Dynamic Programming for Knapsack Problems

John E. Mitchell

1 An example problem

The federal government has $100 million to invest in improving life expectancies. They consider 4 types of projects, each with a financial cost (measured in tens of millions of dollars) and a benefit (measured in weeks of increase in average life expectancy).

<table>
<thead>
<tr>
<th>project</th>
<th>index $k$</th>
<th>cost $c_k$</th>
<th>benefit $v_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>school</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>hospital</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>prison</td>
<td>3</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>industrial park</td>
<td>4</td>
<td>6</td>
<td>9</td>
</tr>
</tbody>
</table>

More than one of any of the types of projects can be undertaken, if desired. This problem is a knapsack problem. We let

$$ x_k = \text{number of projects of type } k \text{ that are built, } k = 1, \ldots, 4 $$

The problem can then be formulated as

$$ \max_{x \in \mathbb{R}^4} \quad 2x_1 + 4x_2 + x_3 + 9x_4 $$

subject to

$$ 2x_1 + 3x_2 + 5x_3 + 6x_4 \leq 10 $$

$$ x_k \geq 0, \text{ integer, } k = 1, \ldots, 4 $$

2 A dynamic programming approach

Consider the process of allocating the money. Can regard it as a sequential process: first decide how many schools to build, then how many hospitals, then how many prisons, then how many industrial parks. Each decision impacts the amount of money available for the remaining items.

Stages: At stage $k$, decide the value of $x_k$, for $k = 1, \ldots, 4$.

States: The state $s$ at stage $k$ is the amount of money left for projects $k, \ldots, 4$.

The value of undertaking $x_k$ projects of type $k$ is equal to

$$ \text{immediate benefit from } x_k \text{ projects} $$

$$ + \text{ subsequent benefit from spending } s - c_k x_k \text{ on projects } k + 1, \ldots, 4 $$

$$ = v_k x_k + f_{k+1}(s - c_k x_k) $$
We use the **backward recursive equations**:

\[ f_k(s) : \text{ value of best way to spend 10s million on projects } k, \ldots, 4. \]

\[ f_k(s, x_k) : \text{ value of best way to spend 10s million on projects } k, \ldots, 4, \]

and given that we undertake \( x_k \) projects of type \( k \).

\[
\begin{align*}
    f_k(s, x_k) &= v_k x_k + f_{k+1}(s - c_k x_k) \\
    f_k(s) &= \max\{ f(s, x_k) : 0 \leq x_k \leq \lfloor \frac{s}{c_k} \rfloor, x_k \text{ integer} \}
\end{align*}
\]

We want to find \( f_1(10) \). We first find \( f_4(s) \) for \( s = 0, \ldots, 10 \), then find \( f_3(s) \), then \( f_2(s) \), and finally \( f_1(10) \).

**Find \( f_4(s) \)**

We undertake as many projects as we can afford of type 4. We have \( c_4 = 6, v_4 = 9 \).

\[
\begin{array}{c|c|c|c|}
  s & f_4(s) = v_4 x_4^* & x_4^* \\
  \hline
  0 & 0 & 0 \\
  1 & 0 & 0 \\
  2 & 0 & 0 \\
  3 & 0 & 0 \\
  4 & 0 & 0 \\
  5 & 0 & 0 \\
  6 & 9 & 1 \\
  7 & 9 & 1 \\
  8 & 9 & 1 \\
  9 & 9 & 1 \\
  10 & 9 & 1 \\
\end{array}
\]

**Find \( f_3(s) \)**

We have \( c_3 = 5, v_3 = 1 \). Recursion: \( f_3(s, x_3) = v_3 x_3 + f_4(s - c_3 x_3) \). Since \( c_3 = 5 \), \( x_3 \) can only take the values 0, 1, 2.

\[
\begin{array}{c|c|c|c|c|c|c|}
  s & f_3(s, x_3) = x_3 + f_4(s - 5 x_3) & f_3(s) = \max\{ f_3(s, x_3) \} & x_3^* \\
  \hline
  0 & 0 + f_4(0) = 0 & 0 & 0 \\
  1 & 0 + f_4(1) = 0 & 0 & 0 \\
  2 & 0 + f_4(2) = 0 & 0 & 0 \\
  3 & 0 + f_4(3) = 0 & 0 & 0 \\
  4 & 0 + f_4(4) = 0 & 0 & 0 \\
  5 & 0 + f_4(5) = 1 + f_4(0) = 1 & 1 & 1 \\
  6 & 0 + f_4(6) = 9 & 9 & 0 \\
  7 & 0 + f_4(7) = 9 & 9 & 0 \\
  8 & 0 + f_4(8) = 9 & 9 & 0 \\
  9 & 0 + f_4(9) = 9 & 9 & 0 \\
  10 & 0 + f_4(10) = 9 & 9 & 0 \\
\end{array}
\]
Find $f_2(s)$
We have $c_2 = 3$, $v_2 = 4$. Recursion: $f_2(s, x_2) = v_2 x_2 + f_3(s - c_2 x_2)$. Since $c_2 = 3$, $x_2$ can only take the values 0, 1, 2, 3.

<table>
<thead>
<tr>
<th>$s$</th>
<th>$x_2 = 0$</th>
<th>$x_2 = 1$</th>
<th>$x_2 = 2$</th>
<th>$x_2 = 3$</th>
<th>$f_2(s) = \max{f_2(s, x_2)}$</th>
<th>$x^*_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$0 + f_3(0) = 0$</td>
<td>$0 + f_3(1) = 0$</td>
<td>$0 + f_3(2) = 0$</td>
<td>$0 + f_3(3) = 0$</td>
<td>$0$</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>$0 + f_3(1) = 0$</td>
<td>$0 + f_3(2) = 0$</td>
<td>$0 + f_3(3) = 0$</td>
<td>$0 + f_3(4) = 0$</td>
<td>$0$</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>$0 + f_3(2) = 0$</td>
<td>$0 + f_3(3) = 0$</td>
<td>$0 + f_3(4) = 0$</td>
<td>$0 + f_3(5) = 0$</td>
<td>$0$</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>$0 + f_3(3) = 0$</td>
<td>$4 + f_3(0) = 4$</td>
<td>$4 + f_3(1) = 4$</td>
<td>$4 + f_3(2) = 4$</td>
<td>$4$</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>$0 + f_3(4) = 0$</td>
<td>$4 + f_3(1) = 4$</td>
<td>$4 + f_3(2) = 4$</td>
<td>$4 + f_3(3) = 4$</td>
<td>$4$</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>$0 + f_3(5) = 0$</td>
<td>$4 + f_3(2) = 4$</td>
<td>$4 + f_3(3) = 4$</td>
<td>$4 + f_3(4) = 4$</td>
<td>$4$</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>$0 + f_3(6) = 0$</td>
<td>$4 + f_3(3) = 4$</td>
<td>$8 + f_3(0) = 8$</td>
<td>$8 + f_3(1) = 8$</td>
<td>$9$</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>$0 + f_3(7) = 0$</td>
<td>$4 + f_3(4) = 4$</td>
<td>$8 + f_3(1) = 8$</td>
<td>$8 + f_3(2) = 8$</td>
<td>$9$</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>$0 + f_3(8) = 0$</td>
<td>$4 + f_3(5) = 4$</td>
<td>$8 + f_3(2) = 8$</td>
<td>$8 + f_3(3) = 8$</td>
<td>$9$</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>$0 + f_3(9) = 0$</td>
<td>$4 + f_3(6) = 4$</td>
<td>$12 + f_3(0) = 12$</td>
<td>$12 + f_3(1) = 12$</td>
<td>$13$</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>$0 + f_3(10) = 0$</td>
<td>$4 + f_3(7) = 4$</td>
<td>$12 + f_3(1) = 12$</td>
<td>$12 + f_3(2) = 12$</td>
<td>$13$</td>
<td>1</td>
</tr>
</tbody>
</table>

Find $f_1(s)$
We have $c_1 = 2$, $v_1 = 2$. Recursion: $f_1(s, x_1) = v_1 x_1 + f_2(s - c_1 x_1)$. Since $c_1 = 2$, $x_1$ can only take the values 0, 1, 2, 3, 4, 5. We only need to find $f_1(10)$.

<table>
<thead>
<tr>
<th>$s$</th>
<th>$x_1 = 0$</th>
<th>$x_1 = 1$</th>
<th>$x_1 = 2$</th>
<th>$x_1 = 3$</th>
<th>$x_1 = 4$</th>
<th>$x_1 = 5$</th>
<th>$f_1(s) = \max{f_1(s, x_1)}$</th>
<th>$x^*_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>$0 + f_2(10) = 13$</td>
<td>$2 + f_2(8) = 11$</td>
<td>$4 + f_2(6) = 13$</td>
<td>$6 + f_2(4) = 13$</td>
<td>$8 + f_2(2) = 7$</td>
<td>$10 + f_2(0) = 8$</td>
<td>$13$</td>
<td>0, 2</td>
</tr>
</tbody>
</table>

Optimal solution
We have two optimal solutions. Can find them by backtracking through the tables.

- $x_1 = 0$, then $x_2 = 1$ from $f_2(10)$, then $x_3 = 0$ from $f_3(7)$, then $x_4 = 1$ from $f_4(7)$. This solution spends $90$ million for a benefit of 13 weeks.

- $x_1 = 2$, then $x_2 = 0$ from $f_2(6)$, then $x_3 = 0$ from $f_3(6)$, then $x_4 = 1$ from $f_4(6)$. This solution spends $100$ million for a benefit of 13 weeks.
3 Computational complexity

We look to solve the generic knapsack problem

$$\max_{x \in \mathbb{R}^n} \quad v^T x$$

subject to

$$c^T x \leq b$$

$$x \geq 0, \text{ integer}$$

The maximum number of rows in each table is $b + 1$. The number of columns to calculate $f_k(s, x_k)$ for each $k$ is no more than $b + 1$. Therefore, the overall complexity is $O(b^2 n)$.

The space required to store the number $b$ on a computer is $O(\log_2 b)$. Now $b = 2^{\log_2 b}$, so the runtime is exponential in the storage.