Dynamic Programming

Optimal linear interpolation

\[ \text{Error} = \sum (y_i - ax_i - b)^2 \]

Optimal linear interpolation with \( K \) segments

\[ \text{Opt}^k[j] = \min_i \{ \text{Opt}^{k-1}[i] + E_{ij} \} \quad \text{for} \quad 0 < i < j \]

Optimal solution with \( k \) segments extends an optimal solution of \( k-1 \) segments on a smaller problem

Subset Sum Problem

- Let \( w_1, \ldots, w_n = \{6, 8, 9, 11, 13, 16, 18, 24\} \)
- Find a subset that has as large a sum as possible, without exceeding 50

Adding a variable for Weight

- \( \text{Opt}[j, K] \) the largest subset of \( \{w_1, \ldots, w_j\} \)
  that sums to at most \( K \)
  - \( \{2, 4, 7, 10\} \)
    - \( \text{Opt}[2, 7] = \)
    - \( \text{Opt}[3, 7] = \)
    - \( \text{Opt}[3,12] = \)
    - \( \text{Opt}[4,12] = \)

Subset Sum Recurrence

- \( \text{Opt}[j, K] \) the largest subset of \( \{w_1, \ldots, w_j\} \)
  that sums to at most \( K \)
Subset Sum Grid

\[
\text{Opt}[j, K] = \max(\text{Opt}[j - 1, K], \text{Opt}[j - 1, K - w_j] + w_j)
\]

\[
\begin{array}{cccccccccccc}
4 & 3 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\{2, 4, 7, 10\}

Subset Sum Code

\[
\text{for } j = 1 \text{ to } n \\
\text{for } k = 1 \text{ to } W \\
\text{Opt}[j, k] = \max(\text{Opt}[j-1, k], \text{Opt}[j-1, k-w_j] + w_j)
\]

Knapsack Problem

- Items have weights and values
- The problem is to maximize total value subject to a bound on weight
- Items \{l_1, l_2, ..., l_n\}
  - Weights \{w_1, w_2, ..., w_n\}
  - Values \{v_1, v_2, ..., v_n\}
  - Bound K
- Find set \(S\) of indices to:
  - Maximize \(\sum_{i \in S} v_i\) such that \(\sum_{i \in S} w_i \leq K\)

Knapsack Recurrence

\[
\text{Subset Sum Recurrence:} \\
\text{Opt}[j, K] = \max(\text{Opt}[j - 1, K], \text{Opt}[j - 1, K - w_j] + v_j)
\]

Knapsack Grid

\[
\text{Opt}[j, K] = \max(\text{Opt}[j - 1, K], \text{Opt}[j - 1, K - w_j] + v_j)
\]

\[
\begin{array}{cccccccccccc}
4 & 3 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Weights \{2, 4, 7, 10\} Values: \{3, 5, 9, 16\}

Dynamic Programming Examples

- Examples
  - Optimal Billboard Placement
    - Text, Solved Exercise, Pg 307
  - Linebreaking with hyphenation
    - Compare with HW problem 6, Pg 317
  - String approximation
    - Text, Solved Exercise, Page 309
Billboard Placement

• Maximize income in placing billboards
  – $b_i = (p_i, v_i)$, $v_i$: value of placing billboard at position $p_i$
• Constraint:
  – At most one billboard every five miles
• Example
  – $\{(6,5), (8,6), (12, 5), (14, 1)\}$

Design a Dynamic Programming Algorithm for Billboard Placement

• Compute $\text{Opt}[1], \text{Opt}[2], \ldots, \text{Opt}[n]$
• What is $\text{Opt}[k]$?

Input $b_1, \ldots, b_n$, where $b_i = (p_i, v_i)$, position and value of billboard $i$

Opt$[k] = \text{fun}(\text{Opt}[0], \ldots, \text{Opt}[k-1])$

• How is the solution determined from subproblems?

Solution

$$j = 0; \quad \text{// } j \text{ is five miles behind the current position}$$

for $k := 1$ to $n$
  while $(P[j] < P[k] - 5)$
    $j := j + 1; \quad \text{// the last valid location for a billboard, if one placed at } P[k]$
  $j := j - 1; \quad \text{// } j \text{ is now the next valid location}$
  $\text{Opt}[k] = \max(\text{Opt}[k-1], v[k] + \text{Opt}[j]);$