A Report on the Cutting Stock Problem

Prepared for: Moshe Sniedovich
MS Software Pty. Ltd.

Submitted on: 8 November 2004

Prepared by: Robert Chilov 58600
Joshua Richmond 109012
Derek Weeks 132817
Executive summary

This is a report on the cutting stock optimisation problem, commissioned by Moshe Sniedovich on behalf of MS Software Pty Ltd. MS Software is interested in a modified version of the cutting stock problem, and hopes to write software to control machines faced with this problem.

In the modified problem, the quality of a piece of stock may vary along its length. The stock is cut to produce smaller items, but the quality of each item depends on what part of the stock it came from. The sale price of an item depends on its quality. The objective is to maximise profit from the items produced.

Our primary aim was to formulate a mathematical model to describe the modified problem. Our secondary aim was to develop a way to solve any given instance of the problem. We achieved both our aims.

We modelled the problem as a mixed linear-integer program. This model is easy to implement using standard optimisation software, and we produced code in the Mosel language for use with the Xpress-MP solver. Thus, we have a working solution that can be applied to a given instance of the problem and will generate an optimal solution.

We recommend that resources are allocated for further work in three areas:

- comparing other optimisation software to Xpress-MP for price and performance;
- testing the performance of the model on large problems which take longer to solve;
- optimising the model itself to improve performance (if necessary).
# Contents

**Executive summary**  
1

## 1 Introduction

1.1 Background ......................................................... 1

1.2 Problem description ........................................... 1

1.3 Aim ............................................................... 2

## 2 Results

2.1 Conclusions .......................................................... 2

2.2 Recommendations .................................................. 3

## 3 Method

3.1 Mathematical model ................................................ 4

3.1.1 Input variables .................................................. 4

3.1.2 Decision variables ............................................. 5

3.1.3 Objective function ............................................ 5

3.1.4 Constraints ....................................................... 6

3.2 Xpress-MP implementation ........................................ 9

3.3 Example problem ................................................... 9

3.3.1 Input parameters ............................................... 9

3.3.2 Optimal solution .............................................. 11

References 12
1 Introduction

This is a report on the investigation into the cutting stock optimisation problem. The report was commissioned by Moshe Sniedovich on behalf of MS Software Pty Ltd.

1.1 Background

The cutting stock problem is a common problem faced in industry. In the classical problem, a number of one-dimensional pieces of stock are cut into several smaller items (Winston 2004). The objective is to minimise the number of pieces of stock that need to be cut up. The items that are produced must each be of some standard length, and a fixed number of items of each standard length must be cut in order to satisfy demand. Often a length at the end of the stock is left over; this is waste.

A variation of the classical problem is to seek to maximise revenue, with no regard for demand. In this case, items of different standard lengths sell for different values. This is identical to the unbounded knapsack problem.

In the classical problem, the stock is assumed to be of uniform quality. In many practical applications, however, this is not a valid assumption. For example, a piece of timber may have a knot at some point along it, or a strip of metal may be damaged or corroded. This significantly complicates the classical problem and is the motivation for this report.

1.2 Problem description

A single instance of the problem involves only one piece of stock of a given length. The piece is to be cut into many smaller items. There is a fixed number of different standard lengths, and each item must be as long as one of the standard lengths. There are no restrictions on the number of items that can be cut to a particular standard length.

The quality of the piece of stock varies along its length, so the piece is divided into segments, with the quality constant in any one segment. Thus, each segment is assigned a quality grade, and there are a fixed number of different quality grades. A segment may be graded as ‘unusable’.

Any item produced may span more than one segment of the stock. The item is assigned a quality grade equal to the poorest quality grade of all the segments it spans. The sale
price of an item depends on its standard length and quality grade.

Any off-cuts that are not a standard length are considered waste. There is a cost associated with disposing of waste which is proportional to the total length of the waste produced. This cost may be negative (e.g. if the waste can be sold).

The objective is to maximise profit, which is the total revenue from the items produced less the cost of disposing of any waste produced.

1.3 Aim

The primary aim of this investigation was to build a mathematical formulation of the problem.

The secondary aim was as follows: First, to determine whether the problem is, in fact, a well-known operations research problem, and search for existing solutions. Second, if the problem is new and unsolved, to develop a way to solve any given instance of the problem. Third, if in general an optimal solution cannot be found in a sensible time period, to develop a heuristic technique that approximates the optimal solution.

If the secondary aim is achieved, MS Software will be able to develop software to control stock cutting machines faced with this problem. Thus, the purpose of the secondary aim is to provide MS Software with a means of solving instances of this problem in real time.

2 Results

2.1 Conclusions

We modelled the problem as a mixed linear-integer program (LIP), achieving our primary aim.

Though the classical cutting stock problem is common and solvers are readily available, the modified version considered here is not examined in the current literature. Our search uncovered no existing software that solves this particular problem.

However, we developed a means of solving the problem. Because we had modelled the problem as an LIP, we were able to implement a solution using standard optimisation
software. We coded our model in the Mosel language and successfully solved examples with the Xpress-MP solver.

We found that the solver was sufficiently fast for solving typical problems on standard PC hardware (within a few seconds). Therefore, we did not attempt to develop any heuristic methods for solving the problem, which may be faster.

Thus, both our primary and secondary aims were met.

2.2 Recommendations

We believe our model and solution provide proof of concept and that MS Software will now be able to build software to control stock cutting machines. We recommend that resources are allocated for further work in three areas:

- comparing other optimisation software to Xpress-MP for price and performance;
- testing the performance of the model on large problems which take longer to solve;
- optimising the model itself to improve performance (if necessary).

The code written in Mosel does not operate without the Xpress-MP solver. However, we do not recommend coding a solution from scratch. Software such as Xpress-MP can compile binaries that can be linked to other programs written, for example, in C. MS Software need only build a user interface to collect the input parameters, pass them to the Xpress-MP binary, and then control the cutting machine based on the solver’s outputs.

3 Method

We followed standard operations research modelling techniques to formulate the LIP. The main advantage of developing an LIP is that it is easy to solve using standard optimisation software. A clear disadvantage, however, is that it is difficult to understand.

In this section we walk through our mathematical model of the problem. Following standard procedure, we identify the input variables, then define the decision variables, objective function, and constraints. There is a large number of variables, so we give a summary at the end of each subsection. An example of an instance of the problem is given in section 3.3 to demonstrate the model.
3.1 Mathematical model

The diagram below represents a generic piece of stock.

![Diagram of stock with segments, quality, and waste]

3.1.1 Input variables

First, we consider the information we are given for a particular instance of the problem.

The piece of stock itself is divided into \( q \) segments, which we label \( 1, \ldots, q \) from ‘left to right’. We let \( d_s \) be the distance from the beginning of the stock to the end of segment \( s \). Thus, each segment \( s \) starts at position \( d_{s-1} \) and ends at position \( d_s \) (with \( d_0 := 0 \)).

For convenience, we define \( L := d_q \), the total length of the stock.

There are \( m \) different quality grades used to rate each segment: \( 1, \ldots, m \) in order of increasing quality. A quality grade of 0 represents the ‘unusable’ quality. We let \( u_s \) be the quality grade of segment \( s \).

There are \( n \) different standard lengths that items are allowed to be: \( 1, \ldots, n \) in order of increasing length. We let \( l_j \) be the length that an item of standard length \( j \) must be.

Note that the number of items produced cannot be greater than \( L/l_1 \) (the length of the stock divided by the shortest standard length). We let \( p := \lfloor L/l_1 \rfloor \) be the maximum number of items that can be produced.

The revenue gained from producing an item depends on the item’s standard length and quality. Thus, we have an \( n \times m \) revenue matrix, with \( r_{jk} \) the revenue for an item whose length is \( l_j \) and quality grade is \( k \).

Finally, we let \( c \) be the cost associated with generating 1 unit length of waste.
Summary of input variables

\( q \) := no. of segments in stock  
\( m \) := no. of quality grades  
\( n \) := no. of standard lengths  
\( d_s \) := distance from start of stock to end of segment \( s \)  
\( u_s \) := quality of segment \( s \)  
\( l_j \) := length of standard length \( j \), with \( l_1 < l_2 < \cdots < l_n \)  
\( L := d_q \) = total length of stock  
\( p := \lfloor \frac{L}{s} \rfloor \) = maximum number of items produced  
\( r_{jk} \) := revenue for item of length \( l_j \) and quality \( k \)  
\( c := \text{cost of 1 unit waste} \)

3.1.2 Decision variables

The only decision we need to make is where to cut the stock. In order to produce a particular item \( i \), we need to make two cuts: one on the left and one on the right. We denote the position of the left and right cuts by \( x_i \) and \( y_i \) respectively.

Note that in practice we may not need to make a cut for every \( x_i \) and \( y_i \). For example, if \( x_1 = 0 \) then item 1 begins at the start of the stock and no cut is needed. Furthermore, if two items are flush, i.e. \( y_i = x_{i+1} \), only one cut is needed at that position. Thus, it is better to think of \( x_i \) and \( y_i \) as the start and end of item \( i \) respectively.

We label the items 1, \ldots, \( p \), from left to right. Now \( p \) is the maximum number of items possible, but an optimal solution may produce less than \( p \) items. To accommodate this, we allow for items of zero length. If \( x_i = y_i \) for some item \( i \), then the item does not exist.

Summary of decision variables

\( x_i \) := distance from start of stock to start of item \( i \)  
\( y_i \) := distance from start of stock to end of item \( i \)

3.1.3 Objective function

The objective function must calculate total profit (revenue less cost). Now, based on the values of the decision variables \( x \) and \( y \), we can deduce the standard length and quality of each item. The revenue matrix, however, does not give a revenue based on the actual
length of an item, but on its standard length, which is an integer from 1 to n. The quality of an item is also an integer, from 1 to m.

To write down the objective function we require an external variable, i.e. a variable that depends on the decision variables. We define the binary variable $z_{ijk}$ to be equal to 1 if and only if item $i$ is of standard length $j$ and quality $k$.

If a particular $z_{ijk} = 1$, then the revenue for item $i$ is $r_{jk} = r_{jk} z_{ijk}$. If a particular $z_{ijk} = 0$, then the revenue for item $i$ is $0 = r_{jk} z_{ijk}$. Thus our total revenue is given by summing $r_{ij} z_{ijk}$ over all items, standard lengths, and qualities.

The waste produced between item $i$ and item $i+1$ is $x_{i+1} - y_i$. Thus, the total cost is the sum of $c (x_{i+1} - y_i)$ over all items, where $y_0 := 0$ and $x_{p+1} := L$. The objective function is given by (1).

$$\max_{xy} \sum_{i=1}^{p} \sum_{j=1}^{n} \sum_{k=1}^{m} r_{jk} z_{ijk} - c \sum_{i=0}^{p} (x_{i+1} - y_i)$$  \hspace{1cm} (1)

### 3.1.4 Constraints

Throughout this section, we have $i \in \{1, \ldots, p\}$, $j \in \{1, \ldots, q\}$, and $k \in \{1, \ldots, m\}$, except where specified otherwise. $M$ is a large positive constant.

We labelled the items 1, \ldots, $p$ from left to right, so we require $0 \leq x_1 \leq y_1 \leq x_2 \leq y_2 \leq \ldots \leq x_p \leq y_p \leq L$. This is achieved by constraints (2), (3), (4), and (5). (This also ensures non-negativity of decision variables.)

\begin{align*}
0 &\leq x_1 \quad (2) \\
x_i &\leq y_i, \quad \forall \ i. \quad (3) \\
y_i &\leq x_{i+1}, \quad \forall \ i \in \{1, \ldots, p-1\}. \quad (4) \\
y_p &\leq L \quad (5)
\end{align*}

We place a binary constraint (6) on each $z_{ijk}$.

$$z_{ijk} \in \{0, 1\}, \quad \forall \ i, j, k. \quad (6)$$

Constraint (7) prevents an item having more than 1 standard length and quality. We do not require that an item has exactly one standard length and quality, as we allow for zero-length items.
\[ \sum_j \sum_k z_{ijk} \leq 1, \quad \forall i. \tag{7} \]

Constraints (8), (9), and (10) force the length of each item to be exactly equal to a standard length, or zero.

For a particular item \( i \) and standard length \( j \), if \( z_{ijk} = 1 \) for some \( k \) then we want \( y_i - x_j = l_j \). We achieve this by saying that the length of item \( i \) is at least \( l_j \) (8) and at most \( l_j \) (9).

On the other hand, for a particular item \( i \) and standard length \( j \), if each \( z_{ijk} = 0 \) then item \( i \) has some length other than \( l_j \) (possibly zero). So the length of \( i \) is at least zero (8) and at most \( M \) (9), i.e. it is unrestricted. Note that \( M \) can be replaced by \( L \) in (9) and (10).

Finally, for a particular item \( i \), if \( z_{ijk} = 0 \) for all \( j \) and \( k \), then constraint (10) forces \( i \) to have zero length; if not it merely restricts \( i \) to be no longer than \( M \).

\begin{align*}
 y_i - x_i & \geq l_j \sum_k z_{ijk}, \quad \forall i, j. \tag{8} \\
 y_i - x_i & \leq l_j \sum_k z_{ijk} + M \left( 1 - \sum_k z_{ijk} \right), \quad \forall i, j. \tag{9} \\
 y_i - x_i & \leq M \sum_j \sum_k z_{ijk}, \quad \forall i. \tag{10}
\end{align*}

The constraints thus far have ensured that each item is of an allowable length. In order to define constraints on the quality of an item, we define another external variable. Let \( v_i \) be the quality grade of item \( i \).

\[ v_i \geq 0, \quad v_i \in \mathbb{Z}, \quad \forall i. \tag{11} \]

In maximising our objective function, we want each item to have the highest possible quality. So the optimal solution using only the constraints given thus far will assign each item \( i \) a quality \( v_i = m \). But we know that if an item spans a quality segment, then its quality should be no better than the quality of that segment.

So for a given segment \( s \) that begins at position \( d_{s-1} \) and ends at position \( d_s \), we say the following: If \( x_i < d_s \) and \( y_i > d_{s-1} \) then \( v_i \leq u_s \) (note the strict inequalities). This is
achieved by constraints (12), (13), and (14). Two external binary variables $a$ and $b$ are defined for every item and segment.

For a particular item $i$ and segment $s$, if $x_i < d_s$ then constraint (12) forces $a_{is} = 1$. Similarly, if $y_i > d_{s-1}$ then (13) forces $b_{is} = 1$. Now if $a = b = 1$, then the RHS of (14) is zero and $v_i \leq u_s$ as required. If $a = 0$ and/or $b = 0$, then $v_i$ is essentially unrestricted.

Note that $M$ can be replaced with $L$ in (12) and (13), and with $m$ in (14).

$$d_s - x_i \leq Ma_{is}, \quad \forall i, s. \quad (12)$$

$$y_i - d_{s-1} \leq Mb_{is}, \quad \forall i, s. \quad (13)$$

$$v_i - u_s \leq M(2 - a_{is} - b_{is}), \quad \forall i, s. \quad (14)$$

$$a_{is}, b_{is} \in \{0, 1\}, \quad \forall i, s. \quad (15)$$

Our constraints now force each $v_i$ to hold the correct value, but $v_i$ is just an external variable. We need to relate the value of each $v_i$ back to our objective function. That is, we must ensure that the correct $z_{ijk}$ is 1 with respect to $k$.

In maximising the objective function, we will seek to assign a high quality to an item, so we need only prevent the system assigning a quality that is not allowed. What we would like to say is this: For a particular item $i$ and quality $k$, $\sum_j z_{ijk} = 1$ only if $v_i = k$. This is achieved by constraints (16) and (17) (note that $M$ can be replaced with $m$).

$$v_i - k \leq M(1 - \sum_j z_{ijk}), \quad \forall i, k. \quad (16)$$

$$v_i - k \geq -M(1 - \sum_j z_{ijk}), \quad \forall i, k. \quad (17)$$

Our final constraint ensures that we do not produce an item containing an unusable segment of the stock. Recall that $u_s = 0$ implies that segment $s$ cannot be used. Constraint (18) says that if $v_i = 0$ then item $i$ cannot exist, i.e. $y_i - x_i = 0$.

$$y_i - x_i \leq Mv_i, \quad \forall i. \quad (18)$$

Note that the $M$ in (18) can be replaced by $L$. This is a very interesting constraint, because it relates the length of an item (LHS) to its quality (RHS).
Summary of external variables

\[ z_{ijk} := \begin{cases} 1 & \text{if } y_i - x_i = l_j \text{ and } v_i = u_k \\ 0 & \text{otherwise} \end{cases} \]

\[ v_i := \text{quality of item } i \]

\[ a_{is}, b_{is} := \text{both equal to 1 iff item } i \text{ spans segment } s \]

3.2 Xpress-MP implementation

We used Xpress-MP to implement our model. The Mosel code is given in appendix B.1. All the variable types and constraints presented in section 3.1 can be used in Mosel, so the implementation was straightforward.

Our implementation was tested using the student version of Xpress-MP. This version allows only 400 variables and 800 constraints, so it can only be used to solve small instances of the problem. For example, if we have \( n = m = 5 \) and \( p = q = 10 \), then there are 480 decision variables (including externals) and 791 constraints (excluding non-negativity, binary, and integer constraints).

3.3 Example problem

In this section we present an example problem that we solved using Xpress-MP. It is a small example, but an optimal solution is hard to deduce by inspection.

3.3.1 Input parameters

We are given a piece of stock 6.6m in length. The stock is divided into four quality segments (\( q = 4 \)) as shown below:
From the diagram, we have \( d_1 = 1.6, d_2 = 3.8, d_3 = 4.0, \) and \( d_4 = 6.6 = L. \)

There are two quality grades \( (m = 2) \) denoted by purple and white, where purple is superior to white. Therefore, we label them quality 1 and quality 2 as follows:

<table>
<thead>
<tr>
<th>quality 1</th>
<th>quality 2</th>
</tr>
</thead>
</table>

Thus, we have the qualities of the different segments given by \( u_1 = 1, u_2 = 2, u_3 = 1, \) and \( u_4 = 2. \) There is no segment of quality 0, as all the stock is useable.

We are given three standard lengths \( (n = 3) \), defined by \( l_1 = 1.0, l_2 = 1.5 \) and \( l_3 = 2.0: \)

**Standard Lengths**

<table>
<thead>
<tr>
<th>Standard Length 1</th>
<th>Standard Length 2</th>
<th>Standard Length 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0m</td>
<td>1.5m</td>
<td>2.0m</td>
</tr>
</tbody>
</table>

We can now calculate \( p = \lfloor L/l_1 \rfloor = \lfloor 6.6/1.0 \rfloor = 6. \) Thus, we can produce up to 6 items from this piece of stock.

The revenue matrix gives the revenue for producing an item of a particular standard length and quality:

<table>
<thead>
<tr>
<th>Standard 1</th>
<th>Standard 2</th>
<th>Standard 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1.00</td>
<td>$2.00</td>
<td>$6.00</td>
</tr>
<tr>
<td>$1.25</td>
<td>$3.00</td>
<td>$7.50</td>
</tr>
</tbody>
</table>

So, for example, we have \( r_{12} = $1.25 \) and \( r_{31} = $6.00. \)

Finally, the cost coefficient for waste is given by \( c = $4.00/m. \)
3.3.2 Optimal solution

To solve this example using our Xpress-MP implementation, we entered the parameters above into the input file given in appendix B.2. Using a desktop PC with a Pentium 4 processor, we obtained an optimal solution in under 1 second. The output generated is given in appendix B.3.

The solution was to make cuts in the locations marked with red lines:

This solution generates four items. Note that items 1 and 2 are not produced, as \( x_1 = y_1 = 0 \) and \( x_2 = y_2 = 0 \). Item 3 begins at \( x_3 = 0.1 \text{m} \) and ends at \( y_3 = 1.6 \text{m} \).

The solution specifies the quality of each item, with \( v_3 = 1 \), \( v_4 = 2 \), \( v_5 = 1 \), and \( v_6 = 2 \). We can confirm that these are correct by inspection. For example, item 5 spans qualities 1 and 2, and is correctly assigned a quality \( v_5 = 1 \).

The revenue for the items produced is summarised below:
The total revenue is $18.00. Now the only waste produced is the 0.1m chunk at the start of the stock (e.g. there is no waste between items 3 and 4, as $y_3 = x_4$). So the total cost of waste is $0.1m \times $4.00/m = $0.40. Thus:

$$\text{Total profit} = \text{revenue} - \text{cost} = $18.00 - $0.40 = $17.60.$$ 

An analysis of the constraints for this example problem is beyond the scope of this report, but it is easy to verify that all constraints hold.

**References**

A Model evolution

This appendix outlines the development of our model and how it changed over time.

First, we noted that the problem could be approximated as a discrete problem. This approach might have been useful had we needed to develop a heuristic method, but it would not always yield an optimal solution so we did not consider it further.

We discussed the option of using dynamic programming to model the problem, but we did not attempt it. We envisaged that a dynamic model would work as follows: Initially, we would consider all possible ways in which we could produce a single item from the stock. Then, at any stage of the dynamic process, we would produce one more item from the parts of the stock not already used. There would typically be an infinite search space at each stage, though we could probably make this finite by only placing items in ‘sensible’ locations.

The main concern we had with a dynamic programming approach was that it might be very difficult to make each decision. That is, at a particular stage, it might be difficult to decide where the next item should go. Furthermore, the problem of evaluating the objective function is still difficult.

Therefore, we pursued a linear programming approach. It was not difficult to write down our problem in English: maximise profit by deciding where to cut the stock, subject to items only being of allowable lengths. The main difficulty was to relate the objective function to the qualities of the items produced. We did this using the external binary variable $z_{ijk}$.

In its early stages, our model used only $z_{ij}$, which did not encapsulate the quality of item $i$. However, despite our objective function being incomplete, we were able to develop constraints enforcing standard lengths using $z_{ij}$. We then introduced the external variable $v_i$ to represent the quality of an item $i$, and constrained this based on the quality segments and cut locations. Then, in trying to relate $v_i$ back to the objective function, we expanded $z_{ij}$ to $z_{ijk}$.

Thus, the model presented in this report is the first and only model that we produced. As the model evolved it did not change so much as expand, as we added new variables and constraints.
B  Xpress-MP implementation

B.1  Mosel code

! File: industrial.mos
! Description: Model for solving the industrial problem
! The meaning of variables and constraints is best understood
! from the mathematical model. All letters have the same meaning.
! Copyright (c) 2004 Robert Chilov, Joshua Richmond, Derek Weeks

model "Industrial"
uses "mmxprs"

parameters
    INPUT  = "eg1.dat"
end-parameters

declarations
    n: integer
    m: integer
    q: integer
    p: integer
    L: real
end-declarations

! We must initialise the array sizes before we declare the arrays
initializations from INPUT
    n  m  q  p  L
end-initializations

declarations
    ! Input parameters
    l: array(1..n) of real
    u: array(1..q) of integer
    d: array(1..q) of real
    r: array(1..n, 1..m) of real
    c: real

    ! Decision variables
    x: array(1..p) of mpvar
    y: array(1..p) of mpvar

    ! External variables (depend on decision variables)
\[
\begin{align*}
z & : \text{array}(1..p, 1..n, 1..m) \text{ of mpvar} \\
v & : \text{array}(1..p) \text{ of mpvar} \\
a & : \text{array}(1..p, 1..q) \text{ of mpvar} \\
b & : \text{array}(1..p, 1..q) \text{ of mpvar}
\end{align*}
\]
end-declarations

initializations from INPUT
\[
\begin{align*}
l & \quad u & \quad d & \quad r & \quad c
\end{align*}
\]
end-initializations

 Objective function: maximise profit
\[
\begin{align*}
\text{revenue} & := \sum(i \text{ in } 1..p, j \text{ in } 1..n, k \text{ in } 1..m) \ r(j,k) \times z(i,j,k) \\
cost & := c \times (x(1) + \sum(i \text{ in } 1..p-1) \ (x(i+1) - y(i)) + L - y(p)) \\
\text{profit} & := \text{revenue} - \text{cost}
\end{align*}
\]

 Binary/integer constraints on external variables
\[
\begin{align*}
\forall(i \text{ in } 1..p, j \text{ in } 1..n, k \text{ in } 1..m) & \ z(i,j,k) \text{ is_binary} \\
\forall(i \text{ in } 1..p, s \text{ in } 1..q) & \ a(i,s) \text{ is_binary} \\
\forall(i \text{ in } 1..p, s \text{ in } 1..q) & \ b(i,s) \text{ is_binary} \\
\forall(i \text{ in } 1..p) & \ v(i) \text{ is_integer}
\end{align*}
\]

 The \(x(i)\) and \(y(i)\) vars must remain in a sensible order
\[
\begin{align*}
0 & \leq x(1) \\
\forall(i \text{ in } 1..p) & \ x(i) \leq y(i) \\
\forall(i \text{ in } 1..p-1) & \ y(i) \leq x(i+1) \\
\quad y(p) & \leq L
\end{align*}
\]

 Upper bound on the quality of each item
\[
\forall(i \text{ in } 1..p) \ v(i) \leq m
\]

 Each item has at most one standard length and quality
\[
\forall(i \text{ in } 1..p) \ \sum(j \text{ in } 1..n, k \text{ in } 1..m) \ z(i,j,k) \leq 1
\]

 These three constraints force the length of an item to be a standard length or 0
\[
\begin{align*}
\forall(i \text{ in } 1..p, j \text{ in } 1..n) & \ y(i) - x(i) \geq l(j) \times \sum(k \text{ in } 1..m) \ z(i,j,k) \\
\forall(i \text{ in } 1..p, j \text{ in } 1..n) & \ y(i) - x(i) \leq l(j) \times \sum(k \text{ in } 1..m) \ z(i,j,k) + \\
& \quad L \times (1 - \sum(k \text{ in } 1..m) \ z(i,j,k)) \\
\forall(i \text{ in } 1..p) & \ y(i) - x(i) \leq L \times \sum(j \text{ in } 1..n, k \text{ in } 1..m) \ z(i,j,k)
\end{align*}
\]

 If \(a(i,s) = 1\) and \(b(i,s) = 1\) then item \(i\) spans segment 2, so \(v(i) \leq u(s)\).
\[
\begin{align*}
\forall(i \text{ in } 1..p, s \text{ in } 1..q) & \ d(s) - x(i) \leq L \times a(i,s) \\
\forall(i \text{ in } 1..p) & \ y(i) - 0 \leq L \times b(i,1) \\
\forall(i \text{ in } 1..p, s \text{ in } 2..q) & \ y(i) - d(s-1) \leq L \times b(i,s) \\
\forall(i \text{ in } 1..p, s \text{ in } 1..q) & \ v(i) - u(s) \leq m \times (2 - a(i,s) - b(i,s))
\end{align*}
\]
Each $z(i,j,k)$ can only be 1 if item $i$ is of quality $k$ (i.e. if $v(i) = k$)
for all $(i \in 1..p, \ k \in 1..m) \ v(i) - k \leq m \ (1 - \sum (j \in 1..n) z(i,j,k))$
for all $(i \in 1..p, \ k \in 1..m) \ v(i) - k \geq -m \ (1 - \sum (j \in 1..n) z(i,j,k))$

If $v(i) = 0$ then the item spans unusable quality, so we need its length $y - x = 0$
for all $(i \in 1..p) \ y(i) - x(i) \leq L \cdot v(i)$

Optimisation statement
maximize(profit)

The remainder of the code produces human-readable output.

Display profit broken up into revenue and cost
write ("Profit = ", strfmt(getsol(revenue), 0, 2))
writeln(" - ", strfmt(getsol(cost), 0, 2))
writeln(" = $", strfmt(getobjval, 0, 2))
writeln

Print the locations of cuts
writeln("Cuts are made at:")
lastcut := 0.0
for all $(i \in 1..p)$ do
  ! Is the next $x(i)$ different to the position of the last cut?
  nextcut := getsol(x(i))
  if nextcut > lastcut then
    writeln(" ", strfmt(nextcut, 0, 2))
    lastcut := nextcut
  end-if

  ! Now check $y(i)$ the same way
  nextcut := getsol(y(i))
  if nextcut > lastcut then
    writeln(" ", strfmt(nextcut, 0, 2))
    lastcut := nextcut
  end-if
end-do
writeln

Print the properties of each non-zero item produced
for all $(i \in 1..p, \ j \in 1..n, \ k \in 1..m)$ do
  if getsol($z(i,j,k)$) = 1 then
    left := getsol($x(i)$)
    right := getsol($y(i)$)
writeln("Item ", i, ":")
writeln(" Start: ", strfmt(left, 0, 2))
writeln(" End: ", strfmt(right, 0, 2))
writeln(" Length: ", strfmt(l(j), 0, 2))
writeln(" Std Length: ", j)
writeln(" Quality: ", k)
writeln(" Revenue: ", strfmt(r(j, k), 0, 2))
writeln
end-if
end-do

! Print the total length of waste
total := getsol(x(1)) - 0.0
forall(i in 2..p) do
    total := total + getsol(x(i)) - getsol(y(i-1))
end-do
total := total + L - getsol(y(p))
writeln("Total waste: ", strfmt(total, 0, 2))
end-model

B.2 Example input file

! File: eg1.dat
! Description: Sample input data for the industrial problem
!
! Copyright (c) 2004 Robert Chilov, Joshua Richmond, Derek Weeks

q: 4 ! there are four segments
d: [1.6, 3.8, 4, 6.6] ! distances to the end of each segment
L: 6.6 ! this is simply the last entry of d
m: 2 ! there are 2 quality grades
u: [1, 2, 1, 2] ! quality grade of each segment
n: 3 ! there are 3 standard lengths
l: [1, 1.5, 2] ! l(j) = standard length j
p: 6 ! maximum number of items = floor( 6.6 / 1)
r: [1, 1.25, 2, 3, 6, 7.5] ! revenue matrix
    ! r[j,k] = revenue from an item of length l(j), quality k.
c: 4 ! cost of disposing of 1 unit length of waste
B.3 Example output file

Profit = 18.00 - 0.40
= $17.60

Cuts are made at:
0.10
1.60
3.60
4.60
6.60

Item 3:
Start: 0.10
End: 1.60
Length: 1.50
Std Length: 2
Quality: 1
Revenue: 2.00

Item 4:
Start: 1.60
End: 3.60
Length: 2.00
Std Length: 3
Quality: 2
Revenue: 7.50

Item 5:
Start: 3.60
End: 4.60
Length: 1.00
Std Length: 1
Quality: 1
Revenue: 1.00

Item 6:
Start: 4.60
End: 6.60
Length: 2.00
Std Length: 3
Quality: 2
Revenue: 7.50

Total waste: 0.10