An algorithm for corrugated paper cutting

DAMIR KALPIC, VEDRAN MORNAR, KRESIMIR FERTALJ
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3, 10000 Zagreb
CROATIA

Abstract: - An automated machine cuts the rolls of corrugated paper longitudinally and splits the paper stripe into multiple conveyors, where in each of them a different equidistant lateral cut can be applied. There is a choice of input rolls of infinite length but different widths. The market requirement for large series of different rectangle-shaped articles has to be met. Upper limits for the articles also exist. The minimum material consumption is the objective of optimisation. A recursive function to generate all the possible cutting schemas is written. It provides for formulation of a linear programming model. The minimisation of machine set-up costs cannot be practically solved by binary variables because of the prohibitive problem size. Instead, an iterative navigation around the achieved optimum solution, using the dual activity values is devised.

Key-Words: - Cutting stock problem, Linear programming, Integer programming, Production planning, Duality

1 Introduction

A factory producing corrugated paper packages applies an automated cutting machine. The input roll of paper is cut longitudinally into at most Q stripes. Adjacent stripes can further proceed through T conveyors. On each of these conveyors the belonging stripes are cut laterally and therefore the articles, which are cut on each conveyor, have the common length as shown in Figure 1.

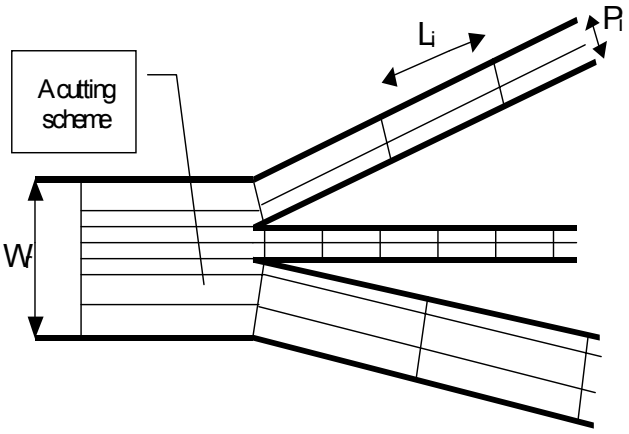

Fig. 1 Cutting machine with 3 conveyors

2 Problem Formulation

The company using described machines has produced the cutting schemes manually and asked for advice weather it could be automatic. The authors considered the problem and the first conceivable solution led to mixed integer programming. The requirements for integer variables derive from the discrete number of schemas to be cut on one hand and from fixed charge property of the cutting technology. A certain minimum length of paper stripe is indispensable for the process to be technologically feasible but also financially, due to the incurred fixed costs required for the machine set up. The idea of integer programming was quickly abandoned due to excessive computational complexity of such models. Rounding off is acceptable since the counts of schemas are generally rather large numbers. The problem of fixed charge is approached via iterative modifications of the linear programming model, as shall be explained further on. Some basic data structures that can be found in any proper textbook [1] were applied to support data input and storage. A recursive algorithm to produce possible cutting schemas was developed. The schemas are mapped into a linear programming model input file aimed for our proprietary software LPE [2]. Iterative interactive iterations are devised to produce a feasible and acceptable cutting plan.

2.1. Input data, sets and lists

The first set of input data consists of the widths of available input corrugated paper rolls. From these data, an ordered set $\boldsymbol{R}$ $\{W_r\}$ is formed, where the length of each roll is assumed to be infinite and the width is W_r, $r = 1\ldots nr$ and $W_r > W_{r+1}$ for $r = 1\ldots nr\text{-}1$. The input roll's usable width is reduced by value WST, an inherent waste due to the applied technology.

The next group of input data consists of triplets: required number of articles, article width and article length. The articles to be produced form the set of pairs $\boldsymbol{P}$ $\{P_p, L_p\}$ for $p = 1 \ldots np$, where P_p is the width and L_p is the length of article p. The production plan requirements form the set $\boldsymbol{B}$ $\{b_p\}$ of the same cardinality np.

2.2. Linear program

The selection of appropriate cutting schemas, such that they shall satisfy the market requirements and consume minimum possible paper area, proceeds by application of linear programming. The linear programming model is constructed from the input data.

2.2.1 Decision variables

n_p is the number of articles p to be produced.
S_{sr} is the number of repetitions for schema s, cut from roll r.

2.2.2 Constraints

The required number of articles must be produced:
$n_p \geq b_p \quad ; \quad p = 1 \ldots np$

The number of excessive articles to be produced is limited by a tolerance that can depend upon the required production quantity and/or upon the article, and it can be generally defined as $T_p \geq 1$.
$n_p \leq T_p \, b_p \quad ; \quad p = 1 \ldots np$

2.2.3 Objective function

The principal objective is to minimise the whole area of used paper.

min z = *area of the consumed paper* $+ z'$

The area of the consumed paper z is calculated within the schema generation algorithm in a later chapter.
z' is a corrective term added to the objective function, aimed to stimulate the production of excessive articles out of the paper that would otherwise be wasted:

$$z' = - f \cdot \sum_p n_p \cdot P_p \cdot L_p / \; W_1$$

The stimulation is significantly, e.g. for f = 0.01 it is at least 100 times less favourable than the penalty for paper consumption. This ensures that excessive production will not occur, unless from paper otherwise wasted.

2.2.4 Formulation of the linear program

To solve the linear program, a proprietary software LPE [2] is used. It does not require the input data to be sorted, nor specially declared as ROWS, COLUMNS, BOUNDS or RHS.

Instead of that, equations, variables and bounds are used in a free order. Therefore, the constraints and the corrective member elements of the linear programming model can be created in parallel to the data input. A sample of the linear programming model data is given, with adjacent comments in mathematical notation:

`LO n001 1950`	$n_1 \geq 1950$
`n001 z -3.362442`	$z' := z' - 3.362442 * n_1$
`UP n001 2145`	$n_1 \leq 2145$
`LO n002 10200`	$n_2 \geq 10200$
`n002 z -2.973837`	$z' := z' - 2.973837 * n_2$
`UP n002 11220`	$n_2 \leq 11220$
`LO n003 4600`	$n_3 \geq 4600$
`n003 z -3.183721`	$z' := z' - 3.183721 * n_3$
`UP n003 5060`	$n_3 \leq 5060$
…	…

2.2.5 Generation of cutting schemas

The articles to be cut form a set of linear lists $\boldsymbol{L}$ $\{\mathbf{L}_k\}$. Each list groups articles of equal lengths because they can be cut laterally together. An atom of the list $\mathbf{L}_k$ contains the article identifier p, its width P_p and its length L_p. Thus, cardinality of $\boldsymbol{L}$ equals to the number of distinct article lengths nl.

Additional set of lists is required: $\boldsymbol{Z}\{\mathbf{M}_c\}$. Each list of articles $\mathbf{M}_c$ is one of the $nc = \binom{nl}{T}$ combinations among the lists $\mathbf{L}$, which all contain elements for possible joint schema generation. The number of combinations may appear prohibitive, but in practical applications, the count of conveyers, T seems to be small, e.g. 2 or 3. The recursive function `gen` to generate the schemas is called for each $\mathbf{M}_c$, and for each available roll. It means that it is called $nr * nc$ times from the main program. In each of these calls, all the technologically feasible cutting schemas are generated. Each of them is treated as a one-dimensional problem of cutting the rolls longitudinally. The technological feasibility check regarding the maximum number of longitudinal cuts Q, as trivial, is left out from the algorithm description. Let $\boldsymbol{A}\{a_p\}$ be the set of numbers of articles p that are produced out of a current cutting schema. The generated cutting schema is not recorded in the central memory but its description is formulated as elements of the linear programming model. Only the schema length is recorded in set $\boldsymbol{D}\{d_s\}$ and used later for presentation of results. For the same reason, the apparently redundant input roll identifier is recorded. The algorithm to generate the schemas and to write the second part of the linear programming model proceeds as follows:

```
s := 0
D := Ø
for r = 1 to nr
  for each Mc ∈ Z
    for each ap ∈ A
      ap := 0
    gen (Mc, Wr - WST, r, 1, A, D, s)

function gen (M, W, r, i, A, D, s)
  if i ≤ |M|
    p := Mi
    for j := ⌊ W / Pp ⌋ down to 0 by step -1
      ap := j
      remainder := W - j * Pp
      if remainder ≥ min_{k∈M} Pk
        gen (M, remainder, r, i+1, A, D, s)
      else
        maxlength := max_{k∈M | ak>0} Lk
        s := s +1
        for k := 1 to |M|
          q := Mk
          if aq > 0
            aq := aq * maxlength / Lq
            write "S", s, r, " n", k, aq
        write "S", s, r, " z", maxlength * Wr
        ds := maxlength
```

A sample of the linear programming model data is given, with adjacent comments in mathematical notation:

S0000101 n001 2.000000	$n_1 := n_1 + 2 * s_{11}$
S0000101 z 1644750	$z := z + 1644750 * s_{11}$
S0000201 n001 2.000000	$n_1 := n_1 + 2 * s_{21}$
S0000201 z 1606500	$z := z + 1606500 * s_{21}$
S0000301 n001 1.000000	$n_1 := n_1 + s_{31}$
S0000301 z 1377000	$z := z + 1377000 * s_{31}$
S0000402 n001 2.000000	$n_1 := n_1 + 2 * s_{42}$
S0000402 z 1644750	$z := z + 1644750 * s_{42}$
S0000502 n001 1.013072	$n_1 := n_1 + 1.013072 * s_{52}$
S0000502 n002 1.000000	$n_2 := n_2 + s_{52}$
S0000502 z 1666250	$z := z + 1666250 * s_{52}$
...	...

For a reminder: n_p is the number of articles p to be produced, S_{sr} is the number of repetitions for schema s, cut from roll r.

In the given sample, the article with identifier 1 can be produced from rolls 1 or 2 within the schemas 1, 2, 3 or 4, 5, respectively. In the schemas 1, 2, 3 and 4 it is the longest article. It is repeated as 2 stripes in schemas 1, 2 and 4, and as a single stripe in the schema 3. The schema 5 achieved the length of article 2, which is longer then the article 1. Therefore, within the schema 5, the repetition factor for article 1 is larger than one and it is not an integer.

3 Solution and interpretation of results

The program LPE [2] processes the generated input data and the results are stored in a file. In principle, the final solution cannot be infeasible, except for trivial reasons, e.g., if a required article is wider than the widest available roll. The roll lengths are supposed to be infinite. The upper bounds for articles cannot cause infeasibility, because the unused paper can be wasted instead. These upper bounds nevertheless make sense because they provide for balance among excessive production, if such occurs.

The results of the optimisation are read from the solution file as values of the variables $S_{sr} > 0$ and are stored into the set of quadruples $\boldsymbol{X}\{s, y_s, l_s, r_s\}$. The set cardinality ns is the count of basic cutting schemas in the optimal solution. The value s denotes schema identifier, y_s is the number of repetitions of schema s in the solution, i.e. it is the solution value for S_{sr}, l_s is the schema length fetched from the set $\boldsymbol{D}$ and r_s is the identifier of the input roll. The value for y_s is rounded off to the next largest integer.

The normative of articles that are cut from the schema s and the identifier r of the roll are reconstructed from the linear programming input file. The set $\boldsymbol{A}$ $\{a_p\}$ is now filled for each schema that appears as basic in the optimal solution, and it contains the count of repetitions of article p within that schema. This data, read from the linear programming model, are stored in $\boldsymbol{A}$ only for basic schemas, one current schema at a time, thus avoiding the excessive RAM consumption.

The number of produced articles p from the current schema s is obtained as

$$LateralCount_p * LongitudinalCount_p$$

For a current schema $s \in \boldsymbol{X}$ & $a_p > 0$, $LateralCount_p = \lfloor a_p / [l_s / L_p + \varepsilon] \rfloor$.

$\varepsilon = 0.000005$ is a numerical error correction.

The term in denominator represents the count of longitudinal article repetitions within a single schema, as a real number. The number of article repetitions within a

schema is divided by that number and the decimals are cut off to obtain the lateral repetition for that article.

$LongitudinalCount_p = y_s * l_s / L_p$ is self-explanatory.

An example for the optimum results in numerical form:

```
_____________________________________________________
# 1 article of dimensions 0945 * 0765 is cut
- laterally 1 time, longitudinally 1950 times (1950
pieces)
# 9 article of dimensions 0965 * 0860 is cut
- laterally 1 time, longitudinally 1735 times (1735
pieces)
Roll width is 2100
Waste width is 190
Unit schema length is 860
Schema #0007002 is repeated 1735 times
The length of the series is 1492100

_____________________________________________________
# 5 article of dimensions 1015 * 0530 is cut
- laterally 1 time, longitudinally 1100 times (1100
pieces)
# 9 article of dimensions 0965 * 0860 is cut
- laterally 1 time, longitudinally (678 pieces)
Roll width is 2100
Waste width is 120
Unit schema length is = 860
Schema #0028302 is repeated 678 times
The length of the series is 583080

_____________________________________________________
The overall length of roll of width 2100 is 10133940
The overall length of roll of width 1800 is 11065075
…
Of the required  1950, article #  1 produced  1950 pieces
Of the required 10200, article #  2 produced 10201 pieces
…
Useful area is   37703819264.000000
Consumed area is 41198407680.000000
Yield is 91.517661%
```

A list of articles **F** is initially formed and filled only with article identifiers. The atoms contain two pointers: to a subsequent article and to the list of all the basic schemas containing the article.

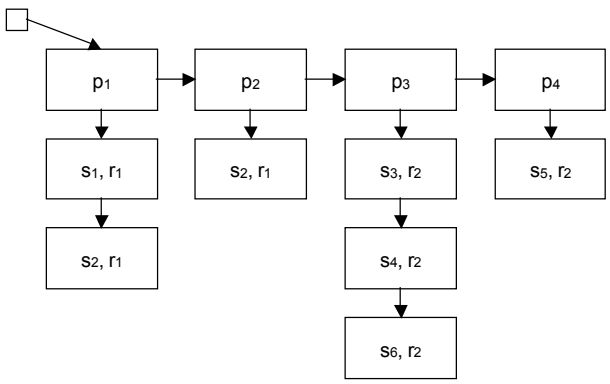

Fig. 2 Articles connected to the cutting schemas

The example in the Fig. 2 shows 4 articles, which are produced from 2 different input rolls and are cut by 6 different schemas. Such a list is formed from the optimisation results.

As already stated, the solution cannot be formally infeasible. However, there may exist a technological infeasibility, as a violation of the user's requirement that no cutting schema series be shorter than the length C. This is an empirical requirement arising from the problem of fixed costs incurred by adjusting the machine. The exact formulation to meet this requirement is mathematically rather simple:

$$y_s * l_s \geq C * \delta_s, \quad \delta_s = 0, 1; \quad \forall s$$
$$y_s * l_s \leq G * \delta_s, \quad \delta_s = 0, 1; \quad \forall s$$

G represents an arbitrary large number.

Expressed verbally, the number of repetitions of a schema *s* multiplied by the schema length must be at least C long, otherwise it cannot be greater than zero.
This simple statement is not easy to implement. Binary program has the *a priori* complexity of $O(2^{\Sigma})$, where Σ is the overall count of generated schemas. This count can easily exceed the value of few thousand. Therefore, introduction of mixed integer or binary programming is not practically feasible.
The possibility to edit interactively the achieved results was implemented instead. Due to the characteristics of linear programming, it may happen that a short series does appear in the optimum solution. It is represented by a basic S_{sr} bearing a relatively small value.
The user is prompted to choose whether s/he wants to reduce the number of schemas. If the answer is "Yes", using the list **F** and the set ***X***, the program searches for the schema applied in the shortest series. Then it checks whether this schema can be omitted from the solution. It cannot be omitted if it is a single schema for production of any article. If so, it also implies that a short series is not due to some peculiarity of linear programming but to a small production requirement for a certain article. However, there is a possibility, if it is a relatively narrow article, that it could be combined with some other article in a longer series. On the other hand, if the shortest series is not cut by a unique schema, there exist a probability that the schema for this shortest series can be omitted from the optimum solution. Formally, the solution will be deteriorated (or strictly speaking, it cannot become better), but the removal of a short series can actually reduce the fixed costs, which were not contained in the linear programming model. In this latter case, the user is advised to eliminate this schema. The

user can accept the advice, if there is any, or overrule it by entering the schema to be eliminated. Again, if such a schema is unique for production of any article in the optimal solution, the elimination is rejected. If the elimination is possible, the linear programming input data are revised by fixing to zero the entered schema, together with all the currently non-basic schemas. This will force the linear program to try to realise the required production with a reduced number of schemas. The result might be feasible and acceptable for the user, but now it can also become infeasible. Then comes the possibility for the user to increase the number of schemas. For the schemas that were fixed to zero, the expressed dual value in the infeasible solution is reviewed. While the solution is infeasible, the program LPE evaluates a composite sum of current coefficients in infeasible rows [3] to reduce the sum of infeasibilities. This sum acts likewise as the dual activity in a feasible basis. Therefore, the minimum negative dual solution is searched for. At least one such value must exist to point to a schema that would, by its unit increase, mostly contribute to the reduction of infeasibility. Such a schema is suggested to be considered to enter the optimisation. The user can accept the advice or overrule it by entering some other schema identification. The optimisation is repeated and the whole procedure can also be repeated. If the solution is feasible and the user wants to add an additional schema, no negative dual value can be encountered and the user is notified that addition of a new schema does not make sense. However, it is allowed, because it overrules the fixation to zero of the entered schema and it may serve on purpose for further navigation around the optimum solution.

4 Conclusion

The program was tested in a plant that produces corrugated paper packages on a cutting machine made by Agnati, s.p.a., Italy, model RDA-14, with two conveyors.

The user prepared some sample data, and the results have shown advantage over the manual schema planning, in reduced workload for the planner and in better material utilisation.

However, one must be aware that the solution of the cutting schemas is only a local optimum. In discussion with the user it has emerged that minimisation of paper consumption is not the proper objective. A proper objective would be the same, as it was in the production planning for another user [4]; the maximisation of contribution, i.e. to maximise the difference between income and variable costs.

A comprehensive model should encompass:

a) Selling prices
b) Fixed orders
c) Quantities for anonymous customer, to be sold with certain probability at a certain price
d) Roll costs
e) Machines productivity
f) Set-up times
g) Waste of material at the set-up
h) Original sources of variable costs (consumption of electricity, oil, steam, water etc.)
i) The prices of electricity, water, oil, steam, water, etc.
j) Workers' engagement
k) Variable part of salaries
l) Stock keeping costs

…and probably much more.

To achieve the proper goal, optimisation of cutting is not a bad starter. However, to take full advantage of optimisation techniques, an integrated information system is a prerequisite.

And finally, although the numerical results have demonstrated advantage of the applied algorithm, the implementation has been postponed since the users were not convinced that implementation of a user-friendly interface is a trivial problem. Many users unfortunately prefer nice looking screens and reports to the value of the embedded algorithm. This aspect should never be forgotten by any operational researcher.

References

[1] Weiss M A, *Data Structures and Algorithm Analysis in C*, Addison-Wesley, 1997

[2] Kalpic D, *Algoritmi linearnog programiranja na malom racunalu (Linear programming algorithms for a small computer) PhD thesis*: Faculty of electrical engineering: Zagreb, Croatia, 1982

[3] Orchard-Hays W, A*dvanced linear-programming computing techniques*, McGraw-Hill, New York., 1968

[4] Kalpic D, Baranovic M and Mornar V, Case study based on a multi-period multi-criteria production planning model. *European Journal of Operational Research* 87, 1995, pp 658-669, 1995