UNIVERSITÄT OSNABRÜCK

Cutting Stock by Iterated Matching

Andreas Fritsch, Oliver Vornberger

University of Osnabrück
Dept. of Math./Computer Science
D-49069 Osnabrück
oliver@informatik.uni-osnabrueck.de
http://brahms.informatik.uni-osnabrueck.de/prakt/prakt.html

Because of the increasing costs of raw material and the need to avoid industrial waste, solving cutting stock problems became one of the most important problems in the area of operations research. Cutting stock problems belong to the class of combinatorial optimization problems, so we have to find a solution among all possible solutions, which optimizes a criterion function subject to a set of constraints. Unfortunately the size of the solution space containing all feasible solutions of an optimization problem in general is enormous. As a consequence an exhaustive search within the whole solution space is impractical and sophisticated heuristics which ensure the calculation of reasonable solutions within limited time are required.

The combinatorial optimization problem considered here is a special two-dimensional cutting stock problem arising in the glass industry. Given a demand of non-oriented small rectangles and a theoretically infinite set of large stock rectangles of given lengths and widths, we are looking for layouts of minimal waste specifying how to cut the demand out of the stock rectangles. The constraints are chosen with respect to the production conditions in the flat glass industry: a 90° rotation of the demand rectangles is allowed and only guillotine cuts are permitted that means the cuts must go from one edge of the rectangle to the opposite edge.

For solving this problem we developed an iterative algorithm based on the following idea: In every iteration step we match some rectangles and consider the matched pair as a new, larger rectangle, a so called meta-rectangle. In the following iteration steps these meta-rectangles can be treated in the same way as ordinary rectangles. During the iteration we have to avoid matchings whose resulting meta-rectangles do not fit in the stock rectangle. The iterative matching automatically takes care of the guillotine structure. Every meta-rectangle can be cut by a guillotine cut. The iteration stops if every meta-rectangle represents a traverse. That means that the length of one side of the meta-rectangle is nearly the same as the width of the stock rectangle and the other side is smaller than the length of the glass sheet. The distribution of the meta-rectangles to the stock rectangles can be formulated as a one-dimensional bin packing problem and heuristically be solved by a first fit decreasing algorithm.

One of two subjections the cuts have to fulfill is that the traverses have to be smaller than a given maximum length. Otherwise the glass sheets can not be splitted on an automatic cutting table. A typical value is a maximal traverse length of 2300 mm for stock sheets of about 6000 x 3200 mm. The second subjection is that the distance between two parallel cuts

has to be of a minimal distance. This is since after the glass has been scratched it has to be broken along the cut. If the distance between two cuts is too small the glass can easily be damaged. The minimal cut distance depends on the thickness of the stock sheets, e.g. 20, 40 or 60 mm for a sheet of 4, 5 or 6 mm thickness.
To store the actual pairs of rectangles in every iteration step we use binary slicing trees. Every leaf of a tree represents a demand rectangle. Two rectangles are paired, if they have the same father who represents the resulted meta-rectangle. The orientations of the rectangles are not fixed at any time of the algorithm. For example, if we match two demand rectangles A and B the resulting meta-rectangle AB has eight essentially different layouts: A lies above or left from B and both rectangles can be rotated. We use shape functions to store every possible layout for every rectangle. A shape function s_R for a rectangle R is a decreasing, piecewise linear function. The interpretation is that $s_R(w)$ is the minimum length of a meta-rectangle that has the width w. We consider the function as a list of four-dimensional vectors, representing the length and width of the rectangle, the cut orientation and the point where the rectangle has to be cut. So during the iteration we calculate the shape function for each rectangle and store it in the representing tree node. After the iteration process has stopped, we use the shape function in the root of a tree to detect how the large stock rectangle has to be cut into two pieces. These pieces are represented by the sons of the root and the shape functions in these nodes give information about the further cutting. So the tree contains information about how the rectangles are paired and how we can get the demand rectangles out of a stock rectangle. At the bottom of this page a layout of a stock rectangle and the appertaining slicing tree are shown.
The problem of finding out which rectangles have to be paired in an iteration step can be solved by using a method of combinatorial optimization: the maximum weight matching. Given a complete, undirected, weighted graph, we have to choose a set of arcs, so that no two arcs incident with the same node and so that the costs of the chosen arcs are maximized. In our problem we interpret the graph nodes as the matchable rectangles, the arcs as the possible matchings and the weights as the benefit for the matching the arcs represent. The benefit is high if the matching causes only little waste, the benefit is small if it causes a lot of waste. To calculate the benefit we use the shape functions again. The benefit also is used to forbit unallowed pairs concerning the minimal cut distance or the maximum traverse length.
The algorithm is programmed in C and implemented on various kinds of computers. It has been tested with several instances taken from real demands of a glass factory. These instances contain between 50 and 160 rectangles and the average waste of the resulting layouts is about 5 percent. These results calculated within a few minutes (on a 486-PC) are even superior to those of a commercial program used by this glass factory.