There are many valid "cutting patterns" which will provide the required lengths, e.g.:

\[
\begin{array}{ccc}
4 \text{ feet} & 4 \text{ feet} & 5 \text{ feet} \\
\end{array}
\]

Example

<table>
<thead>
<tr>
<th>L_k</th>
<th>C_k</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
</tr>
</tbody>
</table>

An order is received:

<table>
<thead>
<tr>
<th>length (ft)</th>
<th>4</th>
<th>5</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td># pieces</td>
<td>30</td>
<td>20</td>
<td>40</td>
</tr>
</tbody>
</table>

How should these required pieces be cut from the standard lengths, so as to minimize the cost?

Notation

- \( L_1, L_2, \ldots, L_k \) standard lengths
- \( C_1, C_2, \ldots, C_k \) costs
- \( \ell_1, \ell_2, \ldots, \ell_k \) required lengths
- \( b_1, b_2, \ldots, b_k \) \# pieces required
- \( A_j \) cutting pattern \#j
- \( k(j) \) index of standard length used by pattern \#j
- \( A_j \) \# pieces of length \( \ell_j \) produced by pattern \#j

Integer LP Model

\[
X_j = \# \text{ of times pattern } j \text{ is used}
\]

Minimize \( \sum_{j=1}^{J} C_{kj} X_j \)

subject to: \( \sum_{j=1}^{J} A_{ij} X_j \geq b_i, i=1,2,\ldots,I \)

\( X_j \geq 0 \) and integer, \( j=1,2,\ldots,J \)

Column Generation Scheme for LP

At each iteration of the revised simplex method, a simplex multiplier \( \pi \) is computed.

A pattern \( \{a_1, a_2, \ldots, a_I\} \) may be entered into the basis if its reduced cost is negative, i.e., if

\[
C_k - \pi a < 0 \quad \text{i.e.,} \quad \sum_{i=1}^{I} \pi a_i > C_k
\]

and feasible, i.e., \( \sum_{i=1}^{I} \ell_i a_i \leq L_k \)

where \( L_k \) is the length used by the cutting pattern.
Column Generation Scheme for LP

Given \( \pi \), solve the knapsack problem, which (if \( \pi \) is a \( \mathbb{C} \)) yields a pattern which may be entered into the solution.

Subproblem:

Maximize \( \sum_{i=1}^{n} \pi_i a_i \)

s.t. \( \sum_{i=1}^{n} \beta_i a_i \leq L_k \)

\( a_1, a_2, \ldots, a_n \geq 0 \) and integer

Solution of Knapsack Subproblems

If dynamic programming is used, then we will obtain a solution of the subproblem for each standard length.

We can then add each pattern to the LP if the pattern's reduced cost is negative.

A heuristic method (which is faster than DP but doesn't guarantee the optimal solution to the knapsack problem) may also produce patterns with negative reduced cost to be added to the LP.

Heuristic Knapsack Filling

Order the requested lengths according to the ratios of "payoff" per unit length, \( \frac{\pi_i}{L_i} \leq \frac{\pi_{i+1}}{L_{i+1}} \leq \cdots \leq \frac{\pi_n}{L_n} \)

Step 1)

Maximize \( \sum_{i=1}^{n} \pi_i a_i \)

s.t. \( \sum_{i=1}^{n} \beta_i a_i \leq L_k \)

\( a_1, a_2, \ldots, a_n \geq 0 \) and integer

Step 2)

Fill the knapsack with the maximum possible number of pieces of length \( L_k \),

\[ a_i = \left\lfloor \frac{L_k}{L_i} \right\rfloor \]

Step 3)

Fill what remains empty \( (L_k-L_i a_i) \) with as many pieces as possible of the next most "valuable" length \( L_{i+1} \), etc., until the knapsack is filled or the last length \( L_n \) has been tried.

DP Solution of Knapsack Problem

Maximize \( \sum_{i=1}^{n} \pi_i a_i \)

s.t. \( \sum_{i=1}^{n} \beta_i a_i \leq L_k \)

\( a_1, a_2, \ldots, a_n \geq 0 \) and integer

Thus, all subproblems have been solved by computing \( f_k(L_k) \) for \( k=1, 2, \ldots, K \).

Solution of Example

<table>
<thead>
<tr>
<th>( k )</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_k )</td>
<td>9</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>( C_k )</td>
<td>5</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( i )</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \beta_i )</td>
<td>4</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>( b_i )</td>
<td>30</td>
<td>20</td>
<td>40</td>
</tr>
</tbody>
</table>

To begin our column generation algorithm, we shall (arbitrarily) propose the following three patterns cut from the 9-feet stock length:

\[ \begin{bmatrix} 2 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \end{bmatrix} \] 4-ft length

\[ \begin{bmatrix} 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \end{bmatrix} \] 5-ft length

\[ \begin{bmatrix} 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \end{bmatrix} \] 7-ft length

Our initial LP is

\[ \begin{align*}
\text{Minimize} & \quad 5X_1 + 5X_2 + 5X_3 \\
\text{s.t.} & \quad 2X_1 + X_3 - S_1 = 30 \\
& \quad -S_2 = 20 \\
& \quad X_3 - S_3 = 40 \\
& \quad X_2 \geq 0, \ k=1,2,3; S_i \geq 0, i=1,2,3
\end{align*} \]

with the optimal solution:

\[ \begin{align*}
X_1 &= 15, X_2=0, X_3=40, S_1=S_2=S_3=0 \\
\pi_1 &= 2.5, \pi_2 = 5, \pi_3 = 5, \text{ cost} = \$375
\end{align*} \]
Let's try generating a cutting pattern from the longest (16-ft) stock using the heuristic method:

- sort the lengths: \( \frac{\pi_2}{2} > \frac{\pi_3}{3} > \frac{\pi_4}{4} \), i.e., \( \frac{5}{2} > \frac{5}{3} > \frac{2.5}{4} \)
- fill the knapsack with as many as possible of the 5-ft lengths, i.e.,
  \[
  a_2 = \frac{16}{5} = 3
  \]
- after including 3 pieces of length 5 feet,
  \( 16 - 3 \times 5 = 1 \) foot of capacity remains.
This is insufficient for any of the other required pieces, i.e., \( a_3 = a_1 = 0 \).

The heuristic method has produced the pattern \( A^4 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \) with reduced cost \( 10 - 10A^4 = 10 - 15 = -5 \)
The reduced cost is negative, so it is worthwhile to add this pattern to the LP.

The new LP is

\[
\begin{align*}
\text{Minimize} & \quad 5X_1 + 5X_2 + 5X_3 + 10X_4 \\
\text{s.t.} & \quad 2X_1 - S_1 = 30 \\
& \quad X_2 + 3X_3 - S_2 = 20 \\
& \quad X_3 - S_3 = 40 \\
& \quad X_1 \geq 0, S_1 \geq 0 \quad k=1,2,3,4; \quad s=1,2,3
\end{align*}
\]
which has the optimal solution:

\[
\begin{align*}
X_1 &= 15, \quad X_2 = 40, \quad X_3 = \frac{20}{3}, \quad X_4 = S_1 = S_2 = S_3 = 0 \\
\pi_1 &= 2.5, \quad \pi_2 = \frac{10}{3}, \quad \pi_3 = 5, \quad \text{cost} = 3341.67
\end{align*}
\]

We fill the 16-ft knapsack with as many of the 7-ft required pieces as possible:

\[
a_3 = \frac{16}{7} = 2
\]
This leaves \( 16 - 2 \times 7 = 2 \) feet capacity, which is insufficient for any of the other required lengths. The pattern generated by the heuristic method is therefore

\[
A = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}
\]
with reduced cost \( 10 - 5 \times 2 = 0 \).

If we apply the heuristic to the 14-ft stock length, we again obtain \( a_1 = a_2 = 0, a_3 = 2, \) but because the cost of the 14-ft stock length is lower, the reduced cost is \( 9 - 5 \times 2 = -1 < 0 \).
Therefore, we add the pattern \( A^5 = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} \) (with \( k=5 \)).

\[
\begin{align*}
\text{Minimize} & \quad 5X_1 + 5X_2 + 5X_3 + 10X_4 + 9X_5 \\
\text{s.t.} & \quad 2X_1 - S_1 = 30 \\
& \quad X_2 + 3X_3 - S_2 = 20 \\
& \quad X_3 + 2X_5 - S_3 = 40 \\
& \quad X_1 \geq 0, S_1 \geq 0 \quad k=1,2,3,4,5; \quad s=1,2,3
\end{align*}
\]

We can include 2 of the 5-ft pieces, leaving 14 - 10 = 4 feet capacity. The next length to try is 7-ft, but there is insufficient capacity; finally, we include 1 of the 4-ft pieces, exactly filling the knapsack.
We have thus generated the pattern \( A^0 = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} \) from the 14-ft stock length. Because the reduced cost of this pattern is

\[
C_i - \sum a_i \pi_i = 9 - \left( \frac{2.5 \times 1}{3} + \frac{10}{3} \times 2 + 4.5 \times 0 \right) = -0.17 < 0
\]

we will add the pattern to the LP:

**Minimize** \( 5X_1 + 5X_2 + 5X_3 + 10X_4 + 9X_5 + 9X_6 \)

**s.t.**

\[
\begin{align*}
2X_1 + 3X_2 + 2X_4 - S_1 &= 30 \\
X_2 + 2X_3 - S_2 &= 20 \\
X_3 + 2X_5 - S_3 &= 40 \\
X_k &\geq 0, k=1,2,3,4,5,6; \\
S_k &\geq 0, k=1,2,3
\end{align*}
\]

**LP**

- **Minimize** \( 5X_1 + 5X_2 + 5X_3 + 10X_4 + 9X_5 + 9X_6 \)
- **s.t.**
  - \( 2X_1 + 3X_2 + 2X_4 - S_1 = 30 \)
  - \( X_2 + 2X_3 - S_2 = 20 \)
  - \( X_3 + 2X_5 - S_3 = 40 \)
  - \( X_k \geq 0, k=1,2,3,4,5,6; \)
  - \( S_k \geq 0, k=1,2,3 \)
- with solution:
  - \( X_1=10, X_2=20, X_4=10 \)
  - \( \pi_1 = 2.5, \pi_2 = 3.25, \pi_3 = 4.5, \text{cost} = \$320 \)

If we apply the heuristic pattern-generating method to the 9-ft stock length, we get the pattern \( A^2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \) with reduced cost

\[
C_i - \sum a_i \pi_i = -0.75
\]

**Heuristic solution of knapsack problem:**

\[
\begin{align*}
\text{Maximize} & \quad 2.5a_1 + \frac{39}{12}a_2 + 4.5a_3 \\
\text{s.t.} & \quad 4a_1 + 5a_2 + 7a_3 \leq 14 \\
& \quad a_i \geq 0 \text{ & integer, } i=1,2,3
\end{align*}
\]

**Sort the pieces:**

\[
\frac{\pi_3}{\pi_2} > \frac{\pi_3}{\pi_1}, \text{i.e., } \frac{4.5}{5} > \frac{2.5}{4}
\]

Using 14-ft stock length, we include \( a_2=2 \) of the 5-ft length, \( a_3=0 \) of the 7-ft length, and \( a_1=1 \) of the 4-ft length; **but this was pattern #6** (which now has reduced cost 0)

**Minimize** \( 5X_1 + 5X_2 + 5X_3 + 10X_4 + 9X_5 + 9X_6 \)

**s.t.**

\[
\begin{align*}
2X_1 + 3X_2 + 2X_4 - S_1 &= 30 \\
X_2 + 2X_3 - S_2 &= 20 \\
X_3 + 2X_5 - S_3 &= 40 \\
X_k &\geq 0, k=1,2,3,4,5,6; \\
S_k &\geq 0, k=1,2,3
\end{align*}
\]

**LP Solution:**

\[
\begin{cases}
X_1=5, X_2=20, X_4=20 \\
\pi_1 = 2.5, \pi_2 = 2.5, \pi_3 = 4.5, \text{cost} = \$305
\end{cases}
\]

**Column-generating subproblem:**

\[
\begin{align*}
\text{Max} & \quad 2.5a_1 + 2.5a_2 + 4.5a_3 \\
\text{s.t.} & \quad 4a_1 + 5a_2 + 7a_3 \leq 14 \\
a_i &\geq 0 \text{ & integer, } i=1,2,3
\end{align*}
\]

The heuristic method is unable to generate a pattern having a negative reduced cost; however, because it doesn’t in general optimize the knapsack problem, there may be other patterns which do have negative reduced costs.

Therefore, we must now apply the DP (dynamic programming) algorithm.

Heuristic solution of knapsack problem:

<table>
<thead>
<tr>
<th>( a_1 )</th>
<th>( a_2 )</th>
<th>( a_3 )</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>2.5</td>
<td>4.5</td>
<td>305</td>
</tr>
</tbody>
</table>

**Maximize** \( 2.5a_1 + 2.5a_2 + 4.5a_3 \)

**s.t.**

\[
\begin{align*}
4a_1 + 5a_2 + 7a_3 &\leq L \\
a_i &\geq 0 \text{ & integer, } i=1,2,3
\end{align*}
\]

**Sorting the reg’d pieces:**

\[
\frac{\pi_3}{\pi_2} > \frac{\pi_3}{\pi_1}, \text{i.e., } \frac{4.5}{5} > \frac{2.5}{4}
\]

Using \( L=9 \), we get \( a_3 = 1, a_2 = 0, a_3 = 0 \), which is pattern #3 again, now with reduced cost 0

Using \( L=14 \), we get \( a_3 = 2, a_1 = a_2 = 0 \) with reduced cost 0

Using \( L=16 \), we get \( a_3 = 2, a_1 = a_2 = 0 \), which has reduced cost 0

**Stock Piece 1 (Length 5):**

- **Optimal Knapsack Value is 5**
- **There are 3 optimal patterns from this stock piece**

\[
\begin{align*}
\text{Optimal Pattern #1 from this stock piece:} & \\
1 & 1 & 0 & \text{Scrap remaining: 1} \\
1 & 4 & 0 & \text{Scrap remaining: 0} \\
1 & 7 & 0 & \text{Scrap remaining: 0}
\end{align*}
\]

Reduced Cost: 0
Stock Piece 2 (Length 14)

*** Optimal Knapsack Value is 9 ***

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>a</th>
<th>4</th>
<th>0</th>
<th>Scrap remaining: 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reduced Cost: 0

Stock Piece 3 (Length 16)

*** Optimal Knapsack Value is 10 ***

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

Reduced Cost: 0

Stage 1

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-999.99</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-999.99</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>-999.99</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>-999.99</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>4</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>5</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>6</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>7</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>8</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>9</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>10</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>11</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>12</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>13</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>14</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>15</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>16</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
</tbody>
</table>

Stage 2

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.00</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>1</td>
<td>0.00</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>2</td>
<td>0.00</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>3</td>
<td>0.00</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>4</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>5</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>6</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>7</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>8</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>9</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>10</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>11</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>12</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>13</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>14</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>15</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
<tr>
<td>16</td>
<td>2.50</td>
<td>-999.99</td>
<td>-999.99</td>
</tr>
</tbody>
</table>

Stage 3

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.00</td>
<td>-999.99</td>
</tr>
<tr>
<td>1</td>
<td>0.00</td>
<td>-999.99</td>
</tr>
<tr>
<td>2</td>
<td>0.00</td>
<td>-999.99</td>
</tr>
<tr>
<td>3</td>
<td>0.00</td>
<td>-999.99</td>
</tr>
<tr>
<td>4</td>
<td>2.50</td>
<td>-999.99</td>
</tr>
<tr>
<td>5</td>
<td>2.50</td>
<td>-999.99</td>
</tr>
<tr>
<td>6</td>
<td>2.50</td>
<td>-999.99</td>
</tr>
<tr>
<td>7</td>
<td>2.50</td>
<td>-999.99</td>
</tr>
<tr>
<td>8</td>
<td>2.50</td>
<td>-999.99</td>
</tr>
<tr>
<td>9</td>
<td>2.50</td>
<td>-999.99</td>
</tr>
<tr>
<td>10</td>
<td>2.50</td>
<td>-999.99</td>
</tr>
<tr>
<td>11</td>
<td>2.50</td>
<td>-999.99</td>
</tr>
<tr>
<td>12</td>
<td>2.50</td>
<td>-999.99</td>
</tr>
<tr>
<td>13</td>
<td>2.50</td>
<td>-999.99</td>
</tr>
<tr>
<td>14</td>
<td>2.50</td>
<td>-999.99</td>
</tr>
<tr>
<td>15</td>
<td>2.50</td>
<td>-999.99</td>
</tr>
<tr>
<td>16</td>
<td>2.50</td>
<td>-999.99</td>
</tr>
</tbody>
</table>

Optimal Solution of LP

- \( X_1 \) - 5 feet
- \( X_5 \) - 7 feet
- \( X_7 \) - 4 feet