THE UNIVERSITY of ADELAIDE

INGENUITY CHALLENGE 2020

This challenge is the combination of two well-known problems: the Knapsack Problem and the Travelling Salesperson Problem.

Formally, we define this problem as follows. We are given a set of **n=433** locations with **x-y** coordinates, and each location has **10** items. As the starting and end location (with identifier “1”) does not have any items, this leaves us with a total of **4320** items.

Each item **k** is defined by a profit p_k and a weight w_k. We must visit all locations exactly once, pick up items, and return to the starting city. At each location we can pick up 0 to 10, but we can only obviously pick up each item once.

Our rented knapsack has a capacity limit of **W**, i.e. the total weight of the collected items must not exceed W. In addition, we consider a renting rate **R** that we must pay at the end of the treasure hunt, and the maximum and minimum velocities denoted v_{max} and v_{min} respectively.

A solution to our challenge is coded in two parts: the tour $X = (x_1, \ldots, x_n)$ is a vector containing the ordered list of locations 1 to 433, and the picking plan $Z = (z_1, \ldots, z_m)$ is a binary vector representing the states of items (1 for packed, 0 for unpacked).

What makes this problem challenging is that our speed changes according to the knapsack weight: our velocity at location **x** is defined as;

$$v_x = v_{max} - C * w_x$$

where

$$C = (v_{max} - v_{min})/W$$

is a constant value, and w_x is the weight of the knapsack at city **x**.

The total value of items is

$$g(Z) = \sum_m p_m * z_m,$$

such that

$$\sum_m w_m * z_m \leq W.$$

The total travel time is

$$f(X, Z) = \sum_{i=1}^{n-1} t(x_i, x_{i+1}) + t(x_n, x_1),$$

where

$$t(x_i, x_{i+1}) = d(x_i, x_{i+1}) / v_{x\_i}$$

is the travel time from x_i to x_{i+1}, **d** is the rounded-up (“ceil”) Euclidean distance between x_i, and x_{i+1}.

Our objective is to maximise our profit, which is the total profit of all items minus the travel time multiplied with the renting rate:

$$F(X, Z) = g(Z) - f(X, Z) * R.$$

To help you get started, we provide code in Java, C# and Matlab.

To submit solutions, you will need to create files containing the following information as comma-separated values in square brackets: the permutation of the cities in the first line (note: the first city is "1", do not print the "1" at the end), and the list of picked items in the second line (the numbering of the items starts with "1"). For example:

[1,5,4,3,2]

[21,313]

which means that only the items with the numbers 21 and 313 are picked, and the sequence of visited cities is <1,5,4,3,2,1>. Note that this format can easily be achieved in Java with the function Arrays.toString(...). The Java code provided below also contains a function to produce solutions files for you. For the code written in other languages, we are sure you can create solution files in the correct format.

- Java
- C#
- Matlab:
- Instance file