Matlab Project: Knapsack Problem

Matlab and its Applications in Experiments of Logistics Management

Learn by yourself on the 0-1 knapsack problem given below, and complete the Matlab exercise and submit the report on a group basis. Please note that you need to figure out on your own for the exact description of task, target, detailed plan, and conclusion. **Submit this report, together with the previous one on DELS project, before the noon on April 17, 2019.**

Given a set of items, each with a weight and a value, we want to select some items and pack them into a knapsack with limited total weight. How can we determine these items so as to maximize the total value? Here are the mathematical settings:

- collection of items $\{1, 2, \cdots, n\}$;
- item i has a value v_i and a weight w_i for each $i = 1, \cdots, n$;
- the maximum weight in the knapsack (i.e., capacity) is x;
- the objective is to maximize the total value of selected items.

On basis of the above settings, we can formulate the problem as below:

$$g(x) = \max_{y_i \in \{0,1\}, \forall i} \left\{ \sum_{i=1}^{n} v_i y_i : \text{such that} \sum_{i=1}^{n} w_i y_i \leq x \right\}, \tag{1}$$

where y_i indicates whether item i is selected ($y_i = 1$) or not ($y_i = 0$), and $g(x)$ denote the maximum value corresponding to capacity x. For simplicity, we assume that capacity x all weights w_i are non-negative integers. In the following, denote by $\boldsymbol{v} = [v_1, \cdots, v_n], \boldsymbol{w} = [w_1, \cdots, w_n]$ and corresponding problem Knpsk$(x, \boldsymbol{v}, \boldsymbol{w})$. What's the feasible set of g for a specific problem instance Knpsk$(x, \boldsymbol{v}, \boldsymbol{w})$?

Dynamic programming (DP) problem

The main idea is to inductively calculate $g(0), g(1), \cdots, g(x)$. Firstly observe that $g(0) = 0$, i.e., if the knapsack allows only 0 maximum weight, then clearly no item can be selected, and hence the total value is 0. For general x, we have that dynamic recursion as below:

$$g(x) = \max_{i: w_i \leq x} \{v_i + g(x - w_i)\}. \tag{2}$$

Then the desirable value $g(x)$ can be founded in an $O(nx)$ running time. Code in Matlab for the DP algorithm and verify the running time.

Heuristic: Greedy approximation algorithm

Sometime we prefer to a heuristic by the following the idea:

1. sort the items in decreasing order of value per unit of weight v_i/w_i as

$$v_1/w_1 \geq v_2/w_2 \geq \cdots \geq v_n/w_n.$$

2. select items starting from the one with highest v_i/w_i until the total weight exceeds x

$$\max_n \left\{ \sum_{i=1}^n v_i/w_i : \sum_{i=1}^n w_i \leq x \right\}.$$

This is a greedy approximation algorithm. Code in Matlab for this heuristic, and then develop a numerical study and report the performance. (how should we understand the term "performance"?)

Alternative DP problem

As a way other than (1) to build the DP recursion for this problem, we can take into consideration the index of each item. Specifically, sort all items by some rule such that the i^{th} item, denoted by *item* i, has the specific value v_i and weight w_i. Moreover, let $G(m, x)$ be the maximum value of selecting **the first** m item into a knapsack of capacity x. Then the DP problem is

$$G(m, x) = \max\{G(m-1, x), G(m-1, x-w_m) + v_m\}. \tag{3}$$

How should we understand the above DP formulation? What' the feasible set of function G? Is it the "same" as the one given by (2)? Code it in Matlab and be careful on its connection to (2), then compare the two DP formulae and report your understanding.