the items included in the knapsack $v = x_1 \times v_1 + x_2 \times v_2 \cdots + x_n \times v_n$, and (ii) the total weight of the included items $w = x_1 \times w_1 + x_2 \times w_2 \cdots + x_n \times w_n$. Then the evaluation of a solution is calculated according to the following formula:

  $$v - p \times \max(0, w - W),$$

  where p is a penalty multiplier. So if the evaluated solution satisfies the original constraint, i.e. the total weight is less than the capacity $w \leq W$, then the second term is 0, and there is no penalty, while a total weight greater than W would result in penalizing the total value of the included items. Finally, in each step of the iteration the neighbour with the highest evaluation value is chosen as new new starting point.

As a summary, the input to the implementations includes the value and weight of the items, the total capacity of the knapsack, the maximum number of iterations, the memory of the tabu list, and the penalty multiplier.

To evaluate your implementation, try it on the following data with 15 items (the maximal total value that can be reached is 1458):

- The value of the items: 135, 139, 149, 150, 156, 163, 173, 184, 192, 201, 210, 214, 221, 229, 240
- The weight of the items: 70, 73, 77, 80, 82, 87, 90, 94, 98, 106, 110, 113, 115, 118, 120
- The total capacity: 750