PATTERN REDUCTION IN PAPER CUTTING

report submitted to

Indian institute of technology, madras

For the partial fulfilment of the course

of

Modelling in industry

by

Pawan Jiyalal Yadav

Diganta Naskar

M Sowmia

Shweta Saraf

Under the Supervision of

Prof.S. Sunder

Department of Mathematics

DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY, MADRAS

MARCH 2020

ACKNOWLEDGEMENT

This thesis owes its existence to the help, support and inspiration of several people. It brings us an immense pleasure to express our deepest sense of gratitude to our project **Supervisor Prof. S. Sunder**, Department of Mathematics, Indian Institute of Technology, Madras for his eminent guidance and to give us the freedom for choosing Pattern Reduction in Paper Cutting as a research topic for modelling in industry. His wonderful social behaviour made him a visionary and inspiring teacher and an ideal supervisor for our team.

We are ineffably indebted to our **Faculty Advisor-Prof. SivaramAmbikasaran**, Department of Mathematics, Indian Institute of Technology, Madras for his invaluable direction, encouragement, support and for providing us all the necessary facilities for carrying out this work well before time.

Finally, our deepest gratitude goes to our family for their unflagging love and unconditional support throughout our life and studies.

Date:

Place: IIT Madras

ABSTRACT

A large part of the paper industry involves supplying customers with reels of specified width in specified quantities. These 'customer reels' must be cut from a set of wider 'jumbo reels', in as economical a way as possible. The first priority is to minimize the waste, i.e. to satisfy the customer demands using as few jumbo reels as possible. Within the class of solutions having minimum waste, a secondary problem is to minimize the number of different patterns needed. The motivation here is to reduce the number of times that the knives have to be reset on the paper-cutting machine.

Keywords

$\{r_i\}_{i=1}^{s}$ - distinct customer reel width**.**

$\{d_i\}_{i=1}^{s}$ - corresponding quantities of customer reels to be produced.

Contents

1. General Introduction

The paper cutting problem is an example of the “one-dimensional cutting stock problem". Customers order a number of reels that have to be cut from larger reels. The latter reels are called “jumbo rolls". The problem is to keep the waste as small as possible. Fortunately, there are “good" algorithms for solving this minimum-waste problem for paper cutting.

However, there is an extra problem in practice. The jumbo rolls are cut into smaller reels by slitting knives which must be moved each time a new arrangement of customer reels is to be cut from a jumbo roll. The reels of paper can move through some of the modern paper cutting machines at a rate of 60 miles/hour. Since this process is done at such a high speed, re-positioning the slitting knives slows the process down considerably and is therefore undesirable.

The Sub-Problem of Pattern Reduction

The goal of the pattern reduction problem is to reduce the number of times the slitting knives must be repositioned. This is done by taking a minimum-waste solution and then trying to reduce the number of patterns used. Of course, this reduces the number of slitting knives that must be repositioned. It is important that we start by taking a minimum-waste solution, because the minimum-waste is more important than the reduction of patterns. Therefore, it is not an option to have fewer patterns at the expense of not having a minimum-waste solution.

In general, for a minimum waste solution it is too computationally expensive to find the minimum number of patterns that can be used to cut the jumbo rolls. This is the reason that one tries to use heuristics to reduce the number of patterns in a given minimum-waste solution.

2. Problem Statement

In this report we consider two problems of so-called paper cutting problem.

1. Cutting Stock – minimal waste solution
2. Pattern reduction in a given minimal waste solution

1. CUTTING STOCK

Input: positive integer J (the jumbo width), distinct positive integers $r_1, r_2 \ldots\ldots, r_s$ (the various customer reel widths), and positive integers $d_{1,} d_2 \ldots\ldots, d_s$(the quantity of each customer width that must be produced).

Task: use as few jumbos of width J as possible to satisfy the demand for d_α customer reels of width r_α (for each α = 1…., s).

A solution to CUTTING STOCK consists of a series of 'patterns', each corresponding to a different (unordered) set of customer reels that may be cut from a single jumbo.

The solution also specifies how many jumbos must be cut according to each pattern to satisfy all demands. Typically, the number of patterns in a solution are in the range 5-30, with occasional large problems of up to 80 patterns. Each pattern consists of 2-10 customer reels and is used for 1-1.5 jumbos.

Within the class of solutions having minimum waste, a secondary problem is to minimize the number of different patterns needed. The motivation here is to reduce the number of times that the knives have to be reset on the paper-cutting machine. The specification now becomes

2. PATTERN REDUCTION

Input: positive integer J, distinct positive integers $r_1, r_2 \dots\dots, r_s$ and positive integers $d_{1,} d_2 \dots\dots, d_s$

Task: find a minimum-waste solution to the corresponding instance of CUTTING STOCK, which further minimizes the number of patterns used.

3. Cutting stock – minimal waste solution

The successful solution of large-scale mixed integer programming (MIP) problems requires solving linear programming (LP) relaxations that have huge number of variables. One of several reasons for considering LP with huge number of variables is that compact formulation of a MIP may have a weak LP relaxation. Frequently the relaxation can be tightened by using huge number of variables.

3.1 Cutting stock problem

The cutting stock problem arises from many physical applications in industry. For example: in a paper mill, there are a number of rolls of paper of fixed width waiting to be cut(these rolls are called rows), yet different manufacturers want different numbers of rolls of various-sized widths (these rolls are called finals). How should the rolls be cut so that the least amount of left-overs are wasted? or rather, least number of rolls are cut? This turns out to be an optimization problem, or more specifically, an integer linear programming problem.

For solving the problem, the first thing we need is a formulation of problem as an integer program. One model can be formed as follows.

- Problem of cutting an unlimited number of pieces of material (paper rolls, for instance) of width jto produce$d_i$pieces of width r_i, i= 1, 2, …, s.
- The objective is to minimize the number of pieces of material to meet the demands.
- **Note:**
  minimize number of pieces ≡ minimize total waste
- Cutting pattern P_j(j = 1, 2, …, NJ) corresponds to a specific way of cutting a piece of material:
  a_{ij}= number of pieces of width $r_i$produced with cutting patternP_j

where $a_{ij} \geq 0$ and integer, $i = 1, 2, \ldots, s$

and $\sum_{i=1}^{s} a_{ij}\, r_i \leq j$

Mathematical Model:

$$\text{Min} \sum_{j=1}^{NJ} x_j$$

$$\text{Subject to:} \sum_{j=1}^{NJ} a_{ij}\, x_j \geq d_i \quad , \text{i} = 1, 2\ldots, \text{s}$$

$$x_j \geq 0, \text{j} = 1, 2\ldots, \text{NJ}$$

where x_j represents the number of pieces of material cut with the pattern j.

4. Pattern reduction in a given minimum waste solution

The goal of the pattern reduction problem is to reduce the number of times the slitting knives must be repositioned. This is done by taking a minimum-waste solution and then trying to reduce the number of patterns used. Of course, this reduces the number of slitting knives that must be repositioned. It is important that we start by taking a minimum-waste solution, because the minimum-waste is more important than the reduction of patterns. Therefore, it is not an option to have fewer patterns at the expense of not having a minimum-waste solution.

In general, for a minimum waste solution it is too computationally expensive to find the minimum number of patterns that can be used to cut the jumbo rolls. This is the reason that one tries to use heuristics to reduce the number of patterns in a given minimum-waste solution.

Suppose a customer wants reels of the following lengths: 1.9m, 1.9m, 1.95m, 1.95m. A minimum-waste solution for jumbo rolls of length 4m is given on the left side of Figure 1. However, another minimum-waste solution is given on the right part of the same figure and this latter solution requires only one pattern. This reduces the number of patterns from two to one and is therefore called a 2→1 reduction. Note that this reduction is always possible: there is no extra check required to see whether the new pattern fits into a jumbo roll.

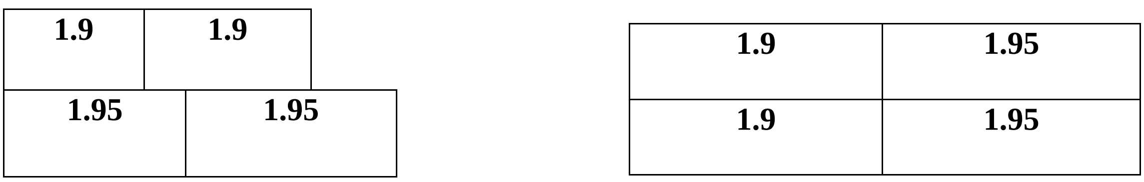

Figure 1: A 2 → 1 reduction

A slightly more complicated example is given in Figure 2 where three patterns are reduced to two. Because of the similarity with a stair-case this is known as a 3 → 2 stair-case reduction. Note that this type of reduction is not always possible, it is necessary to check that the pattern AD fits onto a jumbo roll. These 3 → 2 reductions are quite easy to detect and the more general form of them along with search techniques for finding them are given in Allwood and Goulimis.

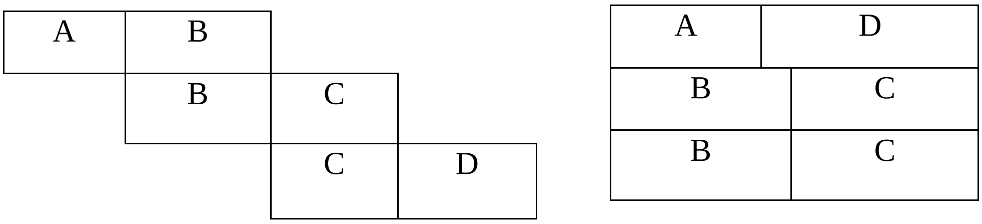

Figure 2: 3→ 2 reduction

Using these simple rules, it is often possible to reduce the number of patterns in a minimum-waste solution. Of course, we would like to know how far we are from the minimum number of patterns required for the minimum-waste solution. Therefore, we would like to have some kind of lower bound on the number of patterns needed. The derivation of a lower bound is not that difficult.

Suppose that for a given paper cutting problem the minimum-waste solution requires N jumbo rolls. Suppose also that we have a minimum-waste solution which uses the minimum number of patterns, n, where 1 ≤ n ≤ N. From these n patterns we can obtain n new patterns just by removing all multiple entries of a customer reel. Then we are left with n patterns in which each customer reel occurs exactly once in precisely one of the n patterns. Letr_i,i=1…., sdenotes the widths of the different customer reels and let W denote the width of a jumbo roll. Since all the different customer reels occur in these n patterns, we have $\sum_{i=1}^{s} r_i/W \leq$ n

Some new algorithms

We describe some new algorithms that are not too difficult to implement, run fast enough to meet the requirements and give useful results. Section 3.1 describes an algorithm for finding all 2 → 1 reduction. In Section 3.2 we develop an algorithm for 3 → 2 reductions. The results are presented in Section 2.3. We conclude this section with some remarks concerning 4 → 3 reductions.

4.1 An algorithm for finding 2 → 1 reduction

After we get an optimal set of patterns, our next aim is to check if these new patterns can be reduced further to a lesser no. keeping the waste same and meeting the demand as well. For that we can study all sorts of reduction like 2 → 1, 3 → 2, 3 → 1, etc.

Let us understand 2 → 1.

In this, we choose any 2 patterns out of optimal set of patterns and see if they can be further reduced to 1 or not. If so, under what circumstances.

This can be seen by the algorithm below.

Steps:

1) Choose a pair of patterns from Optimal Set of Patterns.
2) Check if these 2 patterns can be reduced to just one pattern.
3) If there is no reduction, take the next pair of patterns and go to step 2.
4) In case of reduction, stop now and restart the algorithm.

Let us understand Step 4.

Suppose we've P_1, P_2, P_3...., P_k such optimal patterns. Initially we pick P_1 and P_2 and see if reduction is happening. If not, next choose P_1 and P_3, else suppose P_1 and $P_2$is reduced to some pattern P_{12} then stop the algorithm and restart the algorithm. Again, we start by choosing P_{12} and P_3.

But the most crucial step of our algorithm is Step 2 which checks whether reduction is happening or not.

For that, suppose there are n_1 jumbo rolls using P_1 and n_2 jumbo rolls using P_2.

Let n_3 = n_1 + n_2

Suppose we're getting a reduced pattern P_{12}, n_3 times.

We build a bucket, put every element of P_1 exactly n_1 time and every element of P_2 exactly n_2 times and sort the contents of the bucket width wise.

If every distinct element of this bucket appears a multiple of n_3 times then only a reduction is happening i.e. P_{12} exists and the new pattern P_{12} consists of every n_3^{th} element of the sorted bucket. But why we want every distinct element of the bucket a multiple of n3 times is because otherwise we will have to reposition the knife which we don't want to do.

Let us understand this with the help of 2 examples.

Let Width J of Jumboroll: 4 m

Case I: When reduction isn't happening

Take n_1 = 3, n_2 = 1

Then n_3 = 4

P_1: 1.9 m, 1.9 m

P_2: 1.95 m, 1.95 m

Here 1.9 m is appearing 6 times whereas 1.95 m is appearing 2 times. And since none of them are multiple of 4 so reduction isn't possible.

Case II: When reduction is happening

Take n_1 = 1, n_2 = 1

Then n_3 = 2

P_1: 1.9 m, 1.9 m

P_2: 1.95 m, 1.95 m

Here both 1.9 m and 1.95 m are appearing 2 times. And since both of them are multiple of 42so reduction is possible.

And the new reduced Pattern P_12: 1.9 m, 1.95 m will utilise n3= 2 Jumbo rolls.

4.2 An algorithm for some 3 → 2 reduction

In general, a 3 → 2 reduction reduces n_1 jumbo rolls cut to Pattern 1, n_2 cut to Pattern 2 and n_3 cut to Pattern 3 to n_4 jumbo rolls cut to Pattern 4 and $n_5$cut to Pattern 5 where $n_1 + n_2 + n_3 = n_4 + n_5$. The algorithm described here handles only those cases where $n_1 = n_2 = n_3$, $n_5 = n_1$ and $n_4 = 2 * n_5$.

So, in the first step we form submatrices of the matrix of data containing the minimum waste solution. This is done so that the repetition numbers of the patterns in each of the submatrices are equal. (The repetition number is the number of jumbos to be cut to that pattern.) As we are searching for a 3 → 2 reduction, only those submatrices consisting of three or more lines are of any interest. In the second step we are searching for a 3 → 2 reduction in each submatrix. Within any submatrix we may assume that the patterns in the submatrix are used only once, since the repetition number does not affect the operation of the algorithm once the submatrices have been formed. The outline of the algorithm is quite similar to the case of the 2 → 1 reduction:

1. Select the first submatrix (that with smallest repetition numbers);

2. Take the first triple of patterns of the submatrix;

3. Check if a reduction is possible;

4. If there is no reduction, take the next triple and go to Step 3;

5. If there is a reduction stop searching in this submatrix, select the next submatrix and go to Step 2;

6. If there was a reduction in any submatrix, restart whole algorithm.

To illustrate the workings of the algorithm we present the following example with jumbo roll width = 300:

Pattern 1: 50 50 90 90

Pattern 2: 85 85 105

Pattern 3: 90 105 105

To see if there is a reduction one first puts the three chosen patterns all together into one “bucket” and sorts the reels by width. In our example this gives:

bucket = 50 50 85 85 90 90 90 105 105 105

The “bucket” is to be split between three jumbo rolls, two are to be cut to Pattern 4 and one to Pattern 5. If there is a reduction, then every customer reel cut in Pattern 4 will be produced an even number of times by this pattern. Hence every reel that occurs only once or an odd number of times in the “bucket” must appear in Pattern 5 at least once. So we “extract modulo 2” from the bucket and place these customer reels in Pattern 5.

For example:

bucket = 50 50 85 85 90 90 105 105

Pattern 5 = 90 105

If the length (i.e. the sum of the customer reel widths) of Pattern 5 is larger than the jumbo roll width, then we can immediately stop as there is no reduction. Otherwise, as above, the length of Pattern 5 is smaller than the jumbo roll width.

As every reel in the bucket now appears there an even number of times, we can construct Pattern 4 by taking every second reel of the bucket. Note that Pattern 4 may actually be too long to fit into a jumbo roll at this stage.

Pattern 4 = 50 85 90 105

Next, we calculate the total waste in the original three patterns. In this example it is 45. In any new solution with a reduced number of patterns we will cut the same number of customer reels from the same number of jumbo rolls so the total waste is a constant. If the length of Pattern 5 plus the total waste is greater than or equal to the width of a jumbo roll, we are done and we have a reduction with the new patterns Pattern 4 and Pattern 5. (In the case of equality Pattern 4 completely fills a jumbo roll, otherwise there is some waste in Pattern 4.)

Most often at this stage the length of Pattern 5 is too small, or equivalently, Pattern 4 is too big to fit into a jumbo roll. This is the case in our example, ($90 + 105 + 45 = 240 < 300$). So, the next thing to try is to move customer reels from Pattern 4 into Pattern 5. Since Pattern 4 is cut twice as many times as Pattern 5, any reel moved from Pattern 4 into Pattern 5 will necessarily appear there twice. We hope that after this rearrangement the length of Pattern 5 is still smaller than (or equal to) the width of a jumbo roll and the length of Pattern 5 plus the total waste is larger than or equal to the width of a jumbo roll. In our example we are lucky, since we can take 50 from Pattern 4 and put it into Pattern 5 (where it must appear twice) to give:

Pattern 4: 85 90 105

Pattern 5: 50 50 90 105

Obviously $50 + 50 + 90 + 105 = 295 \leq 300$ and $295 + 45 = 340 \geq 300$. So, in this case we found a pattern reduction. If moving one customer reel from Pattern 4 into Pattern 5 still does not produce a solution, then we try to move a pair of reels from Pattern 4 into Pattern 5. One could also try to move triples and quadruples and so on from Pattern 4 to Pattern 5 but the computation time increases rapidly. Restricting ourselves to pairs of customer reels means that the algorithm is not too difficult to implement and computation time is still reasonable. Also, as there are only between three and eight customer reels per pattern for most paper cutting problems it is reasonable to stop after considering pairs of elements. It is advisable to restart the algorithm if a reduction is found, since it is possible that the new patterns can be combined with other (old) patterns to find even more pattern reductions. Hence the last step of the algorithm.

4.3An outline for a general 3 → 2 algorithm:

We are ready to present an algorithm that will find 3 → 2 pattern reductions with no restrictions on the number of jumbo rolls that are cut according to each pattern. It is thus an extension of the algorithm as described in previous section. Although there are no restrictions on the number of rolls cut, the time-consumption of the algorithm will increase with increasing number of jumbo rolls cut.

The algorithm takes three patterns and also requires the number of rolls that are cut to each of these patterns. We can write this in the following form:

No of jumbo rolls	Width of customer reels in pattern
n_1	A B C D …..
n_2	E F G H …..
n_3	I J K L …..

With this input, we make a (sorted) list of all the customer reels where we also record the number of customer reels required with each of these widths. We make sure that all customer reels of equal width are collected together. Say, for example that A and B are reels of equal width, our list of widths would look like this:

A	C	D	E	F	……
2*n_1	n_1	n_1	n_2	n_2	……

If there exists a pattern reduction, it must be of the form:

No. of jumbo rolls	Widths of customer reels in pattern
η_1	M N O P …
η_2	Q R S T …

where $\eta_1 + \eta_2 = n_1 + n_2 + n_3$, i.e. the same number of jumbo rolls are used. This preserves the minimum waste solution. If we call the total number of jumbo rolls that are cut $n_{tot} = n_1 + n_2 + n_3$, we see that the numbers η_1 and η_2 take on only a distinct number of values, namely:

η_1=	n_{tot}	n_{tot} -1	n_{tot} -2	……	0
η_2=	0	1	2	…..	n_{tot}

Setting η_1 = 0 or η_2 = 0 would give a 3 → 1 reduction. We easily see that we do not need to check more than half of the combinations because of the symmetry.

E.g. η_1 = 5 and η_2 = 3 would find the same pattern reduction as η_1 = 3 and η_2 = 5. So we can assume that in the following steps that $\eta_1 \geq \eta_2$.

We make two new lists, "Line X" and "Line Y", corresponding to η_1 and η_2 jumbo rolls respectively by taking all reels which appear as multiples of n1 from our complete list of customer reels and putting them in Line X. The leftovers are put in Line Y.

Suppose η_1 = 3 and η_2 = 2. We could have:

Customer reel widths	A	B	C	D	E	…..
Line X: η_1 =3 Line Y: η_2 =2	3 0	6 2	3 1	0 1	3 0	…… ……

Now, all entries of Line X are multiples of η_1. We must also make all entries of Line Y multiples of η_2, so we do the following for each entry corresponding to a customer reel in Line Y:

- Check if it is a multiple of η_2. If it is then go to the next entry. If not, go on to the next step.
- Subtract η_1 reels from Line X and add them to Line Y. Go back to the previous step. If there are no reels left in Line X to subtract, then no reduction is possible given these values of η_1 and η_2. (There is no point in trying to subtract from Line X and add to Line Y if η_1 is a multiple of η_2 (and $\eta_2 \neq 1$), that is if η_1 = k * η_2 for some integer k).

If we use our previous example, doing the above would give us:

Customer reel widths	A	B	C	D	E	…..
Line X: η_1 =3 Line Y: η_2 =2	3 0	6 2	0 4	0 1	3 0	…… ……

Here the reduction is not possible.

Thus, both Line X and Line Y are multiples of η_1 and η_2, respectively. Let x denote the least common multiple of η_1 and η_2. We must preserve the property that all elements in Line X are a multiple of η_1 and that all elements in Line Y are a multiple of η_2. So, the only alterations that can be made to Lines X and Y now are those which take multiples of x copies of some customer reel width from Line X and add them to Line Y.

Let the total waste of the three original patterns be $waste_{tot}$, and let the length of one jumbo roll be l_{jumbo}. We define the function sum to be the sum of all the widths of customer reels. For example, in the previous case sum (Line X) = 3 * A + 6 * B + 3 * E + …Also, let waste (Line Y) be n2 * l jumbo sum (Line Y).

The algorithm for finding a possible reduction is now:

1. If sum (Line Y) >η_2 * l_{jumbo} then no reduction is possible given these values of η_1 and η_2. (Line Y will never fit into η_2 jumbo rolls).

2. If sum (Line Y) ≤ η_2 *l_{jumbo} and waste (Line Y) ≤ $waste_{tot}$ we have a reduction. (The Line Y does fit into η_2 jumbo rolls and since the waste is less than the total waste there has to be some waste in Line X too. Thus, Line X fits into η_1 jumbo rolls.)

3. If Steps 1 and 2 give no result, try moving x copies of some customer reel from Line X to Line Y. Start with the smallest customer reels. Do Steps 1 and 2 again. If there is no result, put the customer reels back into Line X and try moving another bundle of x customer reels. To find all possible reductions, one must try all combinations of customer reels in Line X. So, the algorithm is more practical with an end criterion. For example, this could be to stop after trying all combinations of two customer reels.

Test Problems And Computational Results

We have provided several real problems, which have been used to make a preliminary evaluation of the effectiveness of the proposed algorithm. The input to each problem was a set of patterns generated by a waste-minimization algorithm accompanied by preliminary pattern reduction, although we found that not all 2 → 1 and 3 → 2 staircase reductions had been made.

Some samples are given below, giving just the input and final output, together with the number of moves used. Some of the lengths have been rescaled to reduce the numerical values of the r_α and J. Execution times were well within the allocated computing budget, with typical problems requiring only a few seconds on a modest PC.

The scope for pattern reduction seems sensitive to the waste percentage of the input. If the waste percentage is small, say below 1%, then there are generally few feasible moves available, but if the percentage approaches 5% then the possible pattern reductions can be quite dramatic.

PROBLEM 1:

Jumbo size: 870

customer	170	180	200	220	250	260	266

reels							
demand	1	1	1	1	1	1	1

Using minimal waste algorithm:

THE JUMBO WIDTH=870

THE OPTIMAL SET OF PATTERNS GENERATED BY MATLAB SOLVER IS:

PATTERN: 21 [266, 266, 266] repeats 1.0

PATTERN: 37 [250, 260, 260] repeats 1.0

PATTERN: 63 [180, 220, 220, 250] repeats 1.0

PATTERN: 71 [170, 200, 250, 250] repeats 1.0

PATTERN: 151 [170, 180, 200, 266] repeats 1.0

THE TOTAL NUMBER OF JUMBOS USED= 5.0

THE TOTAL WASTAGE= 226.0

THE TOTAL PERCENTAGE OF WASTAGE= 5.195402298850575 %

Using Pattern reduction algorithm:

AT THE END OF A 3-2 PATTERN REDUCTION ENCOUNTERED, WE HAVE:

PATTERN [266, 266, 260] repeated 2

PATTERN [170, 250, 250, 180] repeated 2

PATTERN [180, 180, 220, 220] repeated 1

PROBLEM 2:

Jumbo size: 4350

Customer reels	850	900	1000	1100	1250	1270	1300	1330	1340	1345
demand	4	4	4	4	8	1	4	8	1	1

Using minimal waste algorithm:

THE JUMBO WIDTH=4350

THE OPTIMAL SET OF PATTERNS GENERATED BY MATLAB SOLVER IS:

PATTERN: 18 [1250, 1330, 1330] repeats 1.0

PATTERN: 66 [1300, 1300, 1345] repeats 1.0

PATTERN: 131 [850, 900, 1250, 1330] repeats 1.0

PATTERN: 139 [1270, 1300, 1340] repeats 1.0

```
PATTERN: 224 [900, 900, 900, 1330] repeats 1.0
PATTERN: 294 [1300, 1330, 1330] repeats 1.0
PATTERN: 323 [850, 1100, 1100, 1250] repeats 1.0
PATTERN: 336 [850, 1000, 1250, 1250] repeats 2.0
PATTERN: 355 [1000, 1000, 1100, 1250] repeats 1.0
PATTERN: 362 [1100, 1330, 1330] repeats 1.0
THE TOTAL NUMBER OF JUMBOS USED= 11.0
THE TOTAL WASTAGE= 2655.0
THE TOTAL PERCENTAGE OF WASTAGE= 5.54858934169279 %
```

Using Pattern reduction algorithm:

```
AT THE END OF A 3-2 PATTERN REDUCTION ENCOUNTERED, WE HAVE:
PATTERN [1250, 1300, 1330] repeated 4
PATTERN [1270, 1340, 1345] repeated 1
PATTERN [850, 1000, 1100, 1330] repeated 4
PATTERN [900, 1250] repeated 2
```

PROBLEM 3:

Customer reels	16	17	18	20	21	24	27	28	30	31	36	38	41	43	45
demand	**6**	**4**	**21**	**10**	**38**	**30**	**14**	**27**	**1**	**10**	**21**	**14**	**17**	**2**	**10**

Using minimal waste algorithm:

```
THE JUMBO WIDTH=85
THE OPTIMAL SET OF PATTERNS GENERATED BY MATLAB SOLVER IS:
PATTERN: 33 [16, 24, 45] repeats 5.0
PATTERN: 64 [41, 43] repeats 2.0
PATTERN: 141 [24, 24, 36] repeats 7.0
PATTERN: 168 [18, 31, 36] repeats 4.0
PATTERN: 215 [20, 24, 41] repeats 10.0
PATTERN: 242 [18, 28, 38] repeats 9.0
PATTERN: 346 [17, 27, 41] repeats 4.0
PATTERN: 362 [28, 28, 28] repeats 1.0
```

PATTERN: 421 [21, 28, 36] repeats 10.0

PATTERN: 458 [16, 28, 41] repeats 1.0

PATTERN: 529 [21, 21, 21, 21] repeats 6.0

PATTERN: 547 [24, 30, 31] repeats 1.0

PATTERN: 553 [27, 27, 31] repeats 5.0

PATTERN: 596 [38, 45] repeats 5.0

PATTERN: 643 [18, 18, 21, 28] repeats 4.0

THE TOTAL NUMBER OF JUMBOS USED= 74.0

THE TOTAL WASTAGE= 11.0

THE TOTAL PERCENTAGE OF WASTAGE= 0.17488076311605724 %

Using Pattern reduction algorithm:

For the above example , we have not found any pattern reduction

Conclusions

We have first obtained the minimal waste solution in the paper cutting problem for a given set of customer orders. Then we have seen the level of human intervention in the paper cutting process. Our solution comes in the two parts summarised below.

1. From the obtained minimum-waste solution, we construct a new minimum-waste solution that requires fewer patterns by using one of our algorithms or a combination of them .

2. We do a smart reordering within the set of patterns to reduce the number of blade changes. Together, these algorithms can restrict the level of human involvement in the paper cutting process. When implemented on a computer, these algorithms could be of great help to paper cutting companies.

Code:

```
            # determine feasible set of patterns-PROBLEM1
                    #ASSUME JUMBO WIDTH IS 870m
import itertools
from itertools import chain, combinations
            #CUSTOMER WIDTHS AND DEMAND
r=[170,180,200,220,250,260,266]
d=[2,2,2,2,4,2,4]
min_width=min(r)
max_subset_len=int(870/min_width)
def powerset(iterable):
    s = list(iterable)
    return  chain.from_iterable(combinations(s,  r)  for  r  in
range(1,max_subset_len+1))
def patterns():
    l=[]
i=0
all_patterns=[]
    for j in d:
for k in range(j):
l.append(r[i])
i=i+1
all_patterns=list(powerset(l))
feasible_patterns=[]
    for i in range(len(all_patterns)):
        if(sum(all_patterns[i])<=870):
feasible_patterns.append(all_patterns[i])
feasible_patterns=[list(x)  for  x  in  set(tuple(x)  for  x  in
feasible_patterns)]
    return(feasible_patterns)
print("set of all fesible patterns are:")
list1=patterns()
```

```
for i in range(len(list1)):

print("the pattern",i+1,"is:",list1[i])
```

#CREATING OCCURANCE MATRIX - ie NO OF OCCURANCES OF WIDTH r_i in pattern j AND EXPORTING TO MATLAB

```
import numpy

import numpy, scipy.io

def pattern_matrix():

    pattern=patterns()

count_matrix=numpy.zeros(((len(pattern),len(r))))

    for i in range(len(pattern)):

        for j in range(len(r)):

count_matrix[i][j]=pattern[i].count(r[j])

    return(count_matrix)

print(pattern_matrix())

count=pattern_matrix()

numpy.savetxt('occurance_matrix1.mat',count)
```

#PROBLEM-1 -optimal soln matlab code

%all set of patterns imported from python

```
my_patterns = load('occurance_matrix1.mat', '-ASCII');

%now using matlsb's optimization tool to solve the optimisation problem

%Write the objective function vector and vector of integer variables.

f=ones([1,size(my_patterns,1)]);

intcon=1:size(my_patterns,1);

%Write the linear equality constraints.

my_patterns1=transpose(my_patterns);

%there are no inequality constraints.so A and b are empty matrices ([]).

A=[];

b=[];
```

```
%creating matrices for equality constrints
Aeq=my_patterns1;
beq=[2;2;2;2;4;2;4];
%Each variable is bounded below by zero.
lb = zeros(1,size(my_patterns,1));
% No upper bound on variables.
%call the solver.
lb = zeros(size(my_patterns,1),1);
[x,fval] = intlinprog(f,intcon,[],[],my_patterns1,beq,lb);
%printing the solution
disp(x);
disp(fval);
optimal_set=[];
reps=[];
for i=1:size(x,1)
    if x(i)~=0
optimal_set=[optimal_set,i];
        reps=[reps,x(i)];
    end
end
disp(optimal_set)
disp(reps)
save prob1_optimal.mat optimal_setreps ;
```

#IMPORTING MINIMAL WASTE SOLUTION RETURNED BY MATLAB FOR PATTERNS REUDCTION

```
import scipy.io as sio
import numpy as np
import math
mat_contents=sio.loadmat('prob1_optimal.mat')
sorted(mat_contents.keys())
optimal_set=np.array(mat_contents['optimal_set'])
repetitions=np.array(mat_contents['reps'])
repetitions=[round(i) for i in repetitions[0]]
```

```
print("THE JUMBO WIDTH=870 \n")

print("THE OPTIMAL SET OF PATTERNS GENERATED BY MATLAB SOLVER IS:\n")

j=0

used=0

for i in optimal_set[0]:

    print("PATTERN:",i,list1[i-1],"repeats",repetitions[j])

    used+=sum(list1[i-1])*repetitions[j]

    j=j+1

print("\n THE TOTAL NUMBER OF JUMBOS USED=",sum((repetitions)),"\n")

print("THE TOTAL WASTAGE=",-used+sum(repetitions)*870,"\n")

print("THE TOTAL PERCENTAGE OF WASTAGE=",((-used+sum(repetitions)*870)*100/(sum(repetitions)*870)),"%\n")
```

#PROBLEM-1 PATTERN REDUCTION(3 to 2 pattern reduction algorithm)

```
optimal_patterns1=[]

for i in optimal_set[0]:

    optimal_patterns1.append(list1[i-1])

sets=[2,1]

for t in sets:

    select=optimal_patterns1

    ele_3=list(combinations(select, 3))

    reduced=1           #boolean value to detect if there is a reduction or not

    merge=ele_3[t][0]+ele_3[t][1]+ele_3[t][2]

    pattern5=[]

    l=0

    while(l<len(merge)):

        if(merge.count(merge[l])%2==1):

            pattern5.append(merge[l])

merge.pop(l)

        else:
```

```
            l+=1
    pattern5
    if(sum(pattern5)>870):
        reduced=0                      #means no reduction is possible
    pattern4=[merge[2*i] for i in range(int(len(merge)/2) )]# taking every second occuring element in patter4
    if(sum(pattern4)>870):
        less=sum(pattern4)-870
ket=0
        while(ket<less):
ket+=pattern4[0]
ans=pattern4.pop(0)
            pattern5.append(ans)
            pattern5.append(ans)
index_remove=[optimal_patterns1.index(ele_3[t][n])  for  n  in range(3)]
    optimal_patterns1.pop(index_remove[0])
    optimal_patterns1.pop(index_remove[1]-1)
    optimal_patterns1.pop(index_remove[2]-2)
repetitions.pop(index_remove[0])
repetitions.pop(index_remove[1]-1)
repetitions.pop(index_remove[2]-2)
    optimal_patterns1.append(pattern4)
    optimal_patterns1.append(pattern5)
repetitions.append(2)
repetitions.append(1)
print("AT THE END OF  A 3-2 PATTERN REDUCTION ENCOUNTERED,WE HAVE:\n")
    for i in range(len(optimal_patterns1)):
        print("PATTERN",optimal_patterns1[i],"repeated ",repetitions[i])
    print("\n\n")
```