Fractional knapsack

Vassos Hadzilacos

A thief breaks into a store holding a knapsack that can carry up to a maximum weight $W > 0$. The store contains items $1, 2, \ldots, n$, where item t has value $v_t > 0$ and weight $w_t > 0$. The thief can steal some fraction x_t of item t, where $0 \leq x_t \leq 1$; the value of this fraction of item t is $x_t \cdot v_t$. The thief must decide what fraction of each item to steal, so as to maximize the total value of the stolen goods, subject to the constraint that their total weight must not exceed the knapsack's capacity W.

More precisely, a ***knapsack*** is a vector $S = (x_1, \ldots, x_n)$ that satisfies the following feasibility constraints: (a) for each item t, $0 \leq x_t \leq 1$, and (b) $\sum_{t=1}^{n} w_t x_t \leq W$ — i.e., the total weight of the (fractions of) stolen items does not exceed the capacity of the knapsack. The ***value*** of S, denoted $\mathcal{V}(S)$, is defined as $\mathcal{V}(S) = \sum_{t=1}^{n} v_t x_t$ — i.e., the total value of the (fractions of) stolen items. S is an ***optimal*** knapsack if it is a knapsack of maximum value; i.e., for all knapsacks S', $\mathcal{V}(S) \geq \mathcal{V}(S')$. The ***fractional knapsack problem*** can now be specified as follows:

INPUT. $W > 0$, and $v_t > 0, w_t > 0$ for each t, $1 \leq t \leq n$.

OUTPUT. An optimal knapsack.

The ***0-1 knapsack problem*** (also known as the ***discrete knapsack problem***) is the version of this problem where we require that $x_t = 1$ or $x_t = 0$, for each item t; in other words, the thief can either steal an item in its entirety or not at all; partial items have no value. This makes sense if the items are, say, shirts (half of a shirt has no value) as opposed to flour (half a kilo of flour has half the value of one kilo of flour). The 0-1 knapsack problem seems to be computationally very hard. It belongs to the famous class of ***NP-complete problems***,[1] which has the following property: (a) we don't know how to solve any NP-complete problem efficiently (i.e., in time that is a polynomial in the size of the input), and (b) if any one of these problems has an efficient solution, then they all do! The class of NP-complete problems includes many optimization problems that are important in practice.

In contrast to the 0-1 knapsack problem, the fractional knapsack problem can be solved by means of a simple and efficient greedy algorithm. Informally, the algorithm is as follows: Consider the items in decreasing value-to-weight ratio. Add whole items to the knapsack one at a time, in this order, until we reach an item whose addition would cause the knapsack's capacity W to be exceeded; add the largest fraction of that item that fits into the knapsack, and stop. This is expressed in pseudocode below:

```
1   sort the items so that v_1/w_1 ≥ v_2/w_2 ≥ ... ≥ v_n/w_n
2   s := 0 # total weight of stolen items in knapsack so far
3   k := 1 # next item to be considered
4   while k ≤ n and s + w_k ≤ W do
5       x_k := 1
6       s := s + w_k
7       k := k + 1
8   if k ≤ n then
9       x_k := (W − s)/w_k
10      for t = k + 1 to n do x_t := 0
11  return (x_1, ..., x_n)
```

[1] More precisely, the ***decision*** version of the 0-1 knapsack problem is NP-complete: Given the capacity W, the values and weights of the items, ***and a target value*** v, is there a knapsack whose total value is at least v? The output here is just "yes" or "no". What is defined above is the ***optimization*** version of the problem, which is obviously at least as hard.

It is straightforward to see that this algorithm can be implemented so that it runs in $O(n \log n)$ time. We now prove that the knapsack returned by this algorithm is optimal.

PROOF OF OPTIMALITY. Without loss of generality, assume that the items are indexed in the order in which they are sorted by the algorithm; thus,

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

If $\sum_{t=1}^{n} w_t \leq W$ (i.e., the knapsack can accommodate all items), then clearly the optimal knapsack is $(1, 1, \ldots, 1)$, and this is what the algorithm returns in this case. So, in the rest of the proof we assume that $\sum_{t=1}^{n} w_i > W$.

Lemma 1 *Let $S = (x_1, \ldots, x_n)$ be a knapsack, and i, j be items such that $i < j$, $x_i < 1$, and $x_j > 0$. Then there is a knapsack $S' = (x'_1, \ldots, x'_n)$ such that $\mathcal{V}(S') \geq \mathcal{V}(S)$, S' is identical to S in all items except i and j, and either $x'_i = 1$ or $x'_j = 0$.*

PROOF. Intuitively we construct S' by transferring some of the weight occupied by item j to item i, which is at least as valuable per unit of weight. There are two exhaustive cases, depending on whether we can take all of the weight occupied by item j and devote it to stealing more of item i (in which case $x'_j = 0$), or we can take only part of the weight occupied by item j and devote it to stealing all of the rest of item i (in which case $x'_i = 1$).

CASE 1. $w_j x_j \leq w_i(1 - x_i)$ (i.e., we can transfer all of the weight occupied by item j). Then let $S' = (x'_1, \ldots, x'_n)$ where

$$\begin{aligned} x'_t &= x_t, \quad \text{for all } t \neq i, j \\ x'_i &= x_i + (w_j/w_i)x_j \\ x'_j &= 0, \end{aligned}$$

First we verify that this is indeed a knapsack, i.e., it satisfies the two feasibility constraints (a) and (b). For (a), it is obvious that $0 \leq x'_t \leq 1$ for all $t \neq i$, and that $x'_i \geq 0$. Furthermore, by the definition of x'_i and the hypothesis of Case 1, we have

$$x'_i = x_i + (w_j/w_i)x_j \leq x_i + w_i(1 - x_i)/w_i = 1,$$

so $x'_i \leq 1$, as wanted. For (b), by the definition of x'_i and x'_j, we have

$$\sum_{1 \leq t \leq n} w_t x'_t = \left(\sum_{t \neq i,j} w_t x'_t \right) + w_i x'_i + w_j x'_j = \left(\sum_{t \neq i,j} w_t x_t \right) + w_i\big(x_i + (w_j/w_i)x_j\big) = \sum_{1 \leq t \leq n} w_t x_t \leq W,$$

as wanted. It remains to prove that $\mathcal{V}(S') \geq \mathcal{V}(S)$. Indeed, using the definition of x'_i and x'_j we have

$$\mathcal{V}(S') - \mathcal{V}(S) = v_i x'_i + v_j x'_j - v_i x_i - v_j x_j = v_i(x_i + \frac{w_j}{w_i} x_j - x_i) - v_j x_j = \frac{v_i}{w_i} w_j x_j - v_j x_j \geq 0$$

where the last inequality follows from the fact that $v_i/w_i \geq v_j/w_j$ (since $i < j$) by multiplying both sides by $w_j x_j$. Therefore $\mathcal{V}(S') \geq \mathcal{V}(S)$.

CASE 2. $w_j x_j > w_i(1 - x_i)$ (i.e., we can only transfer part of the weight occupied by item j before exceeding the remaining weight of item i). Then let $S' = (x'_1, \ldots, x'_n)$ where

$$\begin{aligned} x'_t &= x_t, \quad \text{for all } t \neq i, j \\ x'_i &= 1, \\ x'_j &= x_j + (w_i/w_j)(1 - x_i), \end{aligned}$$

Using algebra as in Case 1, we can verify that S' is a knapsack, and that $\mathcal{V}(S') \geq \mathcal{V}(S)$. □

The next lemma states something obvious: An optimal knapsack cannot have any unused capacity.

Lemma 2 *If $S = (x_1, \ldots, x_n)$ is an optimal knapsack then $\sum_{t=1}^{n} w_t x_t = W$.*

PROOF. We prove the contrapositive: Suppose $\sum_{t=1}^{n} w_t x_t \neq W$. So S has some surplus capacity $c = W - \sum_{t=1}^{n} w_t x_t > 0$. Since $\sum_{t=1}^{n} w_t > W$, it cannot be that $x_t = 1$ for all items t, so there is some item k such that $x_k < 1$. By increasing x_k to $x_k + \min(c/w_k, 1 - x_k)$ (i.e., using up the surplus capacity c to steal more of item k, up to stealing all of it) we obtain a knapsack of greater value that S. (Since $c > 0$ and $x_k < 1$, $\min(c/w_k, 1 - x_k) > 0$, so the fraction of item k is strictly increased.) So S is not optimal. □

Let us define an ***anomaly*** in a knapsack $(x_1, \ldots, x_n)$ to be a pair of items i and j such that $i < j$, $x_i < 1$ and $x_j > 0$. The next lemma states that the knapsack returned by the greedy algorithm has no anomalies and no unused capacity.

Lemma 3 *The knapsack $G = (x_1^g, \ldots, x_n^g)$ returned by the greedy algorithm satisfies the following:*
(a) There is some item k such that $x_t^g = 1$ for all items $t < k$, and $x_t^g = 0$ for all items $t > k$.
(b) $\sum_{t=1}^{n} w_t x_t^g = W$.

PROOF. From the algorithm's description it is clear that, for some k, the algorithm assigns $x_k = 1$ to the first $k - 1$ items in line 5, assigns some part (or all) of item k's weight to x_k in line 9, and assigns 0 to all remaining items in line 10. Thus, part (a) of the lemma holds. A straightforward induction shows that, at the end of the i-th iteration of the loop in lines 4–7, $s = \sum_{j=1}^{i} w_j$. Since, by assumption, $\sum_{i=1}^{n} w_i > W$, the algorithm exits the while loop with $i \leq n$. So, by the assignments in lines 9 and 10, $\sum_{i=1}^{n} w_i x_i = W$. □

There is only one knapsack with no anomalies and no unused capacity: if there were two different ones, the 1s in one would be a proper prefix of the 1s in the other, which implies that the first has unused capacity. By Lemma 3, that is precisely the knapsack G returned by the greedy algorithm. By Lemmas 1 and 2, there is an optimal knapsack with no anomalies and no unused capacity. But since there is only one such knapsack, namely G, it follows that G is optimal.