Cross Cut Optimization of Wooden Boards Based on Automatic Defect Detection

Erik Åstrand¹ and Mikael Rönqvist²
Dept. of Electrical Engineering¹
Dept. of Mathematics²
Linköping University, S-581 83 Linköping, SWEDEN

Abstract

A fully automatic cross cut saw, based on automatic detection of all relevant wood defects, gives completely new opportunities to determine the optimum use of a wooden board. In this paper, a quality model is given to match the requirements of the end products to the quality of the raw material. The quality model is well suited for efficient computer implementation. Furthermore, an optimization procedure based on this model is proposed, giving an integrated system where defect data is input in one end and final cutting lists are output in the other. The procedure produces optimal cutting lists from the given input data. Tests with real data from a Swedish window manufacturer show that the average output value is 7% higher using this procedure than what is obtained from their existing commercial saw. The procedure is also very efficient. The tests show that it is capable of producing optimal cutting lists for their given product mix within in average 17 ms on an ordinary PC.

1 Introduction

The wood industry is a field where different forms of visual inspection take place at a number of stages of the production. Machine vision systems are beginning to find their way into wood applications, but the nature of wood makes the inspection tasks very difficult to automate. The subject of this paper is cross cutting of boards in the secondary wood industry. In a traditional optimizing cross cut saw line, a human grader inspects the board and marks zones of different qualities with a special fluorescent crayon. The quality of each zone is determined from the contents of various defects such as knots, checks, pitch pockets etc. The marked board is fed into the saw which detects the positions of the crayon marks and calculates a saw pattern based on a number of fixed lengths having different values.

When machine vision systems were first introduced for cross cut operations, they were installed together with optimizing saws designed for crayon marking, and the quality data sent to the saw were in the same format as if the information was sent from a crayon mark camera. However, this scheme lacks the ability in the optimization phase to utilize information about the type, position and size of each individual defect. By using an automatic defect detection closely integrated with the optimization procedure, much more complex quality rules can be handled. In this paper, which is a short version of [1], a quality model is described together with a fully integrated optimization procedure. The purpose of the model is to reduce the computational efforts needed during the actual production phase by preprocessing the quality rules into special data structures. The optimization part is
based on the same data structures and produces optimal cutting lists within limited time. The algorithm is straightforward and easy to implement, e.g. on an ordinary PC.

In this paper, it is assumed that an image processing system is available which is capable of accurate detection and classification of all relevant defects on the boards. This is a somewhat unrealistic assumption as today's commercially available wood inspection systems still have difficulties with some classes of defects, but new and improved methods and systems are continuously being reported in the literature [2].

1.1 Quality Rules

Below an example of quality rules from a Swedish window manufacturer is given. Several different dimensions are cut, one example being the 35x45 mm dimension, which is used to produce three different window casement profiles as shown in Figure 1. The windows are produced in different sizes, in some cases even according to customer specific orders. This means that the casement pieces are to be cut in a number of different lengths.

![Figure 1. Window casement profiles from the dimension 35x45 mm.](image)

Two of the profiles in Figure 1 are used as inner and outer casement, respectively, in a double glass window, while the third profile is the outer casement in a triple glass window. Depending on the position within the window, a certain side will either be visible (marked B) or hidden (marked C). On the hidden sides, more defects will be allowed than on the visible sides. The corner in which the fold for the glazing is routed, marked D in Figure 1, may contain virtually any kind of defects except large checks. Furthermore, the first and last 50 mm of each fix-length must be free of defects on all sides, corresponding to the leftmost profile in Figure 1. In addition to the window casements, the 35x45 mm dimension may also be used as the interior of a door frame. In this case, it will be completely hidden and its sides, marked E, may contain any defects. Hence, five different quality areas are defined, and an example of possible quality rules for each area is given in Table 1.

<table>
<thead>
<tr>
<th>Defect</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black knots</td>
<td>-</td>
<td>&lt; 10 mm</td>
<td>&lt; 20 mm</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>Sound knots</td>
<td>-</td>
<td>&lt; 30 mm</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>Checks</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>&lt; 0.5 mm</td>
<td>OK</td>
</tr>
<tr>
<td>Pith</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>Pitch Pockets</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>Wane</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>OK</td>
<td>OK</td>
</tr>
</tbody>
</table>

Table 1. Sample quality rules for different areas.

Depending on the symmetry of the profiles, the quality requirements of the fix-length may be matched to the defects on the board using different orientations. This kind of complicated quality rules can not be handled in a traditional optimizing saw. There is no way that the operator, doing the crayon marking, can supply the optimization with enough information about the defects. He can only mark larger defects and assign general qualities to the
areas in between. If he finds a small defect near a corner he may disregard it, assuming that it will be possible to turn the piece so that the glazing fold ends up in that particular corner. But what if he sees two such defects in opposite corners some distance apart? If the board is cut between the two defects everything is OK, but there is no way the operator can force this to happen. Neither can he handle defects that are forbidden near a cut but otherwise allowed. The only way to handle this kind of complex quality rules is to use an automatic defect detection. This is also the conclusion of other investigations [6]. To illustrate the model and the optimization procedure, the following simple example will be used:

- Boards with a square cross section, say 10x10 cm.
- Two fix-lengths, 40 and 60 cm respectively. Value 10 and 20.
- Knots smaller than 30 mm are allowed on one side, but not within the first and last 10 cm.
- One low quality fix-length without knot restrictions, 80 cm value 5.

2 Quality Model

There are several possible models to describe the quality of a board. In the usage bit model, see Astrand [1], the different possible cross sections, as shown in Figure 1, are represented by bits in a bit field, one bit for each possible orientation. The number of bits required depends on the symmetry of the cross sections. The processing of the defect list results in a bit list for the entire board. A one bit in a particular zone of the board indicates that this particular piece of wood may be used as a part of a fix-length with the given cross section. In order to be allowed to allocate a fix-length at a given position, the corresponding usage bit, or bits if we have different requirements in the lengthwise direction, must be set along the entire fix-length.

In this paper, we instead use what we call the allocation bit model. Here a single bit set in a certain zone indicates that the starting point of a given fix-length with a given orientation may be allocated in that zone. This means that the number of required bits increases compared to the usage bit model as we now need separate bits for the different lengths of otherwise identical products. However, the optimization part will become quite simple using the allocation bit model as input.

Before we start the scanning of the board, we assume that it is completely free of defects. This means that all allocation bits will be set to one along the entire board as long as they fit within the total length. In our previous example, we had three different lengths, one totally symmetric and two which can be turned in four different ways. In this example we need a total of nine allocation bits, one for the 80 cm piece and four for the 60 and 40 cm respectively.

The scanning of the board results in a list of defects, all with a given type, size and position. Each such defect disqualifies one or more allocation bits in one or more zones of the board. When we have gone through the entire defect list, the result is the final allocation list which is passed to the optimization. A sample board and its final allocation list is shown in Figure 2. For simplicity, the board only contains defects on the two visible sides.

Figure 2. Final allocation list.
In order to apply the sometimes rather complicated quality rules with enough efficiency to the allocation bit list of the board, some kind of preprocessing of the quality rules is necessary. The objective is to reduce the amount of computations needed during the actual production phase. This can be done by transforming the quality rules into disqualifying bit lists, which can be merged into the allocation bit list of the board using simple logical AND operations. The effect of a certain defect on the different products can thus be established in parallel, limited only by the word length of the computer used. An example of a disqualifying list is shown in Figure 3. The example corresponds to a small knot on one certain side in our example.

![Figure 3. Disqualifying bit list for a defect.](image)

During the preprocessing of the quality rules, a number of such disqualifying lists are calculated depending on defect type, size and crosswise position. As shown in Figure 3, all lengthwise positions within the list are relative to either the start or the stop position of the defect. During the production, these values are taken from the input list of defects. The final disqualifying list can then be established and merged into the allocation list of the board. Note that if the defect covers a longer zone, positions in the list may change places. There are also some other parameters which must be taken into account, such as saw kerf width and end trim cuts. More details about this procedure is given in [1].

3 Optimization

3.1 Mathematical Model

The mathematical formulation of the decision problem is critical since the model should describe the real world problem accurately enough while being solvable in reasonable time. In this particular application, the time constraint is crucial. The problem can be formulated using a number of different mathematical models, which are more or less suitable for a given solution method. One formulation of the problem, but without any solution procedure, is given by Zscheile [7]. Our formulation is based on the allocation bit model described in Section 2. A discrete model is chosen where the board is divided into \( n \) discrete points. The size of each step could vary but is typically 1 mm as this is reasonable resolution for both the image processing system as well as the saw. Moreover, the \( m \) fix-lengths in the cutting list are also normally given in mm. Introduce the following notations:

\[
\begin{align*}
  x_{ij} &= \begin{cases} 
    1 & \text{if fix-length } i \text{ is allocated at position } j \quad i = 1, \ldots, m \quad j = 1, \ldots, n \\
    0 & \text{otherwise}
  \end{cases} \\
  a_{ip} &= \begin{cases} 
    1 & \text{if fix-length } i \text{ which is allocated at position } j \text{ also covers position } p \quad p = 1, \ldots, n \\
    0 & \text{otherwise}
  \end{cases} \\
  c_i &= \text{value of fix-length } i \quad i = 1, \ldots, m
\end{align*}
\]
The problem can now be formulated as

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

s.t. $$\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} x_{ij} \leq 1 \quad p = 1, \ldots, n$$

$$x_{ij} \in \{0,1\} \quad i = 1, \ldots, m \quad j = 1, \ldots, n$$

This problem is a so called set packing problem, see e.g. Nemhauser and Wolsey [4]. Various general techniques for its solution can be adopted, e.g. Branch and Bound. Another possibility is to use heuristics which are able to generate near optimal solutions. The constraints ensure that any position is covered by at most one allocation. Note that no constraints are included to explicitly ensure that an allocation does not violate the quality requirements. This is because the feasibility of each $$x_{ij}$$ variable can be determined a priori from the board’s allocation bit list. Those $$x_{ij}$$ corresponding to inadmissable allocations are simply not regarded as part of the problem and will never be included in any solution procedure.

In our application, $$a_{ij}$$ has a certain structure which can be exploited in the solution procedure.

$$a_{ij} = \begin{cases} 1 & \text{if } j \leq p \leq j + l_i \\ 0 & \text{otherwise} \end{cases} \quad i = 1, \ldots, m \quad j = 1, \ldots, n \quad p = 1, \ldots, n$$

where $$l_i$$ = length of fix-length $$i$$

3.2 Solution Procedure

The solution procedure is based on a forward dynamic programming approach, which is guaranteed to produce optimal solutions, see e.g. Dreyfus and Law [3]. In our formulation we have a discretization of $$n$$ elements, and each such element will give a step in the solution procedure. Introduce the following notation:

$$S(k) = \text{the value of the maximum allocation pattern in the interval } [1,k-1]$$
$$R(k) = \text{start point of the allocation ending in } k-1 \text{ giving } S(k)$$
$$S_{\text{max}} = \text{total maximum value}$$
$$R_{\text{max}} = \text{position giving max value}$$

A first straightforward algorithm is given by:

Initialize $$S(k) = 0 \forall k, S_{\text{max}} = 0, R_{\text{max}} = 0$$

for $$k = 1$$ to $$n + 1$$

if $$S(k) > S_{\text{max}}$$ then

$$S_{\text{max}} = S(k), R_{\text{max}} = k$$

else

$$S(k) = S_{\text{max}}, R(k) = R_{\text{max}}$$

for each admissible allocation $$i$$ at position $$k$$

if $$S_{\text{max}} + c_i > S(k + l_i)$$ then

$$S(k + l_i) = S_{\text{max}} + c_i, R(k + l_i) = k$$

endfor

endfor

When the algorithm terminates, $$S_{\text{max}}$$ gives the optimal objective value, and $$R_{\text{max}}$$ is the position of the last cut. The complete cutting list, i.e. the final $$x_{ij}$$ values, is determined by following $$R(k)$$ backwards. The algorithm can be interpreted as walking along the board
from left to right. At each position, or element, all admissible alternatives are tested and new values are updated at their end points respectively. The new value for each alternative is the sum of the maximum value reached so far and the value of the fix-length. If when we arrive at a new position, the recorded value is lower than the maximum value achieved so far, the recorded allocation is replaced by a zero value waste section from the position of the maximum value to the current position. This is done in the else branch in the first if.

The number of operations in the procedure is proportional to the length of the board (number of elements) and the number of fix-lengths. The worst case is if the board is completely free from defects, which means that all fix-lengths are allowed at all positions. The number of required operations could however be drastically reduced by a simple observation. If we at any position of the board allocate a fix-length following a waste section, then there can never be any point in not allocating it as early as possible, i.e. making the waste section as small as possible. This rather self-evident observation means that the only positions we have to investigate are positions where new possible allocations start and all the subsequent positions representing all length combinations of admissible fix-lengths. In terms of the procedure above, only positions which previously have received values and positions where new allocation possibilities start need to be investigated. Furthermore, if a position has a recorded value but this value is lower than the maximum value achieved so far, then this value can be disregarded. Note also that if a position is investigated only because of new possibilities, only the new ones need to be tested.

The procedure can be improved even further by observing that if a number of allocations with equal lengths and equal values are admissible at a given position, only one of them has to be tested. This situation will frequently occur as the allocations often are different orientation of each other. Equivalently, an allocation does not need to be tested if another allocation with a shorter (or equal) length and a higher (or equal) value is admissible at the same position. This can be included in the algorithm by associating an exclusion set with each fix-length \(i\). Each time an allocation is tested, its exclusion set is subtracted from the remaining set of allocations to test at that position. By implementing the set operations using bit manipulations, this can be done very efficiently. Note that the order in which the allocations are tested is significant. The exclusion sets and the best order of the fix-lengths can be determined when the quality rules are preprocessed. Care must however be taken if the values of the fix-lengths are updated dynamically during the production, e.g. if the value of a certain fix-length depends on the number produced so far. Solution schemes for these improvements can be found in Åstrand and Rönqvist [1].

### 3.3 Numerical Results

The implementation of the algorithm is straightforward. In our tests, the main part of the optimization was implemented using only some 40 lines of C-code. The procedure was evaluated using defect data, quality rules and fix-length lists from the Swedish window manufacturer mentioned in Section 1. This particular manufacturer has installed a Woodeye system [5] connected to a commercial optimizing cross cut saw based on the usage bit model.

20 boards with varying defect contents were run through their existing system and the total value for each board were recorded. The fix-length list used included five different lengths of casement parts and two different low quality frame parts. Both inner and outer casement parts were included, which means that there were eight different ways to cut each casement fix-length. All together, the number of possible fix-length alternatives was 42. The value and number of combinations for each fix-length is given in Table 2.
The output usage bit lists from the Woodeye were converted to allocation bit lists using the same fix-length list and the same parameter values for end trim cut (5 mm) and kerf width (4 mm). These lists were then run through our optimization procedure. The test was run on a 33 MHz 486 PC and for each board, the total value and execution time were recorded. The results are shown in Table 3. The execution time, given in ms, is measured for the main part of the procedure including initialization but excluding the output of the cutting list. The resolution was 1 mm, i.e. the number of discrete points equals the length of each board. The old value is the value given by the existing commercial cut saw.

Table 3 shows that our proposed algorithm finds better solutions than the commercial system in 17 out of 20 boards. Only in three cases were the optimal solution found by both systems. In this particular installation, or at least this particular cutting situation, the average profit could be increased by nearly 7%. Table 3 also shows that the procedure is quite fast. The key to the efficiency is the structure of the input data, combined with the small number of allocations that really need to be investigated.

<table>
<thead>
<tr>
<th>Board no.</th>
<th>Length (mm)</th>
<th>Value</th>
<th>Old value</th>
<th>Time (ms)</th>
<th>Board no.</th>
<th>Length (mm)</th>
<th>Value</th>
<th>Old value</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4496</td>
<td>412</td>
<td>402</td>
<td>16</td>
<td>11</td>
<td>4189</td>
<td>424</td>
<td>364</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>4796</td>
<td>427</td>
<td>396</td>
<td>17</td>
<td>12</td>
<td>5090</td>
<td>500</td>
<td>478</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>5094</td>
<td>579</td>
<td>569</td>
<td>23</td>
<td>13</td>
<td>4493</td>
<td>399</td>
<td>397</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>4195</td>
<td>420</td>
<td>420</td>
<td>17</td>
<td>14</td>
<td>3899</td>
<td>373</td>
<td>367</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>5387</td>
<td>602</td>
<td>569</td>
<td>20</td>
<td>15</td>
<td>5091</td>
<td>430</td>
<td>334</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>4799</td>
<td>455</td>
<td>419</td>
<td>15</td>
<td>16</td>
<td>5386</td>
<td>185</td>
<td>184</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>4495</td>
<td>463</td>
<td>421</td>
<td>15</td>
<td>17</td>
<td>4495</td>
<td>473</td>
<td>471</td>
<td>17</td>
</tr>
<tr>
<td>8</td>
<td>4492</td>
<td>453</td>
<td>341</td>
<td>19</td>
<td>18</td>
<td>4793</td>
<td>522</td>
<td>522</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>5086</td>
<td>288</td>
<td>266</td>
<td>14</td>
<td>19</td>
<td>5385</td>
<td>472</td>
<td>419</td>
<td>18</td>
</tr>
<tr>
<td>10</td>
<td>4193</td>
<td>449</td>
<td>447</td>
<td>19</td>
<td>20</td>
<td>4794</td>
<td>105</td>
<td>105</td>
<td>13</td>
</tr>
</tbody>
</table>

Table 3. Results from tests with real production data.

Table 3 shows that our proposed algorithm finds better solutions than the commercial system in 17 out of 20 boards. Only in three cases were the optimal solution found by both systems. In this particular installation, or at least this particular cutting situation, the average profit could be increased by nearly 7%. Table 3 also shows that the procedure is quite fast. The key to the efficiency is the structure of the input data, combined with the small number of allocations that really need to be investigated.

4 Conclusions

The following conclusions can be drawn from the model, the optimizing procedure and the numerical tests:

1. An automatic wood inspection system can do much more than just replacing an existing manual operation. In a crosscut saw, information about every individual defect can be taken into account and the automated system can handle a greater variety of product combinations simultaneously. This enables a higher output yield and the use of raw material of lower quality.
2. A good quality description model is essential in order to match the quality demands of the end product to the quality of the raw material. The proposed model can handle nearly all relevant quality requirements, while being suitable for efficient computer implementation.

3. New rather complex optimization problems arise from the new product combination possibilities. Careful structuring of the input data in combination with efficient algorithms can however solve these optimization problems, using cheap standard computer hardware, well within the time limits given by the industrial requirements.

References


