Alternative formulations for a layout problem in the fashion industry

By Z. Degraeve, W. Gochet and R. Jans in European Journal of Operational Research (article in press, on line at http://www.sciencedirect.com)

It is very exciting to see how operations research methods and models can be implemented in various disciplines and areas of industry. In this paper, two integer programming (IP) models are proposed to solve the layout problem in the fashion industry. Before cutting, several layers of cloth are put on a cutting table and several templates, indicating how to cut all material for a specific size, are fixed on top of the stack. The problem consists of finding good combinations of templates and the associated stack heights of the cloth to satisfy demand while minimizing total excess production.

The original layout problem is very similar to the fixed charge cutting stock problem (FCCSP). However, in the FCCSP, the cost of trim loss is minimized, whereas the initial objective function of the problem in this paper minimizes the total cost of cutting operations and excess production. New improved optimal procedures are developed for the problem instead of focusing on heuristic approaches. We also have to know here that for low-demand, high fashion clothing the cost of being near optimal, i.e. too much overproduction, can be very high, whereas for the high demand clothing industry this is not so much a problem. This cost issue, together with the fact that we are dealing with real life problems, justifies the search for better optimal solutions.

In the first alternative formulation, the objective function remains as the minimization of total production. The constraints are the demand constraints, the constraints modeling the maximal number of patterns to be selected and the maximal height of the stack. Besides these constraints, a set of constraints eliminates alternative optima among patterns by imposing an ordering of the sizes within each pattern. It is arbitrarily selected to arrange the sizes in a pattern in increasing order. One possible way to impose this ordering introduces two new constraints, where one constraint ensures that there is at most one stencil per position, and whereas the other constraint enforces the size in the previous position to be smaller than or equal to the size in the current position. Next alternatives across patterns are eliminated. With these constraints, an empty position, which has no size assigned to it, is placed at the end of a pattern. After ordering sizes within each pattern, the patterns are ordered in a lexicographic way by decreasing number of times size one appears in the pattern. If there is a tie, the order is arranged by decreasing number of times size two appears, and so on. If there is a tie for the first size, the first term of the left- and right-hand side will cancel and consequently only the number of times size two appears will be decisive. If there is no tie, the number of times size one appears will be decisive. This is ensured by multiplying with a specific
factor the number of times size one. And, finally, the integrality constraints are added.

The second alternative formulation enumerates all possible patterns and assigns a binary variable to each one, indicating that the pattern will or will not be used. A pattern variable has as many indices as the number of possible sizes. Each index indicates how many times a specific size appears in the pattern. This model has the same objective function as the previous model, i.e. minimizing total production and the usual constraints: the demand constraints, the constraints modeling the maximal number of patterns to be selected and the maximal height of the stack. A new constraint imposes the minimal number of stencils to be used for each size. This constraint is not necessary for the logical formulation of the problem, but it speeds up computation times.

In addition to the well-outlined formulations, one can check to see how well the algorithms perform. A random problem generator is proposed to generate a given set of parameters. The results of the experiments with these randomly generated parameters indicate that the new formulations outperform the original problem. The results are available in a table form. Comparing computation speeds of these formulations and the original model, the first formulation is approximately 4 times faster than the original model. The model where all the patterns are enumerated, the second formulation, is almost 6 times as fast as the original model. One interesting result is that the maximum height has no substantial effect on the CPU time or the average percentage excess production.

In conclusion, this paper is a good example of observing how integer programming methods can be modified to better formulations, according to the nature of the problem. The two new models presented here perform better than the original model. We can also see in this example how certain characteristics of the problem have an effect on the difficulty of the problem. In our case, from an extended computational experiment, it can be concluded that the number of patterns, positions, sizes and free positions determine the difficulty of the problem. Adding more patterns, positions or increasing the number of free positions generally decrease the average percentage excess production. Adding a size increases the average percentage excess production. The maximum height has no substantial effect on the CPU time or the average percentage excess production. We can see that different integer linear programming formulations exist for this problem. The differences in performance among these formulations indicate that it is important to be creative in the modeling process and look for alternative formulations.