Constrained Optimization Problems

A problem in which some function of certain variables (called the optimization or *objective function*) is to be optimized (usually minimized or maximized) subject to some *constraints*.

Types of solutions:

- **Feasible solution**: Any assignment of values to the variables that satisfies the given constraints.
- **Optimal solution**: A feasible solution that optimizes the objective function.

Greedy Algorithms

- At each step in the algorithm, one of several choices can be made.
- **Greedy Strategy**: make the choice that is the best at the moment.
- After making a choice, we are left with **one subproblem** to solve.
- The solution is created by making a sequence of **locally optimal** choices.

Greedy Algorithms: Optimality Conditions

Greedy Choice property:
A globally optimal solution can be arrived at by making a locally optimal (greedy) choice.

Optimal Substructure:
An optimal solution to the problem contains within it optimal solutions to subproblems.

Greedy Algorithms: Examples

- Prim's algorithm: Each step, include a new edge into the set A. Greedy criterion: select the **minimum-weight** edge connecting a vertex inside A and a vertex outside A (i.e., select a vertex that has smallest key value).
- Kruskal's algorithm: Each step, include a new edge into the set A. Greedy criterion: select the **minimum-weight** edge connecting two trees in A.
- Dijkstra's algorithm: Each step, include a new vertex into the set S. Greedy criterion: select the vertex with **smallest** d[u] value (i.e., the vertex that is closest to the source s).

Fractional Knapsack Problem

A thief considers stealing m pounds of merchandise. The loot is in the form of n items, each with weight w_i and value p_i. Any amount of an item can be put in the knapsack as long as the weight limit m is not exceeded.

Knapsack Problem: Formal Description

- Input: n objects and a knapsack.
- Each object i has a weight w_i, a value p_i and the knapsack has a capacity m.
- A fraction of object $x_i, 0 \le x_i \le 1$ yields a profit of $p_i \cdot x_i$.
- Objective is to obtain a filling that maximizes the profit, under the weight constraint of m.
- **Optimization Problem:** find $x_1, x_2, ..., x_n$, such that:

$$\begin{cases} \text{maximize:} & \sum_{i=1}^{n} p_i \cdot x_i \\ \text{subject to:} & \sum_{i=1}^{n} w_i \cdot x_i \le m \\ & \text{and } 0 \le x_i \le 1, 1 \le i \le n \end{cases}$$

Two Observations

Lemma 1 In case $\sum_{i=1}^{n} w_i \leq m$, then $x_i = 1, 1 \leq i \leq n$ is an optimal solution.

Lemma 2 In case $\sum_{i=1}^{n} w_i \geq m$, all optimal solutions will fit the knapsack exactly.

Problem Instance

$n = 3, m = 20, P = (25, 24, 15)$ and $W = (18, 15, 10)$.

Solution 1: $x_1 = 0.5, x_2 = \frac{1}{3}, x_3 = \frac{1}{4}$

$$\underbrace{\sum w_i \cdot x_i = 16.5}_{\text{a feasible solution}} \Rightarrow \text{Total profits} = 24.25$$

Solution 2: $x_1 = 0.0, x_2 = 1.0, x_3 = \frac{1}{2}$

$$\underbrace{\sum w_i \cdot x_i = 20}_{\text{a feasible solution}} \Rightarrow \text{Total profits} = 31.5$$

Possible Greedy Strategies

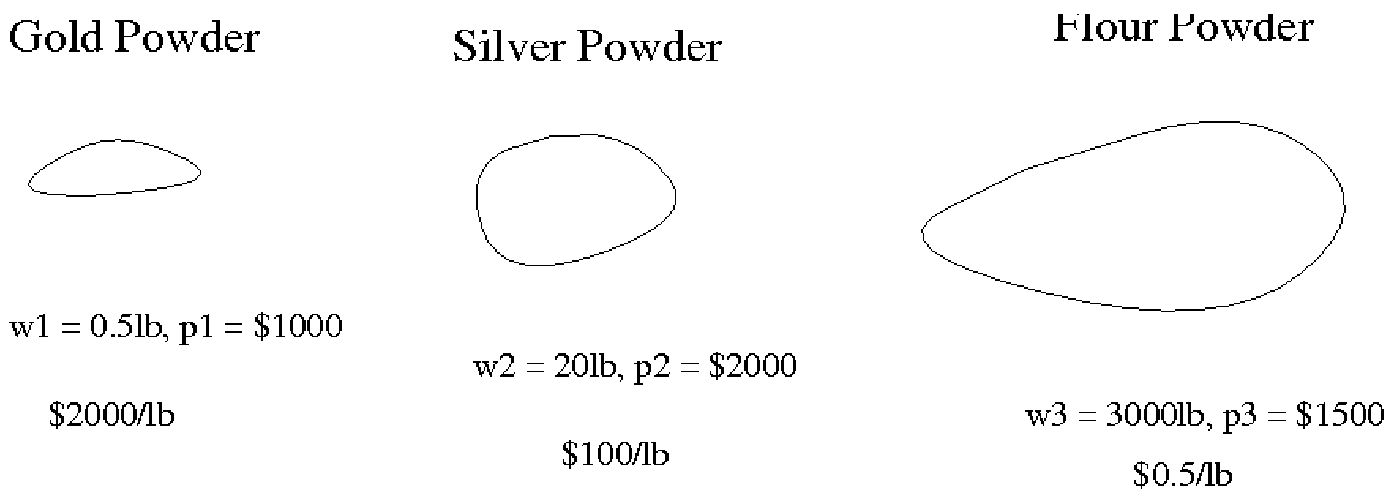
Strategy 1: Pick the max-value object first.
Choose the object in nonincreasing order of value.

$$x_1 = 1,\ x_2 = \frac{2}{15},\ x_3 = 0 \Rightarrow \sum p_i \cdot x_i = 28.2$$

Strategy 2: Pick the lightest object first.
Choose the object in nondecreasing order of weight.

$$x_3 = 1,\ x_2 = \frac{2}{3},\ x_1 = 0 \Rightarrow \sum p_i \cdot x_i = 31$$

Pick the object with the maximum value per pound



Strategy 3: Choose the object in nonincreasing order of $\frac{p_i}{w_i}$

$$\frac{p_i}{w_i} = (\frac{25}{18}, \frac{24}{15}, \frac{15}{10}) = (1.39, 1.60, 1.5)$$

so $x_2 = 1, x_3 = \frac{1}{2}, x_1 = 0 \Rightarrow \sum p_i \cdot x_i = 31.5$

Greedy Knapsack

```
void GreedyKnapsack(float m, int n)
// p[1..n] and w[1..n] contain the profits and weights
// respectively of the n objects ordered such that
// p[i]/w[i] ≥ p[i+1]/w[i+1]. m is the knapsack
// capacity and x[1..n] is the solution vector.

    for i := 1 to n   x[i] = 0.0;       // initialize x

    U := m;
    for i := 1 to n
        if (w[i] > U) break;
        x[i] := 1.0;                    // put the whole object in
        U := U - w[i];

    if (i ≤ n) x[i] := U/w[i];        // the last object to be put in
```

Proving the correctness of a Greedy algorithm is not trivial

- Prim's algorithm: Corollary 23.2 proves $A \cup u$ is still a subset of certain MST.
- Kruskal's algorithm: Corollary 23.2 proves $A \cup u$ is still a subset of certain MST.
- Dijkstra's algorithm: Theorem 24.6 proves that when we insert a vertex u into the set S, it's shortest path is determined, $d[u] = \sigma[s, u]$.

Note: Optimal solutions are not unique in some cases.

Correctness of Greedy Strategy

Theorem: If objects are included in the nonincreasing order of p_i/w_i, then this results in an optimal solution to the knapsack problem.

Proof Sketch: We use the following technique, which is typically useful in proving optimality of greedy algorithms.

Compare the greedy solution with the optimal. If the two solutions differ, then find the first x_i at which they differ. Then show how to make x_i in the optimal solution equal to that of the greedy solution without loss of the total value. Show that the greedy solution is optimal by repeatedly using this transformation.

Proof of Correctness

Let $x = (x_1, ..., x_n)$ be the solution generated by the greedy algorithm.
If $x_i = 1$ for all i, then clearly the solution is optimal. Let j be the first index such that $x_j \neq 1$. Then:

- $x_i = 1$ for $i \in [1, j)$
- $x_j \in [0, 1)$
- $x_i = 0$ for $i \in (j, n]$.

Let $(y_1, ..., y_n)$ be an optimal solution. Then $\sum w_i y_i = m$, by Lemma 2.
Let k be the least index such that $y_k \neq x_k$. Then we can prove $y_k < x_k$, by considering the three possibilities below:

- If $k < j$, then $x_k = 1$. Then $y_k < x_k$, since $y_k \neq x_k$.
- If $k = j$, then since $\sum_{i=1}^{j} w_i x_i = m$ and $y_i = x_i$ for all $1 \leq i < j$, we obtain $y_k = x_k$ (contradiction), otherwise we would have $\sum w_i y_i \neq m$.
- If $k > j$, then $y_k = 0 = x_k$ (contradiction), otherwise we would have $\sum w_i y_i > m$.

Proof of Correctness

Suppose we increase y_k to x_k and decrease as many of $(y_{k+1}, ..., y_n)$ as necessary. This results in a new solution $(z_1, ..., z_n)$ with $z_i = x_i$, for $1 \leq i \leq k$ and:

$$\sum_{k<i\leq n} w_i(y_i - z_i) = w_k(z_k - y_k).$$

Then the total profit for z is:

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

Proof of Correctness

Hence, $\sum p_i z_i \geq \sum p_i y_i$. There are two possible cases:

1. $\sum p_i z_i > \sum p_i y_i$, which means that y cannot be optimal, which is a contradiction, because y was chosen to be an optimal solution. Therefore our assumption (that there is an index k such that $x_k \neq y_k$, where y was an optimal solution) is false, which means that x is an optimal solution.

2. $\sum p_i z_i = \sum p_i y_i$, which means that we made the y_k in the optimal solution equal with the x_k in the greedy solution without loss of the total value. Substitute y with z and repeat the entire procedure for $x_{k+1}, ..., x_n$. We will either exit through case 1, obtaining a contradiction, or end up with an optimal solution z that is the same as x, in which case x is an optimal solution.