The Fractional Knapsack Problem

Jianwen Zhao

Department of Computer Science and Engineering
The Chinese University of Hong Kong

Fractional Knapsack Problem

Suppose there are n gold bricks, where the i-th gold brick g_i weighs $p_i > 0$ pounds and is worth $d_i > 0$ dollars. Given a knapsack with capacity $W > 0$, our goal is to put as much gold as possible into the knapsack such that the total value we can gain is maximized.

Different from the 0-1 Knapsack Problem (which has been discussed in the special exercise list 3), in this fractional variant, each gold brick is allowed to be broken into smaller pieces, i.e., we may take any fraction x_i ($0 \leq x_i \leq 1$) of g_i, then g_i will contribute the weight $p_i \cdot x_i$ to the total weight in the knapsack and the value $d_i \cdot x_i$ to the total value.

Fractional Knapsack Problem

More formally, let $g_i = (d_i, p_i)$ represent the value d_i and the weight p_i of g_i, x_i be the fraction taken from g_i, and W be the capacity of the knapsack. Our mission is to find a solution, denoted by a vector $X = (x_1, x_2, \cdots, x_n)$, to the following optimization problem:

$$\max \sum_{i=1}^{n} d_i \cdot x_i$$

$$s.t. \quad \sum_{i=1}^{n} p_i \cdot x_i \leq W$$

Example

Assume there are 4 gold bricks $\{(280, 40), (100, 10), (120, 20), (120, 24)\}$ and a knapsack with capacity 60. It can be verified that:

$X_1 = (1, 0, 1, 0)$ is a solution with

$$ToatalValue = 280 \times 1 + 100 \times 0 + 120 \times 1 + 120 \times 0 = 400$$
$$ToatalWeight = 40 \times 1 + 10 \times 0 + 20 \times 1 + 24 \times 0 = 60$$

And similarly, $X_2 = (1, 1, 0.5, 0)$ is another solution with

$$ToatalValue = 280 \times 1 + 100 \times 1 + 120 \times 0.5 + 120 \times 0 = 440$$
$$ToatalWeight = 40 \times 1 + 10 \times 1 + 20 \times 0.5 + 24 \times 0 = 60$$

Obviously, X_2 is preferred since it achieves higher value.

Algorithm

Next, we will present a simple greedy algorithm for solving the fractional knapsack problem.

Define $v_i = d_i/p_i$ to be the value-per-pound for the i-th gold brick g_i. The first step of our algorithm is to calculate v_i for all $i = 1, \cdots, n$ and sort all gold bricks by v_i in descending order.

For simplicity, let us assume $v_1 \geq v_2 \geq \cdots \geq v_n$, namely, the n gold bricks have already been sorted according to the value-per-pound.

Algorithm

The algorithm initializes $x_i = 0$ for all $i = 1, \cdots, n$ and then do the following:

1. **for** $i = 1$ **to** n
2. **if** $p_i \leq W$
3. $x_i = 1,\ W \leftarrow W - p_i$
4. **else**
5. $x_i = W/p_i$, **break**
6. **return** $X = (x_1 \ \cdots \ x_n)$

Example

Consider again the previous example with the set of gold bricks $S = \{(280, 40), (100, 10), (120, 20), (120, 24)\}$ and $W = 60$. After sorting by v_i, we have $S' = \{(100, 10), (280, 40), (120, 20), (120, 24)\}$.

Then the greedy algorithm runs as follows (based on S').

1. Since $p_1 = 10 < W$, set $x_1 = 1$ and $W = 60 - 10 = 50$.
2. Since $p_2 = 40 < W$, set $x_2 = 1$ and $W = 50 - 40 = 10$.
3. Since $p_3 = 20 > W$, set $x_3 = W/p_3 = 0.5$.
4. return $X = [1 \ \ 1 \ \ 0.5 \ \ 0]$ as the final solution.

Note that this solution gives total value 440 and total weight 60.

Analysis

Next, we prove that the greedy algorithm presented before gives an optimal solution to the fractional knapsack problem, namely, the solution that achieves the maximum value among all the possible choices.

Again, without loss of generality, for the n gold bricks, let us assume that $v_1 \geq v_2 \geq \cdots \geq v_n$.

Define T to be the set of gold pieces collected by an optimal solution, henceforth, instead of using X, for clarity, we will use T to denote an optimal solution.

Analysis

Lemma 1. There exists an optimal solution that selects exactly the same fraction x_1 $(0 < x_1 \leq 1)$ of g_1 as the greedy choice did.

Proof. Let T^* be an arbitrary optimal solution that does not select the same fraction of g_1 as the greedy choice did. We will turn T^* into another optimal solution T that selects the same fraction of g_1 as the greedy choice did and thereby finish the proof.

Suppose the greedy algorithm selects a fraction x_1 of g_1, which implies $W \geq p_1 \cdot x_1$. Since T^* does not take the greedy choice, it can only take less than x_1 fraction of g_1.

Analysis

Proof (cont.). Now, from T^*, we can take away some gold pieces that totally weigh $p_1 \cdot x_1$ pounds and replace them by the x_1 fraction of g_1, which does not violate the capacity requirement, and hence yields another solution T.

Since g_1 has the maximum value-per-pound, T is not worse than T^*. Therefore, T is another optimal solution that selects the same fraction x_1 of g_1 as the greedy choice did.

□

Analysis

Lemma 2. Let $S = \{g_1, \cdots, g_n\}$ be a set of n gold bricks satisfying $v_1 \geq v_2 \geq \cdots \geq v_n$, and $S' = S - \{g_1\}$. Given a capacity $W \geq p_1$, suppose T' is an optimal solution to the fractional knapsack problem on S' and $W - p_1$, then $T' \cup \{g_1\}$ is an optimal solution to the fractional knapsack problem on S and W.

Proof. We will prove by contradiction. Suppose that $T' \cup \{g_1\}$ is not an optimal solution to the fractional knapsack problem on S and W. By Lemma 1, there exists an optimal solution T to the fractional knapsack problem on S and W that selects g_1. Denote by $V(T)$ the total value achieved by the solution T, then

$$V(T' \cup \{g_1\}) < V(T)$$

Analysis

Proof (cont.). On the other hand, we know that all the other gold pieces in $T - \{g_1\}$ come from S' and since T' is an optimal solution to the fractional knapsack problem on S' and $W - p_1$, we have:

$$\begin{aligned} & V(T') \geq V(T - \{g_1\}) \\ \Rightarrow\ & V(T' \cup \{g_1\}) \geq V(T) \end{aligned}$$

and thus giving a contradiction. □

Analysis

Theorem. The greedy algorithm gives the optimal solution to the fractional knapsack problem.

Proof. We will prove by induction on the number n of the gold bricks.

Base Case. $n = 1$, the algorithm is obviously optimal.

Inductive Step. Assuming that the algorithm is correct for all $n \leq k$. We need to prove that it is also correct for $n = k + 1$, and this directly follows from Lemma 2.

□