A Prolog Program for Decomposing Clothing Orders into Lays

David A. Rosenblueth

Universidad Nacional Autónoma de México

Abstract

We describe a Prolog program for solving a problem occurring in the clothing industry. The input of the problem is a clothing "order," consisting of a number of requested garments for each size. The output is a decomposition of the given order into "lays," producing exactly the requested number of garments per size. We wish to minimize both the fabric and the labor costs. For each lay we are only interested in knowing how many copies of the set of pieces of each size are to be placed (we do not deal with the geometrical problem of placing such pieces). Computing an optimal decomposition can be viewed as a generalization of the bin-packing problem.

Our system uses an enumerative, branch-and-bound method to traverse a search space. This approach resulted practical only for orders with up to seven sizes. However, since most of the orders in the firm where we found this problem have exactly this number of sizes, our system is useful in spite of having this limitation.

Keywords: Clothing production, cutting stock, branch-and-bound, Prolog.

Introduction

There is a process in the clothing industry involving the decomposition of an "order" into "lays," trying to minimize both the fabric and the labor costs. An order consists of a certain number of requested garments for each size. A lay contains a certain number layers of fabric, as well as a certain number of copies of the set of pieces of each size. This problem can be viewed as a generalization of the bin-packing problem and is therefore computationally intensive, especially when using an enumerative approach. Such an approach may not be practical for orders having more a certain number of sizes. We wrote a Prolog program traversing a search space with an enumerative, branch-and-bound method and found that such a program was practical for decomposing orders with up to seven sizes. Most orders in the clothing firm where we found this problem have exactly this number of sizes. Hence, our system is useful for such a firm. We believe that having chosen Prolog for developing our system resulted in an important advantage over more conventional programming languages, since both the description and the traversal of the search space can naturally be done using Prolog's nondeterminism and search strategy.

Problem description

We are concerned with the following problem, occurring in certain clothing firm.[1] Orders are for garments of specified style, material, color, and size. An order has only one style and one material, and includes a certain number of garments for each color and size. For simplicity, we will concentrate here on orders having a single color. (We plan to describe a method for decomposing multiple-color orders in a forthcoming paper.) Typically, the number of sizes in an order of the studied firm is seven, but occasionally may be up to 15. The number of requested garments for each size is forced to be a multiple of 10 so as to simplify several tasks, like manually counting the number of produced garments. An example single-color order is:

size						
30	*31*	*32*	*33*	*34*	*36*	*38*
70	80	100	80	100	100	70

[1] We have based our problem description on that of [2].

The stock material is rolls of fabric, available in varying materials, colors and designs. Rolls are on the average 100 m. long, and may vary in width from 1.30 m. to 1.90 m., depending on the type of fabric. Many rolls may be used in satisfying one order.

When laying out fabric, stacks are be abutted to give a total *lay* up to 12 m. in length. When cutting a lay into its components, duplicates of the same components are produced. The height of each lay is limited by the cutting equipment. If measured in layers of fabric, the maximum lay height depends on the fabric type, and varies between 60 to 120 layers. The number of layers in a lay is always a multiple of 10.

Each garment consists of many pieces, typically nine to 18. The set of pieces of a conceptual garment whose copy is duplicated in a lay is called a *mold*. Note that the number of produced garments of a certain size in one lay is equal to the number of layers times the number of molds of that size.

The problem we wish to solve is that of decomposing an order into lays in such a way as to produce exactly the number of requested garments for each size (and color). An example of a decomposition of the order given above, where a blank space means zero molds, is:

		size						
lay	*layers*	*30*	*31*	*32*	*33*	*34*	*36*	*38*
1	50	1	1	2	1	2	2	1
2	20	1	1		1			1
3	10		1		1			

Observe now that some decompositions may be preferable to others. A property of the production process is that for a fixed number of layers, the amount of wasted fabric per garment decreases as the length of the lay increases. To see this, assume first a lay with only one (irregular) piece. Then, all the area around that piece in a minimal rectangle circumscribing such a piece will translate into wasted fabric. The more pieces we fit into the lay, the closer we can get to a jigsaw puzzle, with little area per garment in between and around the pieces, if we position such pieces appropriately. (See [2, Fig. 2] for typical layouts.)

A human-based, computer-aided design system is utilized to determine the position of each piece of a mold. A human operator places a copy of each piece on a screen, and the computer system simply verifies that the pieces do not overlap. Unlike other firms [1, 2], where existing patterns of lays are kept in a database, the studied firm has operators who design each layout afresh.

Formulation

Let us formalize the problem. For each size, the number of produced garments must be equal to the number of requested garments:

$$\sum_{i=1}^{k} l_i a_{ij} = r_j \qquad \forall j$$

where

- k is the number of lays in a decomposition,
- l_i is the number of layers in lay i,
- a_{ij} is the number of molds in lay i of size j, and
- r_j is the required number of garments of size j.

The values of l_i, a_{ij}, and r_j are non-negative integers. Note that unlike other optimization problems where the number of constraints is fixed, here the value of k is unknown.

Let us now turn to the dimensions of a lay. The table where the fabric is layed is bounded in length, which we found convenient to approximate by bounding the number of molds in a lay. This constraint can be stated as:

$$\sum_{j=1}^{n} a_{ij} \leq m_{max} \qquad \forall i$$

where

- n is the number of sizes of an order.

Also, depending on the type of fabric, the number of layers in a lay is bounded:

$$l_i \leq l_{max} \qquad \forall i$$

Let us establish an objective function. The cost of a decomposition comprises that of the fabric as well as that of the labor. Consider the fabric cost first. It is possible to determine, from previous layouts, f, the *amount of wasted fabric per garment*. This amount depends on various factors, such as the style and the size of the different molds in a lay, but it mainly depends on A_i, the number of molds in a lay, where

$$A_i = \sum_{j=1}^{n} a_{ij}$$

We will assume that f only depends on A_i. In this case, $f(A_i)$ decreases monotonically with A_i, varying as recorded in old layouts.

For a given lay i, the amount of wasted fabric is obtained by multiplying $f(A_i)$ by $l_i A_i$, the number of garments in lay i:

$$W_i = f(A_i) l_i A_i$$

Consider now the labor cost of a lay. This cost depends on the number of both layers and molds. However, in this model we will approximate the cost associated with a lay with a constant q. We thus arrive at the objective function (cost of a given decomposition):

$$Z = \sum_{i=1}^{k} (W_i + q)$$

Observe that Z is monotonic in the number of lays of a decomposition, so that it is possible to use a branch-and-bound method to traverse a search space determined by the gradual addition of lays to partial decompositions.

Properties of the objective function

We will now observe that the two components $\Sigma_{i=1}^{k} W_i$ and $\Sigma_{i=1}^{k} q$ of the objective function are somewhat independent of each other. Also we will remark that whereas it is easy to (approximately) minimize $\Sigma_{i=1}^{k} W_i$, it is hard to minimize $\Sigma_{i=1}^{k} q$.

Note first that every decomposition d_W in which each lay has m_{max} molds is optimal with respect to $\Sigma_{i=1}^{k} W_i$. The reasons are that (1) $f(A_i)$ will have the smallest possible value $f(m_{max})$ in each lay, and (2) regardless of how many lays k' the decomposition has, the total number garments $\Sigma_{i=1}^{k'} l_i A_i$ is constant, so that $\Sigma_{i=1}^{k'} f(A_i) l_i A_i = f(m_{max}) \Sigma_{i=1}^{k'} l_i m_{max}$.

As an example, consider the following order:

size			
30	*31*	*32*	*33*
10	20	30	60

with $m_{max} = 4$. A decomposition for this order, of the form d_W, is:

		size			
lay	*layers*	*30*	*31*	*32*	*33*
1	10	1	1	1	1
2	10		1	1	2
3	10			1	3

Note now that if we have a decomposition d_q that is optimal with respect to the number of lays, we may be able to increase the number of molds in each lay and yet keep constant the number of garments in such a lay by decreasing the number of layers. If it were possible to have a non-integer number of layers and molds, then we could have exactly m_{max} molds in each lay of the new decomposition, which would be of the form d_W. However, since the number of layers and molds are constrained to be integers, we may not be able to have exactly the maximum number of possible molds in each lay, but we may approximate such a value. Thus, the resulting decomposition will approximate a decomposition of the form d_W.

Continuing with the previous example, consider a decomposition of the form d_q:

		size			
lay	*layers*	*30*	*31*	*32*	*33*
1	30			1	2
2	10	1	2		

which is clearly not optimal with respect to ΣW_i, because each lay has only three molds. We can obtain four molds in each lay by allowing non-integers:

		size			
lay	*layers*	*30*	*31*	*32*	*33*
1	30(3/4)			4/3	8/3
2	10(3/4)	4/3	8/3		

Since this is not possible, in general we may only be able to approach m_{max}.

As a conclusion, we can conceptually seek to minimize only Σq, and then reduce the number of layers in each lay so as to use as many molds as possible, which would (approximately) minimize ΣW_i.

Minimizing Σq, however, may be complex. In fact, this latter subproblem can be viewed as a generalization of the bin-packing problem [3]. In such a problem we wish place a given set of objects in the minimum number of bins as possible. All the bins have the same, constant capacity, and the objects may not be partitioned. If each lay of our problem corresponds to a bin, then the (variable) number of layers l_i in a lay i translates to a *variable* capacity of bins. Such a capacity is $l_i m_{max}$ units (with l_i a non-negative integer). Also, if we regard each garment size as an object and the number of required garments of that size as the capacity used by the corresponding object, then unlike the bin-packing problem, we *can* partition objects, provided that we accommodate exactly $a_{ij} l_i$ units of object j in bin i, where a_{ij} is a non-negative integer.

To illustrate the minimization of Σq consider the order:

size			
30	*31*	*32*	*33*
10	20	50	100

with $m_{max} = 3$.

A decomposition that is optimal w.r.t. ΣW_i and yet bad w.r.t. Σq is:

		size			
lay	*layers*	*30*	*31*	*32*	*33*
1	10	1	1	1	
2	10		1	2	
3	20			1	2
4	20				3

The above decomposition can easily be made worse w.r.t. Σq by splitting lays *3* and *4* into lays with 10 layers. We did not do so, to illustrate that it is important to select the appropriate sizes that have a number of molds greater than zero in each lay. If we are not careful in such a selection, even if we use as many layers as possible in each lay, the resulting decomposition may be bad w.r.t. Σq.

An optimal decomposition w.r.t. both ΣW_i and Σq is:

		size			
lay	*layers*	*30*	*31*	*32*	*33*
1	10	1	2		
2	50			1	2

Additional constraint

There is an additional important constraint resulting from the amount of available fabric. Enough rolls of fabric are available so that the material is barely enough for a given order. (Although the exact amount of needed fabric is not known *a priori*, this amount can be estimated from previous decompositions.) As a consequence, the lays cut last have to be satisfied with shorter segments of fabric than the lays cut first. As a convention, we assume that the lays are cut in the same order in which they occur in the corresponding decomposition. Hence, the number of maximum molds in a lay occurring near the end must be less than that of a lay occurring in the beginning of a decomposition. Usually 18 molds cat fit in a lay occurring first (limited by the size of the table) and up to six molds are adequate for lays occurring last (limited by the available amount of fabric). In case the last fabric segments are shorter than estimated, the operator can mentally split a six-mold lay into two or three lays with fewer molds.

Since we construct decompositions by adding lays in the same order in which the lays are cut, we incorporated this constraint by varying the maximum number of molds with respect to the percentage of the number of cut garments. For example, the maximum number of molds in a lay may be 18 for the first 95% of the garments, and six for the last 5%.

Method

We traverse a search space of partial (i.e. either complete or incomplete) decompositions. The root of such a space is the empty decomposition. Each child of an incomplete decomposition is obtained by adding a different lay to such a decomposition. The branch-and-bound strategy cuts subtrees of the search space which are rooted in nodes that are worse than the best node known so far. We mark the incomplete decompositions that are roots of such unwanted subtrees. Lays are gradually added to the empty unmarked decomposition as follows:

1. (branch) Nondeterministically select an unmarked incomplete (and possibly empty) decomposition called *current decomposition*, if such a decomposition exists. If no unmarked incomplete decomposition exists, then halt. Add one lay that has not been added before to the *current decomposition*, if possible, obtaining a new unmarked *current decomposition* and goto (2). If it is not possible to add such a lay, then mark the *current decomposition* and goto (1).
2. (bound)
   (a) if no complete decomposition has been constructed or if the *current decomposition* is better than the best complete decomposition so far constructed, then
      i. if the *current decomposition* is complete, then record the *current decomposition* as the best one so far, and goto (1)
      ii. otherwise, goto (1)
   (b) otherwise, mark the *current decomposition* and goto (1)

We traverse this search space following the standard Prolog regime: depth-first, from left to right.

Now the problem has been transformed into that of adding a lay to an incomplete decomposition.

A lay consists of a number of layers and a (possibly zero) number of molds for each size. The analysis in the previous section simplifies to some extent computing these values, since such an analysis justifies using as many molds as possible in each lay (if we are willing to only approximate to an optimum).

We can conceptually further simplify the problem of constructing a lay by nondeterministically selecting the subset S of sizes which will have a number of molds strictly greater than zero in such a lay.

We will next distinguish between two cases:

1. If the number of garments of sizes in S is sufficiently small, then the number of molds of each size is determined simply by "filling in" the table (and producing a residual number of garments possibly greater than zero due to the variables being discrete).
2. If the number of garments in S is not sufficiently small, then we face the additional difficulty of deciding how many molds the lay will have for each size.

Let Gs be a set of pairs $(size, garments)$. If there exists a lay with L layers such that (a) the total number A_i of molds does not exceed m_{max} and (b) the number of remaining garments of each size does not exceed L (i.e. the number of molds for each size is constrained by the number of requested garments and not by the length of the table), then we say that Gs is *non-saturated*. Otherwise, we say that Gs is *saturated*.

A saturated (subset of an) order would be:

size	
30	*31*
100	430

with $m_{max} = 4$ and $l_{max} = 20$.

If Gs is non-saturated, we can compute each a_{ij} as $a_{ij} = G_j \; div \; L$ and each remainder G'_j as $G'_j = G_j \; mod \; L$. If Gs is saturated, however, we can "fill in" the table and still leave residuals greater than the number of layers, so that it is not obvious in which sizes to leave such residuals. The number of choices in constructing a lay can then be enormous. We opted for reducing the number of garments for each size in a manner *proportional* to the requested number of garments of that size. This heuristic may be justified by observing

first that if the number of garments of all sizes is much greater than the maximum number of possible layers in a lay, then the distribution of the molds among the sizes in the first lay does not have an important effect over the optimality of the whole decomposition. Second, by reducing the number of garments of each size proportionally to the requested number of garments of that size, we may be able to combine molds of different sizes once we arrive at a non-saturated order. Third, combining molds of different sizes once the order is non-saturated may result in an advantage if the number of garments of such sizes are multiples of each other.

As a counterexample of this heuristic, consider the previous saturated order, satisfying first all garments of the size *30*, and then all garments of size *31*. One such decomposition is:

		size	
lay	*layers*	*30*	*31*
1	20	4	
2	10	1	
3	20		4
4	20		4
5	20		4
6	20		4
7	20		4
8	10		3

Note that this method does not exploit the fact that lays *2* and *8* can be combined into a single lay.

The construction of a lay for a non-saturated order is based on the following Prolog predicate.

$lay([\,], L, [\,], [\,]) \leftarrow$

$lay([g(S,G)|Gs], L, [a(S,A)|As], [g(S,G')|Gs']) \leftarrow$
$\quad A \ is \ G \ div \ L, \ G' \ is \ G \ mod \ L, \ G' > 0,$
$\quad lay(Gs, L, As, Gs')$

$lay([g(S,G)|Gs], L, [a(S,A)|As], Gs') \leftarrow$
$\quad A \ is \ G \ div \ L, \ 0 \ is \ G \ mod \ L,$
$\quad lay(Gs, L, As, Gs')$

Given a list Gs of pairs $g(size, garments)$ and a number L of layers, the predicate $lay(Gs, L, As, Gs')$ can be used to compute the number A of molds and the remainder number G' of garments for each size of a lay.

We also assume the following predicates:

1. $min_g(Gs, G_{min})$ that holds if Gs is a list of pairs $g(S,G)$ and G_{min} is the minimum of all G's; and
2. $num_molds(As, M)$ that holds if As is a list of pairs $a(S,A)$ and M is the sum of the A's.

A non-saturated order is defined as:

$non_saturated(Gs) \leftarrow$
$\quad min_g(Gs, G_{min}), \ min(G_{min}, l_{max}, L),$
$\quad lay(Gs, L, As, _), \ num_molds(As, M),$
$\quad M \leq m_{max}$

Addition of a lay to a partial decomposition

Input: a set Gs of pairs $g(size, garments)$.

Output: a lay $l(L, As)$ and a (possibly empty) residual order Gs'' where L is the number of layers, As is a list of pairs $a(size, molds)$ and Gs'' is a set of pairs $g(size, garments)$.

Method: Nondeterministically select a subset $Gs' \subseteq Gs$, such that $|Gs'| \leq m_{max}$ (i.e. such that we can use at least one mold for each size in Gs'; if $|Gs'| > m_{max}$, we better choose a Gs' with fewer sizes).

1. If $non_saturated(Gs)$, then
   (a) Compute the least L such that
   $$lay(Gs', L, As, Gs''), \ num_molds(As, M), \ M \leq m_{max}$$
   holds.
   (b) Return $l(L, As)$ and Gs''.
2. Otherwise,
   (a) Compute L, where
   $$min_g(Gs, G_{min}), \ min(G_{min}, l_{max}, L)$$
   holds, compute As as a scaled-down version of Gs, and compute Gs'' as the remainder order.
   (b) Return $l(L, As)$ and Gs''.

□

Implementation

The described system was written in Arity/Prolog32. Using an Intel Pentium MMX CPU at 233 MHz, we obtained the following execution times in seconds for a non-saturated order. (For saturated orders the execution times were only slightly greater.)

	number of sizes			
	5	6	7	8
exhaustive	0.27	3.73	163	6221
b. and b.	0.05	1.21	22	736

The row labeled "*exhaustive*" records the time taken to traverse the whole search space, whereas the row labeled "*b. and b.*" shows how the time decreases when such a space is curtailed with the branch-and-bound strategy. Observe that for up to seven sizes the execution time is less than one minute (22 s). For orders with more sizes, however, the execution time may be intolerable. Hence, a different method must be adopted for such orders. Similarly, here we have omitted orders with several colors. We plan to describe both generalizations in another paper.

Concluding remarks

The described system has been in use for several months now, with satisfactory results. This system uses a branch-and-bound strategy, with an objective function considering both the fabric cost and the labor cost. We observed that these costs are independent of each other to some extent. Minimization of the fabric cost can approximately be achieved by using as many molds as possible in each lay, because the area in between and around the pieces, divided by the number of molds, decreases as we increase the number of molds. Minimization of the labor cost, however, may be extremely complex, and can be viewed as a generalization of the bin-packing problem. We use an enumerative approach for minimizing this cost, and only orders with up to seven sizes can be solved within a reasonable amount time (less than one minute). Since most of the orders in the studied firm have exactly seven sizes, this system is useful in spite of its limitations.

Acknowledgment

We are grateful to Luis Silva, Mateo Beja and the employees who patiently explained the problem to us.

The author can be contacted at:
IIMAS, UNAM
Apdo. 20-726
01000 Mexico D.F.
Mexico
email: `drosenbl@servidor.unam.mx`

References

[1] A. Elomri, P. Morel, L. Pun, and G. Doumeingts. Solving a cutting problem based on existing patterns. *European Journal of Operational Research*, 77:169–178, 1994.

[2] Alan A. Farley. Mathematical programming models for cutting-stock problems in the clothing industry. *Journal of the Operational Research Society*, 39(1):41–53, 1988.

[3] Michael R. Garey and David S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, 1979.