Design and Analysis of Algorithms

Mohammad GANJTABESH

mgtabesh@ut.ac.ir

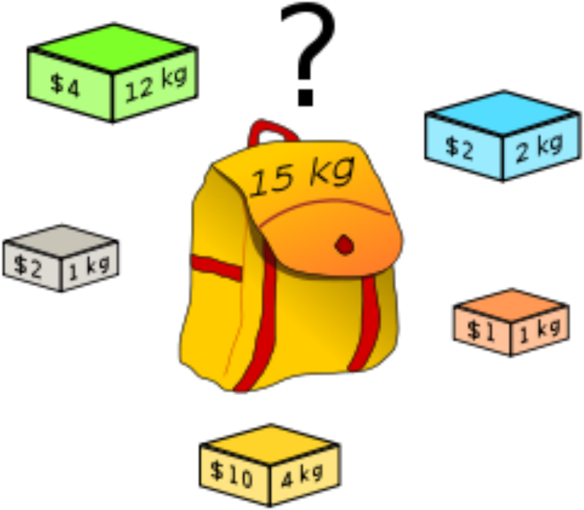
School of Mathematics, Statistics and Computer Science,
University of Tehran,
Tehran, Iran.

Techniques for the design of Algorithms

The classical techniques are as follows:

1. Divide and Conquer
2. Dynamic Programming
3. Greedy Algorithms
4. Backtracking Algorithms
5. Branch and Bound Algorithms

Knapsack Problem



Knapsack Problem

Definition

Suppose that we have n objects, say o_i ($i = 1, 2, \cdots, n$), each with corresponding weight (w_i) and profit (p_i), and a weight bound b. The goal of this problem is to find an $X = (x_1, x_2, \cdots, x_n)$ that maximize $\sum_{i=1}^{n} x_i p_i$ with respect to $\sum_{i=1}^{n} x_i w_i \leq b$.

- if $x_i \in \{0, 1\}$ the this problem is called 0/1-Knapsack.
- if $x_i \in [0, 1]$ the this problem is called fractional-Knapsack.

Knapsack Problem

Definition

Suppose that we have n objects, say o_i ($i = 1, 2, \cdots, n$), each with corresponding weight (w_i) and profit (p_i), and a weight bound b. The goal of this problem is to find an $X = (x_1, x_2, \cdots, x_n)$ that maximize $\sum_{i=1}^{n} x_i p_i$ with respect to $\sum_{i=1}^{n} x_i w_i \leq b$.

- if $x_i \in \{0, 1\}$ the this problem is called 0/1-Knapsack.
- if $x_i \in [0, 1]$ the this problem is called fractional-Knapsack.

For any optimization problem, we have two kinds of condition:

- Feasibility: ask whether a solution is feasible.
- Optimality: ask whether a solution is optimal.

0/1-Knapsack Problem: Dynamic Programming

Suppose that $P[i,j]$ denotes the maximum profit obtained by the first i objects where the sum of their weights is at most j. The goal is to calculate $P[n,b]$.

0/1-Knapsack Problem: Dynamic Programming

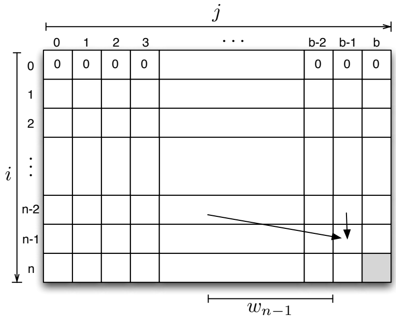
Suppose that $P[i,j]$ denotes the maximum profit obtained by the first i objects where the sum of their weights is at most j. The goal is to calculate $P[n,b]$. To do this, the following formula can be used:

$$P[i,j] = \begin{cases} -\infty & \textit{if } j < 0, \\ 0 & \textit{if } i = 0, \\ max\{P[i-1,j], P[i-1,j-w_i] + p_i\} & \textit{otherwise.} \end{cases}$$

0/1-Knapsack Problem: Dynamic Programming

Suppose that $P[i,j]$ denotes the maximum profit obtained by the first i objects where the sum of their weights is at most j. The goal is to calculate $P[n,b]$. To do this, the following formula can be used:

$$P[i,j] = \begin{cases} -\infty & \textit{if } j < 0, \\ 0 & \textit{if } i = 0, \\ max\{P[i-1,j], P[i-1,j-w_i] + p_i\} & \textit{otherwise.} \end{cases}$$



0/1-Knapsack Problem: Dynamic Programming

```
Dynamic-Programming-Knapsack(n, p[1···n], w[1···n], b){
    for j ← 0 to b do{
        P[0, j] ← 0;
    }
    for i ← 1 to n do{
        for j ← 0 to b do{
            if w_i ≤ j then
                P[i, j] ← max{P[i−1, j], P[i−1, j−w_i] + p_i};
            else
                P[i, j] ← P[i−1, j];
        }
    }
    return(P[n, b]);
}
```

Knapsack Problem

Definition (review)

Suppose that we have n objects, say o_i ($i = 1, 2, \cdots, n$), each with corresponding weight (w_i) and profit (p_i), and a weight bound b. The goal of this problem is to find an $X = (x_1, x_2, \cdots, x_n)$ that maximize $\sum_{i=1}^{n} x_i p_i$ with respect to $\sum_{i=1}^{n} x_i w_i \leq b$.

- if $x_i \in \{0, 1\}$ the this problem is called 0/1-Knapsack.
- if $x_i \in [0, 1]$ the this problem is called fractional-Knapsack.

Knapsack Problem

Definition (review)

Suppose that we have n objects, say o_i ($i = 1, 2, \cdots, n$), each with corresponding weight (w_i) and profit (p_i), and a weight bound b. The goal of this problem is to find an $X = (x_1, x_2, \cdots, x_n)$ that maximize $\sum_{i=1}^{n} x_i p_i$ with respect to $\sum_{i=1}^{n} x_i w_i \leq b$.

- if $x_i \in \{0, 1\}$ the this problem is called 0/1-Knapsack.
- if $x_i \in [0, 1]$ the this problem is called fractional-Knapsack.

For any optimization problem, we have two kinds of condition:

- Feasibility: ask whether a solution is feasible.
- Optimality: ask whether a solution is optimal.

Example of fractional-Knapsack Problem

Example

Suppose that $n = 3$, $b = 20$, $(p_1, p_2, p_3) = (25, 24, 15)$, and $(w_1, w_2, w_3) = (18, 15, 10)$. Then some of the feasible solutions are as follows:

Selection Strategy	x_1	x_2	x_3	$\sum_{i=1}^{3} x_i w_i$	$\sum_{i=1}^{3} x_i p_i$

Example of fractional-Knapsack Problem

Example

Suppose that $n = 3$, $b = 20$, $(p_1, p_2, p_3) = (25, 24, 15)$, and $(w_1, w_2, w_3) = (18, 15, 10)$. Then some of the feasible solutions are as follows:

Selection Strategy	x_1	x_2	x_3	$\sum_{i=1}^{3} x_i w_i$	$\sum_{i=1}^{3} x_i p_i$
Random	$1/2$	$1/3$	$1/4$	16.5	24.25

Example of fractional-Knapsack Problem

Example

Suppose that $n = 3$, $b = 20$, $(p_1, p_2, p_3) = (25, 24, 15)$, and $(w_1, w_2, w_3) = (18, 15, 10)$. Then some of the feasible solutions are as follows:

Selection Strategy	x_1	x_2	x_3	$\sum_{i=1}^{3} x_i w_i$	$\sum_{i=1}^{3} x_i p_i$
Random	$1/2$	$1/3$	$1/4$	16.5	24.25
maximum profit	1	$2/15$	0	20	28.2

Example of fractional-Knapsack Problem

Example

Suppose that $n = 3$, $b = 20$, $(p_1, p_2, p_3) = (25, 24, 15)$, and $(w_1, w_2, w_3) = (18, 15, 10)$. Then some of the feasible solutions are as follows:

Selection Strategy	x_1	x_2	x_3	$\sum_{i=1}^{3} x_i w_i$	$\sum_{i=1}^{3} x_i p_i$
Random	$1/2$	$1/3$	$1/4$	16.5	24.25
maximum profit	1	$2/15$	0	20	28.2
minimum weight	0	$2/3$	1	20	31

Example of fractional-Knapsack Problem

Example

Suppose that $n = 3$, $b = 20$, $(p_1, p_2, p_3) = (25, 24, 15)$, and $(w_1, w_2, w_3) = (18, 15, 10)$. Then some of the feasible solutions are as follows:

Selection Strategy	x_1	x_2	x_3	$\sum_{i=1}^{3} x_i w_i$	$\sum_{i=1}^{3} x_i p_i$
Random	$1/2$	$1/3$	$1/4$	16.5	24.25
maximum profit	1	$2/15$	0	20	28.2
minimum weight	0	$2/3$	1	20	31
maximum profit per unit	0	1	$1/2$	20	31.5

Greedy Algorithm for fractional-Knapsack Problem

Greedy-fractional-Knapsack-Algorithm$(n, b, p[1 \cdots n], w[1 \cdots n])\{$

  Sort objects in nondecreasing order with respect to $\frac{p_i}{w_i} \geq \frac{p_{i+1}}{w_{i+1}}$;

  $X \leftarrow 0$;

  $rw \leftarrow b$;

  for $i \leftarrow 1$ *to* n *do*$\{$

    if $w_i < rw$ then$\{$

      $x_i \leftarrow 1$;

      $rw \leftarrow rw - w_i$;

    $\}$

    else

      break;

  $\}$

  if $i < n$ then $x_i \leftarrow \frac{rw}{w_i}$;

  return(X);

$\}$

Greedy Algorithm for fractional-Knapsack Problem

Theorem

If $p_1/w_1 \geq p_2/w_2 \geq \cdots \geq p_n/w_n$ *then the procedure Greedy-fractional-Knapsack-Algorithm always return an optimal solution and its time complexity is* $O(n \log n)$*.*

Greedy Algorithm for fractional-Knapsack Problem

Theorem

If $p_1/w_1 \geq p_2/w_2 \geq \cdots \geq p_n/w_n$ *then the procedure Greedy-fractional-Knapsack-Algorithm always return an optimal solution and its time complexity is* $O(n \log n)$*.*

Proof.

It is obvious that the time complexity is $O(n \log n)$. The form of X is as follows:

$$X = [1, 1, \cdots, 1, x_j, 0, 0, \cdots, 0], \text{ where } 0 \leq x_j < 1$$

Greedy Algorithm for fractional-Knapsack Problem

Theorem

If $p_1/w_1 \geq p_2/w_2 \geq \cdots \geq p_n/w_n$ then the procedure Greedy-fractional-Knapsack-Algorithm always return an optimal solution and its time complexity is $O(n \log n)$.

Proof.

It is obvious that the time complexity is $O(n \log n)$. The form of X is as follows:

$$X = [1, 1, \cdots, 1, x_j, 0, 0, \cdots, 0], \text{ where } 0 \leq x_j < 1$$

Suppose that $Y = [y_1, y_2, \cdots y_n]$ be an optimal solution. Let k be the smallest index such that $x_k \neq y_k$. First we prove that $y_k \leq x_k$:

- $k < j$: in this case $x_k = 1$ and so $y_k \leq x_k$.
- $k = j$: since $\sum_{i=1}^{n} x_i w_i = b$ so $y_k \leq x_k$ (otherwise $\sum_{i=1}^{n} y_i w_i > b$).
- $k > j$: same as the previous case.

Greedy Algorithm for fractional-Knapsack Problem

Theorem

If $p_1/w_1 \geq p_2/w_2 \geq \cdots \geq p_n/w_n$ then the procedure Greedy-fractional-Knapsack-Algorithm always return an optimal solution and its time complexity is $O(n \log n)$.

Proof.

It is obvious that the time complexity is $O(n \log n)$. The form of X is as follows:

$$X = [1, 1, \cdots, 1, x_j, 0, 0, \cdots, 0], \text{ where } 0 \leq x_j < 1$$

Suppose that $Y = [y_1, y_2, \cdots y_n]$ be an optimal solution. Let k be the smallest index such that $x_k \neq y_k$. First we prove that $y_k \leq x_k$:

- $k < j$: in this case $x_k = 1$ and so $y_k \leq x_k$.
- $k = j$: since $\sum_{i=1}^{n} x_i w_i = b$ so $y_k \leq x_k$ (otherwise $\sum_{i=1}^{n} y_i w_i > b$).
- $k > j$: same as the previous case.

Increasing y_k to x_k produces another solution $Z = [z_1, z_2, \cdots z_n]$, where $z_k = x_k$ and

$$(z_k - y_k) w_k = \sum_{i=k+1}^{n} (y_i - z_i) w_i$$

Greedy Algorithm for fractional-Knapsack Problem

Proof (cont.)

$$\sum_{i=1}^{n} z_i p_i = \sum_{i=1}^{n} y_i p_i + (z_k - y_k) p_k - \sum_{i=k+1}^{n} (y_i - z_i) p_i$$

Greedy Algorithm for fractional-Knapsack Problem

Proof (cont.)

$$
\begin{aligned}
\sum_{i=1}^{n} z_i p_i &= \sum_{i=1}^{n} y_i p_i + (z_k - y_k) p_k - \sum_{i=k+1}^{n} (y_i - z_i) p_i \\
&= \sum_{i=1}^{n} y_i p_i + (z_k - y_k) p_k \frac{w_k}{w_k} - \sum_{i=k+1}^{n} (y_i - z_i) p_i \frac{w_i}{w_i}
\end{aligned}
$$

Greedy Algorithm for fractional-Knapsack Problem

Proof (cont.)

$$
\begin{aligned}
\sum_{i=1}^{n} z_i p_i &= \sum_{i=1}^{n} y_i p_i + (z_k - y_k) p_k - \sum_{i=k+1}^{n} (y_i - z_i) p_i \\
&= \sum_{i=1}^{n} y_i p_i + (z_k - y_k) p_k \frac{w_k}{w_k} - \sum_{i=k+1}^{n} (y_i - z_i) p_i \frac{w_i}{w_i} \\
&\geq \sum_{i=1}^{n} y_i p_i + \frac{p_k}{w_k} \left[(z_k - y_k) w_k - \sum_{i=k+1}^{n} (y_i - z_i) w_i \right]
\end{aligned}
$$

Greedy Algorithm for fractional-Knapsack Problem

Proof (cont.)

$$\begin{aligned}
\sum_{i=1}^{n} z_i p_i &= \sum_{i=1}^{n} y_i p_i + (z_k - y_k) p_k - \sum_{i=k+1}^{n} (y_i - z_i) p_i \\
&= \sum_{i=1}^{n} y_i p_i + (z_k - y_k) p_k \frac{w_k}{w_k} - \sum_{i=k+1}^{n} (y_i - z_i) p_i \frac{w_i}{w_i} \\
&\geq \sum_{i=1}^{n} y_i p_i + \frac{p_k}{w_k} \left[(z_k - y_k) w_k - \sum_{i=k+1}^{n} (y_i - z_i) w_i \right] \\
&\geq \sum_{i=1}^{n} y_i p_i
\end{aligned}$$

Since Y is an optimal solution, so we have $\sum_{i=1}^{n} z_i p_i = \sum_{i=1}^{n} y_i p_i$. We can do the same calculation for other indices and finally we obtain $X = Y$. $\square$

Exercises

1. Suppose that in a $0/1$-knapsack problem, the order of the items when sorted by increasing weight is the same as their order when sorted by decreasing value. Give an efficient algorithm to find an optimal solution to this variant of the knapsack problem, and argue that your algorithm is correct.
2. Describe an efficient algorithm that, given a set $\{x_1, x_2, \cdots, x_n\}$ of points on the real line, determines the smallest set of unit-length closed intervals that contains all of the given points. Argue that your algorithm is correct.
3. Suppose you are given two sets A and B, each containing n positive integers. You can choose to reorder each set however you like. After reordering, let a_i be the ith element of set A, and let b_i be the ith element of set B. You then receive a payoff of $\prod_{i=1}^{n} {a_i}^{b_i}$. Give an algorithm that will maximize your payoff. Prove that your algorithm maximizes the payoff, and state its running time.