UNIVERSITÀ DEGLI STUDI DI PADOVA

Head Office: Università degli Studi di Padova
Department of Mathematics "Tullio Levi-Civita"

Ph.D. COURSE IN: Mathematical Sciences
CURRICULUM: Computational Mathematics
SERIES 36°

Flexible log cutting in sawmill: an approach based on Value Maps and Derivative-free optimization

Thesis written with the financial contribution of Microtec Srl

Coordinator: Prof. Giovanni Colombo
Supervisor: Prof. Luigi De Giovanni
Co-Supervisor: Prof. Francesco Rinaldi
Company Supervisor: Enrico Ursella

Ph.D. student: Enrico Vicario

Abstract

The log-to-board cutting process within sawmills represents a significant opportunity for optimization, especially with the advent of advanced measurement technologies such as computed tomography (CT). Despite the availability of these cutting-edge technologies, their full potential remains underutilized within flexible cutting lines. In fact, a very accurate model of the log and its internal defects can be derived from the tomographic data. This model can be used to obtain an estimate of the value of each possible board that can be cut from the log. An accurate estimate can be beneficial to the solution of the problem of defining an optimal cutting pattern. However, the evaluation process is too computationally intensive to be directly applied in optimization procedures for sawmills with flexible saw lines, where we have several degrees of freedom in determining feasible cutting patterns, and a huge number of estimations are required. This study proposes a realistic and innovative solution to fasten the board valorization process, using value maps generated by convolutional neural networks to dramatically reduce the time required to valorize a board. Furthermore, the study introduces a comprehensive methodology for addressing the flexible log optimization problem, leveraging a dynamic programming algorithm integrated with black-box optimization techniques for effective exploration of the cutting pattern solution space.

Contents

List of Figures

Chapter 1

Introduction

The cutting of logs in sawmills is a key process in the wood processing chain, originating from forest trees and culminating in a diverse range of finished products, tailored to various destinations and applications, each requiring specific qualities and characteristics. It is within the phase of selecting and cutting logs into boards that irreversible decisions are made, limiting the potential final destinations of these boards. This process has evolved over time, driven by advancements not only in control and cutting technology but also in the precision and potential of measurement systems.

In particular, in recent years, the integration of tomographic scanners into production lines has made it possible to accurately measuring the internal quality of logs. These scanners provide a precise spatial model of the internal log features in an automated manner. The model offers valuable insights for log cut optimization by accurately predicting the characteristics of individual boards that can be derived from the log, based on their positioning.

The log-cutting optimization problem belongs to a special class of three-dimensional bin-packing problems (3DBP). Given a set of parallelepipeds, the problem, in its basic version, asks to find an optimal arrangement of them within one or more bins that are also characterized by a shape defined by a parallelepiped. In the problem of cutting a log, the boards that can be obtained are assimilated to parallelepipeds. However, compared to the characteristics usually found in 3DBP problems, the problem addressed in this thesis has many peculiarities. For example, the shape of the bin is not a parallelepiped; it may be assimilable to a cylinder at first analysis, but in practice it is an irregular shape resulting from a measurement. Another fundamental peculiarity is the value of the objects, which is strongly affected by their placement inside the bin. In fact, this is the key contribution to optimization due to the inclusion of the tomographic model of the log and the calculation of the value of the boards in each position. Moreover, the positioning of boards is strongly constrained by the characteristics of the cutting line. Besides being limited to guillotine cuts (a case considered in the literature in the context of cutting problems), the cutting machines add other specific constraints depending on the cut breakdown sequence.

There are many specific studies in the literature on the optimization of sawmill log cutting, presenting solutions with different approaches, among which those based on dynamic programming are prominent. However, to the best of our knowledge, these studies do not take into account all the peculiarities of the problem. In particular, in relation to the positional value of boards, there is a lack of fast algorithmic solutions that are suitable for inline integration in production lines.

In this thesis, the problem of sawmill log cutting is first addressed by proposing a formal definition of the problem that integrates the constraints of more flexible cutting lines and takes into account the boards valorization according to their position. The problem thus formulated is called Flexible Log Optimization (FLO). This problem definition allows taking into account the contribution of the obtained log model from the tomographic scanner, which is a key element of this thesis.

1.1 Context and objectives

This thesis comes at the conclusion of an industrial PhD supported by the company Microtec Srl Gmbh. The company, founded in 1980 in Italy, is today a leading provider of measurement and optimization solutions in the woodworking industry with offices in Europe, Canada and the US. In 2011, it installed its first CT Log scanner, an industrial tomographic scanner able to work 24/7 on log production lines with speeds up to 180 m/min.

Before the beginning of this PhD, I already collaborated, as an employee of the company, in research and development to achieve the implementation of this technological advancement for sawmills. In parallel with the development of the tomographic scanner, I contributed to the creation of the algorithms and technological solutions necessary for its use in the optimization process. Specifically, this previous work involves the development of algorithms, even based on machine learning, that extract relevant features from tomographic data and generate a three-dimensional model of the log, describing the external shape, pith, knots, heartwood, internal splits etc.

In parallel, I also followed the development of the optimization software Maxicut Pro. The software makes use of the information from the tomographic log model and provides solutions to the log cutting problem with speeds compatible with the production line. In fact, from the three-dimensional model of the log, it is possible to obtain a theoretical evaluation of any board that may be cut. This is done by creating a virtual board model obtained from the 3D intersection with the various features within the log.

Maxicut Pro software is installed to date in many sawmills in which the tomographic scanner is present. This optimization software, although capable of providing an improved solution compared to an optimization that neglects tomographic data, fails to exploit the full potential of the tomographic log model, in particular when more flexible cutting lines are considered. As we will detail later in the thesis, the approach adopted by Maxicut Pro is based on testing a set of predefined cutting patterns. These cutting patterns are manually determined, according to operator experience, and define the main boards that will be cut from the log. The optimization software selects the best cutting pattern from those available and its best alignment and rotation, in a discretized set, with respect to the log. Due to the computational resources required by the evaluation process, the number of pre-set cutting patterns has to be taken limited, as to preserve running times suitable for inline use of the overall optimization procedure.

The fundamental objective of this thesis is to propose a valid solution approach to the FLO problem. The approach should make the best use of the information provided by the log model obtained from tomographic scanning, while preserving to the best extent all the degrees of freedom allowed by flexible cutting lines. It must also be compatible, in terms of computation time and required hardware, with an inline installation.

1.2 Content and contributions of the thesis

The remaining chapters of this thesis are organized as follows.

Chapter 2 provides all relevant basic notions of the industrial log cutting process. It describes the different ways in which the cutting process takes place in an industrial sawmill and the most commonly used measurement techniques, with a specific focus in X-ray tomography. The concept of board valorization through rule checking is introduced and how it translates into the valorization of a virtual board obtained on the basis of tomographic data. Based on these notions, a general specification of the log cutting optimization problem, object of this research, is given. The chapter also discusses how this problem has been addressed so far by Microtec's Maxicut Pro software, currently installed in mills where the tomography system has been implemented, and the limitations of this solution. A review of relevant literature on the subject is presented at the end of the chapter.

Chapter 3 formally defines the optimization problem object of the thesis, namely, the Flexible Log Optimization (FLO). In particular, we define what we mean by a flexible cutting pattern and which degrees of freedom are considered. In the definitions, we include the concept of positional board valorization, which characterizes the FLO problem and makes the optimization problem more challenging than what is generally found in the literature and for which, to our knowledge, there are no effective approaches yet. The contribution is the formalization of the FLO problem, according to a general enough definition to include the cutting pattern constraints of most of the case studies in literature. The chapter then introduces and sketches our proposed approach to FLO, a search procedure based on three interacting phases, which are then described in detail in subsequent chapters.

Chapter 4 describes the first of these phases, dealing with the board valorization. This is one of the most important contributions of the thesis. In fact, one of the main challenges that makes the FLO problem difficult to handle is related to the computational time required for the valorization of a virtual board. This is the main ingredient of search-based optimization algorithms, and a very large number of individual board valorizations are needed, in particular in flexible cutting contexts, as for FLO. Two approaches are proposed for a dramatic reduction of the time needed to valorize the boards. The first approach defines a set of Rule Maps (maps that model the log defects in 2D), and methods are presented to include them in the optimization procedure to speed up the evaluation of rules for virtual boards. However, board valorization using Rules Maps has significant limitations that are highlighted in the chapter. The second approach is the most effective in accuracy and reducing the time to calculate the board value. It is based on the creation of Value Maps (2D maps that allow to directly obtain the value of the boards in the cutting pattern), which can be computed in optimization-compliant times using convolutional neural networks. In particular, for this second approach, its integration into the optimization process is described, taking into account the need for an offline training phase. A verification of the performance of these two approaches, both in terms of computation time and accuracy in estimating the value of the solution, is given at the end of the chapter. For the approach based on Value Maps and CNN, a patent has been filed, considering its industrial potential and innovation [42].

Chapter 5 presents the second phase of the proposed optimization approach, dealing with the optimal completion of a partial solution to FLO. In fact, the solution approach is based on the decomposition of the decision variables into two sets: a driving set, called *seed*, that defines a partial solution, and the set of remaining variables, that are determined after a seed is fixed. The seed includes, e.g., decisions on the number and type of cuts that define a pattern, or its rotation. The remaining variables determine, e.g., the sequence of boards to include in the pattern. The seed-constrained optimization of the remaining decision variables is a combination of two nested algorithms. The innermost one is based on one-dimensional dynamic programming and it is able to find optimal sequences of stacked boards constrained by a starting positioning. The outermost algorithm is a step search that make use of the first algorithm to test different combinations of positions of stacked boards to find the best solution. For the inner algorithm, inspired to dynamic programming procedures from literature, we propose a new implementation based on wide interval partitioning, preserving position accuracy, and allowing for speedup at the expense of optimality. A trade-off choice for this partitioning step of the algorithm is then experimentally set and analyzed at the end of the chapter. In addition to wide interval partitioning, we also extend the innermost algorithm to consider sequences of boards with fixed position. The extensions to this algorithm, together with the verification of the tradeoff between computational speed and optimality, is a significant contribution of the thesis and could also find application in other contexts.

Chapter 6 presents the highest-level phase of the proposed optimization methodology, concerning the selection of the seed that lead to the best overall FLO solution after the seed-constrained optimization. Seed selection is based on a search procedure through the seed space. Different approaches are considered for both defining and exploring the domain of the variables included in the seed. A first approach is an heuristic developed as a baseline for the comparison, based on experience in manually defining cutting patterns, that creates an ordered list of cutting patterns and explores them sequentially in a limited set of positions and rotations. However, the most interesting approach we propose for this phase of the optimization is based on black-box optimization and direct search algorithms. In fact, the objective function of the problem, corresponding to the complete optimization of the seed, is computationally expensive and strongly discontinuous with respect to the variables. These are typical conditions for the application of such algorithms. In particular, the performance of the Mesh Adaptive Direct Search (MADS) algorithm and the Non-Monotone Black-Box Optimization for Constrained Problems (NM-BBOA_CP) algorithm are compared. All approaches are tested within time limits considered compatible with their actual inline application (less than one minute), considering a dataset of 50 logs, and the comparative results of the experiments are presented. The performance of these derivative-free algorithms, especially when combined with an initial starting point selection step consisting of a preliminary random exploration of the solution space, proved to be significantly superior to the heuristic in our experiments. The development of the black-box optimization procedures as well as the demonstration of the effectiveness of the overall approach as a viable method for FLO are among the most important contributions of the thesis, and represent an accomplishment of the main objective of this research.

Chapter 7 is devoted to the Ideal Cutting Pattern Problem (ICPP), dealing with a specific problem of determining ideal cutting patterns in a circle. This problem, differently from FLO, does not aim to effectively optimize an individual log, but to provide the basis for building a library of standard cutting patterns, given a diameter and a product list; this library is intended for a higher level of production optimization (see, for example, [15, 34], who propose integer programming models for this task). This type of approach is therefore best suited for fast cutting lines, where the cutting pattern does not change from one log to the other. A heuristic for this problem has already been developed by Microtec. In this chapter, we show how the ICPP problem can be approached with a mixed integer linear programming model. The model was implemented and solved using the Gurobi [22] solver, and tested on scenarios with different diameters and quantities of products used.

As the number of products increases, the MILP approach proves to be less effective than heuristics in obtaining a good solution in a time compatible for use in the application context. Nevertheless, the MILP approach proves useful for providing upper bounds, useful for verification purposes. A paper is under preparation for submission to an international journal, concerning the study presented in this chapter.

The last Chapter 8 provides some concluding remarks and suggests possible lines for future research.

Chapter 2

Introduction to industrial log cutting

Sawmill Log Cutting is a process that can be performed with different cutting technologies and different levels of automation. This process is divided in two main sawing stages, the first and second breakdown. The first breakdown is the cutting of logs into intermediate pieces, called flitches or cants, that requires an additional step, the second breakdown, in order to obtain the final boards. These tasks are accomplished through the use of milling machines and circular or band saws, which may incorporate measurement systems based on laser triangulation or other technologies to detect log size, position, and other characteristics. The selection of the appropriate cutting technology depends on the production requirements and desired quality standards of the wood being processed.

The choice of log cutting positions at various stages of the sawmill process, once reliant on the operators' experience to align the saw blades or the log based on visual inspection, is increasingly being delegated to automated systems. These systems enable optimal choices to be made quickly, in line with the speed of automated production lines. Consequently, the decision-making process for cutting at sawmills is strongly supported by optimization software. In these environments, the operator has to properly configure the optimization software itself by defining the constraints that cutting patterns must satisfy as well as the characteristics and value of the boards to be obtained from the logs. In more automated lines, which are of greater interest in this thesis, the operators' interaction typically reduces to mere supervision and maintenance tasks, with limited decision-making in the cutting process driven by unforeseen issues in the normal workflow.

Simpler cutting lines prefer cutting speed over flexibility. In these cases, logs are typically pre-graded into homogeneous groups by diameter, and the cutting pattern for each group is fixed and remains the same for the entire production run. In other lines, the main part of the cutting pattern remains fixed, but there are degrees of freedom that can be optimized, such as the rotation and orientation of the log, or the orientation and size of the outermost boards.

More flexible cutting lines, which are certainly more interesting from an optimization point of view, allow for choosing size and the position of each individual cut made on each log. The greater the flexibility of the line and the degrees of freedom available, the more an accurate measurement system, coupled with a proper optimization, will allow the line to reach its full potential.

In this chapter, we will overview the main technologies for log cutting and measurement that are of interest for the optimization problem studied in this thesis. The problem will be throughly described in Section 2.3, then, we will provide the relevant literature to our study.

2.1 Technologies related to log cutting

Although there is a wide variety of possible layouts for a sawmill, three main types of log cutting lines can be distinguished, with different typical processing speeds and varying degrees of flexibility in the choice of cut.

All the layouts considered here have in common a guillotine cutting procedure, i.e., all the boards obtained from the log are produced by a sequence of full cuts starting from one side of the log or residual block to the other side.

Figure 2.1 shows three examples of log-to-board breakdown. The first sequence is a typical fast cut sequence, where the log is cut entirely by passing through two consecutive saw blocks, with a 90° rotation in between. The second is a sequence that requires a saw carriage and many cutting steps. The last one requires many cuts and rotations, and is often associated with a cutting line with a merry-go-round system. The next sections will explain these configurations in more detail.

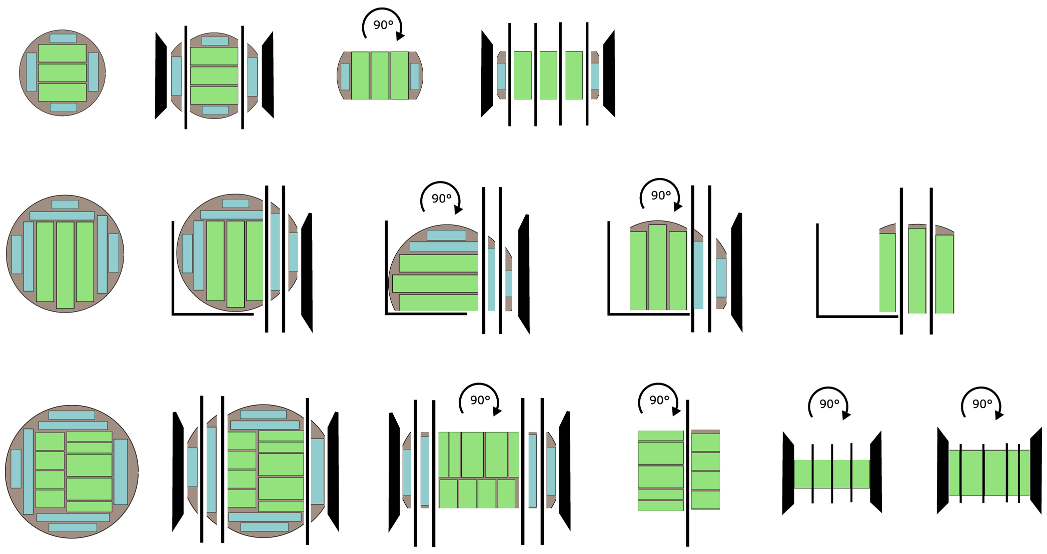


Figure 2.1: Examples of log breakdown: a typical two-pass fast cut sequence (on top), a sequence for saw carriage (in the middle) and a sequence with multiple passages in the saw block (on bottom)

2.1.1 Lines with saw carriage

A saw carriage is a machine that supports and guides the log through a cutting blade along its entire length (Figure 2.2). Each complete pass through the cutting zone produces a board, after which guides on the carriage advance the log transversely, by a controlled amount, to make the next cut.

The handling can be fully automated, controlled by an operator, or in a mixed mode, where the operator only controls the advancement of the log, while the transverse positioning, which determines the thickness of each cut, is handled automatically by a control software. The speed of this line is limited by the fact that each log requires several passes to complete the cut, interrupted by 90° or 180° rotations of the log on the carriage. In addition, the phase of loading and measuring the log on the carriage can cause additional slowdowns. These solutions often include equipment to improve the performance of the line. In some configurations, the saw blades are doubled or cut is performed even during the return movement, so that multiple cuts can be made during a single forward and backward pass. In other line configurations, only a few main cuts are made on the carriage, while splitting into thinner cuts is delegated to later machines.

Ideally, the sawmill carriage enables maximum flexibility in the choice of cut and the orientation and alignment of the log. It also allows the possibility of making some decisions about the cuts during the execution of the cut itself, depending on the internal quality that gradually becomes apparent during the cutting.

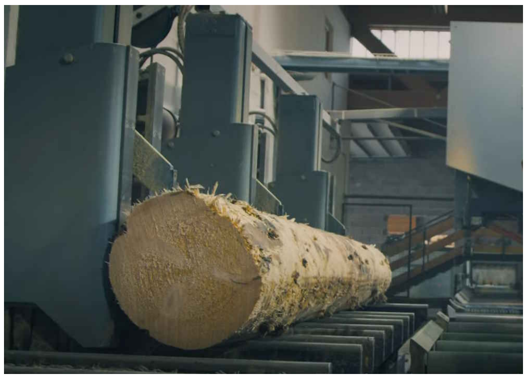


Figure 2.2: A log loaded in a saw carriage

2.1.2 Fast cutting lines

Fast cutting lines, usually designed for logs with diameters that are not too large, typically aim to maximize cutting speed (even over 120 m/min) at the expense of flexibility (see Figure 2.3). Typically, cutting is done with circular saws that do not change position between logs, so the line is fed with pre-selected logs that are all within the same very narrow diameter range. The distance between one log and the next can also be minimized to maximize efficiency, since the saw-block does not require significant changes between logs. It is usually possible to automatically control the centering and rotation of each log before it enters the cutting machine, and in some cases it is possible to vary the width or slope of only the outermost boards in the cutting pattern.

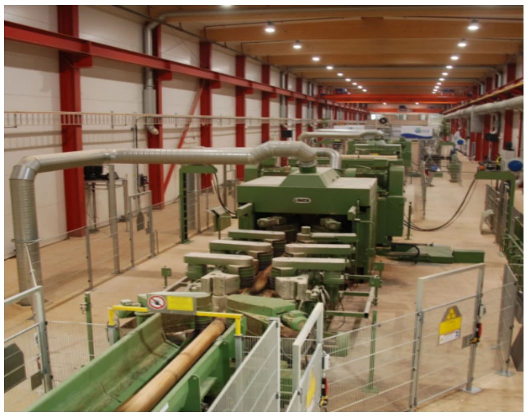


Figure 2.3: A typical high speed log cutting line

2.1.3 Lines with merry-go-round

Cutting lines with merry-go-round trade off the speed and the flexibility of the two previous solutions. The line is structured in such a way that the block of wood obtained from a first pass of the log through the saw-block can circulate again in the same saw-block, turned 90° or not, to complete the cutting pattern. The block of wood can, in principle, undergo several iterations in the same saw-block. There can be many different configurations for this type of line, with more or fewer stages to complete the cut after the first saw-block. Multiple passages of log parts inevitably results in a reduction in line efficiency, as the log infeed line must be stopped from time to time to process the returning parts. The merry-go-round line could in principle be replaced by additional stages all in line, but, typically, logs passing through these lines have different diameters and different cutting patterns, which may or may not require multiple passages. As a consequence, a long line with multiple stages would be underutilized, and this makes the merry-go-round concept a more convenient choice. This type of solution enables more elaborate cutting patterns, which is especially useful for larger logs.

2.1.4 Precision of the cut

Over time, technological evolution has led to greater precision in the execution of the log cut. Accuracy in the order of tenths of a millimeter is now usual in the positioning of the saw blades at this stage of the wood production chain. However, the most critical phase is the first cut made on the log (and possibly the first after a rotation of 90°). The reason for this is that subsequent cuts will refer to the already cut surface, thanks to well defined support points. In turn, the first cut is subject to an alignment of the log that depends on the accuracy of the measurement of the irregular external surface, and therefore on the calibration and resolution of the measuring system, but also and above all on the mechanics that must be able to quickly rotate and align the (heavy and irregular) log as it moves towards the cutting blade.

2.2 Technologies related to log measurement

The accuracy and type of measurement system play a key role in the process of automated log optimization and cutting. Here we describe three main types of measurement that can offer different optimization opportunities:

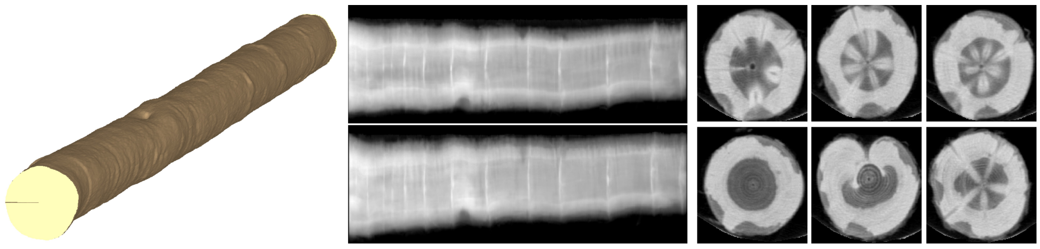


Figure 2.4: Log measurements: external shape (left), X-ray projections (center), slices from CT (right)

1. **external log shape**. This is undoubtedly the most common type of measurement in industrial sawmills. The simplest and most widely used technology is laser triangulation, which uses lasers and calibrated cameras to obtain a complete measurement of the external shape as the log moves forward, slice by slice (in the left in Figure 2.4);

2. **X-ray projections**. One or more X-ray projections are taken along the entire length of the log as it advances (in the center in Figure 2.4). The system consists of one or more fixed X-ray sources at different angles, each coupled to a linear sensor. A scanner for external shape measurement is usually associated with the X-ray system to complete the measure. This type of measurement provides important information about the internal quality of the log, such as the size of the heartwood and the distance of knots whorls. This additional information is very useful to select a cutting pattern for optimization that is best suited to the quality of the specific log.

3. **computed tomography (CT)**. This type of measurement (see Figure 2.5) provides very accurate information about the log, similarly to CT scanning in the medical field. The system consists of a rotating X-ray source combined with a wide sensor (cone beam configuration) that acquires a tomographic projection with a helical trajectory as the log moves forward. A tomographic inversion algorithm provides an accurate density measurement at each point within the log volume (in the right

in Figure 2.4). By processing this three-dimensional image, it is possible to identify defects within the log and create an accurate model of the log that can be used for log quality selection, as well as for very accurate cutting optimization.

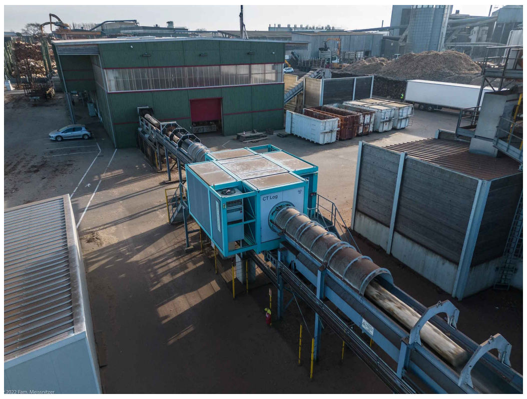


Figure 2.5: A CT scanner installed in a sawmill

2.3 Sawmill cutting optimization

The main objective of log cutting optimization is to maximize the value of the produced boards, trying to make the best use of all the available information provided by the measuring system and satisfying the constraints imposed by the specific cutting line. The value of a board depends on various aspects that are taken into account by applying complex rules, as explained in the following.

2.3.1 Valorization of a board

The grading of a board is usually done by testing rules based on the measurement of visible defects on its surface. Grading can be done according to standards that define norms and may vary from country to country (e.g. the DIN 4074-1:2012-06 standard popular in Europe, considered in [36], or the National Grading Rules for Dimension Lumber in the United States of

America [45]) or other more sawmill-specific production requirements. The grading of the board characterizes its quality and therefore its value.

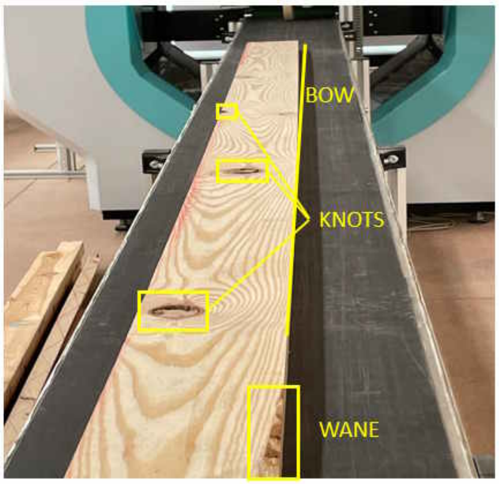


Figure 2.6: A board in the entrance of a board scanner.

Figure 2.6 shows a board on a belt at the entrance of a scanner. Some of the typical defects are highlighted, such as knots and wane. Another type of geometric defect, bowing, is also highlighted. Some types of defects, such as bowing, although important in the classification of the board, if due to errors in the cutting stages, drying stages, or handling of the boards, do not directly affect the log optimization process, so that they are not of particular interest in this discussion.

Some of the identifiable features on the board that can be checked for rules are listed below:

- **wane**. This defect is a consequence of the cutting position of the board in relation to the external shape of the log, and is therefore only present on boards that are cut closer to the log surface. Rules for this defect typically refer to its width, depth, and length. The rules may have different thresholds of validity, referring to the single out-of-threshold wane section or to a maximum total length allowed; sometimes a greater amount of wane is allowed at the head or tail of the board; sometimes the wane rules are evaluated individually for each side, other times the combination of both sides is considered;

- **knots**. Knots are the result of the intersection of the board with the part of the tree branches inside the log. They appear with a rounded or elongated shape, depending on how the surface of the board intersects the knot itself. It is often important to distinguish between a healthy knot and a dead knot. The knot on the board surface is healthy if it is well integrated with the surrounding wood and corresponds to the intersection of the board with a part of the branch that is still alive. On the other hand, a knot is dead if its contour is well separated from the board and tends to become detached. This means that the intersection with the board occurred at a point on the branch where it was already dead and the tree was growing around it. The rules expressed about knots mainly concern their dimensions, usually referring to their diameter, and their classification as healthy or dead. Often the rules refer to their position on the board, whether they are in the center or on the sides, their absolute number on the board, or their distribution;

- **pith**. The pith is the center of the annual rings of the log. Its presence on the surface of the board, and therefore only on boards that have been cut in the center of the log, is usually an aesthetic defect. Rules can be applied to its presence in terms of length on the surface;

- **heartwood**. Heartwood is the dense, inner part of a tree trunk, formed as the tree ages. It is characterized by its durability, resistance to decay, and often a distinct color. Depending on the species, the board produced inside the heartwood can have an higher value.

The grading of the board can be done by considering the board along the whole length of the log, or by considering the possibility of reducing its length. In the latter case, we may remove defects that would have caused the entire board to be downgraded, resulting in a shorter board with a higher market value. In the following discussion, the value associated to a board will be the value of the product (full-length or not) that can be obtained from the board itself, following the product-specific rules.

2.3.2 CT based log model

The tomographic reconstruction, which can be acquired for each individual log in a line where a CT scanner is installed, provides an accurate three-dimensional map of the internal density of the log. This map typically has a longitudinal (along the log length) resolution of 10 mm (one log slice per centimeter), and a transverse resolution of 1 mm. Figure 2.7 shows 4 slices taken from a log where some typical features such as the external shape of the log, knots, heartwood and pith can be easily identified.

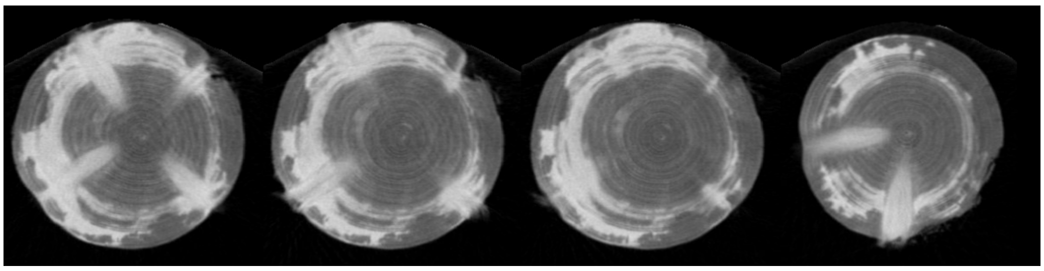


Figure 2.7: Example of different slices of the tomographic data of one log

The extraction of all of these and other features is done inline by an automated software that is able to provide the precise model of the log, i.e. a higher-level version of these features placed in a single reference system. The standard tomographic data acquired by the tomographic scanner corresponds to a sequence of images of log slices. The reference system for the three-dimensional model of the log is centered on the first of these images, as shown in Figure 2.8.

For each slice, the outer shape is given, defined by 360 points on the slice, and the same for the heartwood/sapwood boundary. The pith is given by its position on each slice. Splits, i.e. internal wood cracks usually starting from the pith, when detected, are given by a list of segments for each slice. Figure

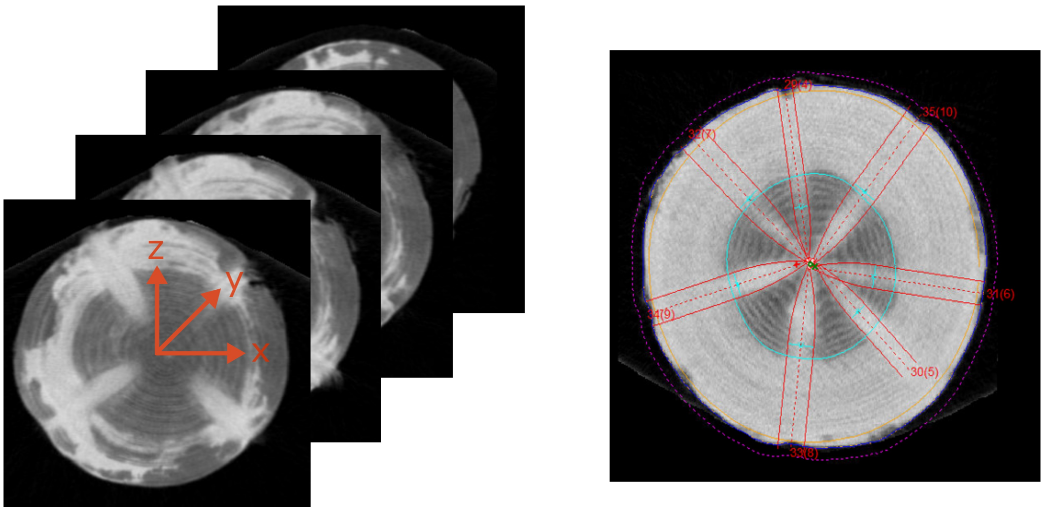


Figure 2.8: Reference for the log model, centered in the first slice image (left) and extracted defects superimposed on a slice (right)

2.8 shows some automatically extracted defects overlapped with one slice image. Some defects, however, are not defined based on slices, but a three-dimensional model is needed. To this end, tomographic data are processed to obtain a full *tree-dimensional log model.* For example, the knots are modeled by a set of parameters that characterize their position and the evolution of their diameter growth, starting from the pith [32]. This knot model also includes an estimate of the point at which the knot goes from healthy to dead (see, for example, the division of knots in Figure 2.9 into yellow healthy knots and red dead knots).

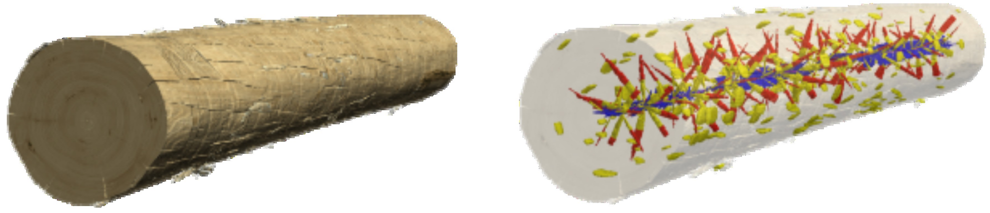


Figure 2.9: 3D volume from CT data (left) and extracted log model (right)

2.3.3 Virtual board

The three-dimensional log model makes it easy to obtain an evaluation of the overall quality of the log, which can be made by associating a measure to the identified characteristics and considering their average or other aggregated values. For example, it is possible to consider the average size of the knots, the number of knots with a dead part, the presence or not of internal splits in the log, the average size of the heartwood, the regularity of the pith, etc. These evaluations are undoubtedly useful for a quality-based selection of the log, and consequently also for an evaluation of a cutting pattern that is good, on average, for logs with certain features. However, the most important opportunity that this log model offers from an optimization point of view is to estimate (e.g., by simulating the cutting on the log model) the actual value that can be obtained from a board cut in that log at any desired position. This allows a software that performs cutting simulation and optimization to predict the value of each board obtained for the different cutting patterns it can construct. Once the positioning of a board is determined in the reference of the log model, a virtual board can be obtained, i.e. a representation of the defects on the four surfaces of the board resulting from the intersection of these surfaces with the model of the log (see Figure 2.10).

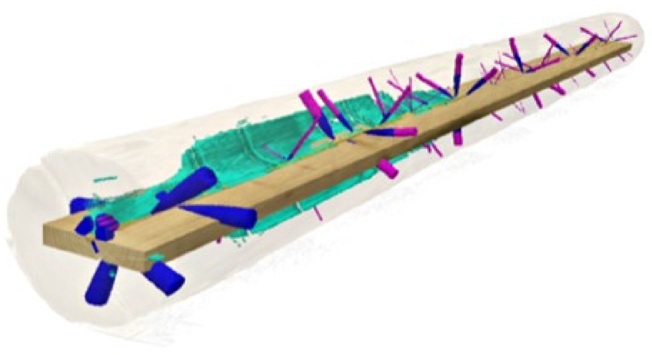


Figure 2.10: Virtual board on CT log model

The same rules mentioned in Section 2.3.1 can be applied to the virtual board (rather than the actual board), allowing its valorization. In principle, the configuration required by a board scanner to automatically assess the quality of a board can be integrated into the cutting optimization software before the board is physically produced, enabling advanced optimization beforehand. Figure 2.11 illustrates a comparison between the defects identified in a virtual board and the corresponding board actually obtained by the cutting process: the lower image is acquired from a board scanner, exhibiting perfect alignment with the upper image generated prior to cutting, where defects extracted from the intersection with the virtual log model are highlighted. It is on these defects that quality assessment rules for the virtual board will be applied.

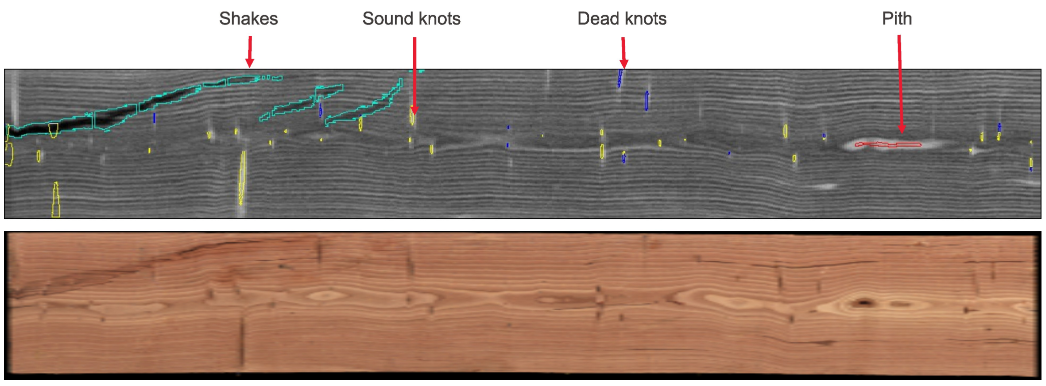


Figure 2.11: Comparison of the surface of the virtual board (top) with the color image of the real board surface, acquired after the cut (bottom). The defects of the virtual board are highlighted over the grayscale image corresponding to the density from the tomographic data

2.3.4 Flexible Log Optimization

In this section, we introduce the sawmill cutting optimization problem that will be considered in this thesis, which consists of creating an optimal cutting pattern for selecting the cut of a given log. Here we will identify the main elements of the problem, a more formal definition for the Flexible Log Optimization problem is given in Section 3.1. As introduced in Section 2.1, the variety of cutting patterns that can be used is very large, especially when slower and more flexible cutting lines are considered. In this research, we will focus on a particular type of cutting pattern that still provides a high degree of flexibility and usually requires a line with a merry-go-round system to complete the cut. Figure 2.12 shows some examples of cutting patterns that fall into this type. The cutting pattern always includes a block of *main boards*, shown in green in the figure. These are the main boards, cut full length in the log. The cuts related to main boards are called *main cuts*. The boards can be stacked in a single cut (patterns a, b, and c in the figure), or they can be arranged in several main cuts next to each other (patterns d and e). To the left and right of this block, there may be additional cuts with orthogonal boards (shown in blue in the figure). Above and below the block of main boards there may be additional cuts with boards placed side by side (in orange in the figure). The last two types of cut consist of boards called *side boards*, and related cuts are called *side cuts*. Side boards are defined as those boards that are usually less thick and may be shorter than the length of the log due to its irregular or cone-like shape, and they allow better filling of the log volume.

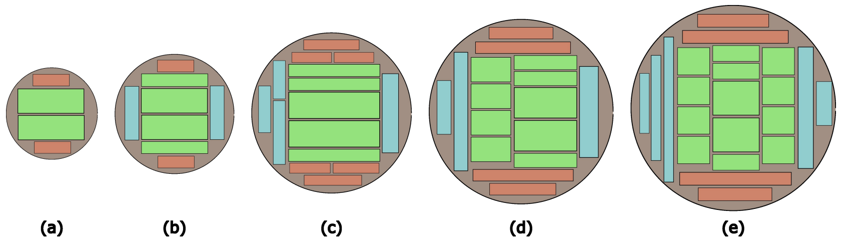


Figure 2.12: Examples of feasible flexible cutting patterns

The cutting pattern is constrained by a guillotine cutting sequence. A single cut separates the different blocks of boards, and a single cut separates two boards within each block. In addition to constructing the cutting pattern as a combination of cuts and boards, the log optimization problem must determine their position, rotation, and skew with respect to the log. In fact, we remark that, as we have seen in Section 2.3.3, the value of each board depends on its 3D position in the log. The objective is to maximize the total value of the resulting boards.

2.3.5 Fixed cutting pattern optimization approach

In this section, we describe the fixed cutting pattern approach to the log optimization problem sketched in Section 2.3.4, as currently implemented by the Maxicut Pro software. The Maxicut Pro software developed by Microtec is a log optimization software that is now installed in many cutting lines in Europe and United States in combination with the tomographic scanner. The software makes the choice of a cutting pattern for each log based on an accurate estimation of the value of each board produced. The estimation is obtained by creating virtual boards, as described in Section 2.3.3, and by intersecting them with the log model received from the log scanner. The software requires the setup of all possible products to be considered by the optimization, defined by a rectangular cross section and a set of possible lengths. Each product is associated with rules that define its quality and must be verified to assign its predetermined value. If a product does not satisfy the rules in a given placement in the log, it will not be considered valid by the optimizer. The optimization approach adopted by Maxicut Pro is based on a library of possible fixed cutting patterns, which must be preconstructed by the user from the list of board sections, depending on the possible products, using a graphical interface (see Figure 2.13).

The cutting pattern is constructed by inserting one or more main cuts side by side with a width equal to the width of the main boards to be inserted in the cut. The cutting pattern construction process then involves filling the main cuts with boards that are compatible in width, but whose thickness may vary from board to board. The construction of the cutting pattern then continues with the insertion of additional side cuts, to the left and right of the main block, whose width corresponds to the thickness of the side boards that are then inserted. Additional side cuts may also be inserted above and below each of the main blocks. Alternatively, the side cuts can be undefined

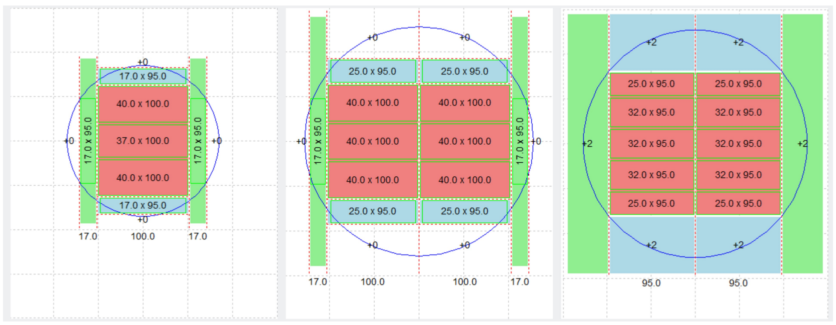


Figure 2.13: Examples of cutting pattern in Maxicut Pro: with a single main cut (left), with a double main cut (center), with a double main cut and free side cuts (right)

(we refer to them as free side cuts) or partially defined (i.e., only the width and number). In this case, the optimizer will take care of adding them during optimization, taking longer to calculate. The number of cuts and products used in the construction of the cutting pattern is free, but must clearly satisfy the constraints of the machine that will perform the cutting. The construction of the cutting pattern leads the software to determine a diameter range for which this cutting pattern can be valid (alternatively, the range can be assigned manually). This diameter will be important in the Maxicut Pro optimization procedure to select the cutting patterns (a limited number to maintain an acceptable computation time) that are most compatible with the log for which the placement attempt will actually be made. The placement attempt is defined in terms of $x - y$ position (or centering), rotation and skew of the whole pattern.

The fixed cutting pattern optimization procedure, executed by the Maxicut Pro optimization software, is shown in Figure 2.16. Upon receiving a new log to be optimized, the optimizer decides on a limited set of rotation angles. Based on the diameter of the log, a subset of cutting patterns is selected from the available patterns that are compatible with that diameter. For each rotation angle, the log pattern is rotated and centered; then each cutting pattern from the set of selected patterns is evaluated independently:

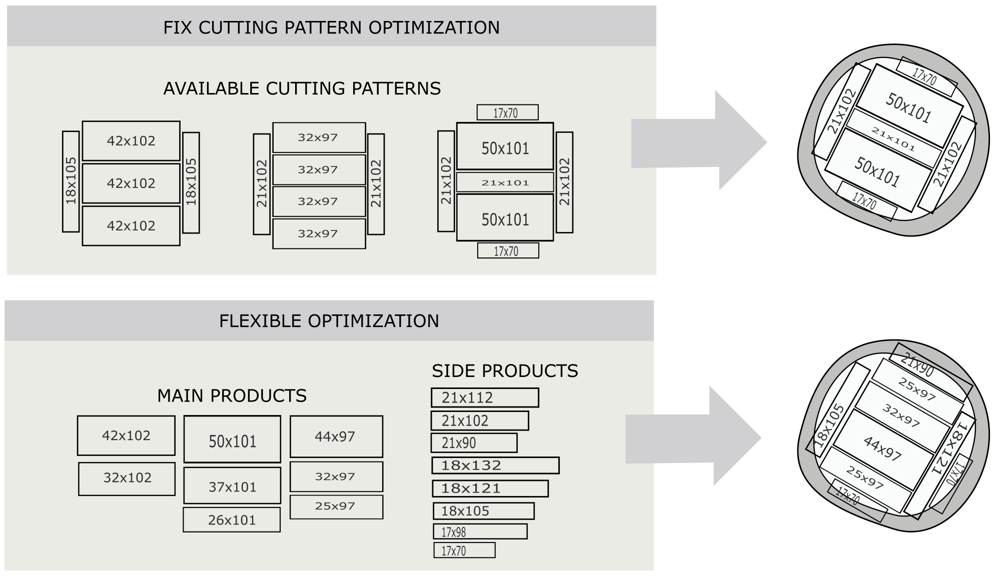


Figure 2.14: Fixed and flexible optimization approaches

the boards of each cut are tried considering a limited number of shifts and inclinations within the cut, compatible with the degrees of freedom of the line. For each pattern placement, the value of the boards is determined using the virtual board approach. Moreover, for each pattern placement, if necessary, a heuristic algorithm completes the cut by adding one or more side cuts and adds the value of the corresponding boards to the solution.

The approach presented here (fixed cutting pattern) remains a viable solution for fast cutting lines such as those in Section 2.1.2. Under these conditions, with few cutting patterns to test and a limited number of rotations and shifts, the total number of virtual boards to be computed during optimization is limited to a few thousand or tens of thousands. This has made the system suitable for inline use, simply by parallelizing the computations on multiple computers and reasonably limiting the steps of this exhaustive heuristic.

The aim of this research is to find a solution to extend optimization based on internal quality to cases of flexible cutting lines, for which the fixed cutting pattern approach is possible, but is a very limited solution compared to the potential of the line. In fact, flexible optimization starts from the products list to freely construct cutting patterns according to log characteristics, instead of just considering a limited number of predefined cutting patterns. This allows a better use of the degrees of freedom offered by the cutting line (see Figure 2.14).

2.4 Literature review

The optimization of the first and second breakdown in the sawmill, i.e. the two successive steps of cutting a log, first into cants and flitches, and then into boards, has been a problem addressed for a long time because of its high value-added margin. The research on this subject is still very important, because it is necessary to follow the evolution of the cutting processes, and the new opportunities offered by the measurement technologies.

At first glance, the log-cutting optimization problem belongs to the family of 3D packing problems, a widely studied optimization problem for its implications, e.g., in logistics. This problem, in its basic version, requires to place parallelepipeds of different sizes within a domain also defined as a parallelepiped. Several solution approaches have been proposed in the literature, relying on both exact and heuristic approaches and considering application-related variants of the basic problem (see [1, 11, 13, 9], among others). However, even if the boards inside a log can be assimilated to parallelepipeds, their arrangement in a cutting pattern is more related to a two-dimensional packing problem, as we have seen in Section 2.3.4. In fact, the cutting pattern is defined along the log section, and the impact of the third longitudinal direction, following the board valorization as defined in Section 2.3.1, is summarized in the board skew, since no further packing is required along the log length. It is thus interesting to explore literature related to the 2D packing problem. In the classical two-dimensional bin packing problem (2BP), a set of rectangular items is given. The problem is to allocate all the items, without overlapping, in a minimal number of rectangular bins of the same size. Even in this case, several variants are considered, including the ones related to cutting optimization: here, the allocated items define a cutting pattern to be executed in order to obtain the items themselves. In the seminal work [17], a first model to face this type of problem is given, denoting that in most of the common practical applications in industry, due to their specific constraints, the problem can be approached by mathematical programming. A survey on more recent advances for this type of problem can be found in [28], where several exact and heuristic approaches are reviewed.

It easy to see that the approaches proposed in literature for the basic versions of 2D or 3D packing problems can be hardly adapted to the log optimization problem we deal with. The same applies to the studied variants of these problems, most suited for application arising in logistics. Moreover, log cutting optimization presents specific constraints and features with respect to problems arising in other cutting contexts. In particular, the problem must consider not only a guillotine cutting mode, but also other constraints between cuts that depend on mechanical aspects that are specific to sawmills. Many studies have been done over the years regarding log cutting optimization in its several variants, and a very extensive survey on the topic has been recently presented in [24]. As early as the 1980s, solutions are being proposed to deal with this stage of the log processing at the sawmill. In [16], for example, a method based on two levels of dynamic programming is proposed for cutting patterns consisting of first-level side-by-side vertical cuts (divided by vertical lines in Figure 2.15), each one split into further second-level side-by-side cuts. In [4], in addition to the case of vertical side-by-side cuts (live sawing), the case of cutting patterns with a central block (cant sawing) and optimization with successive rotations (grade sawing) is analyzed, applying algorithms based on dynamic programming in all cases. In [12] a nested dynamic programming approach is extended to consider the log bucking, the preliminary stage of sawmilling where long logs are cut to standard sizes, together with the first breakdown optimization. In [23], the problem of filling a circumference with rectangles is addressed by setting up a Mixed Integer

Nonlinear Programming (MINLP) model, suitable for small logs, and specific heuristics for large ones. The proposed approach untangles the problem from the cutting patterns flexibility that can be actually achieved by automated cutting lines. As a consequence, some relevant constraints are neglected and the obtained solutions may require a degree of flexibility that is not compliant with real automated cutting lines. In [7], a heuristic to fill a circle with rectangles is proposed, using variable neighbourhood search and simulated annealing, but without considering typical cutting pattern constraints. Reference [10] deals with the optimization of the cutting pattern for Pinus Radiata, with the objective of improving the work time and the volumetric use of the log. It proposes a procedure based on dynamic programming, and considers a model of the internal defective core, estimated by the log external surface.

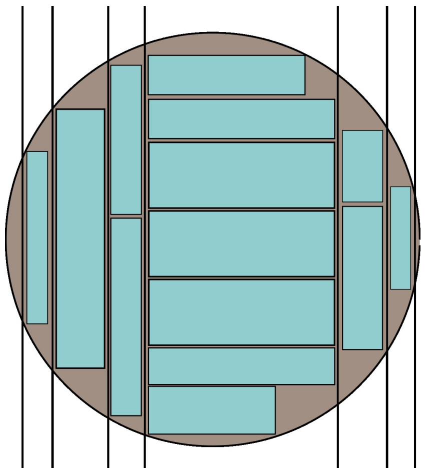
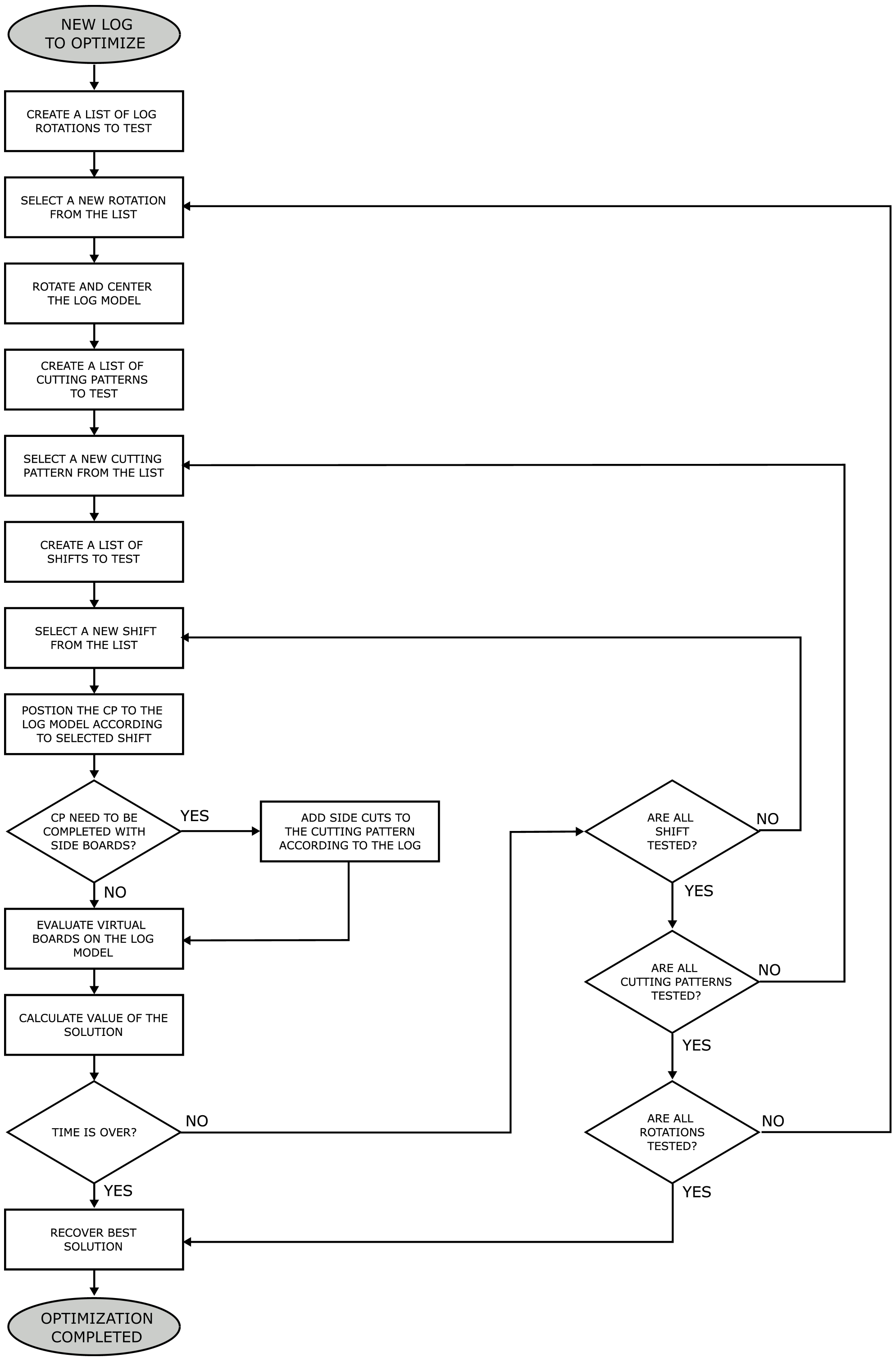


Figure 2.15: Example of two-levels pattern

The advent of *in-line* tomographic measurement of logs [19, 41] has increased the focus on how to integrate this measurement into the cutting optimization. Before in-line scanning, literature in this domain was based on log information from *off-line* tomographic scanners, usually collected in repositories of tomographic data, e.g., the database of Swedish logs presented in [21]. In general, the analysis of an optimization approach able to exploit the information available from tomographic acquisition requires the preliminary extraction of internal features that influence the value of logs and boards. Many studies have been done on modeling [32] and detecting [40, 33] knots, one of the most important internal log features from an optimization point of view. An approach to knots detection based on deep learning is described in [18], which also provides an execution speed compatible with the production line scanner. Other studies deal with automatic detection, based on tomographic reconstruction, of other internal log features, such as pith position [29] or internal splits [43].

Tomography-based cutting optimization enables a significant increase in value recovery, and this is the focus of many studies. For example, reference [5] shows a potential increment in boards value for Scandinavian sawmill processing of pine and spruce by applying an optimized rotation for each log instead of using the standard horns down angle. In the same contest, reference [14] presents an investigation that shows increments in value up to 21%, obtained by adjusting rotation, horizontal displacement and skew of the cutting pattern. Further researches that confirm the added value of an optimization based on tomographic data are for example [30, 31, 38]. A more recent work [36] presents a sawing simulation that takes into account rotation handling and knot detection accuracy, and shows an average value increase of 20% in an cutting optimization for strength-graded Douglas-fir timber. The importance of considering measurement and handling errors is also emphasized in [8], where a detailed analysis of the impact on the estimated board value due to errors in the measurement of the knots parameters is made. In [4], mathematical models and problem formulations related to the optimization of the sawing process of hardwood logs are presented. In this work, a dynamic programming approach is considered for the optimization, and an internal log defects model, based on tomographic data, is used for estimating the added value.

The Flexible Log Optimization problem defined in this thesis includes in its formulation most of the attributes of the sawmill cutting optimization problems analyzed by previous literature. From an optimization perspective, the main peculiarity of the problem relies in the combination of both the requirement to define an optimal cutting pattern following technology-specific constraints and the valorization of each board based on a complex procedure that depends not only on the item itself, but also on its position, hence on the cutting pattern itself. These features represent an optimization challenge that, to the best of our knowledge, has not been considered before. Some preliminary work, mostly related to the validation of the machine-learning based approach to board valorization presented in this thesis, has been the object of an internship at Microtec [6]. The development of the solution approach to flexible log optimization problem proposed in this thesis relies on state-of-the-art black-box optimization algorithms, namely, the Mesh Adaptive Direct Search (MADS) [2, 3] and the NM-BBOA_CP algorithm from the Derivative Free Library [26].



NEW LOG TO OPTIMIZE
CREATE A LIST OF LOG ROTATIONS TO TEST
SELECT A NEW ROTATION FROM THE LIST
ROTATE AND CENTER THE LOG MODEL
CREATE A LIST OF CUTTING PATTERNS TO TEST
SELECT A NEW CUTTING PATTERN FROM THE LIST
CREATE A LIST OF SHIFTS TO TEST
SELECT A NEW SHIFT FROM THE LIST
POSTION THE CP TO THE LOG MODEL ACCORDING TO SELECTED SHIFT
CP NEED TO BE COMPLETED WITH SIDE BOARDS?
YES
ADD SIDE CUTS TO THE CUTTING PATTERN ACCORDING TO THE LOG
NO
EVALUATE VIRTUAL BOARDS ON THE LOG MODEL
CALCULATE VALUE OF THE SOLUTION
TIME IS OVER?
NO
YES
RECOVER BEST SOLUTION
OPTIMIZATION COMPLETED
ARE ALL SHIFT TESTED?
NO
YES
ARE ALL CUTTING PATTERNS TESTED?
NO
YES
ARE ALL ROTATIONS TESTED?
NO
YES

Figure 2.16: Maxicut Pro: Fixed cutting pattern optimization procedure

Chapter 3

Specification of the flexible log optimization problem

In this chapter, we define the rules that, starting from a given list of possible boards, allow specifying feasible cutting patterns in the flexible cutting environment of reference for this thesis. The flexibility should not be interpreted as the possibility to place any board anywhere in the log, but the cutting pattern construction must comply with the restrictions imposed by the cutting line. A flexible cutting line (as described in Section 2.1.3) is used as a reference for defining cutting patterns for flexible optimization, which involves performing the log-to-board breakdown in a few sequential steps. Some examples of cutting patterns, from the simplest to the most complex, that fall under this definition, are shown in Figure 2.12, and a related cutting sequence is shown in Figure 2.1 (excluding the saw carriage breakdown). This chapter provides the basic definitions of product, board, and board valorization. They are necessary to define the different types of cuts, composed by stacked or side by side boards, and the cutting pattern, as a composition of these cuts. This will lead to the formal definition of the Flexible Log Optimization (FLO) problem, the subject of this thesis. The last section of this chapter presents the proposed solution approach, based on the decomposition of the problem into three interacting phases, namely board valorization, seed-constrained optimization and seed selection. We remark that not all cutting modes described in Section 2.1 are included in the definition of FLO, for example, saw-carriage cutting may allow a greater flexibility than FLO. However, the ideas behind the presented approach, e.g. the distinction into three phases, remain valid and can be adapted to specific applications with few adjustments of the last two phases.

3.1 Flexible Log Optimization problem

We consider a three-dimensional Euclidean space and assume that the log for which we want to construct a cutting pattern is placed in this space with one face in the xz plane and the other face in another parallel plane with $y = l$, where l is the length of the log. The y coordinate then represents the direction of the log length, as in Figure 3.1.

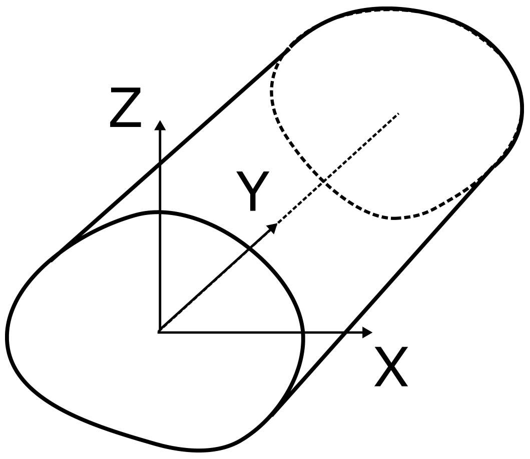


Figure 3.1: 3D Euclidean space with the log placed, where y axis corresponds to the length direction

The definition of the cutting pattern given in this section is intended to allow the cutting pattern to be constructed in 2 dimensions, on the xz plane, considering the boards as simple rectangles with sides parallel to the axes and spaced by a constant saw blade thickness.

3.1.1 Basic definitions

Definition 3.1.1 (section)**.** Let a section S be defined by the pair (w, t), where w is the width and t is the thickness of a rectangle.

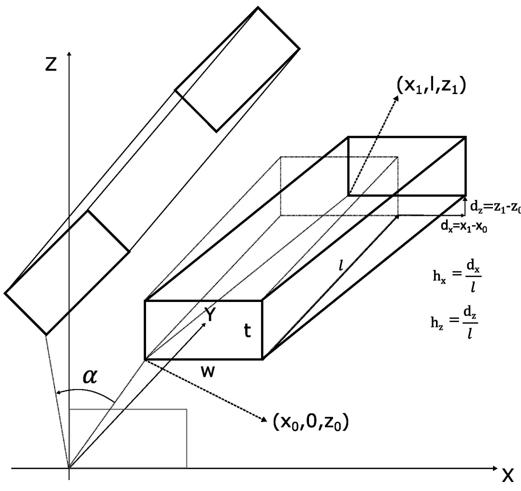


Figure 3.2: Graphical representation of section parameters (w, t) and position $(x_0, z_0, h_x, h_z, \alpha)$ defining a board in space

Definition 3.1.2 (board)**.** Given a section $S = (w, t)$ and x_0, z_0, x_1, $z_1 \in \mathbb{R}$, let

$$R_0 = [x_0, x_0 + w] \times [0] \times [z_0, z_0 + t],\ \ R_l = [x_1, x_1 + w] \times [l] \times [z_1, z_1 + t] \quad (3.1)$$

be two rectangles, R_0 in the plane $y = 0$, R_l in the plane $y = l$. A board B with section S and length l, generated by R_0 and R_l and rotated by an angle α, is identified by the set:

$$B = R_y \cdot (R_0 + \gamma\bar{v}), \quad (3.2)$$

where $0 \leq \gamma \leq l$, $\bar{v}$ is the vector $((x_1 - x_0)/l, 1, (z_1 - z_0)/l)$ and R_y is the rotation matrix of angle α around the y axis, i.e.:

$$R_y = \begin{pmatrix} cos(\alpha) & 0 & -sin(\alpha) \\ 0 & 1 & 0 \\ sin(\alpha) & 0 & cos(\alpha) \end{pmatrix} \quad (3.3)$$

Notice that, according to this definition, rectangles $R_y \cdot R_0$ e $R_y \cdot R_l$ correspond to the opposite top and end faces of the board B.

Definition 3.1.3 (board slopes)**.** Let h_x and h_z be, respectively, the first and third component of the vector $\bar{v}$ in (3.2), i.e., $h_x = (x_1 - x_0)/l$, $h_z = (z_1 - z_0)/l$.

Definition 3.1.4 (board position)**.** Let a board position L be a tuple $(x_0, z_0, h_x, h_z, \alpha)$, with x_0, z_0 position on the xz plane of the reference rectangle R_0 in (3.1), h_x and h_z respectively the slopes on the xy and zy plane as defined in (3.1.3), α the rotation angle around the y axis.

The above definition allow us to identify a board by a pair (S, L), indicating the region of space defined by a section S and its positioning L.

Considering a board as a parallelepiped in the xyz space, it can be seen that its width and thickness do not coincide with w and t defined by section S, if $h_x \neq 0$ or $h_z \neq 0$. In fact, the sizes w and t correspond to the width and height of the rectangle given by the intersection of the parallelepiped with the xz plane, whereas the width (resp. the height) of the board is larger than w (resp. t) if h_x (resp. h_y) is not zero: in presence of a slope, the board axis does not coincide with the log axis y, hence the section orthogonal to the board axis is not parallel to the xz plane. The choice to refer to the xz plane is dictated by implementation practices and measurement systems, which typically consider objects as logs and boards divided into slices along the y-direction. However, we assume that slopes are enough small that approximating the board section with its intersection with the xz plane does not have a significant impact on the optimization result. We thus assume

$$|h_x| \leq h_{max} \ , \ |h_z| \leq h_{max} \tag{3.4}$$

where $h_{max} > 0$ and is sufficiently small.

3.1.2 Board values

We remark that the definition of a board refers to a portion of the log volume, independently from the actual product (with related commercial value) that will be obtained from it. We thus formalize the following definitions.

Definition 3.1.5 (product)**.** An industrial board product p is defined by the tuple (S^p, v^p, R^p), where S^p is a section, $v^p > 0$ is a number that indicates the value of the product and R^p is a set of product rules that must be verified to validate the product, as defined in Section 2.3.1.

Definition 3.1.6 (product validation function)**.** Given an industrial board product $p = (S^p, v^p, R^p)$, a product validation function is defined as the binary function $f^P(p, L)$ that returns 1 if all related product validation rules are satisfied, i.e. the product p can be obtained from the board B defined by (S^p, L).

Definition 3.1.7 (set of section products)**.** Given a section S, let P_S be the set of all industrial product $p = (S^p, v^p, R^p)$ with $S^p = S$.

Definition 3.1.8 (board value function)**.** We define a board value function as a function $V(S, L) \geq 0$ that gives the value of the board defined by the section S and the position L.

Definition 3.1.9 (standard board value function)**.** A standard board value function V^{std} is defined as a function that assigns the value to a board based on checking the validity of products compatible with its section, and maximizing the value itself, more formally:

$$V^{std}(S, L) = \max_{p \in P_S} f^P(p, L) v_p \tag{3.5}$$

Function 3.5 represents the standard way of evaluating a board: among all defined products that can be obtained from this board (and which can be even shorter than the board itself), one searches for the highest value product and assigns this value to the entire board.

3.1.3 Type of cuts in the cutting pattern

We now define how boards can be stacked or placed side by side to create some basic cuts that will later be combined to build a complete cutting pattern. In the following definition, we denote by s the saw blade thickness.

Definition 3.1.10 (main cut)**.** A main cut C^M (shown in Figure 3.3) is defined by a width w^M, a position x^M, slopes h_x^M and h_z^M, vertical limits z_B^M, z_T^M, a value V^M, and a set of N^{TOT} boards $B_i = (S_i, L_i)$, with sections $S_i = (w_i, t_i)$ and board positions $L_i = (x_{0,i}, z_{0,i}, h_{x,i}, h_{z,i}, \alpha_i)$, $i = 1 \dots N^{TOT}$. Boards are divided in three groups: N^B bottom boards, N^M central boards, N^T top boards, where $N^B + N^M + N^T = N^{TOT}$.

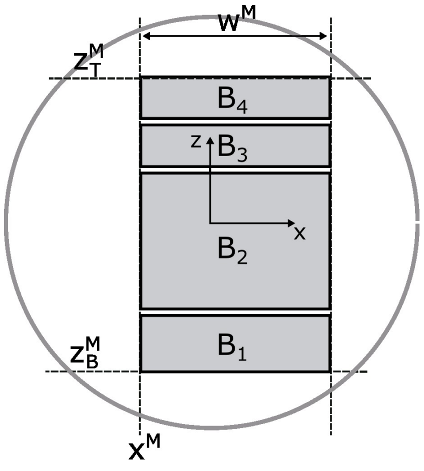


Figure 3.3: Main cut structure

The main cut must satisfy the following constraints:

- Consistency with respect to cut width:

$$w_i \leq w^M,\ i = 1...N^B \text{ or } i = (N^B + N^M + 1)...N^{TOT} \tag{3.6}$$

$$w_i = w^M,\ i = (N^B + 1)...(N^B + N^M) \tag{3.7}$$

- Boards inside the cut x-axis (also in l-position on the y-axis):

$$x^M \leq\ x_{0,i} \leq\ x^M +\ w^M - w_i,\ i = 1...N^{TOT} \tag{3.8}$$

$$x^M + h_x^M l \leq\ x_{0,i} + h_{x,i}\, l \leq\ x^M + h_x^M l +\ w^M - w_i,\ i = 1...N^{TOT} \tag{3.9}$$

- Slope h_z compatible for all boards:

$$h_{z,i}\ = h_z^M,\ i = 1...N^{TOT} \tag{3.10}$$

- Boards stacked on top of each other:

$$z_{0,i} = z_{0,i-1} + t_{i-1} + s,\ i = 2...N^{TOT} \tag{3.11}$$

- Cut vertical limits:

$$z_{0,1} = z_B^M, \qquad z_{0,N^{TOT}} = z_T^M \tag{3.12}$$

- Boards validity check based on the value:

$$V(S_i, L_i) > 0,\ i = 1...N^{TOT} \tag{3.13}$$

  where V is the board value function in Definition 3.1.8.

The value of the cut is defined as:

$$V(C^M) = \sum_{i=1}^{N^M} V(S_i, L_i) \tag{3.14}$$

The pattern can also contain side cuts, where the board widths are "orthogonal" to the board widths of the main cut. We define these cuts as follows.

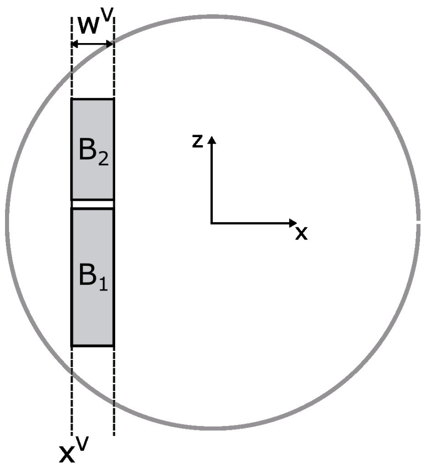


Figure 3.4: Vertical cut structure

Definition 3.1.11 (vertical cut)**.** A vertical cut C^V (shown in Figure 3.4) is defined by a width w^V, a position x^V, slopes h_x^V and h_z^V, a value V^V, a set of N^V boards $B_i = (S_i, L_i)$, with sections $S_i = (w_i, t_i)$ and board positions $L_i = (x_{0,i}, z_{0,i}, h_{x,i}, h_{z,i}, \alpha_i)$, $i = 1 \dots N^{TOT}$, that satisfy the following constraints:

1. Consistency with respect to cut width:

$$t_i = w^V,\ i = 1...N^V \tag{3.15}$$

2. Boards inside the cut with respect to x-axis:

$$-(t_i + z_{0,i}) = x^V,\ i = 1...N^V \tag{3.16}$$

3. Slope h_x and h_z compatible for all boards:

$$h_{x,i} = h_z^V,\ i = 1...N^V \tag{3.17}$$

$$h_{z,i} = -h_x^V,\ i = 1...N^V \tag{3.18}$$

4. Boards are positioned side by side:

$$x_{0,i} = x_{0,i-1} + w_{i-1} + s,\ i = 2...N^V \tag{3.19}$$

5. Boards validity check based on the value:

$$V(S_i, L_i) > 0,\ i = 1...N^V \tag{3.20}$$

In this cut we assume that boards are vertical, i.e., rotated by 90°, according to (3.3). The value of the cut is defined as:

$$V(C^V) = \sum_{i=1}^{N^V} V(S_i, L_i) \tag{3.21}$$

The pattern may also contain additional cuts at the top and bottom of the main cuts. We define these cuts as follows:

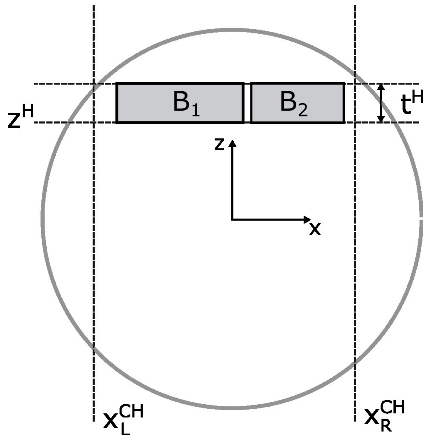


Figure 3.5: Horizontal cut structure

Definition 3.1.12 (horizontal cut)**.** An horizontal cut C^H (shown in Figure 3.5)is defined by a thickness t^H, a position z^H, slopes h_x^H and h_z^H, limits x_L^{CH},x_R^{CH}, a value V^H, a set of N^H boards $B_i = (S_i, L_i)$, with sections $S_i = (w_i, t_i)$ and board positions $L_i = (x_{0,i}, z_{0,i}, h_{x,i}, h_{z,i}, \alpha_i)$, $i = 1 \dots N^{TOT}$, that satisfy the following constraints:

1. Consistency with respect to cut thickness:

$$t_i = t^H,\ i = 1...N^H \tag{3.22}$$

2. Boards inside the cut with respect to z-axis:

$$z_{0,i} = z^H,\ i = 1...N^H \tag{3.23}$$

3. Boards inside the cut with respect to x-axis:

$$x_{0,1} \geq x_L^{CH}, \quad x_{0,N^H} \leq x_{RL}^{CH} \tag{3.24}$$

$$x_{0,1} + h_{x,1} l \geq\ x_L^{CH}\ + s_x^H l \tag{3.25}$$

$$x_{0,N^H} + w_{N^H}\ + h_{x,N^H} l \leq x_{RL}^{CH}\ + s_x^H l \tag{3.26}$$

4. Slope h_x and h_z compatible for all boards:

$$h_{x,i}\ = h_{x,i-1},\ i = 2...N^H \tag{3.27}$$

$$h_{z,i}\ = h_z^H,\ i = 1...N^H \tag{3.28}$$

5. Boards side by side:

$$x_{0,i} = x_{0,i-1} + w_{i-1} + s,\ i = 2...N^H \tag{3.29}$$

6. Boards validity check based on the value:

$$V(S_i, L_i) > 0,\ i = 1...N^H \tag{3.30}$$

The value of the cut is defined as:

$$V(C^H) = \sum_{i=1}^{N^H} V(S_i, L_i) \tag{3.31}$$

3.1.4 Flexible cutting pattern

A cutting pattern is given by the combination of several cuts of the types defined in Section 3.1.3, taking into account the constraints of a cutting system. The cutting pattern template defined here covers many possible cutting configurations. It consists of left and right vertical cuts and central main cuts with a limited number of horizontal cuts above and below. An example is shown in Figure 3.6.

Definition 3.1.13 (cutting pattern)**.** A cutting pattern is defined as a set of left vertical cuts C_i^{VL} for $i = 1...N^{VL}$, main cuts C_i^M for $i = 1...N^M$, right vertical cuts C_i^{VR} for $i = 1...N^{VR}$, top horizontal cuts C_i^{HT} for $i = 1...N^{HT}$, bottom horizontal cuts C_i^{HB} for $i = 1...N^{HB}$ with

$$N^{VL} \geq 0,\ N^{VR} \geq 0,\ N^M \geq 1,\ N^{HT} \geq 0,\ N^{HB} \geq 0 \tag{3.32}$$

and slopes h_x^{CP} e h_z^{CP}.

In order for the cutting pattern to be consistent with the cutting machine technology, the cuts that create the pattern must be subject to the specific constraints described in the following.

We extend the notation to include the features of the different type of cuts. For example, the position x^V of the left vertical cut C_i^{VL} is denoted by x_i^{VL}.

A cutting pattern must satisfy the following constraints:

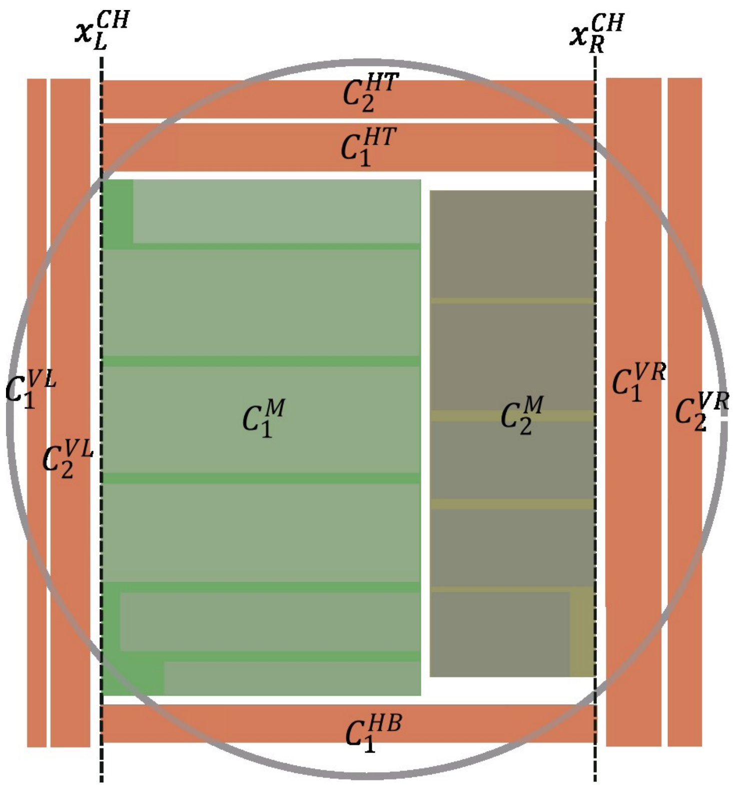


Figure 3.6: Example of cutting pattern with different cuts highlighted

1. Sequencing of vertical and main cuts:

$$x_i^{VL} = x_{i-1}^{VL} + w_{i-1}^{VL} + s,\ i = 2...N^{VL},\ \text{if } N^{VL} \geq 2 \tag{3.33}$$

$$x_1^M = x_N^{VL} + w_N^{VL} + s,\ \text{if } N^{VL} \geq 1 \tag{3.34}$$

$$x_i^M = x_{i-1}^M + w_{i-1}^M + s,\ i = 2...N^M \tag{3.35}$$

$$x_1^{VR} = x_N^M + w_N^M + s,\ \text{if } N^{VR} \geq 1 \tag{3.36}$$

$$x_i^{VR} = x_{i-1}^{VR} + w_{i-1}^{VR} + s,\ \ i = 2...N^{VR},\ \text{if } N^{VR} \geq 2 \tag{3.37}$$

2. Consistency of horizontal slope of main and vertical cuts:

$$h_{x,i}^{VL} = h_{x,j}^M = \ h_{x,k}^{VR} = \ h_x^{CP},\ i = 1...N^{VL}, j = 1...N^M, k = 1...N^{VR} \tag{3.38}$$

3. Consistency of vertical slope of main and horizontal cuts:

$$h_{z,i}^{HB} = h_{z,j}^{M} = h_{z,k}^{HT} = h_{z}^{CP}, \; i = 1...N^{HB}, j = 1...N^{M}, k = 1...N^{HT} \tag{3.39}$$

4. Sequencing of horizontal and main cuts:

$$z_i^{HB} = z_{i-1}^{HB} + t_{i-1}^{HB} + s, \; i = 2...N^{HB}, \text{ if } N^{HB} \geq 2 \tag{3.40}$$

$$\min_{i \leq N^M} z_{B,i}^{M} = z_N^{HB} + t_N^{HB} + s, \text{ if } N^{HB} \geq 1 \tag{3.41}$$

$$z_i^{M} = z_{i-1}^{M} + t_{i-1}^{M} + s, \quad i = 2...N^{M} \tag{3.42}$$

$$z_1^{HT} = \max_{i \leq N^M} z_{B,i}^{M} + s, \text{ if } N^{HT} \geq 1 \tag{3.43}$$

$$z_i^{HT} = z_{i-1}^{HT} + z_{i-1}^{HT} + s, \; i = 2...N^{HT}, \text{ if } N^{HT} \geq 2 \tag{3.44}$$

5. Limits of horizontal cuts:

$$x_L^{CH} = x_1^{M} \tag{3.45}$$

$$x_R^{CH} = x_N^{M} + w_N^{M} \tag{3.46}$$

It is also assumed, in each of the previous constraints, that the rotation α of each board considered in the various cuts and in the cutting pattern is always the same, with an offset of 90° for the boards in the vertical cuts.

Definition 3.1.14 (cutting pattern value)**.** The value V^{CP} of a cutting pattern is defined as the value given by the sum of the value of all the boards contained in it, that is:

$$V^{CP} = \sum_{i=1}^{N^{VL}} V(C_i^{VL}) + \sum_{i=1}^{N^{HB}} V(C_i^{HB}) + \sum_{i=1}^{N^{M}} V(C_i^{M}) + \sum_{i=1}^{N^{HT}} V(C_i^{HT}) + \sum_{i=1}^{N^{VR}} V(C_i^{VR}) \tag{3.47}$$

The board value function, which must be defined for each possible board used in the optimization, implicitly defines the validity range of the cutting pattern (the value is strictly positive).

The goal of flexible optimization is to find the cutting pattern that gives the maximum value among all the cutting patterns that can be created according to the previous definitions and constraints, given a certain number of possible sections.

Some additional constraints can be taken into account, however, they are out of the scope of this thesis. For example, additional constraints may be related to:

1. Symmetry. A certain symmetry is required between the side cuts for the stability of the cut;
2. Cutters. Vertical and horizontal cuts, when milled directly in-line to obtain the final boards in a single pass, are constrained by the mechanics of the cutters. For example, it may be necessary, for the outermost board, to be completely contained within the profile of the innermost board;
3. Core boards. It is required that some boards, defined by a specific products set, are positioned in the center of the log;
4. Necessity of a centrally positioned cut. A cut is required in the center of the log between two boards.

3.1.5 Flexible Log Optimization problem definition

In the following, we define the Flexible Log Optimization problem:

Definition 3.1.15 (Flexible Log Optimization - FLO)**.** Let P be a set of industrial products. Let V be a board value function, defined for each section S^p such that $p = (S^p, v^p, R^p) \in P$, and for each possible board position L. We want to determine the cutting pattern CP, as from Definition (3.1.13), with the maximum value V^{CP}.

Depending on the cutting line, additional constraints can be imposed on the construction of the pattern, for example, more restrictive limits on the number of cuts allowed for each type of cut that constitutes the cutting pattern, like

$$N^V_{min} \leq N^{VL} \leq N^V_{max},\ N^V_{min} \leq N^{VR} \leq N^V_{max} \tag{3.48}$$

$$N^M \leq N^M_{max} \tag{3.49}$$

$$N^{HT} \leq N^H_{max},\ N^{HB} \leq N^H_{max} \tag{3.50}$$

or other restrictions that affect the positional degrees of freedom (e.g., restrictions on the allowed values of slopes h^{CP}_x or h^{CP}_z, or on the rotation of the entire cutting pattern). Another type of restriction, strictly related to the saw line technology, involves the maximum number of boards that can be included in each cut. For example, in some lines, the maximum number of boards for each vertical or horizontal cut is set to 1.

Note that this problem definition does not make any specific reference to the log to be optimized. Information about the log, its length, its internal properties and its placement are all summarized in the valorization function V (see Definition 3.1.8).

3.2 A three-phases optimization approach to Flexible Log Optimization

The approach we propose to address this problem is based on its decomposition into two distinct decision levels. At the outermost level, some elements of the pattern are selected, such as the widths of the vertical and main cuts, and their global positioning and rotation variables. This first decision, which actually sets some initial constraints on the pattern, is what we will call *seed*. The second level is the completion of the optimization after fixing the seed's variables, which requires to insert and valorize the individual boards in the cutting pattern. The decomposition leads to the proposed approach to FLO, which is structured as a composition of the following three interacting phases, as schematized in Figure 3.7:

1. Board valorization. The first, most basic phase is the evaluation of a board. Whenever the optimizer decides to check the position of a board in the log, it needs the most accurate estimate of its value (ideally its actual value);

2. Seed-constrained optimization. The second phase corresponds to the optimization of all the second level's decision variables, assuming that the seed is fixed. In our implementation, as we will detail in Chapter 5, the second level's variables describe the vertical sections of the log, whereas the seed describes the horizontal position of the vertical and main cuts of a pattern. This phase can be seen as the *evaluation* of a given seed, in particular of the main cuts that define it, and represents the most computationally intensive task, as explained in Chapter 5.

3. Seed selection. The third phase corresponds to an exploration of the seeds' space. Each of the explored seeds is the starting point of the second phase. In particular, a seed includes the variables that define the subdivision of the pattern into vertical and main cuts and the remaining variables related to the positioning of the pattern, such as the rotation α around the y-axis and the slope h_x common to all cuts in the log.

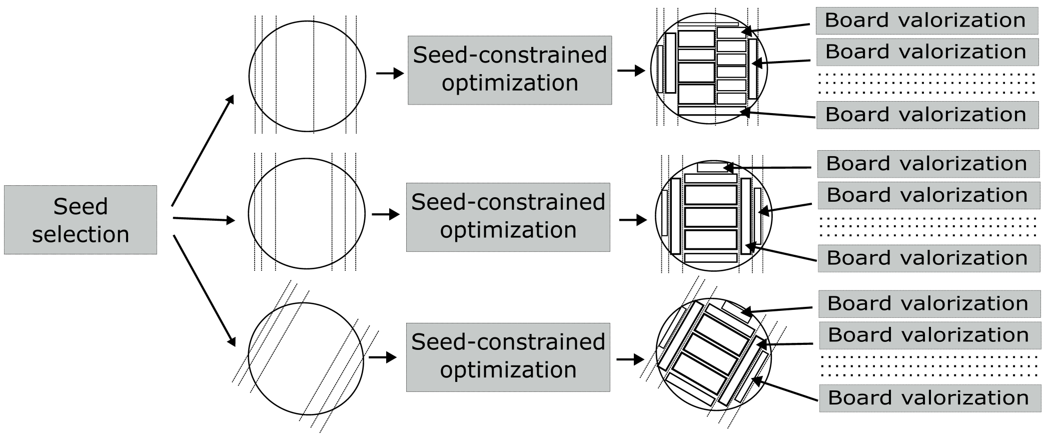


Figure 3.7: Diagram of the three-phases FLO approach

The three phases optimisation approach is schematically represented in Figure 3.7. In principle, the approach pre-selects a list of seeds or search them throughout the optimization process (seed selection phase). Each of these seeds is passed to a procedure that searches for the highest value solution obtainable from that seed (seed-constrained optimization phase) until some termination conditions are satisfied (e.g., maximum running time). During the second phase, we expect a large number of requests to the first phase, to obtain the value of the boards at the required positions.

The following chapters will explore and evaluate different approaches to each phase.

Chapter 4

Board valorization

As introduced in Section 2.3.3, the valorization of a board from tomographic data can be performed by creating a virtual board and validating the individual products that can be derived from the board by multiple simple rule checking.

We are not concerned in this discussion with the details of the implementation of the valorization functions (see Definitions 3.1.6 and 3.1.8) based on the creation of a virtual board and by 3D intersection of defects, but we assume that these are available functions that we will use as a reference in the performance evaluation.

However, as we can easily guess and as we will see in the comparative analysis, the evaluation based on virtual boards is computationally very expensive and its intensive use within an optimization algorithm severely limits its exploration capabilities. Therefore, two different approaches have been developed to trade off evaluation accuracy and speed, which are presented and experimentally compared in the following sections.

4.1 Rules Map (R-Map) approach

This approach aims to speed up the evaluation of individual rules defined for board products. A *Rules Map* is associated with a matrix whose values are computed to allow quantitative evaluations of defects in the log. The maps compress the defect-related information along the longitudinal dimension of the log, providing, for positions in the xz space, aggregated values such as the number of intersected knots or the maximum length available inside the log for each cross-section position. An important feature of these maps is that they are computed independently of the product section S, in the sense that the same maps are potentially valid for every section, but they depend in part on the positioning L (and more precisely on the slopes and angle of rotation) for which an evaluation is required from time to time.

In the following definitions, we assume that the two-dimensional maps are computed with a fixed discretization step with respect to the x, z axes, using the same reference system as the one used for the cutting pattern definition given in Section 3.1.4 (roughly speaking, the origin correspond to the center of the first log slice). Let $\delta_{xz} \in \mathbb{R}, \delta_{xz} > 0$ be a discretization step (valid for both axes), $N \geq 1$ an integer value denoting the number of sampled positions in each direction, i.e., each map will include a $N \times N$ matrix. In other words, each element (i, j) of the matrix is related to a point $(x, 0, z)$ in the xz plane with coordinates:

$$x = (i - N/2)\delta_{xz} \tag{4.1}$$

$$y = (j - N/2)\delta_{xz} \tag{4.2}$$

To retrieve values related to any point in the xz plane, we make use of interpolating functions, that can be defined as follows.

Definition 4.1.1 (interpolating function). An interpolating function $f^I(M, x, z)$, given a point $(x, 0, z)$ in the xz plane, returns an interpolated value over multiple elements of a matrix $M \in \mathbb{R}^{N \times N}$, the ones corresponding to the sampled points that are close to the required point $(x, 0, z)$.

Definition 4.1.2 (*R-Map*). Let a *R-Map* be a tuple $(M, \delta_{xz}, \alpha, h_x, h_z)$, where $M \in \mathbb{R}^{N \times N}$ is a matrix reporting the discretized information on aggregated value of a given defect, δ_{xz} a discretization step and α, h_x, h_z the rotation and slope variables related to a board positioning L as in Definition 3.1.4.

The meaning of the values of the matrix M depends on the defect the map wants to model and the mode of modeling. In general, given a board $B = (S, L)$, with $L = (x_0, z_0, h_x, h_z, \alpha)$, the element (i, j) of a map computed for angle α and slopes h_x, h_z, corresponds to the modeled value of the defect aggregated along the longitudinal direction of the board at the position $((i - N/2)\delta_{xz} - x_0, (j - N/2)\delta_{xz} - z_0)$, relative to a reference system aligned to the board.

Assuming that one has implemented a suitable set of R-Maps, one can obtain an alternative product validation function $f_M^P(p, L)$ that is directly based on these maps. Consequently, one can define an alternative board value function that assign a value to a board based on checking the validity of products compatible with its section, as in Definition (3.1.9), using R-Maps:

$$V_M(S, L) = \max_{p \in P_S} f_M^P(p, L) v_p \tag{4.3}$$

where, we recall, P_S is the set of products compatible with the section S, and v_p the value of the product p.

We notice that the use of the R-Maps approach during optimization, requires the pre-computation of maps for all possible α rotations and h_x, h_z slopes of the cutting patterns that can be constructed during optimization (a choice that, as we will detail in Chapters 5 and 6, depends on the seed selection phase of the optimization approach defined in Section 3.2).

The following subsections describe two examples of R-Maps implemented for fast estimation of rules related to the presence of wane on board edges and to the number of knots on surfaces.

4.1.1 Wane rule's evaluation with R-Map

The choice of implementation for this type of map is related to the verification of product rules related to the presence of wane. The value of each map element corresponds to the total log length crossed by the line passing through the point related to the map position, and can be calculated as a function of the considered angle α and the slope h_x, h_z. Note that the length shown by the map depends on the shape of the log, indicating, as a first approximation, for a board edge in that position, the distance of the point where the wane would begin on that edge with respect to one end of the board. The line where this length is calculated is obtained directly from the Equation 3.2:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = R_y \begin{pmatrix} ((i - N/2)\delta_{xz} \\ 0 \\ (j - N/2)\delta_{xz} \end{pmatrix} + kR_y \begin{pmatrix} h_x \\ 1 \\ h_z \end{pmatrix} \tag{4.4}$$

where R_y is the rotation matrix along the y axis:

$$R_y = \begin{pmatrix} cos(\alpha) & 0 & -sin(\alpha) \\ 0 & 1 & 0 \\ sin(\alpha) & 0 & cos(\alpha) \end{pmatrix} \tag{4.5}$$

It is clear from the definition of this type of map that the values of the matrix M_w, the one included in the definition of the wane R-Map, will be between 0 and l. See for example Figure 4.1. The central part of the map has a constant value of l, corresponding to the presence of the log along the corresponding lines between $y = 0$ and $y = l$. The outermost region of the map is constant at 0, meaning that there is no intersection with the log for the lines associated with those points in the map. On the other hand, the values in the transition zone, highlighted in the right image, depends on the intersection of the lines with the outer shape of the log.

Let us now consider a product of length l that must satisfy the following two wane-related rules:

- no wane larger than d_w is allowed in the top or bottom of the product;
- the maximum total length of the wane on the top or bottom of the product is equal to l_w.

Given a board $B = (S, L)$, with $S = (w, t)$ and $L = (x, z, h_x, h_z, \alpha)$, the validity check for that product will be based on the wane map $(M_w, \delta_{xz}, \alpha, h_x, h_z)$,

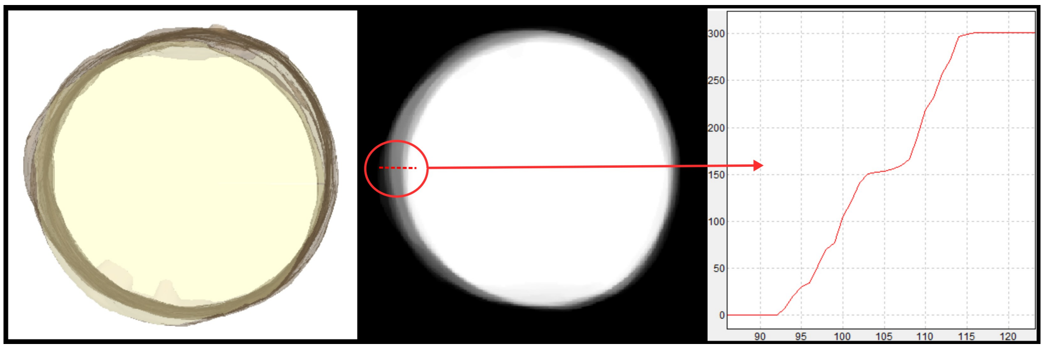


Figure 4.1: R-Map for wane. Representation of the 2D projection of the log shape (left), full length wane map (center), map value in the highlighted region (right)

after verifying the following conditions:

$$f^I(M_w^r, x + d_w, z) = l \tag{4.6}$$

$$f^I(M_w^r, x + w - d_w, z) = l \tag{4.7}$$

$$f^I(M_w^r, x + d_w, z + t) = l \tag{4.8}$$

$$f^I(M_w^r, x + w - d_w, z + t) = l \tag{4.9}$$

$$f^I(M_w^r, x, z) \geq l - l_w \tag{4.10}$$

$$f^I(M_w^r, x + w, z) \geq l - l_w \tag{4.11}$$

$$f^I(M_w^r, x, z + t) \geq l - l_w \tag{4.12}$$

$$f^I(M_w^r, x + w, z + t) \geq l - l_w \tag{4.13}$$

where the function $f^I(M, x, z)$ is a suitable interpolating function as from Definition 4.1.1.

4.1.2 Knots rule's evaluation with R-Map

We present here a type of map designed to check some specific product rules related to possible knots on the surface of the board. The value of each element $M_{i,j}$ of the matrix corresponds to the number of knots with a width in the x-direction greater than a given threshold d_k at their intersection with the plane of the equation $z = z_0$ and with the x-coordinate of the knot intersection center $x < x_0$, where $x_0 = (i - N/2)\delta_{xz}$ and $z_0 = (j - N/2)\delta_{xz}$, rotated and tilted according to the usual variables α, h_x, h_z for which the map is computed, according to Equation 3.2.

Figure 4.2 shows an example of knot R-Maps. The image in the middle reports, in gray scale, the number of knots with a width greater than 10 mm, calculated for $\alpha = 0, h_x = 0, h_z = 0$. The image on the right shows the map for knots larger than 20 mm. Note that the values are lower (darker colors) because the first map also counts all knots in the second map.

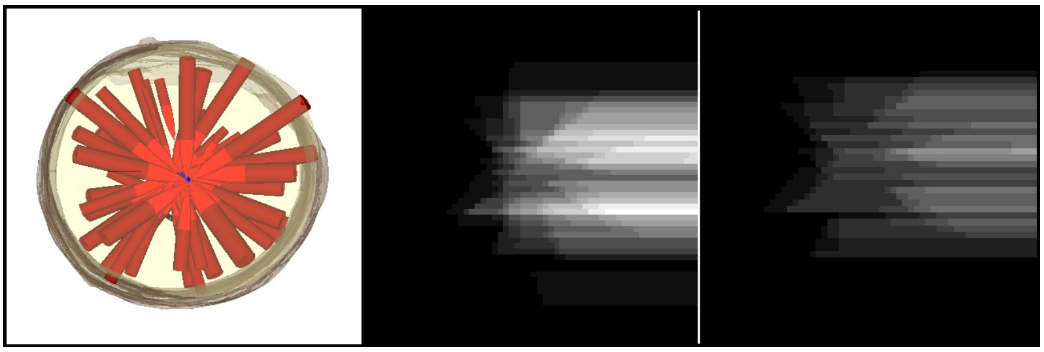


Figure 4.2: R-Map. Log model with knots (left), knot map for knot width > 10 mm (middle), knot map for knot width > 20 mm (right)

Let us now consider a product of length l that must satisfy the following two rules regarding knots:

- no knots with width greater than d_{k1} are allowed in the top or bottom side of the product;
- at most 3 knots with width greater than d_{k2} are allowed in the upper or lower face of the product.

Given a board $B = (S, L)$, with $S = (w, t)$ e $L = (x, z, h_x, h_z, \alpha)$, the validity check for that product will be based on knot maps $(M_{k1}, \delta_{xz}, \alpha, h_x, h_z)$

and $(M_{k2}, \delta_{xz}, \alpha, h_x, h_z)$, including the following conditions:

$$f^I(M_{k1}, x+w, z) - f^I(M_{k1}, x, z) = 0 \tag{4.14}$$

$$f^I(M_{k1}, x+w, z+t) - f^I(M_{k1}, x, z+t) = 0 \tag{4.15}$$

$$f^I(M_{k2}, x+w, z) - f^I(M_{k2}, x, z) \leq 3 \tag{4.16}$$

$$f^I(M_{k2}, x+w, z+t) - f^I(M_{k2}, x, z+t) \leq 3 \tag{4.17}$$

where the function $f^I(M, x, z)$ is an interpolating function as from Definition 4.1.1.

In order to evaluate the presence of knots in the sides of the board, in a similar way, it is necessary to calculate other R-Maps related to the intersection of the knots with the plane of equation $x = x_0$ to evaluate their width with respect to the direction z.

4.1.3 Other R-Maps examples

Other R-Maps have been developed as well. We describe them more synthetically and show them as examples in Figure 4.3:

- R-Maps for dead knots. In this case, the maps are computed in the same way as the knot maps seen in Section 4.1.2, but, for the calculation of the maps, only the intersection of the dead part at the end of the knots, if present, is considered.

- R-Maps for heartwood. Since the boundary between the heartwood and sapwood inside the log is modeled as the outer shape, the maps are calculated in the same way as the wane (see Section 4.1.1);

- R-Maps for pith. The pith in the log model is defined by a single coordinate (x, z) for each longitudinal position. The total length of the pith in the y-direction, corresponding to the length of the log, is distributed among the different elements of the map matrix based on the discretization performed by the δ_{xz} step.

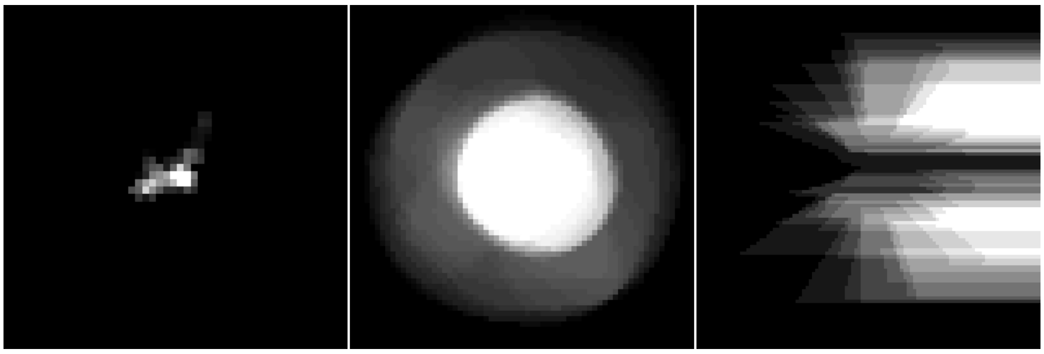


Figure 4.3: Example of R-Maps for other defects. From the left: pith length, heartwood length, dead knots count

4.1.4 Weaknesses of the R-Maps approach

There are two major limitations associated with this R-Map-based approach:

- the approximation of rules by maps;
- the lack of suitable maps for some rules.

The first point concerns the fact that the estimation with the map is an approximation. For example, if the rule about knots has a threshold of a certain width (e.g., no knots wider than 25 mm are allowed on the surface), in order to validate it, we have to look for the precomputed map with the closest knot size. If the rule is specific to the diameter of the knot, and the map is only able to estimate its width, as in the example presented in Section 4.1.2, the evaluation computed using the map may be inaccurate.

Another level of inaccuracy of the map concerns the longitudinal position of the defects. For example, a product to be evaluated could be shorter than the length of the board and could be obtained by trimming the board. In this case, defects should only be evaluated on the useful portion of the board. This effect can be partially mitigated by computing multiple maps for different longitudinal portions of the log, but it cannot be eliminated unless maps are computed for every possible longitudinal location of every possible product length.

The second point concerns the fact that new or specific rules or defects to be taken into account during optimization may require additional maps that are not yet implemented. Adding rules or defects requires specific analysis and implementation, and we cannot exclude cases in which the nature of the rule or of the defect prevents the application of the R-Maps approach. In this case, rules are ignored in the initial phase of the optimization, where a large number of solutions are analyzed, and are considered only in the final phase, where the best solutions are verified by the slower method based on virtual board construction.

These two critical issues together lead the optimizer to make board evaluations that may even differ significantly from the actual value. Even if the evaluation using these maps is actually used to select a limited subset of cutting solutions that will later be re-evaluated more precisely, these errors can often prevent the exploration of potentially better cutting solutions than those that are finally selected.

From the computational perspective, by using R-Maps, the evaluation of a board takes much less time than the more accurate evaluation by virtual board creation, but still requires multiple memory reads in the execution of the procedures that verify all the rules of the possible products, as needed to give a final value to each board.

4.2 Value Map (V-Map) approach

An ideal solution to speed up the board valorization phase would directly provide the value of a board given its location within the log, without the need for further processing, by simply reading its value from a special map associated with the position and orientation of the board. This is the idea behind the Value Map (V-Map).

4.2.1 V-Map definition

The formal definition of V-Maps is similar to R-Maps, with the important difference that for V-Maps the value of each matrix element does not correspond to some approximation of a log defect, but directly to the value of a board positioned with its center in the point associated to the map element. It follows that the V-Map also depends on a specific section S of a board. Figure 4.4 illustrates this concept.

Definition 4.2.1 (*V-Map*)**.** Let a *V-Map* be a tuple $(M, S, \delta_{xz}, \alpha, h_x, h_z)$, where $M \in \mathbb{R}^{N\times N}$, S is a section (w, t), δ_{xz} a discretization step and $M_{i,j} = V^{std}(S, L)$ (see Definition 3.1.9), with $L = ((i-N/2)\delta_{xz}+w/2, (j-N/2)\delta_{xz}+t/2, \alpha, h_x, h_z)$.

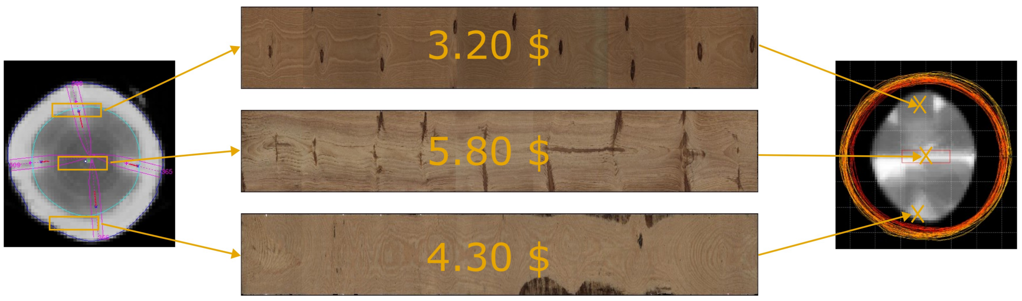


Figure 4.4: Example of a V-Map. On the left, a tomographic section of the measured log with defects highlighted and three possible placements of a board. In the middle, the surface of the board obtained from each of these three placements, where the corresponding value is shown. On the right, the value map related to this board, where the value of the three points shown corresponds to the value of the boards (black points identify value 0, white points the maximum value).

We notice that a V-Map refers to a specific cross section S and to specific slopes h_x, h_z and rotation α. As a consequence, to perform an optimization based on these maps, it will be necessary to precompute all the necessary V-Maps with respect to the discretization steps chosen for these variables. Therefore, the procedure for calculating value maps is as follows:

Std dev. of knot diameter measure error	2 mm
Std dev. of knot dead knot border measure error	10 mm
Std dev. of knot length measure error	5 mm
Knot measure detection rate	98%
Std dev. of rotation error	3°
Std dev. of alignment error	4 mm

Table 4.1: Errors considered in knot measure and log handling

- For each section S, slopes h_x, h_z and rotation α considered by the optimization:
  - For each element $M^v_{i,j}$ related to the map $(M^v, S, \delta_{xz}, \alpha, h_x, h_z)$
    - Create the virtual board $B = (S, L)$ where $L = ((i - N/2)\delta_{xz} + w/2, (j - N/2)\delta_{xz} + t/2, \alpha, h_x, h_z)$.
    - Assign to $M^v_{i,j}$ the value $V^{std}(S, L)$, by verifying the validity of all products with section S and by selecting the validated one with higher value.

We remark that the V-Map directly provides the value of a board, which corresponds to the value of the best product that can be obtained from it, without the need for further verification of individual products.

4.2.2 Stochastic V-Map

The previously defined value maps do not take into account the unavoidable inaccuracy of the log measurement process and the handling error during the cutting phase. To account for these errors, a Monte Carlo approach based on repeated evaluations of each virtual board is proposed, taking into account experimentally verified measurement and handling errors, modeled as Gaussian-type, zero mean and variance errors, as shown, e.g., in Table 4.1.

As an illustrative example, we consider V-Maps influenced by knots. For each evaluation of a virtual board, the parameters of each knot in the log model (see Section 2.3.2) are randomly perturbed, in such a way that the knot diameter, dead knot border and length are changed according to a normal distribution of mean 0 and standard deviation as in Table 4.1. The dead knot border is defined in the knot model as the position of the point, relative to the knot origin point, at which the knot changes from being sound to dead. The given knot detection rate is used by eliminating from the log model each knot with a probability of 2%. The rotation and alignment error impact the entire log model by rotating it and applying shifts in the x-direction and z-direction according to the parameters in Table 4.1.

At the moment, the stochastic model only accounts for errors associated with knot measurement and log handling. The parameters of the stochastic model were based on assumptions and field experience, and their refinement will be part of future work.

Figure 4.5 shows an example of an ideal V-Map and a corresponding stochastic V-Map, obtained with 25 Monte Carlo evaluations changing the log model according to the error values reported in Table 4.1.

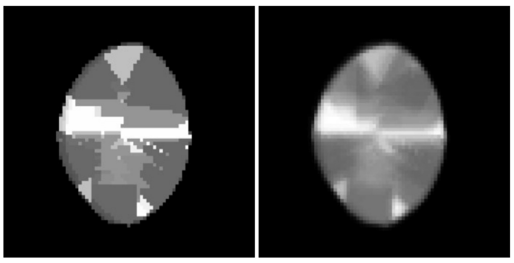


Figure 4.5: Example of ideal value map (left), and stochastic value map with 25 launches (right)

4.3 Value Maps through Convolutional Neural Networks

Evaluating the value maps for each optimization requires a very large amount of computational resources. In fact, a single map relative to a single section requires the evaluation of $N \times N$ virtual boards on which to evaluate all the product rules associated with that section. Using the stochastic approach, this number is multiplied by a further factor. Assuming an angle step of $d_\alpha = 5°$, it will be necessary to evaluate $N_R = 180°/d_\alpha = 36$ rotations; a reasonable number of slope steps is given by $N_{sx} = 5$, $N_{sz} = 5$; a sufficient number of sections involved in the optimization is $N_T = 14$; the chosen number of repetitions for the stochastic approach is $N_{STOC} = 25$; considering an average time $T_{VB} = 1ms$ for the evaluation of a virtual board, it is easy to estimate that the creation of these maps for each log may require a computing time of the order of

$$T_{MAP} = T_{VB} N^2 N_T N_R N_{sx} N_{sz} N_{STOC} \simeq 1300 \text{ hours}. \tag{4.18}$$

Therefore, the possibility of obtaining an approximation of the value maps in a faster way is proposed, using an approach based on Convolutional Neural Networks (CNN) [25].

CNNs are machine learning tools suitable for image processing [25]. The basis for obtaining good results from this type of approach usually requires having a significant amount of data available for training and validation. In our case, we have a set of images processed from the tomographic scan of the log (as we will detail in Section 4.3.1) as input of the CNN, and the value maps as reference output (ground truth). At present, we have a large number of tomographic scans of logs available, with different characteristics in terms of size, quality and species, since tomographic scanners have been installed in many sawmills for years. In addition, once the optimization specifications are defined in terms of products and rules, value maps and network inputs can be automatically calculated, without the need for supervision or manual marking. This is an ideal situation for a deep learning approach.

4.3.1 CNN definition

CNNs are a deep learning model primarily designed for working with images. In fact, CNNs are very effective in tasks such as classification, segmentation or object detection [39, 37]. Their structure is designed to emulate the biological processes involved in the processing of the visual field by the visual cortex, by means of a layered structure based on small convolutional filters. Each of these filters specializes during the learning process to identify specific patterns. The classical structure of a CNN network consists of these elements:

- **Convolutional layers**: these layers consist of sets of convolutional filters that slide across the entire image;
- **Pooling layers**: these layers reduce the spatial dimension of the output after the convolutional layers. They do this by applying maximum or average functions to reduced windows;
- **Activation functions**: non-linear activation function (as the classical ReLU) are applied after convolutional and pooling layers;
- **Fully connected layers**: layers typically at the end of the CNN that connect all the neurons of the previous layer with all the neurons of a new layer;
- **Output layer**: it depends on the task of the network; for example, in the case of classification, the output layer returns the probability values for each class considered.

For our purposes, the CNN network architecture must be able to produce as output a value map, which is an image related to the xz section of the log. To this end, it is necessary to determine how to encode the log model information in order to provide it as input to the network. The entire tomographic reconstruction of the log, typically a volume of 700x700x400 voxels, is likely to be unmanageable in data size and more difficult for the network to interpret defects. Therefore, we start from the log model, described in Section 2.3.2, in particular by creating several two-dimensional maps containing the information of individual features and defects. We call these maps *D-Maps*, and their definition is similar to the R-Maps described in Section 4.1. Since we need to determine a correspondence between input images and output images, the type of task required from the CNN is a pixel-level regression [46], which leans the choice for the CNN towards a U-net architecture [35], as schematized in Figure 4.6.

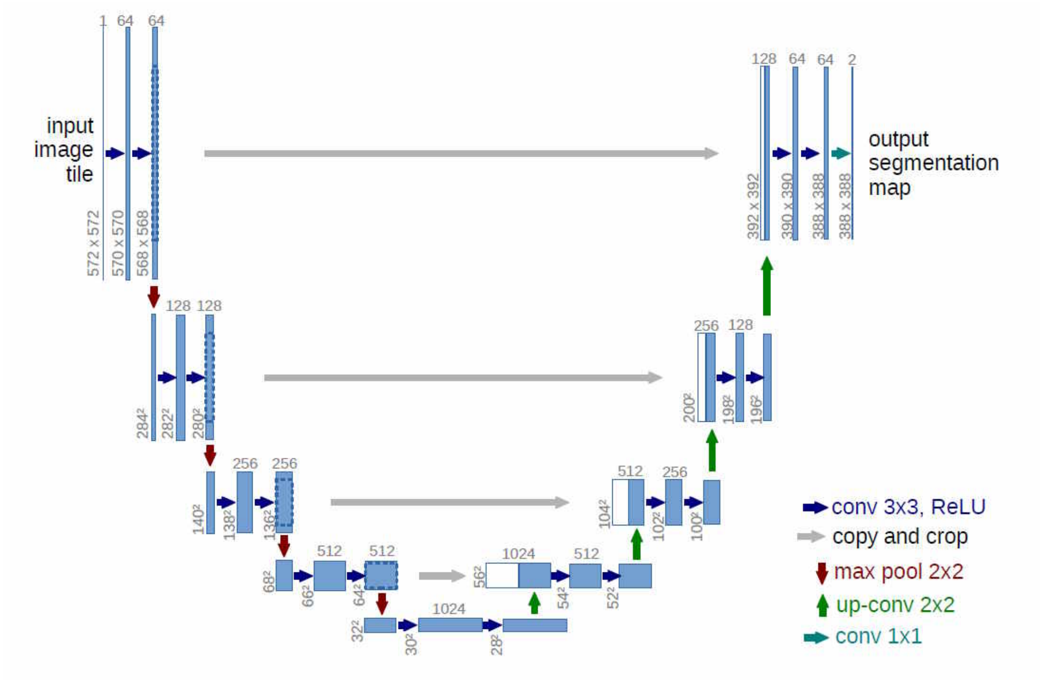


Figure 4.6: U-net architecture (example from [35])

We here provide an overall description of the implementation of the proposed CNN, details can be found in [6].

For each section $S = (w, t)$ needed for the optimization, it will be necessary to train a specific network, whose output will be the value map for the S section and whose input will be a set of D-Maps. The D-Maps we used for the validation of the approach are calculated for sections at 200 mm step along the length of the log, leading to, e.g., a total of 21 elements per type of D-Map, to cover a standard log length of 4 meters. We consider the following D-Map types:

- D-Maps for shape: same as R-Map for wane in Section 4.1.1, but calculated in portion of 200 mm length (see left image in Figure 4.7);

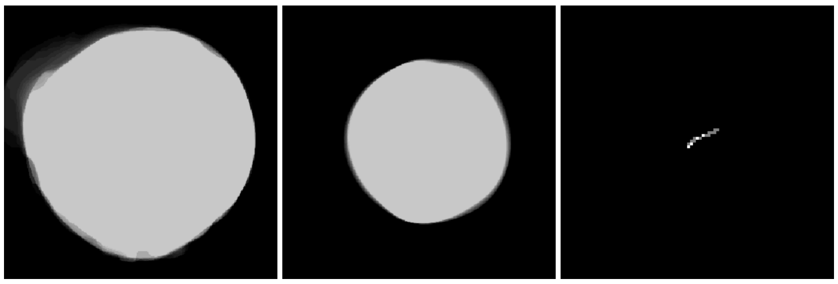


Figure 4.7: Examples of D-Maps: Shape (left), Heartwood (center) and Pith (right)

- D-Maps for knots: an overlapped mask of knot presence. Multiple knot maps are generated for different sizes of knots, e.g., diameter greater than 10, 20, 30, 40 and 50 mm (see left image in Figure 4.8);
- D-Maps for dead knots: same as previous, but considering only the dead portion of each knot (see right image in Figure 4.8);
- D-Maps for heartwood: same as shape D-Maps, but considering the shape of the heartwood region instead of external shape (see center image in Figure 4.7);
- D-Maps for pith: projection of the pith position on the 200 mm length (see right image in Figure 4.7).

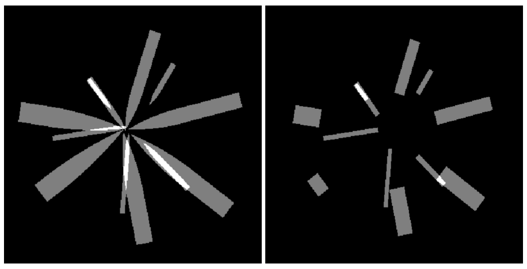


Figure 4.8: Examples of D-Maps: knots (left) and dead knots (right)

Each CNN has been trained with different inputs, depending on the type of rules it has to check (e.g. if there are no pith or heartwood rules for a specific section, these D-Maps are not added to the network input). In general, the input of each network is an image with many channels (even more than 100), 21 for each type of D-Map considered useful.

We remark that it is not necessary to perform specific training on the rotation angle α and possible slopes h_x, h_z of the pattern. In fact, in the inference phase, it will be sufficient to rotate or skew the log model to generate the rotated or skewed D-Maps as input to the CNN. For example, Figure 4.9 shows the output of the inference of the same CNN network applied to the log model, rotated at different angles, to obtain the V-Maps corresponding to these rotations. This assumption makes it more likely that this approach can be used in practice, as it reduces the time to compute the single value map required to train each section to about 5 minutes per log, according to Equation 4.18. These operations can be further accelerated and parallelized to obtain the V-maps of all the logs needed for proper training (about 3000 in our experiments) in a reasonable time.

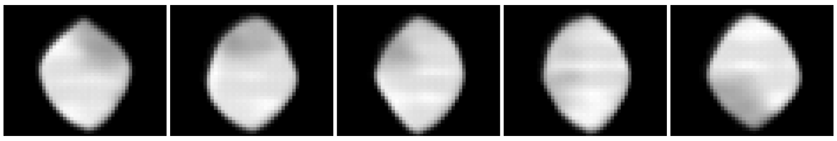


Figure 4.9: Example of V-Map inference results using a 45° rotation step, for the same log and same section

4.3.2 Integration of the CNN-based V-Maps into the optimization process

Figure 4.10 summarizes the log optimization process where CNNs are used to compute V-Maps. The process is divided into two stages, an offline preprocessing step, and an inline stage where the optimization procedure is executed after a new log is measured. More in details, we have:

- **Offline preprocessing**

  Depending on the set of instances that the optimizer has to be able to

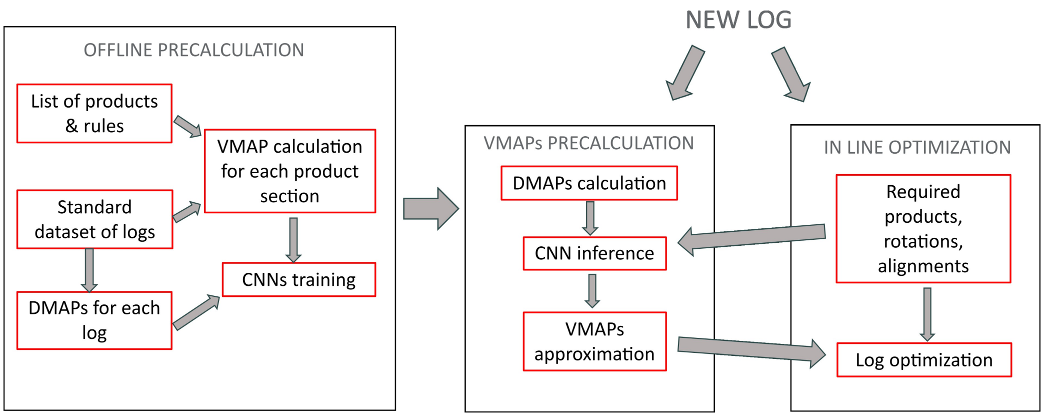


Figure 4.10: Optimization process integrating V-Maps training

face, and thus on the list of all possible products and their rules, all CNN networks needed for inline inference have to be trained. Changes to the rules or products require new training to update the corresponding networks. Training requires computing the ground truth and input of the networks based on a standard log dataset. The input corresponds to D-Maps, which can be computed once and for all, since they do not depend on the optimization settings. The ground truth corresponds to stochastic value maps (see Section 4.2.2), which can be computed point by point using the virtual board approach. The time required can be remarkable, but the system can be designed appropriately considering that this preparation can be done offline. For example, one can think of a centralized server that provides these specific evaluation networks to the various online optimization instances, while maintaining an up-to-date library of these networks over time;

- **Inline optimization**
  In the production line, upon arrival of each log that is required to be optimized, it will first be necessary to calculate the D-Maps for that log. Depending on the procedure of the optimization and the sections needed, the inference of the CNN networks is made to obtain the approximate V-Maps. Based on these maps, the optimization can then

proceed with the most appropriate algorithm (in particular, the three-phases approach described in Section 3.2). The computation times for the creation of the D-Maps and the inference of the networks, as verified in the experimental evaluation, are already of an acceptable order of magnitude to be used inline.

4.4 Numerical results

The methods presented in this chapter provide an implementation of the first phase (board valorization) of the overall three-phase approach presented in Section 3.2. The goal of this first phase is to provide a fast and reliable board evaluation. In this section, we first report on a computational analysis of the effectiveness of the different valorization approaches. To this end, in Section 4.4.1, we compare the results of a same optimization procedure for FLO that embeds the different board evaluation approaches proposed in this chapter. In Section 4.4.2, we provide an empirical analysis of the timing associated to CNN-based V-maps, towards their embedding in optimization procedures in real cutting environments.

4.4.1 Results of log optimization with different valorization approaches

A test has been performed using a data set of 122 logs measured with the tomographic scanner, from which the log model has been derived. In Figures 4.11, 4.12 and 4.13, a characterization of this dataset is shown based on the diameter, length and species distributions.

In terms of products and rules, the optimization setup is similar to that described in [6], specifically 4 sections are defined for main products and 2 sections for side products with multiple lengths. The inserted rules cover only the 4 defects for which both V-Maps and R-Maps approaches are currently developed, and these are wane rules, rules on maximum size and number of

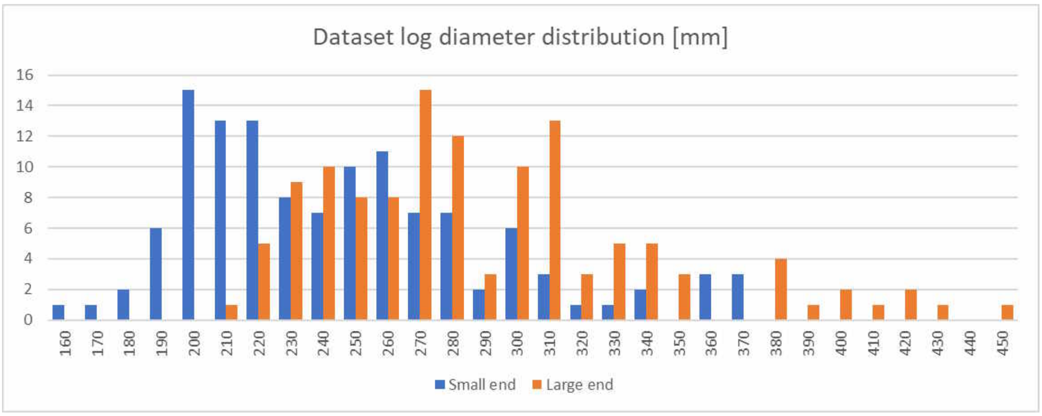


Figure 4.11: Dataset logs diameter distribution. Small end refers to the diameter of the smallest end of the log, large end to the largest

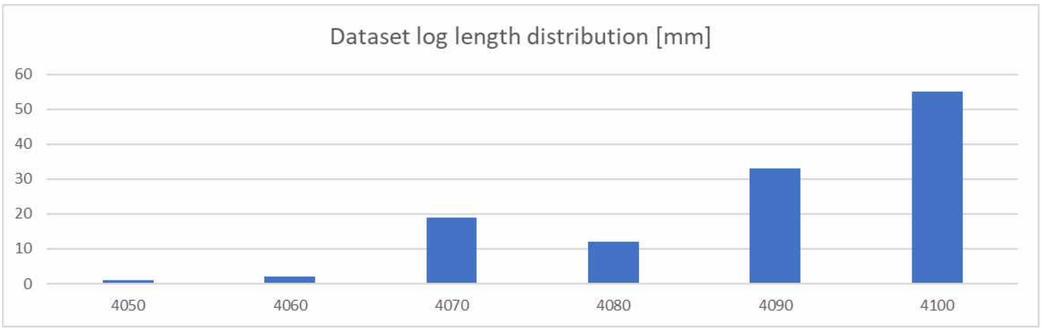


Figure 4.12: Dataset logs length distribution [mm]

healthy and dead knots, percentage of heartwood, and presence of surface pith.

Compared to the model defined in Section 3.1, the optimization procedure used for the test reported in this section considers fewer degrees of freedom, because a large number of value maps must be precomputed as a reference for the optimization. In particular, all boards in the cutting pattern have the same slopes h_x and h_z and rotation $\alpha_{CP} = 0$. Optimization based on stochastic V-maps, computed exhaustively for each log and section, is considered as the reference solution against which to compare the other two approaches to board valorization defined in this chapter, namely R-Maps and CNN V-Maps.

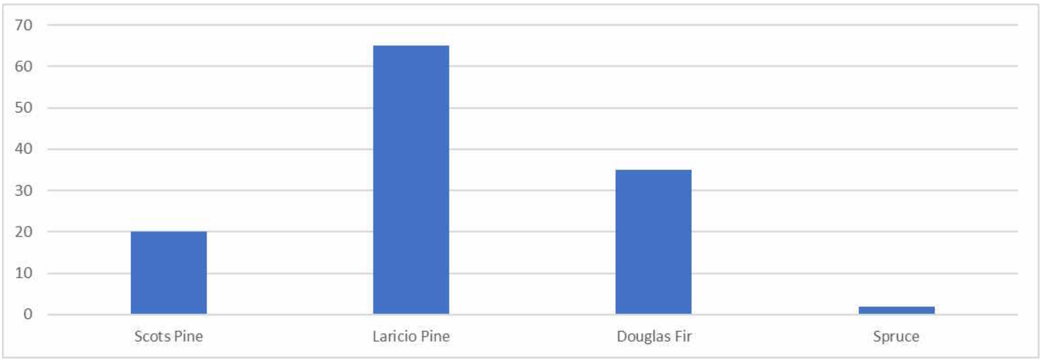


Figure 4.13: Dataset logs species distribution

Test mode	**Value**	**Value %**	**Avg time (s)**
Stoc. V-Map (reference)	2135.46	100.0%	7.4
R-Maps	1976.11	92.5%	14.5
CNN V-Maps	2112.15	98.9%	8.5

Table 4.2: Optimization results with different approaches for board valorization

Table 4.2 summarizes the results of our experiments. Column *Value* reports the total value of the boards obtained by the optimization of all logs in the test set. Column *Value %* reports the percentage of value obtained with respect to the baseline, that is given by the Stochastic V-Maps approach. This approach, as shown by the calculation in Equation 4.18, is not suitable for inline use, due to the long generation times needed by related V-Maps. The last column shows the average computation time for one log. Note that no rotations or slopes of the cutting pattern are taken into account, meaning that, while this time is significant for comparison, it is smaller than the time required for a full optimization. The CNN approach achieves 98.9% of the baseline value with almost the same computation time (which excludes the time for network inference, which is not counted here but evaluated in Section 4.4.2). The R-Maps approach achieves only 92.5% of the baseline value, with an even longer time despite a reduced number of shifts (the value related to the pattern centering shifts tried during the optimization, as explained in Section 5.2.4).

4.4.2 Timing verification

Table 4.3 reports on the inference times of CNN networks. The first column contains the sections considered; the second column contains the number of channels of the input image for the CNN; the third column reports the type of D-Maps used as input; in case of knot D-Maps, the diameter of the knots used in calculation is added to the name. D-Knot refers to the dead knot D-Map, HW to the heartwood D-Map, as specified in Section 4.3.1. The last column reports the inference time, estimated using an Nvidia RTX-A5000 GPU card. Reported times refer to the average for a single inference in condition where multiple instances are launched in parallel to take full advantage of the GPU computing power. For sections where the number of channel in input is the same (e.g. 25x95 and 25x180), the inference time is the same as long as the structure and dimension of the trained CNN network is exactly the same.

Section t x w (mm)	**N. of channel**	**Type of input D-Maps**	**Inference time(ms)**
45 x 100	63	Shape, Knot-10, D-Knot-10	6
25 x 95-180	84	Shape, Knot-30-50, D-Knot-30	7
37 x 100	84	Shape, Knot-50, D-Knot-30, HW	7
40 x 100-200	105	Shape, Knot-30-50, D-Knot-30, Pith	9

Table 4.3: Inference time for the different CNNs trained for different sections

We remark that the computation time does not depend on the type of input, but only on the number of D-Maps considered (number of channels). In our experiments, we have sections that need from 3 to 5 types of D-Maps, each type of D-Map is used 21 times in the input to cover the length of the log (4 meters by 200 mm steps). Therefore, considering the requirements of a typical optimization setting, similarly to Equation 4.18, setting $N_T = 14$

sections, $N_R = 36$ rotations, $N_{sx} = 5$ and $N_{sz} = 5$ slopes, and considering an average inference time $T_{INF} = 7ms$, we obtain an estimate of the total inference time needed for optimization equal to:

$$T_{TOT} = T_{INF} N_T N_R N_{sx} N_{sz} \simeq 90 \text{ seconds} \tag{4.19}$$

The estimated time, while significant, is in the same order of magnitude as the reasonable time for log optimization, considering that we can easily parallelize inference computations on multiple GPUs and different log optimizations on multiple computers.

In fact, the reasonable average time for the optimization of a standard 4 meter log in the sawmill, considering a line with a feed rate of 100 m/min and a distance between logs of about 1 meter, is about 3 seconds, a factor of 30 over the estimated time for inference. It is reasonable to think that a rack of 15 industrial PCs (a configuration already installed by Microtec in several plants just for optimization with tomographic scanners), equipped with 2 GPUs equivalent to the A5000 tested, can perform flexible optimization with the approach based on V-Maps and CNN networks. The GPU is dedicated to the inference of the CNN networks only, since the rest of the optimization algorithm is currently planned to run in the CPU. In this calculation, each PC has an average of 45 seconds of CPU time available for each log to run the optimization algorithms based on the computed V-Maps, with the opportunity to parallelize on internal CPU cores.

Chapter 5

Seed-constrained optimization

The solution of the FLO problem, defined in Section 3.1, consists in the construction of the maximum value cutting pattern based on the valorization of the contained boards, as defined in Chapter 4. The proposed approach involves the decomposition of the problem into two decision levels, which corresponds to a division of the cutting pattern's variables into two parts. This chapter identifies the first-level cutting pattern variables that lead to the definition of the seed, and describes a procedure to optimally complete the cutting pattern from the seed, i.e., the procedure that, having fixed the first-level variables (the seed), defines the second-level variables that provide the highest overall cutting pattern value. The proposed procedure corresponds to the second phase of the solution approach sketched in Section 3.2 for FLO (seed-constrained optimization). It is based on dynamic programming algorithms and is inspired by similar optimization procedures in the literature. In particular, we will see how this approach can be adapted for filling the main cuts defined by seed with stacked boards, taking into account the specific features of the FLO problem and the integration with the procedures that optimize the remaining second-level decision variables. An experimental verification of the performance of the proposed algorithm for seed-constrained optimization is presented, with the aim of calibrating its parameters towards the best trade-off between computational load and performance. The tuning of this algorithm is indeed crucial, since it is called several times in different positions by a second, higher-level algorithm.

5.1 Seed definition

The choice of what to include in the seed is the basis of the proposed approach to the FLO problem. The goal is to divide the optimization into two levels. At the innermost level, we want to keep the possibility of using very efficient algorithms for filling cuts with boards, as we will see in this chapter, and leave the other decision variables related to the cutting pattern to a higher level of optimization. The choice of the first-level variables stems from the availability of dynamic-programming procedures that deals with the filling of a single cut with boards [16], assuming a fixed position for the cut itself. The variables that we consider at the outermost decision level refer to a global positioning of the cutting pattern, in particular, they are:

- the horizontal shift, calculated with respect to reference centering of the cutting pattern. We assume that, for a null horizontal shift, the cutting pattern is horizontally centered with respect to the total width of the vertical and main cuts;
- the rotation, which is common to all cuts and all boards;
- the horizontal slope, common to all vertical and main cuts, possibly different only for the horizontal cuts (in this case, it will be optimized during the seed-constrained optimization);
- the vertical slope, common to all main and horizontal cuts, possibly different only for the vertical cuts (in this case, it will be optimized during the seed-constrained optimization).

From an optimization perspective, the choice of the first-level variables aims at defining a suitable optimization domain, where the search methods used to explore the seed space (as we will detail in Chapter 6) are able to identify promising search directions after perturbation of the seed variables. Notice that any change in the above listed variables changes the position of the cuts in the log and requires, in principle, a complete reoptimization. In fact, the values of the contained boards depend on their position and they are all potentially affected.

Definition 5.1.1 (seed)**.** We define a seed as the tuple

$$(N^{VL}, N^{M}, N^{VR}, w_1^L \dots w_{N^{VL}}^L, w_1^M \dots w_{N^M}^M, w_1^R \dots w_{N^{VR}}^R, c_x, h_x, h_z, \alpha) \quad (5.1)$$

where N^{VL}, N^M, N^{VR} are the number of vertical and main cuts of the pattern, w_i^L, w_i^M, w_i^R are the widths of each cut i in the related sets, c_x is a shift, in x direction, of the whole pattern with respect to a common centering of the cuts, h_x is the common slope, in the x direction, of the vertical and main cuts, h_z the slope, in the z direction, of the main cuts, α the rotation of the whole pattern.

Let x_0 be the position of the first cut of the pattern. It corresponds to x_1^V, position of the first vertical cut (see Definition 3.1.11) if $N^{VL} > 0$, otherwise to x_1^M, position of the first main cut (see Definition 3.1.10), always present in the pattern. The position x_0 is then determined by:

$$x_0 = -(\sum_{i=1}^{N^{VL}} w_i^L + \sum_{i=1}^{N^M} w_i^M + \sum_{i=1}^{N^{VR}} w_i^R + s(N^{VL} + N^{VR} + N^{VR} - 1))/2 + c_x \quad (5.2)$$

In other words, it is assumed that, if $c_x = 0$, the start of the first cut and the end of the last cut are symmetrical with respect to the center of the reference system, which is also assumed to be a central position with respect to the log to be optimized. Although not required by the cutting pattern definitions, it is assumed that the log model is properly centered before optimization and that the log axis is aligned with respect to the y axis of the reference system. Under this assumption, we minimize the size of all maps defined in Chapter 4, since the maps are centered in the origin by construction, according to Definitions 4.1.2 and 4.2.1. The term $s(N^{VL} + N^{VR} + N^{VR} - 1)$ takes into account the saw blades between all the cuts.

5.1.1 Empirical seed symmetry

All the proposed optimization procedures benefit from the following observation. As a general heuristic, and following current practice, we restrict the rotation to angles between 0° and 180° for the definition of the seed. The seed defined by Equation 5.1 for the rotation α is similar to the following seed defined for the rotation $\alpha + 180°$:

$$(N^{VR}, N^{M}, N^{VL}, w^{R}_{N^{VR}} \ldots w^{R}_{1}, w^{M}_{N^{M}} \ldots w^{M}_{1}, w^{L}_{N^{VL}} \ldots w^{L}_{1}, -c_x, -h_x, -h_z, \alpha{+}\pi) \tag{5.3}$$

where the order of the cuts is reversed, and slopes and centering have opposite sign. The restriction comes from the fact that current practice in cutting pattern construction often considers these two seeds as “equivalent”, although their valorization may be different. However, the empirical assumption is justified by pattern construction procedures, that tend to guarantee a certain degree of symmetry of the pattern with respect to the y axis. As we will see in the following sections, the procedures we devise for seed-constrained optimization and for seed selection also stick to this practice, and provide almost symmetric patterns in the y direction. As a consequence, it is not worth to evaluate two seeds obtained with a rotation of 180° from each other, since they are (almost) symmetric and, hence, provide (almost) the same value. This will enable exploring seeds in other degrees of freedom.

We remark that, once the seed is fixed, the problem of optimizing the remaining decision variables of the cutting pattern, i.e., the seed-constrained optimization problem, can be decomposed into two independent subproblems, each dealing with, respectively, the main and the vertical cuts, as defined in Section 3.1.4. The proposed algorithms are explained in detail in the following sections.

5.2 Main cuts optimization

The optimization of the main cuts defines the main boards obtained from that main log block. It includes the definition of the main horizontal cuts and, usually, it require the most relevant computational effort. The problem has, at first sight, strong affinity with the one presented in [16], where the choice of the best sequence of the main cuts is made in two steps, each based on dynamic programming, with a two-levels logic. The external level works in the horizontal direction, the inner one deals with the identification of the optimal solution of each cut in the vertical direction. However, the main cuts optimization problem we have to solve has some peculiarities that prevent the application of such a nested method. The main reason is the presence of multiple parallel main cuts enclosed between two sets of horizontal cuts. We thus propose a different approach, where a single-level dynamic programming algorithm (inspired by [20]) is executed multiple times to solve a selection of optimization subproblems. In particular, a higher-level heuristic samples, for each of the main cuts, multiple starting positions, and then extracts the best overall combination of these starting positions based on the evaluations made in the suboptimizations.

The details of the approach will be explained in the following subsections. In general, the main cuts optimization procedure we propose combines some of the components presented in [16] for simplest problem versions, which can still find application in some simpler instances and, more in general, defines the building blocks of our overall procedure. In the following subsections, we analyze such blocks, gradually introducing the required levels of complexity.

5.2.1 Knapsack-like formulation for single cut board stack optimization

Let T be the total thickness of the vertical cut we want to fill, t_i and v_i, respectively, the thickness and the value of each board $i = 1 \dots n$, in a given set of n boards, s the thickness of the saw blade.

The problem can be formalized as the following integer linear programming model:

$$f(T+s) = \max \sum_{i=1}^{n} v_i x_i \tag{5.4}$$

subject to

$$\sum_{i=1}^{n} (t_i + s) x_i \leq T + s, \tag{5.5}$$

$$x_i \in \mathbb{Z}^+, i = 1 \dots n \tag{5.6}$$

The variable x_i defines how many units of each board are included in the total thickness.

This problem is a classic knapsack problem and, as it is well known, it can be solved by dynamic programming, i.e., by recursively computing the function f with a discretization step $d_T < T$, where d_T is a divisor of T, s and t_i, for every $i = 1 \dots n$. In particular:

$$f(0) = 0 \tag{5.7}$$

$$f(k) = \max\{f(k-1), v_i + f(k - t_i/d_T)\},\ i = 1 \dots n \tag{5.8}$$

where $k = 1 \dots (T+s)/d_T$

However, this approach is not directly applicable to our problem for two main reasons:

- we need to know the position of the inserted boards, not just their number; in fact, the value of the boards is not a given constant (as for v_i in Equation 5.4), since it depends on the board position in the log;
- we cannot guarantee that there is a common divisor for all boards and saw blade thicknesses. Consequently, unless we introduce approximations whose impact on the objective function could be significant, the step we should use would be equal to the basic resolution of the cutting system (under typical conditions for systems of interest in our study equal to 0.1 mm, see Section 2.1), which may lead to an impractical computational load.

5.2.2 Stack boards optimization with positional value

The solution approach presented in Section 5.2.1 can be extended to consider the positional value of the boards, leading to Algorithm 1. In the algorithm, N^B is the number of boards, $t[i]$ the thickness of board i, $V[i][x]$ the value of board i at position x, $F[x]$ the value of function $f(x)$. Since we cannot rely on a common divisor in our problem, we introduce an approximation by ignoring the step d_T, and we check the function f for all arguments x between 0 and T by a uniform discretization step nominally equal to 1. In other words, we compute the function f by steps of 1 unit of length, x being expressed in the same unit. Notice that no further approximation is introduced if the unit used to express sizes corresponds to the precision of the cutting machinery, normally 0.1 mm. In fact, in this case, all parameters related to size can be assumed to be integer valued, and variables x restricted to the integer domain. Because of the positional value of the boards, we will have to construct "continuous" sequences of boards, where boards are adjacent and separated by the saw blade thickness s.

Algorithm 1 Stack boards with positional value

```
for $x = 0, \ldots, T + s$ do                          ▷ check all position with step 1
    $F[x] \leftarrow 0$
    for $i = 1, \ldots, N^B$ do                        ▷ check all possible boards
        $x0 \leftarrow x - t[i] - s$
        if $x0 \geq 0$ and $V[i][x0] > 0$ and $V[i][x0] + F[x0] \geq F[x]$ then
            $F[x] \leftarrow V[i][x0] + F[x0]$
```

At the end of the algorithm, the vector F contains the value of the optimal sequence of boards starting at each discretized position x. Notice that Algorithm 1 may determine different sequences of boards starting at different positions. Moreover, not necessarily a sequence of boards related to a larger position x leads to a larger total value than a solution relative to a smaller position x. As an illustrative example, Figure 5.1 shows a possible outcome of the algorithm. Let us assume $s = 1$ and that the only validated boards during the algorithm execution are the ones in the figure, with the shown positions and values. There may be values of x for which the function F is 0, since there is no sequence starting at that position, e.g., for $x = 10$. Moreover, the sequence starting at $x = 15$ has a higher value than the sequence starting at $x = 18$.

For the sake of simplicity, the algorithm as written does not keep track of the position x of the best solution, nor of the positions and thicknesses of the placed boards, but with minor modifications it can be adapted for this purpose.

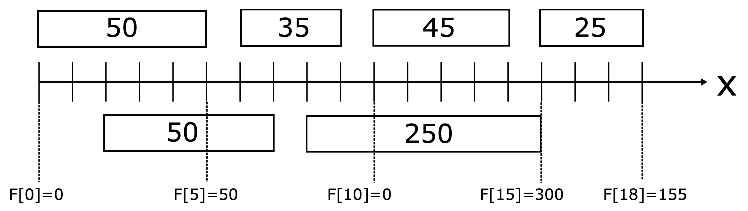


Figure 5.1: Examples of boards sequences for Algorithm 1, and corresponding values of F

Algorithm 1 can be effectively used for solving the main cuts optimization problem when we have single main cuts, i.e., according to the notation of the cutting pattern Definition 3.1.13, main cuts where $N^{HT} = 0$ and $N^{HB} = 0$, which denotes the absence of horizontal cuts. In this case, the optimization of each main cut can be performed independently.

5.2.3 Stack boards optimization with positional value and fixed start

In the condition where $N^{HT} \geq 1$ and $N^{HB} \geq 1$, Algorithm 1 is not directly applicable, since the optimization of each main cut is not independent. As a naive approach, one could check in parallel the partial solutions generated in all main cuts as the position x in the algorithm progresses, but these partial sequences should all somehow be aligned or compatible with the same horizontal cut.

We will then introduce a variant of Algorithm 1 that is able to address a slightly different problem: this new algorithm is general enough and will later be used within the overall algorithm we will propose in Section 5.2.4.

The problem we want to solve is to find the optimal sequence of a maximum number N^S of stacked boards, covering a total thickness of dimension T, whose value is determined as in the problem in Section 5.2.2, but with the constraint that the first board in the sequence is at position 0. The formulation of the problem is as follows.

Again, let T be the total thickness of the vertical cut we want to fill, t_i and v_i, respectively, the thickness and the value of each board $i = 1 \dots N^B$, in a given set of N^B elements, s the thickness of the saw blade. We assume that, for the used unit of measure, $t_i \in \mathbb{Z}^+$. Let $f(i, x)$ be the function that returns the value of the board i at position x, assuming that a board of thickness t positioned in x covers the interval $[x, x + t[$. Let T be the total thickness of a vertical cut we want to fill, N^S the maximum number of stacked boards. We want to find: the length of the sequence of stacked boards $n \leq N^S$; the position x_j of the jth board in the sequence; the board i_j that appears in position j of the sequence, with $j = 1 \dots n$, maximizing the total value of the stacked boards. For explanatory purposes only, the problem can be specified as follows:

$$V(T) = \sum_{j=1}^{n} f(i_j, x_j) \tag{5.9}$$

subject to:

$$x_1 = 0, \tag{5.10}$$

$$x_j = x_{j-1} + t_{i_{j-1}} + s, \ j = 2 \dots n, \tag{5.11}$$

$$x_n + t_{i_n} \leq T, \tag{5.12}$$

The first constraint fixes the positioning of the first board at position 0, the second constraint imposes the continuous sequence of the boards with spacing equal to the saw blade thickness s, and the third constraint guarantees that the whole sequence is completely contained within the total thickness T. We remark that the above formulation is not a mathematical program, because the indexes n, i_n and i_{j-1} are related to decision variables. The formulation can be easily linearized using, for each board and position, a binary variable taking value 1 if the board is in that position, 0 otherwise, and modifying the constraints to take a linear form (which is out of the scope of this thesis).

The problem can be solved trivially by exhaustively exploring all arrangements of N^B boards for N^S cuts. The complexity of such an algorithm is $O({N_B}^{N_S})$. Of course, as the number of available cuts increases, this exhaustive algorithm immediately becomes impractical.

We thus propose a solution approach inspired by Algorithm 1. The algorithm constructs various board sequences and searches for the highest valued solution from a thickness t, but now all sequences are constrained to start exactly at position 0. The algorithm divides the thickness interval $[0, T]$, over which the board sequences are searched, into intervals based on a step δ_x with these specifications:

$$\delta_x > 0 \tag{5.13}$$

$$\delta_x < \min_{1 \leq i \leq N^B} t_i \tag{5.14}$$

Condition 5.14 requires that δ_x is less than the smallest board thickness, to ensure that a board cannot be completely contained in the same interval, otherwise it would not even be possible to fit it into a sequence.

Assuming the all parameters related to size are integer-valued (as observed in Section 5.2.2), the algorithm is exact for $\delta_x = 1$, while for values of $\delta_x > 1$ the algorithm loses the guarantee of optimality with a gain in execution speed. It will therefore be necessary to evaluate a trade-off between execution time and loss of value, as we will see in Section 5.6, devoted to computational experiments.

The idea behind the algorithm is to compute only one value of the function V (see Equation 5.9) for each of the intervals generated by the step δ_x. In fact, the algorithm computes multiple versions of this function, each referring to a specific number of boards contained in the sequence for which each function is computed.

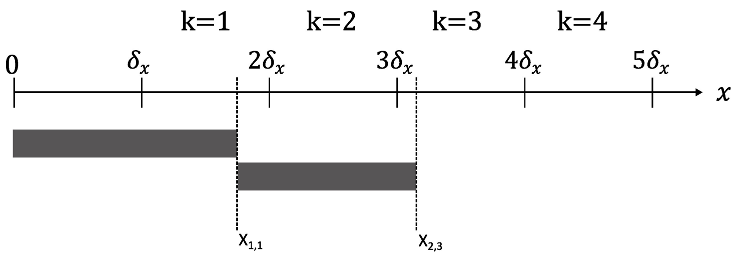


Figure 5.2: Exact position x of the board in the sequence with respect to the interval division

In addition to the value of the function, each time the algorithm inserts a board into a sequence, it also updates the exact position the board reaches in that interval, meaning that the position of the board is not subject to the same discretization of the function V.

To clarify this concept, consider Figure 5.2. The variable $X_{2,3}$ refers to the exact position of the best sequence of exactly two boards, ending in the interval with $k = 3$. The value of $X_{2,3}$ will therefore be within this interval, i.e. greater than or equal to $3\delta_x$ and strictly less than $4\delta_x$. We notice that, while the exact position can be any value in the interval, the gain in speed comes from the fact that proposed algorithm deals with a finite (and fixed) number of function values, one per discretization interval, which speeds up the computation.

We are now ready to describe Algorithm 2, that we propose for the main cuts optimization. In the algorithm, $V^B_{i,x}$ refers to the value of the board $i = 1...N^B$ in position x. $V_{c,k}$, $X_{c,k}$ $I_{c,k}$ are, respectively, the value, the end position and the last board index of a sequence of c boards that ends in the interval $k = [k\delta_x, (k+1)\delta_x[$.

The outermost loop of Algorithm 2 defines the number of boards involved in the sequence. In the first iteration, the values of the function $V_{1,k}$ are assigned, corresponding to the values of a sequence consisting of a single board positioned exactly at 0. The position of the end of the board must be in the interval k, namely in $[k, (k+1)\delta_x[$. The exact value of the position at which the board ends, and from which a second board can be added at the

Algorithm 2 Stack boards with positional value and fixed start

```
for $c = 1, \ldots, N^S$ do
    for $k = 1, \ldots, N^T$ do
        $V_{c,k} \leftarrow 0$
        $X_{c,k} \leftarrow 0$
        $I_{c,k} \leftarrow 0$
    for $i = 1, \ldots, N^B$ do
        if $c = 1$ then
            if $k\delta_x \leq t_i < (k+1)\delta_x$ and $V^B_{i,0} \geq V_{c,k}$ then
                $V_{c,k} \leftarrow V^B_{i,0}$
                $X_{c,k} \leftarrow t_i$
                $I_{c,k} \leftarrow i$
        else
            for $k_{ADD} \in \{0, 1\}$ do
                $k_0 \leftarrow \lfloor (k\delta_x - t_i - s)/\delta_x \rfloor + k_{ADD}$
                $x_0 \leftarrow X_{c-1,k_0} + s$
                if $(k\delta_x \leq x_0 + t_i < (k+1)\delta_x)$ and $V_{c-1,k_0} > 0$
                                and $V^B_{i,x_0} + V_{c-1,k_0} > V_{c,k}$ then
                    $V_{c,k} \leftarrow V^B_{i,x_0} + V_{c-1,k_0}$
                    $X_{c,k} \leftarrow x_0 + t_i$
                    $I_{c,k} \leftarrow i$
```

next iteration of c, is stored in $X_{c,k}$. If the value $V_{1,k}$ is zero, it means that there is no board that starts at 0 and ends in interval k.

For example, suppose we are in the innermost loop at iteration c, k, i, with $c > 1$. The algorithm needs to update the value $V_{c,k}$. It then evaluates whether the board of size t_i can join a sequence of $c-1$ boards (evaluated at the previous iteration of c) and ends up in the interval k obtaining a larger value than previously found. To do this, it must determine the only two possible intervals k_0 and k_1 in which to search for the end of the sequence to which it should append the board of size t_i so that it ends in the interval $[k, (k+1)\delta_x[$. These two intervals are given by $k_0 = \lfloor (k\delta_x - t_i - s)/\delta_x \rfloor$ and $k_1 = k_0 + 1$, the next interval. For each of the two intervals, the algorithm determines if there is a sequence of boards that falls in that interval, and stores in x_0 the position X_{c-1,k_0} (or X_{c-1,k_1}, depending on the interval). If, adding the thickness t_i of the current board to the sequence, the position $x_0 + s + t_i$ is in the interval relative to k, then it is checked whether the new value obtained from $V^B_{i,x_0} + V_{c-1,k_0}$ (or $V^B_{i,x_0} + V_{c-1,k_1}$) is higher than the current one, and, in this case, all variables $V_{c,k}, X_{c,k}, I_{c,k}$ are updated.

The complexity of this algorithm is determined by the three nested cycles, related to the number of boards N^B, the maximum length of the sequence of boards N^S, and the number $N^T = T/d_T$ of intervals into which the total thickness T to be optimized was divided. We then obtain $O(N^B\ N^S\ N^T)$, which is much more convenient than the exhaustive algorithm as the length of the sequence increases.

5.2.4 Overall algorithm for main cuts optimization

The algorithm we propose for the overall optimization of the main cut sequence, in the presence of horizontal cuts, is illustrated in Figure 5.3, and more formally reported in Algorithm 3.

In the first step, the algorithm selects, for each main cut, several possible center split positions based on a predetermined step. For each of these split positions, on each main cut, the Algorithm 2 is applied in the two opposite

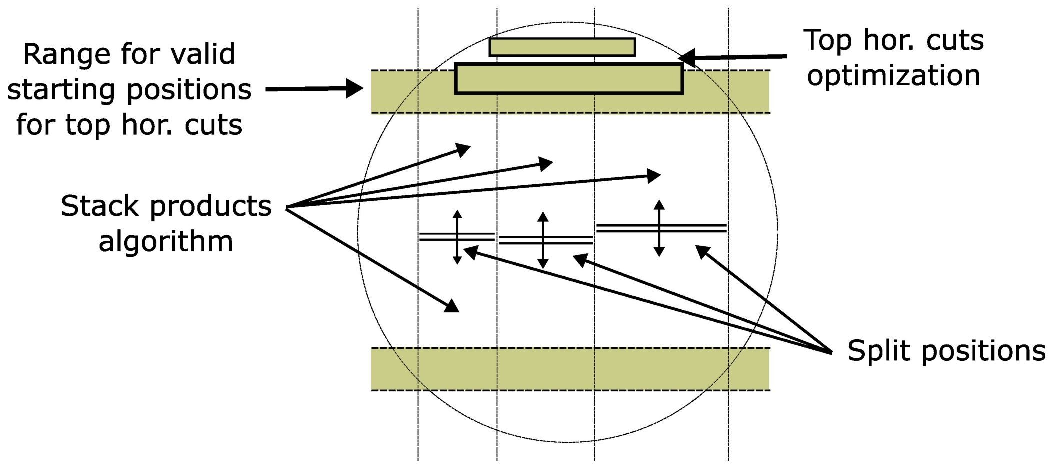

Range for valid starting positions for top hor. cuts
Top hor. cuts optimization
Stack products algorithm
Split positions

Figure 5.3: Overall approach to main cuts optimization

directions. In this way, Algorithm 3 provides, for each main cut, an indication of the possible sequences that, starting from the outside of the log, join exactly at the split position under consideration.

In the second step, Algorithm 3 exhaustively explores the split combinations of the main cuts. For each of these combinations of split positions, it checks the starting position of the horizontal cuts that leads to a larger total value. The search for this starting position is done from the feasible positions of the board sequences found by Algorithm 2, within a heuristically determined range based on the log size. The optimization of horizontal cuts will be described in Section 5.4.

The complexity of Step 1 of Algorithm 3 depends on the number H of split positions to be evaluated for each main cut. The complexity will then simply the one of Algorithm 2 multiplied by this factor.

The complexity of Step 2 is related to the number of split combinations, denoted by H^M, where M is the number of main cuts, and the number of horizontal cut optimizations (as we will detail in Section 5.4) performed for each of them. The number of starting positions for the horizontal cut optimizations depends on the range considered and is reduced by the choice of step δ_x of Algorithm 2, since there will be at most one starting position to consider for each step interval δ_x (for each main cut and for each length

Algorithm 3 Main cuts optimization

```
Step 1: main blocks optimization
for each main cut $C$ do
    for each split position $S$ do
        Using Algorithm 2:
        calculate $V^{C,S,T}$,$X^{C,S,T}$, from top to the split position
        calculate $V^{C,S,B}$,$X^{C,S,B}$, from bottom to the split position

Step 2: extract best solution
for each split combination do
    for each main cut $C$ do
        store valid top start positions for hor. cuts, based on $V^{C,S,T}$,$X^{C,S,T}$
        store valid bottom start pos. for hor. cuts, based on $V^{C,S,B}$,$X^{C,S,B}$
    for each potential top start position for horizontal cuts do
        optimize top horizontal cuts from top start position
        for each main cut $C$ do
            get best value below the top start position from $V^{C,S,T}$,$X^{C,S,T}$
        update the best solution over the splits
    for each potential bottom start position for horizontal cuts do
        optimize bottom horizontal cuts from bottom start position
        for each main cut $C$ do
            get best value over the bottom start position from $V^{C,S,B}$,$X^{C,S,B}$
        update the best solution under the splits
    update the best solution for the main cuts
recover the global best solution for the main cuts
```

of board sequence).

This second step of the algorithm becomes computationally dominant as the number of evaluated split positions increases, especially for seeds where there are many main cuts to consider.

The choice of this approach, based on sequences constrained to a central cutting position, has other significant advantages that are not explicitly derived from the definition of the cutting pattern as from Definition 3.1.13, but are related to operational specifications that sometimes need to be considered depending on the specific cutting environment. These specifications are not detailed here, however, the implementation of Algorithms 2 and 3 can be easily adapted to take them into account. In particular:

- in certain cases, the tool performing the cut is set up with blades of significantly different thicknesses. By starting the dynamic programming algorithm from the center position, the number of the blade that will perform the cut during the dynamic programming algorithm can be uniquely determined. It is then sufficient to consider the actual blade thickness in Algorithm 2.

- in some cutting systems, the outermost blades are in a fixed relative position to the innermost blades. This limits the choice of product thickness that can be used when cutting with these blades. Similarly to the previous point, knowing which blade is doing the cutting allows controlling the selection of valid boards during the iteration of Algorithm 2, by excluding boards in the innermost cycle if not compatible with the current saw blade index of the outermost cycle.

- for some optimization settings, it may be necessary to always make a centered cut within the main cuts, or to start from a pre-configured centered board. These requirements are related to a standard way of cutting logs, dictated by experience, that must be maintained within the optimization, and can be obtained by applying, in Step 1 of Algorithm 3, if a centered board is required, a different split position for top e bottom direction, taking into account the presence of this board.

5.3 Vertical cut optimization

For vertical cuts, the boards are rotated by 90°, i.e. their thickness is equal to the width of the cut. The optimization of the vertical cut is, in general, less complex than the optimization of the main cutting block, so its impact on the overall required optimization computational resources is negligible. In most real-world applications, there are one or two boards side by side in each cut. The degrees of freedom for this optimization are therefore reduced to the selection of these boards and their vertical position in the cut, with the ability to skew them by an angle independent of the main block. A detailed discussion of this phase of the optimization is not of particular interest in this research. For the specific application, we simplify the handling of vertical cuts by limiting them to one board per cut and keeping the skew the same as the main block, thus eliminating the need to compute specific V-Maps. Under these assumptions, a simple heuristic for cut optimization on the basis of value maps is proposed:

- all board sections with thickness equal to the width of the cut are selected;
- V-Maps are calculated for these sections, with angle and slope compatible with the current seed;
- for each section, find the point of maximum in the z direction in the specific interpolated V-Map at the central x position of the cut;
- the maximum value found determines the winning section.

5.4 Horizontal cuts optimization

In the course of optimizing the main cuts, it will also be necessary to repeatedly optimize sequences of horizontal cuts, (see Definition 3.1.12), to complete the top and bottom parts of the pattern. The innermost cut in each of these sequences, as expressed by constraints (3.43) and (3.41), will have a z^H position that depends on the highest and lowest positioned product, relative to all products in the main cuts. As with vertical cuts, we simplify the treatment of horizontal cuts by limiting them to one board per cut and a slope h_x equal to that of the main cuts. The current heuristic takes a greedy approach to completing the horizontal cuts. The second cut, if any, is chosen after the previous cut has been determined. For each cut, every thickness-compatible section is tried, and several positions are tried with a fixed step approach with respect to the position centered on the main cut block. More advanced optimization approaches are certainly possible and will be evaluated in the future.

5.5 Alternative seed definition

In this section, we present an alternative definition of the seed, compared to the one provided in Section 5.1. This definition aims to reduce the computational bargain of optimizing a single seed by moving some degrees of freedom from seed constrained optimization (Phase 2 of the overall optimization approach) to seed selection (Phase 3). Specifically, the alternative seed definition, referred to as *split-seed*, is larger than the previous one, and also includes the positions of the splits for the central cuts.

Definition 5.5.1 (split-seed)**.** We define a split-seed as the tuple

$$(N^{VL}, N^M, N^{VR}, w_1^L \dots w_{N^{VL}}^L, w_1^M \dots w_{N^M}^M, w_1^R \dots w_{N^{VR}}^R, c_x, h_x, h_z, \alpha, S_1 \dots S_{N^M}) \tag{5.15}$$

where the first elements are exactly the same as in Definition 5.1, and $S_1 \dots S_{N^M} \in \mathbb{R}$ correspond to the split position of each main cut as described in Section 5.2.4.

Using this definition, the optimization of the main cuts no longer needs to be evaluated for all possible combinations of splits, thus potentially reducing the complexity of the seed-constrained optimization phase. Algorithm 4, which replaces Algorithm 3 in the case of this different seed definition, simplifies the optimization of the main cuts, as it is now constrained by a specific combination of split positions. We remark that, at step 2, it is still necessary to evaluate the possible starting positions for the horizontal cuts related to the split positions.

However, as will be seen in Section 6.5.5, we anticipate that using this new seed definition seems not to yield any significant advantage in terms of overall optimization performance. Therefore, the recommended approach in this thesis remains the seed definition provided in Section 5.1.

5.6 Numerical results

An analysis of the impact of the choice of the discretization step δ_x of Algorithm 2 is reported in this section. In fact, we want to determine the trade-off between computational speed and the loss of value caused by the suboptimality of Algorithm 2.

The following three different ranges of board thicknesses have been taken into consideration for our experiments:

- scenario with thin boards: thicknesses between 17.0 mm and 40.0 mm;
- scenario with thick boards: thicknesses between 30.0 mm and 60.0 mm;
- scenario with mixed boards: thicknesses between 17.0 mm and 60.0 mm;

These three thickness ranges were used to test whether the δ_x step is more critical with thicknesses closer to each other or with smaller average board thicknesses. Five different values were also evaluated for the number of boards from which the algorithm could choose, varying from 3 to 7, a reasonable range in practice. Finally, a total of 7 possible values were considered for the δ_x parameter, ranging from 0.1 mm (corresponding to the value $\delta_x = 1$, which is the resolution of the system) to 17 mm (corresponding to $\delta_x = 170$),

Algorithm 4 Main cuts optimization with split-seed

```
Step 1: main blocks optimization (single split)
for each main cut $C$ do
    Using Algorithm 2:
    calculate $V^{C,S,T}$,$X^{C,S,T}$, from top to the split position
    calculate $V^{C,S,B}$,$X^{C,S,B}$, from bottom to the split position

Step 2: extract best solution
for each main cut $C$ do
    store valid top start positions for hor. cuts, based on $V^{C,S,T}$,$X^{C,S,T}$
    store valid bottom start pos. for hor. cuts, based on $V^{C,S,B}$,$X^{C,S,B}$
for each potential top start position for horizontal cuts do
    optimize top horizontal cuts from top start position
    for each main cut $C$ do
        get best value below the top start position from $V^{C,S,T}$,$X^{C,S,T}$
    update the best solution over the splits
for each potential bottom start position for horizontal cuts do
    optimize bottom horizontal cuts from bottom start position
    for each main cut $C$ do
        get best value over the bottom start position from $V^{C,S,B}$,$X^{C,S,B}$
    update the best solution under the splits
```

in order to respect the constraint in Equation 5.14. A total of 105 scenarios were then considered from the combination of all these parameters. For each scenario, 1000 simulations were performed, varying the thicknesses involved and the V-Maps used.

Each simulation consists of an optimization of a single main cut, centered with respect to x, on a randomly chosen log in the set considered in Section 4.4; the optimization is based on Algorithm 2, randomly choosing an angle for the cut and the split position within the cut, and running Algorithm 3.

During each simulation run, the number of boards required by the scenario are generated with thicknesses within the required range, but with a minimum difference between the generated thicknesses of at least 2 mm. Each generated board is associated with a V-Map by randomly selecting one from those generated for the experiments reported in Section 4.4, and adjusting it for thickness so that the values are consistent across all boards.

For each simulation with $\delta_x = 1$, because of the significant required computational resources, only two evaluations of the algorithm are made from a central split position. The split position and map rotation are chosen randomly for each run. The returned value is given by the sum of the maximum values obtained by the algorithm in both directions (see Figure 5.4).

For each simulation with $\delta_x > 1$, additional evaluations are performed, each time randomly changing the split position and rotation. The maximum value obtained is taken from all these evaluations. This simulates the possibility of performing more trials by having more time available (as it happens during the overall log optimization, thanks to the seed selection phase). The number of evaluations made for each run is exactly equal to δ_x, the computation time of the algorithm being inversely proportional to this parameter. For this reason, we may expect better results from larger discretization steps, which may be counterintuitive.

The simulation results are shown in Figures 5.5 and 5.6. The graph in Figure 5.5 shows the trend of the average value obtained in the three thickness scenarios as the value of δ_x increases. The lower average value of the scenario

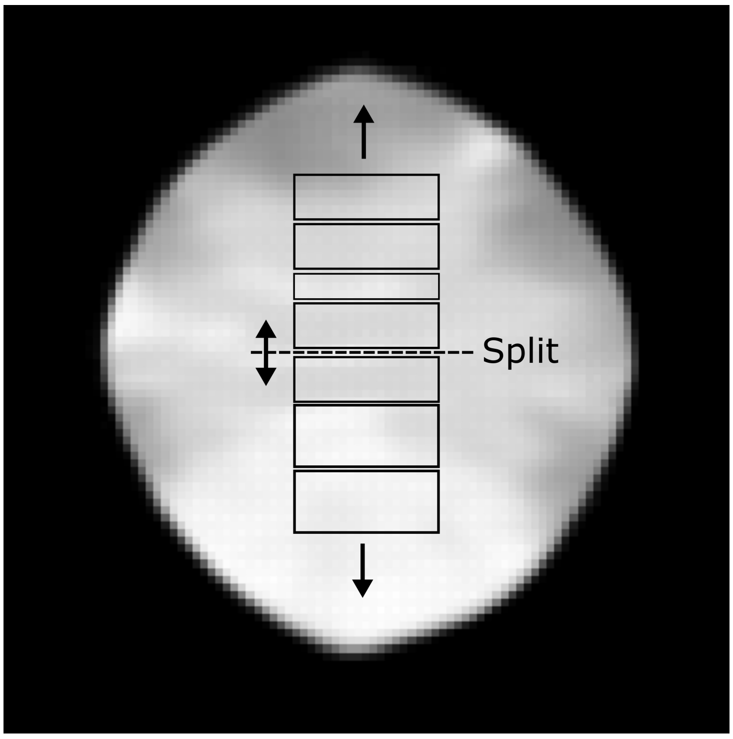


Figure 5.4: Global algorithm for main cuts optimization

with smaller thicknesses can be attributed to a lower possibility of placement on the value map, especially for scenarios where the number of boards is smaller. The graph in Figure 5.6 shows the increase in value compared to the simulation with $\delta_x = 1$. It is clear that there is an initial significant increase in mean value of more than 5% by increasing δ_x. This advantage then decreases for larger values. The initial increase in value is intuitively justified by the fact that the impact of discretization intervals is low, as long as δ_x is sufficiently small, especially for short board sequences. For example, the information of the optimal sequence of length 1 is preserved as long as $\delta_x < d$, where d is the maximum difference between the thicknesses of the boards. As the length of the sequence increases, the probability that two sequences of the same length will end up on the same intervals, and thus be lost, increases. In our experiment, the maximum length of the sequence of boards is limited to 7, a number that is already significant with respect to practical uses. The decrease in value for larger values of δ_x is intuitively due to the fact that it is more and more likely that the sequences of the same length end up in the same intervals, since the differences in thickness between the boards are more likely to be smaller than δ_x. Moreover, the number of discarded sequences increases, thus vanishing the advantage of a larger number of sampled evaluations coming from the increasing δ_x. A good choice of δ_x, based on these simulations, would therefore seem to be between 5 and 10 mm (corresponding to a value between 50 and 100 referring to the resolution used).

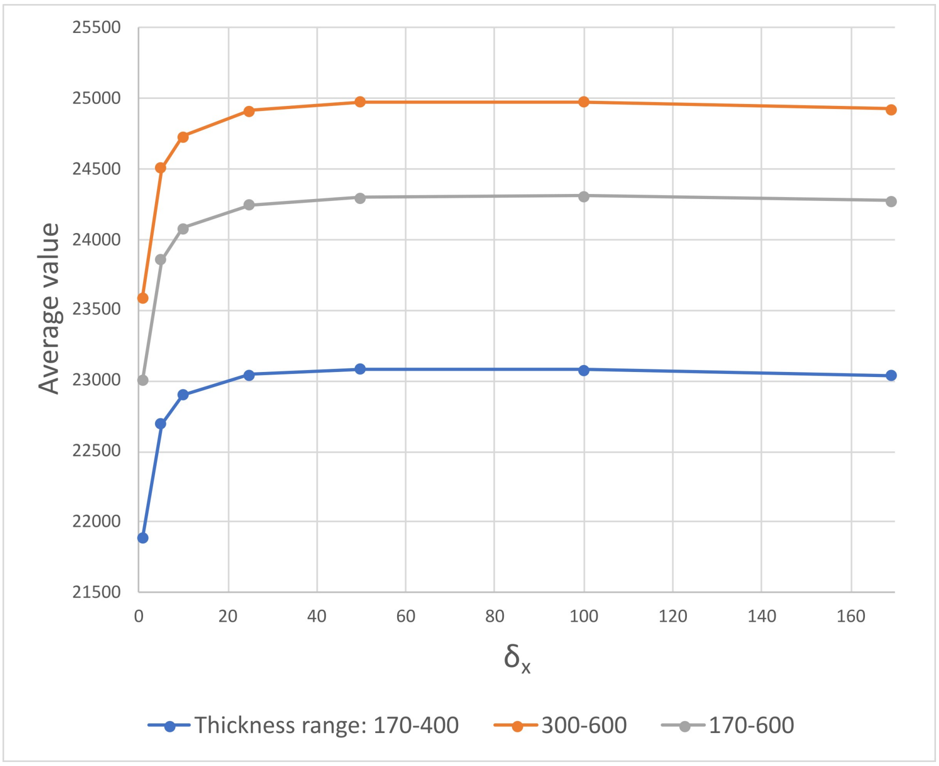


Figure 5.5: Optimization value in function as a function of δ_x: Total values

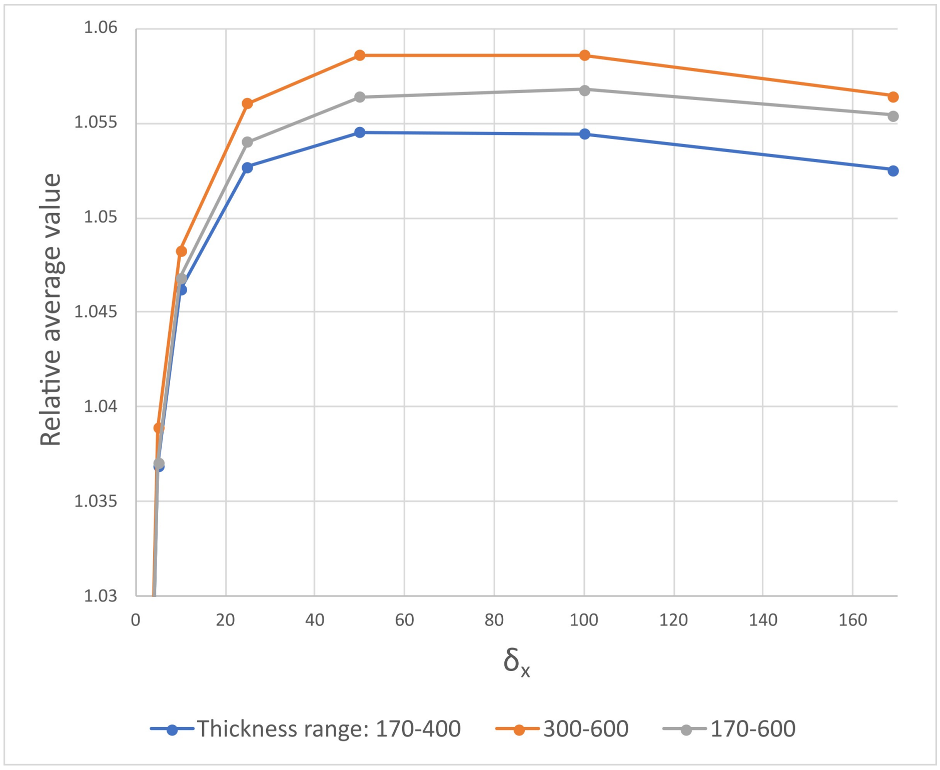


Figure 5.6: Optimization value as a function of δ_x: Relative values with respect to case $\delta_x = 1$

Chapter 6

Seed selection

The seed selection phase of optimization deals with the highest-level decision, namely the choice of an optimal seed. We recall that a seed includes the ordered list of the widths for the vertical cuts and the main cuts of the cutting pattern, an horizontal centering variable, two slope variables, in x and z directions, and the rotation of the whole pattern, as detailed by Definition 5.1.

Following the three-phases approach to FLO defined in Section 3.2, the seed selection phase focuses on the selection of width combinations for the primary vertical cuts in the pattern and their positions. The actual board filling and the addition of horizontal cuts are deferred to the second phase of optimization, as already detailed in Chapter 5.

The number of combinations available for selection can potentially be extensive, especially with an increase in the number of widths available for vertical and main cuts. However, in practical scenarios, the number of widths typically remains manageable, usually not exceeding a dozen, to give an idea of the problem's scale. As from the definition of cutting patterns given in Section 3.1.4, the widths of the side cuts correspond to the possible thicknesses of a set of boards known as *side boards*. These boards usually have reduced thicknesses and variable lengths, enabling optimization of the volume yield of the cut, with better adaptation to the irregular external shape of the log. Additionally, the same set of boards is used for filling the horizontal cuts. In contrast, the widths of the main cuts are sourced from a set of *main boards*, often representing the most significant products in terms of value. These boards are typically of full length, the same of the log, emphasizing their importance in the optimization process.

After an insight into the seed definition variables, the chapter describes the approaches considered for the seed selection phase of the optimization. In particular, in addition to a simple random seed sampling, a heuristic algorithm is explained in detail, and an experimental evaluation is reported to properly tune its parameters. The heuristic has been found inadequate, as even the simple random seed selection algorithm achieves better performance. Two other algorithms are thus described, based on searching procedures that explore the solution space related to the seed variables' domain. The first algorithm applies Mesh Adaptive Direct Search (MADS) [2], the second one applies NM-BBOA_CP from the DFL library [26]. Both approaches, according to literature, are suitable for integer-variable problems such as the one under consideration and belong to the category of derivative-free optimization algorithms. In fact, the nature of the FLO problem falls within the class for which these types of algorithms have been devised, since the problem is characterized by a highly discontinuous and computationally expensive objective function. Experiments show that the performance of these algorithms starting from a single initial solution is not satisfactory. We therefore propose the combination of these algorithms with a multi-start approach, based on an initial random selection of promising solutions. Finally, comparative results of the performance of the different algorithms are presented considering a fixed time limit: they show the ability of the multi-start derivative free search to provide good results with computational resources that are compliant with an inline implementation.

6.1 The seed's decision variables

We recall that, by Definition 5.1, a seed is a tuple of decision variables that we report here for the reader convenience:

$$(N^{VL}, N^{M}, N^{VR}, w_1^L \dots w_{N^{VL}}^L, w_1^M \dots w_{N^M}^M, w_1^R \dots w_{N^{VR}}^R, c_x, h_x, h_z, \alpha)$$

We notice that the selection of cut widths constitutes the combinatorial aspect of the seed, corresponding to its first variables that define a partial cutting pattern; we will refer to them as *pattern variables*. The last four variables of the seed refer to the degrees of freedom for positioning this partial cutting pattern within the log; we will denote them as *positioning variables*. The *positioning variables* are defined within a limited discretization range, in particular:

- c_x corresponds to a horizontal centering of the entire cutting pattern. During cutting, the horizontal centering is typically achieved by shifting the blade block relative to the log alignment. We define the evaluation range of c_x using a step δ_{c_x} and a number of symmetric steps N_{c_x} in the two opposite directions with respect to the plane $x = 0$ (see Figure 3.1), leading to

  $$c_x = k\delta_{c_x},\ -N_{c_x} \leq k \leq N_{c_x},\ k \text{ integer} \tag{6.1}$$

  For $k = 0$ we have $c_x = 0$, that is, the cutting pattern is centered with respect to the original alignment of the log in the reference system (see Section 5.1);

- h_x determines the common horizontal slope across the entire cutting pattern. It is related to an initial alignment of the reference system with respect to a predetermined axis of the log, as mentioned in Section 5.1. This initial alignment already serves as a good heuristic for assigning a value to this variable (corresponding to $h_x = 0$), and the mechanical centering system typically does not even allow for a different alignment.

In fact, such alignment would need to be performed with the log still intact, posing significant mechanical constraints. Nonetheless, we include this variable in the seed definition as it theoretically provides a potential optimization margin. Similar to c_x, we define the evaluation range for this variable using a step δ_{h_x} and a number of steps N_{h_x}, that is,

$$h_x = k\delta_{h_x}, \ -N_{h_x} \leq k \leq N_{h_x}, \ k \text{ integer} \tag{6.2}$$

- h_z determines the vertical slope of the cutting pattern, similarly to h_x. It is also centered with respect to the initial alignment of the log. However, this slope specifically relates to the main cutting block and consequently to any horizontal cuts added during the seed-constrained optimization phase. Vertical cuts, being independent of the main block, may have an independent z-axis slope, which is optimized directly within the seed-constrained optimization process. We define the evaluation range for this variable using a step δ_{h_z} and a number of steps N_{h_z}, so that

$$h_z = k\delta_{h_z}, \ -N_{h_z} \leq k \leq N_{h_z}, \ k \text{ integer} \tag{6.3}$$

- α represents the log rotation and it applies to the entire cutting pattern. Each individual board inserted into the cutting pattern will adhere to this rotation, or at most, 90° + α rotation if placed in vertical cuts. Sometimes, the cutting line imposes constraints on the degrees of freedom of rotation, primarily concerning the stability of the log during the initial cut. Specifically, when the curvature of the log exceeds a certain threshold, the log typically needs to be rotated for the initial cut so that the direction of maximum curvature aligns as vertically as possible. On the basis of the empirical seed symmetry discussed in Section 5.1.1, the useful evaluation range for the rotation angle is between 0 and 180°. After denoting the angular step with δ_α, we have:

$$\alpha = k\delta_\alpha, \ 0 \leq k < \pi/\delta_\alpha, \ k \text{ integer}. \tag{6.4}$$

6.2 Random sampling

The random sampling approach aims to serve as a baseline for other seed selection approaches. The algorithm involves randomly selecting a seed based on the feasible ranges for each individual variables. For all of the algorithms that we will be comparing in this chapter, the same time limit will be taken into account for each of the logs in the test set. Throughout the available time, the random sampling algorithm will generate new seeds and request their evaluation to the seed-constrained optimization phase, storing the best evaluation obtained. There can be various ways to implement this random selection, particularly regarding the number and width of cuts. The chosen method will be detailed in Section 6.4.3, as it will also serve to provide the initial solutions of the multi-start derivative-free search approaches.

6.3 Heuristic approach

The devising of a heuristic procedure for seed selection is driven by its simplicity and its direct reliance to value maps to evaluate a reference width for inserting cuts based on available board sections. Additionally, the proposed heuristic suits the specifications of a realistic flexible cutting line that we will consider in our analysis. This cutting line is characterized by:

- $N^{VL} \in \{1, 2\}$, $N^{VR} \in \{1, 2\}$ (one or two left vertical cuts, and one or two right vertical cuts)
- $N^{M} \in \{1, 2, 3\}$ (one to three main cuts);
- one board only on each verticul cut.

The algorithm runs through two steps:

1. sorting all possible seeds, considering solely on the *pattern variables*, and based on a priority;
2. selecting a limited range for the *positioning variables* of the seed.

In the selection of both left and right vertical cuts, combinations leading to wider outer cuts compared to the inner cuts are excluded. Although this constraint is not enforced by the cutting machine, this choice has long been implemented in software such as Maxicut Pro for optimizing side cuts. The purpose is to reduce, at least partially, the potential number of combinations, aligning with a widely accepted criterion in the practices of cutting patterns construction.

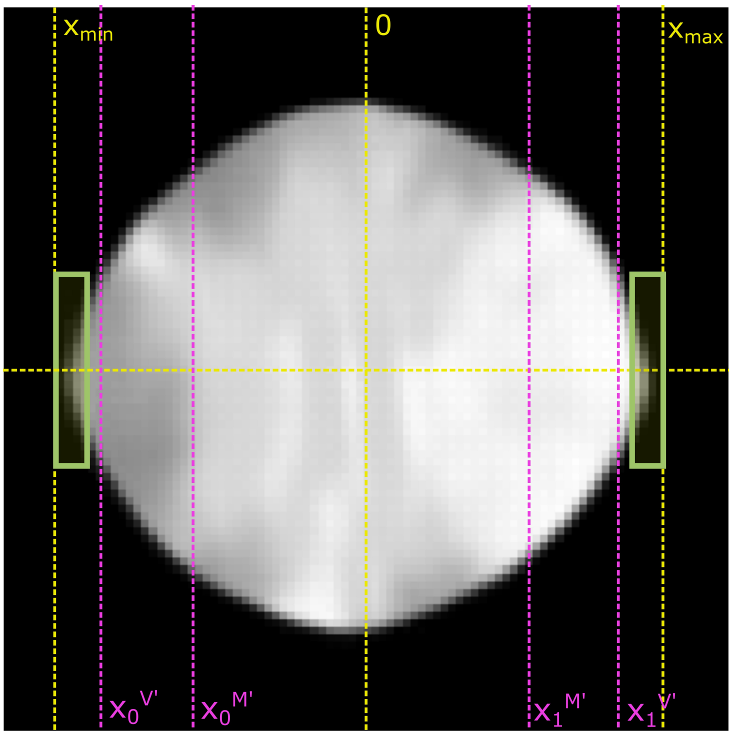


Figure 6.1: Calculation of the expected positions of the vertical and main cuts, based on the value map. Sections at the extreme value positions are highlighted in green, and the expected positions (in purple) are calculated as a percentage based on their positions

The procedure for assigning priority to a partial pattern is based on the comparison of four cutting positions related to the two outermost vertical and main cuts. Position are consider after the assumption that the cutting pattern is centered with respect to the reference system (and thus to the log, as seen in Section 5.1). Based on the value maps, “ideal” positions are obtained considering the actual log. These ideal positions are computed by evaluating, by means of V-Maps, the most extreme positions, with respect to the x-axis, of a board inserted in the outermost left and right vertical cuts. From these extreme positions, the four ideal cutting positions are obtained according to two multiplicative coefficients, one related to the vertical cuts ($0 < k_V < 1$), one related to the main cuts ($0 < k_M < k_V$), determined by experimental tests (see Figure 6.1). The priority is then determined by the distance of the first calculated positions with respect to the ideal ones. More in detail, the procedure is as follows:

1. For each possible side board thickness, among those available, select the section S with the smallest width and choose its V-Map $V = (M^v, S, \delta_{xz}, \alpha, h_x, h_z)$, with $\alpha = \pi/2, h_x = 0, h_z = 0$;

2. based on this V-Map, determine the positions i_{min} and i_{max} for each section in the matrix M^v. Each position is evaluated in the central row $j = N/2$ (corresponding to coordinate $z = 0$). i_{min} and, respectively, i_{max} correspond to the first and last non-zero values in the row of the matrix;

3. for each section S, compute $x_{min} = (i_{min} - N/2)\delta_x - t/2$ and $x_{max} = (i_{max} - N/2)\delta_x + t/2$, where t is the thickness of the section S. x_{min} and x_{max} represent an approximation of the reference positions for vertical cuts that can generate value for the given section;

4. create an exhaustive list of partial patterns (which can be seen as seeds constrained to $c_x = 0, h_x = 0, h_z = 0, \alpha = 0$). To this end, consider: all possible combinations of maximum N^{VL} widths of vertical cuts, based on the section thicknesses of the products considered by the optimization; all combinations of maximum N^M widths of main cuts, based on the section widths of the products; all combinations of maximum N^{VR} widths of vertical cuts. These three combinations of widths are to be

further combined with each other to obtain the exhaustive list of partial patterns;

5. For each of these partial patterns, compute x_0^V, x_0^M, x_1^V, x_1^M, representing respectively the left position of the first vertical cut, the left position of the first main cut, the right position of the last main cut, and the right position of the last vertical cut, using the same method as in Equation 5.2. Therefore, we will have $x_0^V \leq x_0^M \leq x_1^V \leq x_1^M$;

6. based on the thickness of the first and last cut, select the values x_{min} and x_{max} calculated in step 3. Given the two coefficients $0 < k_V < k_M < 1$, determine the expected positions for the limits of the vertical and main cuts:

$$x_0^{V'} = k_V x_{min} \tag{6.5}$$

$$x_0^{M'} = k_M x_{min} \tag{6.6}$$

$$x_1^{M'} = k_M x_{max} \tag{6.7}$$

$$x_1^{V'} = k_V x_{max} \tag{6.8}$$

7. determine the priority of the partial pattern as a measure of the distance between expected and actual positions, in particular:

$$P^S = (x_0^{V'} - x_0^V)^2 + (x_0^{M'} - x_0^M)^2 + (x_1^{M'} - x_1^M)^2 + (x_1^{V'} - x_1^V)^2 \tag{6.9}$$

Once the partial patterns are sorted based on the priority value, the algorithm will evaluate them in ascending order, testing, for each of them, the predefined reduced positioning range, until the available running time expires.

6.4 Derivative free approach

The problem of seed selection can be formalized as a black-box optimization problem, where we lack an explicit mathematical representation of the objective function. The problem under consideration falls within the following general definition:

$$\min f(x), \quad x \in C \subset \mathbb{Z}^n \tag{6.10}$$

where n is the number of variables included in the seed. We recall that both pattern and positioning variables of the seed are constrained to integer values: the former correspond to the choice of a size among the available product sections; the latter are defined after a discretization step.

The seed selection process is characterized by several aspects that make it suitable for a solution approach based on derivative-free optimization [27]. In particular:

- the execution of the objective function is computationally intensive. To obtain the value of a solution, completing the optimization of the constrained seed is required, which entails evaluating a significant number of boards;

- the decision variables are integer, and they cannot be relaxed for the sake of the objective function evaluation; the integer constraint cannot be relaxed;

- the *pattern variables* of the seed have a combinatorial nature, leading to strong discontinuities in the objective function. For example, modifying the width of a main cut not only entails optimizing that specific cut from an entirely different set of main products, but also changes the horizontal positioning of all other cuts based on the new overall width;

- even the *positioning variables* of the seed, c_x, α, h_x and h_z, potentially continuous, are actually discretized with significant steps, because of the need to precalculate V-Maps in a reasonable time, leading to substantial variations in the objective function. For example, some cuts may become invalidated in a single discretization step.

The definition of the set C in Equation 6.10 pertains to the encoding of the solution (solution representation), which affects the definition of the search space and, hence, potentially, the search behavior. This is in fact a crucial aspect, as it distinguishes how the seed information is represented and processed by the search optimization procedure, and the number and the domain of variables depends on this choice. Let w_j^V, with $j = 1 \ldots N^{WV}$, denote the possible widths for vertical cuts and w_j^M, with $j = 1 \ldots N^{WM}$ denote the possible widths for main cuts. Additionally, let N^V be the maximum number of vertical cuts allowed in the cutting pattern, both left and right, and N^M be the maximum number of main cuts. We consider two alternative ways of defining the vector x of decision variables, depending on the chosen domain for the pattern variables of the seed. In particular we have:

1. **binary pattern variables (width-presence mode)**:

$$x = (x^{VL}, x^M, x^{VR}, c_x, h_x, h_z, \alpha) \tag{6.11}$$

where

$$x^{VL} \in \{0,1\}^{N^V \times N^{WV}}, \text{ i.e., } x_{i,j}^{VL} \in \{0,1\}, i = 1...N^V, j = 1...N^{WV} \tag{6.12}$$

$$x^{M} \in \{0,1\}^{N^M \times N^{WM}}, \text{ i.e., } x_{i,j}^{M} \in \{0,1\}, i = 1...N^M, j = 1...N^{WM} \tag{6.13}$$

$$x^{VR} \in \{0,1\}^{N^R \times N^{WR}}, \text{ i.e., } x_{i,j}^{VR} \in \{0,1\}, i = 1...N^V, j = 1...N^{WV} \tag{6.14}$$

Variable $x_{i,j}^{VL}$ takes value 1 if the width w_j^V is selected for the left vertical cut i, 0 otherwise. Remaining variables are defined in a similar way. At most one binary variable in each cut is allowed to take value 1, meaning that the following constraints (where we use $*$ as an abuse of notation) must be satisfied:

$$\sum_{j=1}^{N^{W*}} x_{i,j} \leq 1, \; i = 1...N^*, * \in \{V, M\} \tag{6.15}$$

If all binary variables in one cut have value 0, the cut is not present.

2. **integer pattern variables (width-list mode)**: we assume that the widths w_j^V for $j = 1...N^{WV}$, and w_j^M, for $j = 1...N^{WM}$, are in ascending order. We have:

$$x = (x^{VL}, x^M, x^{VR}, c_x, h_x, h_z, \alpha) \tag{6.16}$$

where

$$x^{VL} \in \{0...N^{WV}\}^{N^V}, \text{ i.e., } x_i^{VL} \in \{0...N^{WV}\}, \; i = 1...N^V \tag{6.17}$$

$$x^{M} \in \{0...N^{WM}\}^{N^M}, \text{ i.e., } x_i^{M} \in \{0...N^{WM}\}, \; i = 1...N^M \tag{6.18}$$

$$x^{VR} \in \{0...N^{WV}\}^{N^V}, \text{ i.e., } x_i^{VR} \in \{0...N^{WV}\}, \; i = 1...N^V \tag{6.19}$$

The value taken by each variables x_i^{VL}, x_i^M, x_i^{VR}, if ≥ 1, directly corresponds to the choice of a width in the ordered set of widths; a value of 0 indicates that the cut is not present.

The set C is defined not only by simple bounds on the variables but, for both modes, some combinations of input variables will not be feasible and are not part of the domain C. For example, a solution where an inner vertical cut is absent while an outer one is present is not allowed.

The *positioning variables*, on the other hand, are managed in a similar way in both modes. The only consideration concerns the handling of rotation, related to the implementation of specific derivative-free algorithms:: to allow optimization algorithms to work in the proximity of angles 0 and π, the domain will be slightly enlarged.

6.4.1 Mesh Adaptive Direct Search - NOMAD

The Mesh Adaptive Direct Search (MADS) algorithm [2, 3] offers a robust approach to optimization and is well-suited for addressing problems characterized by nonlinearity, discontinuity, and lack of smoothness in the objective function. It is part of the family of Direct-search Methods. The basic strategy of the algorithm is to iteratively refine a mesh of points in the solution space, dynamically adapting to the local characteristics of the objective function, decreasing the mesh size to refine a local minimum, or increasing the mesh size to explore other regions, based on a variable set of polling directions. The algorithm is well-documented, implemented, and constantly updated in the NOMAD library, written in C++.

6.4.2 Derivative Free Library - DFL

The other algorithm used for the problem is part of the Derivative Free Library, an open-source library born from a collaboration between CNR and several Italian universities. The algorithm is the *NonMonotone Black-Box Optimization Algorithm for Constrained Problems* (NM-BBOA_CP) [26], specifically designed for black-box integer problems. The algorithm is based on defining primitive directions on which, at each step, the algorithm attempts to find an improved solution relative to a reference value updated considering a fixed-size buffer containing the last useful objective function values, with a non-monotone linear search. The size of this buffer has been the subject of a preliminary analysis aimed at experimentally evaluating an appropriate value tailored to the specific type of optimization problem being analyzed (see Section 6.5.3). After determining a suitable buffer size, this choice was consistently maintained throughout all subsequent experiments to ensure comparability and reliability of the obtained results.

6.4.3 Multi-start approach

The proposed derivative-free algorithms are designed to explore the domain of feasible solutions, starting from a feasible one. To find a feasible solution for both algorithms, we use the same procedure described in the heuristic approach (Section 6.3), constructing a starting solution using the partial pattern with the highest priority, and keeping the other position variables (c_x, h_x, h_z, α, as defined in Section 6.1) to 0. However, the experimental results of these algorithms, simply starting from this first solution, are not satisfactory, as both algorithms do not seem to explore the domain efficiently, in fact they are outperformed by both the heuristic approach and the random search. Therefore, a multi-start approach is proposed for the MADS and the DFL algorithms. An initial exploration phase evaluates possible promising solutions for a short time interval compared to the available time limit. In the remaining time, the derivative-free algorithm is repeated, using as starting solutions those identified by the first phase, and subject to a time limit. In order to provide comparable results to the single start approach, the overall time limit is allocated part to the first sampling phase, and the remaining part is equally divided among all the single searches.

In order to improve the sampling ability of the multi-start approach, the choice of the initial solution is made by a random search. This choice is justified by the fact that, as we will show in Section 6.5, this very simple approach outperforms the heuristic approach, even if the latter has been experimentally tuned (see Section 6.5.1).

The procedure for the multi-start approach is as follows:

1. for a predetermined time interval (in our experiments, 10s), a first phase of random sampling is performed (see Section 6.2);

2. all solutions evaluated by the first phase are sorted according to the value obtained;

3. the best solution is selected from this sorted list. This will be the first starting solution for the derivative-free algorithm;

4. the derivative-free algorithm is executed, from that starting solution, with a predetermined time limit (in our experiments, 5s);

5. a new starting solution is selected, taking the next element in the ordered list identified by step 2; the new solution must be sufficiently distant from those already used as a starting point. This evaluation is done by considering a minimum threshold (in our experiments, equal

to 5) on the distance calculated by:

$$d = ||x^0 - x^1|| \tag{6.20}$$

where x^0 and x^1 are vectors containing the seed variables of 2 solutions (Section 6.1);

6. the procedure is repeated from step 4, until the time limit is reached.

During the random sampling (see Section 6.2), seeds are generated based on a uniform random selection, in the admissible range, of each element of the vector containing the seed variables.

6.5 Numerical results

All the experiments reported in this section are based on a test set of 50 logs, each 4 meters long, with a variable diameter range as shown in Figure 6.2. The logs were acquired in a production line during normal processing using a CT Log scanner at a scanning speed of 130m/min. The log model used to calculate the value maps was also generated directly inline using Microtec CT Pro standard software. The value maps from the log model were calculated using CNN networks specifically trained on the specific products, according to the methods described in Section 4.3.

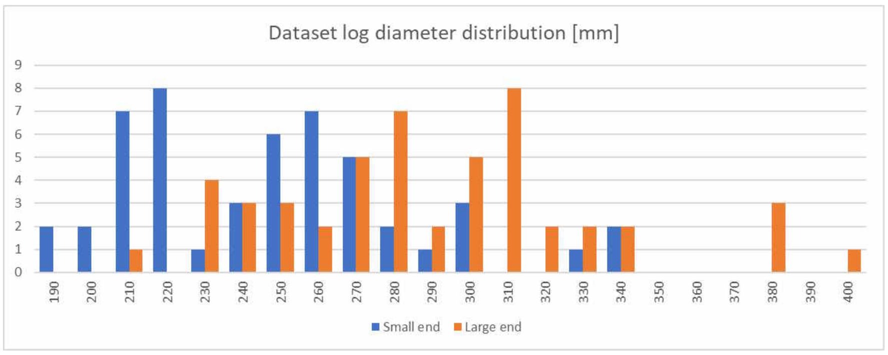


Figure 6.2: Dataset logs diameter distribution. Small end refers to the diameter of the smallest end of the log, large end to the largest

The products used for the test are listed in Table 6.1, which shows the side products, and in Table 6.2, which shows the main products.

Thick [mm]	**Width [mm]**	**Length [mm]**	**N° of qual.**	**Rules considered**	**Prices**
17	95	3000-4000	4	Wane, sound&dead knots	50,70,100,170
17	180	3000-4000	4	Wane, sound&dead knots	50,70,100,170
25	95	3000-4000	4	Wane, sound&dead knots	50,70,100,170
25	180	3000-4000	4	Wane, sound&dead knots	50,70,100,170
32	95	3000-4000	4	Wane, sound&dead knots	50,70,100,170
32	180	3000-4000	4	Wane, sound&dead knots	50,70,100,170

Table 6.1: List of the side products used in the simulation. Each row corresponds to a board section, for which multiple products (with different qualities, lengths and prices) are defined

Each row of the table corresponds to a section, and includes multiple products distinguished by quality, with increasingly restrictive rules and prices that increase accordingly. In the case of side boards, the products are also defined with multiple possible lengths, ranging from 3 to 4 meters. The tables also provide a summary indication of the types of defects considered by the rules of those products. The choice of these products and defects is similar to what may be required for optimization in a sawmill setting.

Furthermore, all simulations are based on the approach for valuing boards using approximate V-Maps generated by convolutional networks (see Section 4.3). The motivation behind this, is the ability to generate all necessary maps for the simulation within reasonable time frames. We can still reasonably assume that the overall result obtained deviates from what could be achieved with original value maps by approximately 1-2%, as empirically verified in Section 4.4. A more detailed verification may be achieved by using non-approximated V-Maps (see Section 4.2). This would require a significant computational effort, and would be out of the scope of our tests, focused on

Thick [mm]	**Width [mm]**	**Length [mm]**	**N° of qual.**	**Rules considered**	**Prices**
32	95	4000	4	Wane, sound&dead knots	50,70,100,170
32	180	4000	4	Wane, sound&dead knots	50,70,100,170
37	60	4000	2	Heartwood, knots	50,170
37	100	4000	2	Heartwood, knots	50,170
40	60	4000	3	Knots, pith	80,120,140
40	100	4000	3	Knots, pith	80,120,140
40	200	4000	3	Knots, pith	80,120,140
45	100	4000	3	Knots	50,85,105

Table 6.2: List of the main products used in the simulation. Each row corresponds to a board section, for which multiple products (differentiated by quality and price) are defined

the effectiveness and efficiency of the proposed optimization methods, and it is left for future analysis.

The products used in the simulation lead to a total of 14 different sections, 6 for the side boards, and 8 for the mainboards. In the V-Maps approach, all products with the same section are summarized into a single map.

The rotation step chosen in the simulations, as defined in Equation 6.4, is $\delta_\alpha = \pi/36$, which is 5°, resulting in a total of 36 possible rotations. Overall, for these simulations, it was necessary to compute 504 value maps for each log in the set, by inferring the specific CNN networks and generating the input D-Maps based on the required rotations.

The horizontal centering step, as defined in Equation 6.1, is set to $\delta_{c_x} = 3$ mm, with the number of symmetric steps set to $N_{c_x} = 3$, for a total of 7 different possible horizontal centerings of the cutting pattern.

For this simulation, the degrees of freedom associated with the slopes were ignored. However, a subsequent study could certainly take this additional degree of freedom into account.

The optimization software was written in C++. For the MADS algorithm, the Nomad library version 4.3.1 was integrated. For the NM-BBOA_CP algorithm, its Python version was used, interfaced with the C++ software through data exchange via shared memory. To make the time comparisons as consistent as possible, the computation times considered for all algorithms are only those related to the calls of the black-box function and recorded by the optimization software in C++.

6.5.1 Estimation of the heuristic search's parameters

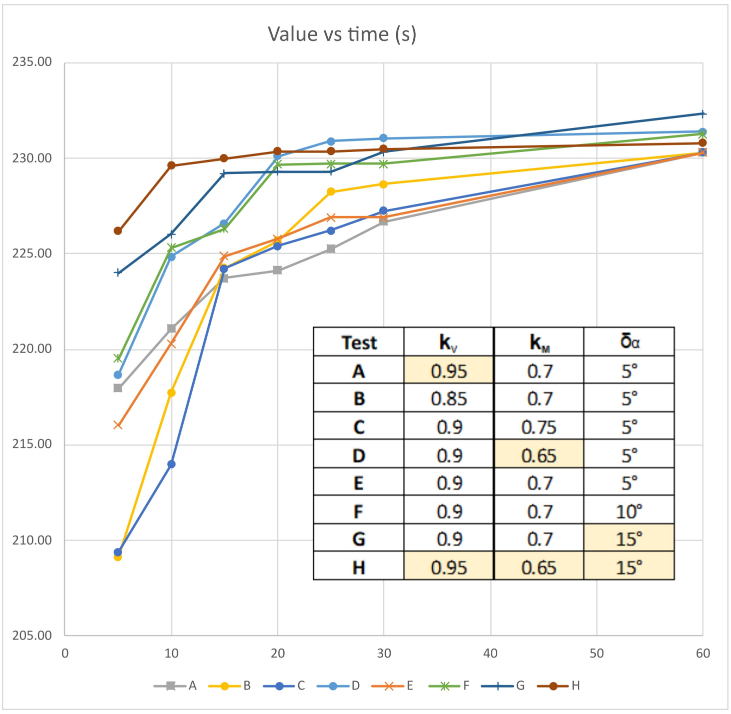
The heuristic approach defined in Section 6.3 includes parameters that impact its performance:

- k_V and k_M, used for the determination of the priority of each partial pattern, as described in Section 6.3;
- δ'_α, multiple of the general δ_α, to reduce the number of computed angles in order to test more partial patterns in the same time range.

Therefore, configurations were compared using a separate dataset of 20 logs to select the configuration that will be subsequently used for comparison with other algorithms. The results of this tuning phase are shown in Figure 6.3, where the tested combinations of parameters' value are reported. The chosen combination (H) is highlighted.

6.5.2 Choice of the pattern variables' domain

A preliminary evaluation was conducted regarding the choice of the seed representation, i.e., the mode for defining the domain of the pattern variables, to be used in the derivative-free approach. We recall that two possible modes have been proposed, namely the *width-presence mode* defined by Equation 6.12, and the *width-list mode* defined by Equation 6.17. The evaluation was performed using only the Nomad library, under slightly different conditions



Test	k_V	k_M	$\delta\alpha$
A	0.95	0.7	5°
B	0.85	0.7	5°
C	0.9	0.75	5°
D	0.9	0.65	5°
E	0.9	0.7	5°
F	0.9	0.7	10°
G	0.9	0.7	15°
H	0.95	0.65	15°

Figure 6.3: Test of different parameter sets for heuristic algorithm.

compared to subsequent tests. In particular, the number of products considered in this early stage of the research, which occurred at a preliminary stage of the research, was a few smaller than the complete configuration considered in later tests. However, the solution based on the width-presence mode proved to be less effective from the outset, with losses of up to 10% of the value. The width-list mode reduces the number of variables and the frequency of domain invalidations as a result of the black-box function, which leaded us to focus on this mode for subsequent tests. We remark that, choosing to sort the list in ascending order of width, is coherent with the physical meaning of the variables, corresponding to an expansion of the cutting pattern.

6.5.3 Selection of the buffer size for NM-BBOA_CP algorithm

The NM-BBOA_CP algorithm requires setting the size of the memory buffer used during the non-monotone line search step. A good choice for the buffer length is important because it affects the algorithm's ability to escape from regions of the solution space that are critical because of the landscape defined by the objective function (such as steep-sided valleys or flat areas). In a preliminary phase of the research, simulations were conducted to define an appropriate value for the buffer size. Starting from a minimum value set to 1, the buffer size was progressively increased by one unit, verifying the performance of the algorithm using different time limits. A peak performance value was determined to be at a buffer size of 8. The test results are shown in Figure 6.4. We remark that the benchmark of this preliminary test slightly differs from the one of the final comparative tests presented in the next Section 6.5.4, in terms of number and value of boards defined in the configuration of the optimization. Based on this preliminary test, the buffer size has been set to 8 for all the following experiments.

6.5.4 Comparative results

Figures 6.5 and 6.6 show the results of tests using different optimization procedures obtained by the following settings:

1. RND: it corresponds to the random approach described in Section 6.2. The choice of the value of the domain variables is random at each iteration. Each variable of the partial pattern is selected uniformly within its validity range, determined by a maximum and minimum value, according to the width-list mode. The same approach is used for the positioning variables, according to Section 6.1;

2. HEU: heuristic search with parameters tuned as in Section 6.5.1;

3. NMD: algorithm MADS implemented in Nomad;

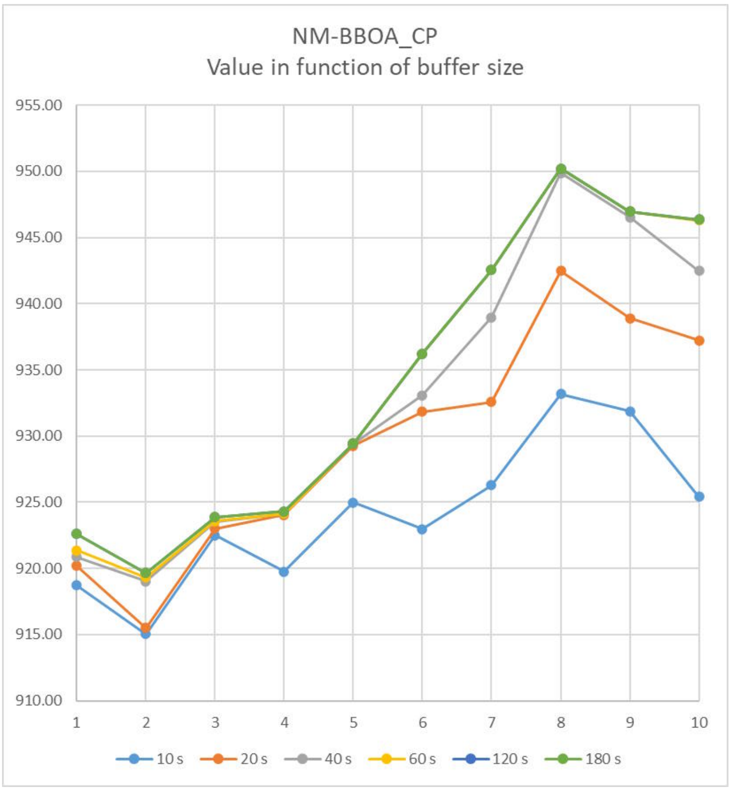


Figure 6.4: Test of NM-BBOA_CP algorithm with different buffer sizes. Each line corresponds to a different execution time limit.

4. DFL: algorithm NM-BBOA_CP from DFL library;
5. R-NMD: algorithm MADS with multiple starting points chosen with random approach (see Section 6.4.3);
6. R-DFL: algorithm NM-BBOA_CP with multiple starting points chosen with random approach (see Section 6.4.3).

The analyzed time range is one minute of overall processing time devoted to the evaluation of the objective function; one minute roughly corresponds to the time we can allocate to optimizing a log, as estimated in Section 4.4. The top line of Figure 6.5 (denoted by MAX-HEU) represents a reference value obtained by the heuristic search, set with very tight discretization steps and a time limit of 60 minutes per log.

In Figure 6.6, the trend relative to this reference solution is reported, showing only the algorithms able to find, in a limited amount of time, the closer value to this reference.

Preliminary considerations can be made based on these results:

- the simple NMD and DFL algorithm without multi-start have proven to be ineffective in exploring the solution space;
- The heuristic search fails to quickly reach a satisfactory result. In fact, the limits imposed to reduce the search in view of the time limit have proven to be too restrictive, making random exploration without specific constraints preferable;
- the R-NMD and R-DFL solutions exhibit similar behaviors and significantly outperform the reference procedure based on intensive and long heuristic search. Both algorithms, after the first 10 seconds devoted to finding the starting solutions for the multi-start approach, quickly obtain values close to the reference solution. The MADS algorithm appears slightly better on the tested instances.

In Figure 6.7, some examples of optimized cutting patterns are shown, reporting the best solution found for the same log by different approaches.

6.5.5 Comparative results with split-seed

Further tests were conducted to verify if the alternative seed definition (split-seed) outlined in Section 5.5 would achieve better performance. Figure 6.8 shows the results obtained using this approach with some of the previously considered algorithms, namely:

- RND-S: random approach with split-seed;
- R-NMD-S: MADS algorithm with multiple random starting points and split-seed;

- R-DFL-S: NM-BBOA_CP algorithm with multiple random starting points and split-seed.

It can be observed that the results of each algorithm tested using the split-seed approach are, after a same running time, approximately 0.5% worse than the same algorithm applied with the standard seed approach. The result can be motivated by the fact that the heuristics for optimizing the main cuts used for standard seed approach (see Section 5.2.4) are more efficient, especially when cutting patterns have multiple main cuts. Indeed, the optimization of the standard seed requires a systematic exploration of the possible combinations of the splits of the main cuts in the application of Algorithm 3. This exploration requires, in turn, repeated calls to Algorithm 2 (stacking of boards with fixed start position), for all the split positions considered in each main cut and in both directions. In contrast, the optimization based on the split-seed requires an evaluation of a reduced number of split combinations (less than 10% compared to the standard seed evaluation in our experiments), since the exploration of the solution space also includes the split positions and it's more specifically guided by the optimization made by Phase 3 (seed selection phase). However, from a computational perspective, each call to Algorithm 4 (which replaces Algorithm 3 when the split-seed is used) requires each time an evaluation from scratch of all the main cuts, i.e., Algorithm 4 calls Algorithm 2 in both directions for all main cuts. In the systematic exploration implemented by Algorithm 3, a same call to Algorithm 2 is conveniently exploited by more combinations of split positions. This issue could be further investigated in the future to explore the possibility of a different implementation of Algorithm 4, as to enable some incremental evaluation that reduces the number of calls to Algorithm 2 avoiding to call it for previously evaluated split positions. The main challenge toward this implementation is the memory required to store the incremental information resulting from the various calls to Algorithm 2. In summary, although incorporating the split positions into the seed provides more focused sequences of tested combinations of split positions, our experiments show that this is not sufficient to compensate for the additional calls to Algorithm 2.

Our tests also show that the performance of the split-seed based optimization tends to stabilize at a worse level than the standard seed approach. The result suggests that it is more advantageous to perform a more comprehensive exploration of split positions given a standard seed, as done by the standard seed constrained optimization embedded in Algorithm 3, compared to the exploration of the split positions performed by a derivative-free algorithm, although the latter approach allows for the exploration of a larger overall number of split-seeds and, hence, recalling that a split-seed is an extension of the standard seed, a larger number standard seeds.

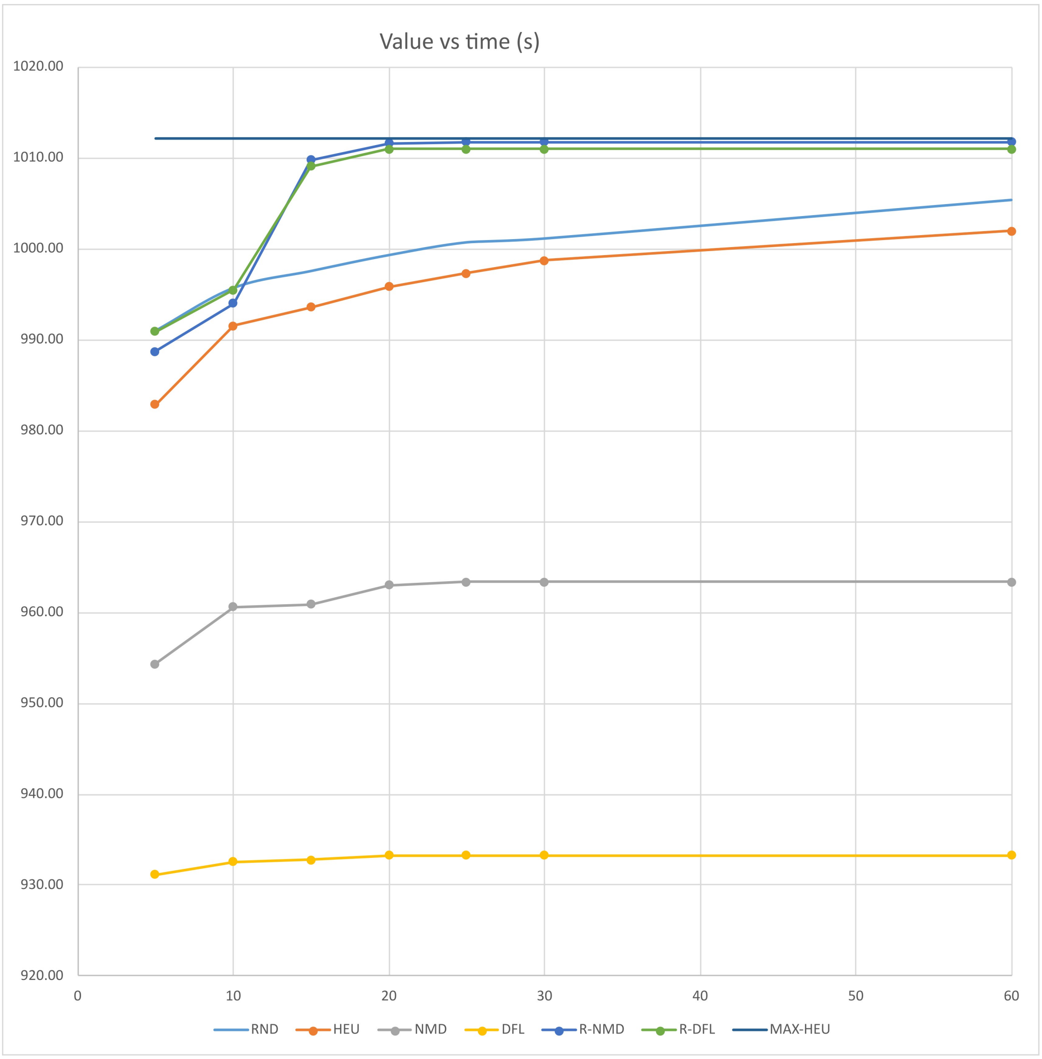


Figure 6.5: Test of different algorithms. The graph reports the total value obtained from each algorithm. The MAX-HEU line is the reference value obtained by the heuristic after a long execution time

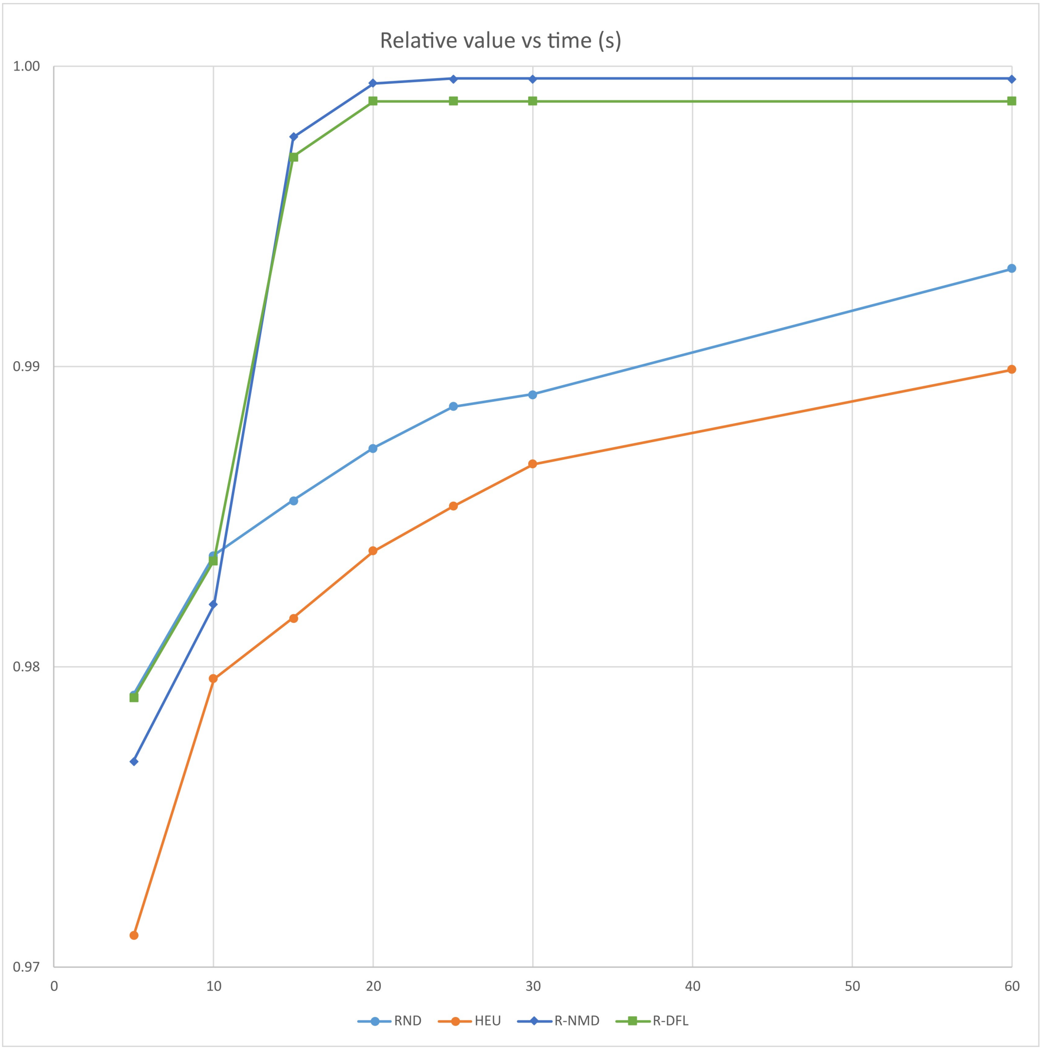


Figure 6.6: Test of different algorithms. The graph reports the relative value obtained from each algorithm with respect to the reference value obtained by the heuristic after a long execution time

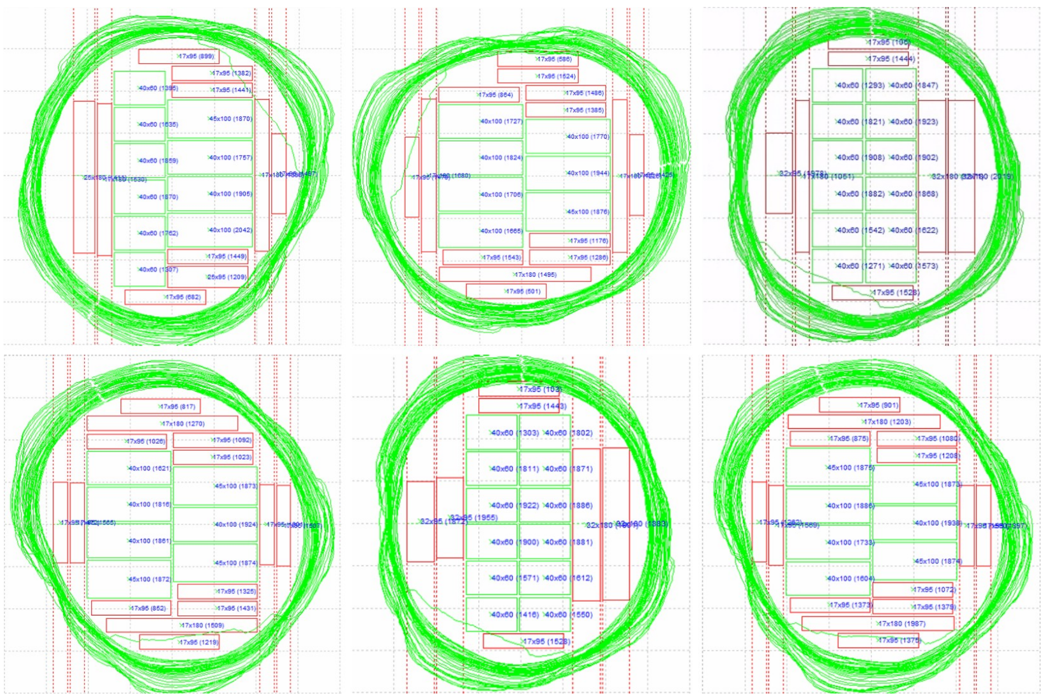


Figure 6.7: Examples of optimization results: from top to bottom, left to right: DFL (score 30.3, angle:120°), NMD (score 30.2, angle:355°), RND (score 31.1, angle:250°), HEU (score 30.6, angle:240°), R-NMD/R-DFL (score 31.2, angle:270°), MAX-HEU (score 31.5, angle:250°)

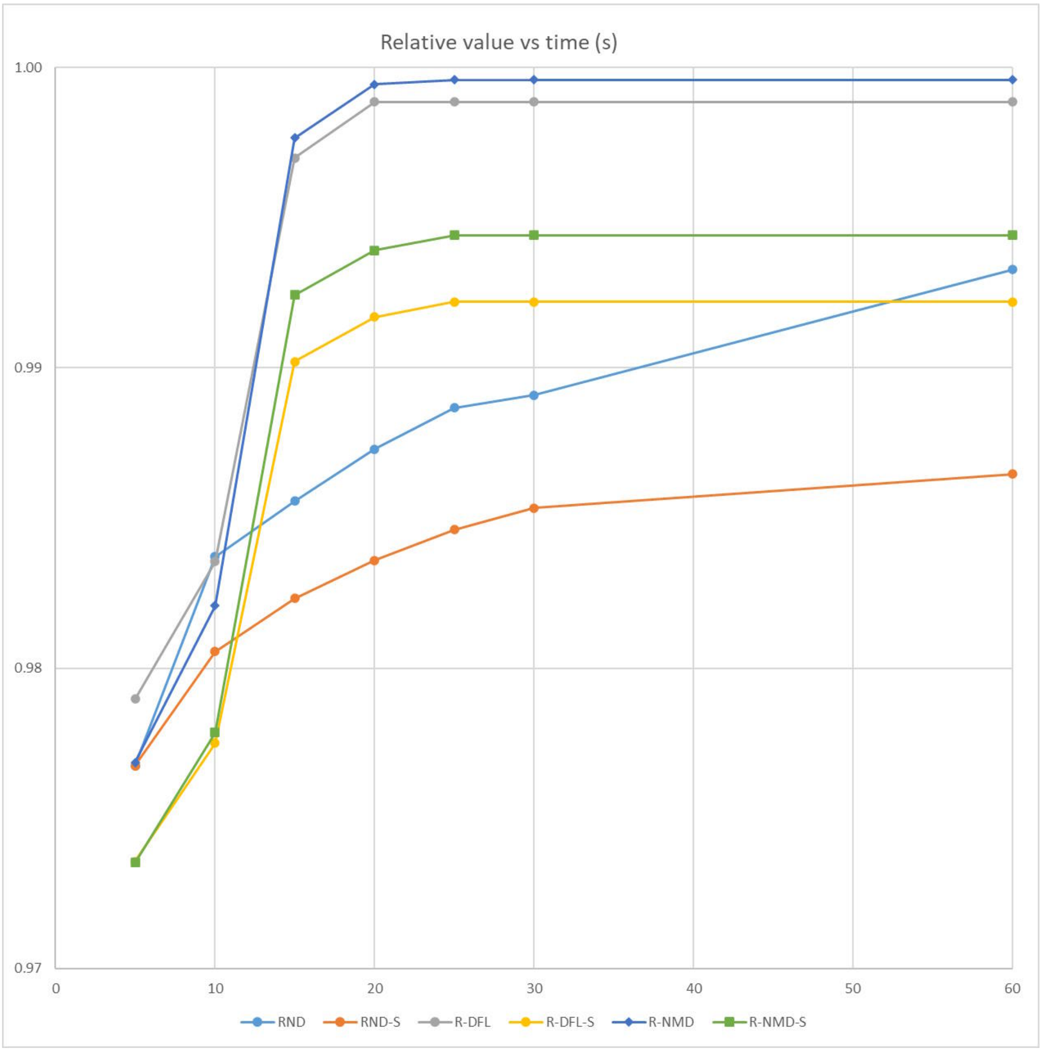


Figure 6.8: Test of algorithms with split-seed approach. The graph reports the relative value obtained from each algorithm with respect to the reference value obtained by the heuristic after a long execution time

Chapter 7

A Mixed Integer Linear Programming approach for the Ideal Cutting Pattern Problem

The log optimization problem considered in this chapter arises from an industrial application where it is necessary to find out the best cutting pattern based only on the log diameter and a set of available products.

The log is assumed as a homogeneous cylinder: this assumption reduces the log optimization problem into a more usual 2D cutting stock problem, since the rotation or inclination of the cutting pattern as well as different slopes of the side cuts do not impact the value of a solution. Under these assumptions, we can model the boards as rectangles and the log shape as a circle of given diameter. In addition, the value of the boards is not dependent on their position inside the log and the boards are valid only if fully contained in the circle, i.e, no wane is allowed.

This problem may seem simpler than the FLO problem discussed in the previous chapters, however, the challenge comes from the fact that the number of products that can be used to construct the cutting pattern is very large. As we will discuss later, the problem considered in this chapter, unlike FLO, is intended for application in contexts where the flexibility and control over cutting individual logs is minimal.

In this chapter, an initial description of the Ideal Cutting Pattern Problem (ICPP) is given. Then the ICPP is modeled as a Mixed Integer Linear Programming (MILP) problem, defining all the constraints and the function to be maximized. For the definition of the constraints and the objective function, the cutting pattern is decomposed into two types of cuts, namely side and main cuts. The individual constraints of the various cuts are then combined for the definition of the full ICPP problem. We will then discuss how the solution of ICPP can be conveniently integrated as a subproblem of an overall production planning problem arising in sawmills, in particular when low degree of flexibility is allowed. Finally, comparative experimental results are presented, considering multiple scenarios with different amounts of main and side products available to the optimization.

7.1 The Ideal Cutting Pattern Problem (ICPP)

We are given two sets of products, main and side. Each product is characterized by a thickness, a width and a value. The problem consists of constructing the best cutting pattern in two dimensions, in order to fill the area defined by a circle, maximizing the overall value of the inserted products. The cutting pattern consists of several cuts, some related to main products, some related to side products, and must satisfy some construction constraints, as shown in Figure 7.1.

A feasible cutting pattern is structured as follows:

1. One or two main product blocks. Each of these blocks consists of a sequence of products of the same size stacked on top of each other. The two blocks may consist of products of different sizes and therefore may have different overall block heights;

2. In the right and left of the two main blocks, there is a sequence of 1 to 3 side product cuts. These cuts can contain one or more products of the same thickness placed side by side (products 1 to 12 in Figure 7.1);

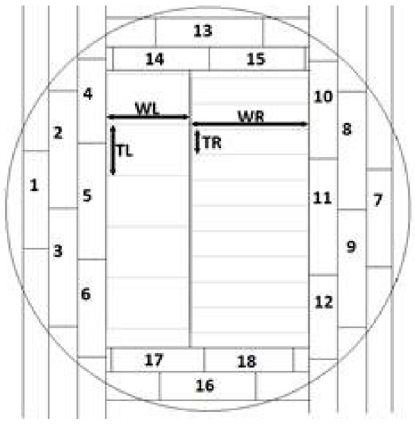


Figure 7.1: Cutting pattern for ICPP

3. At the top and bottom of the main blocks, there are two further sequences of 1 or 2 cuts of side products (products 13 to 18 in Figure 7.1).

The various cuts and products are spaced by a fixed thickness (the thickness of the saw blade that will make the cut).

Additional specific limitations due to the mechanics of the line are ignored: any combination of products that meets what is specified in the constraints described above is considered valid.

The cutting patterns that can be constructed under this definition can be seen as a subset of those that can be constructed under the cutting pattern definition for the FLO problem (see Section 3.1.4). In this case, the constraints can be expressed by limiting the number of allowed cuts and removing some positional degrees of freedom:

$$N^{VL} \leq 3,\ N^{VR} \leq 3,\ N^{M} \leq 2,\ N^{HT} \leq 2,\ N^{HB} \leq 2 \tag{7.1}$$

$$h_x^{CP} = 0,\ h_z^{CP} = 0,\ \alpha^{CP} = 0 \tag{7.2}$$

In addition to these restrictions, it would also be necessary to impose a limit on the number of boards in each cut, and to impose boards of the same size in each of the main cuts. The board valorization function should also simply evaluate the possibility of inserting a board inside a cylinder. In any case, solving ICPP using the approach seen for FLO is not particularly useful, since a suitable heuristic has already been developed under the very stringent constraints of the problem.

7.2 MILP modelling

A dedicated heuristic was developed at Microtec to address ICPP, obtaining at first glance satisfactory results in reasonable time for industrial use. To validate the results obtained by this heuristic, we propose to formulate the problem as a Mixed Integer Linear Programming (MILP) model and to use off-the-shelf solvers. This approach, through constraint relaxation, is able to provide upper bounds for optimization.

7.2.1 General definitions

Let N^M and N^S be the sizes of two lists of products (main products and side products). For each product, a width, a thickness and a value are defined, and they are denoted by, respectively,

$$w_i^M, t_i^M, v_i^M,\ i = 1...N^M \tag{7.3}$$

for main products, and, for side products, by

$$w_i^S, t_i^S, v_i^S,\ i = 1...N^S \tag{7.4}$$

Let R_C be the radius of the circle to be optimized. The circle is assumed centered in $(0,0)$. Let s be the saw blade thickness assumed for simplicity to be the same for all cuts.

The cutting pattern can be conveniently modeled by decomposing it into the 10 single lateral cuts (L, R, T, B) and in the 2 main cuts $(C0, C1)$, as shown in Figure 7.2.

7.2.2 Model for each side cut

We define, for each side cut (Figure 7.3), the following variables:

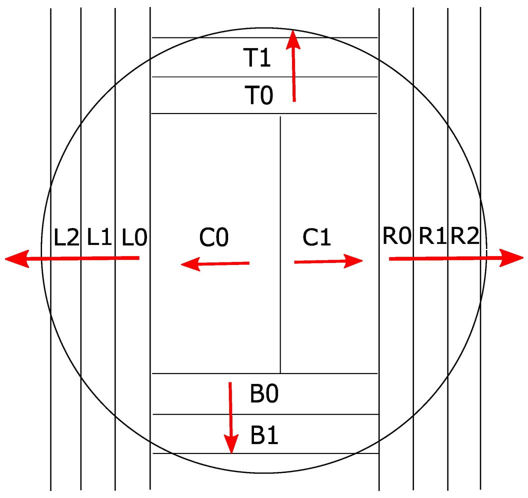


Figure 7.2: Cut types and order

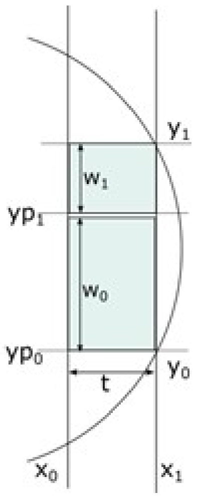


Figure 7.3: Side cut model

- cut limits

$$x_0, x_1 \in [0, R_C],\ y_0, y_1 \in [-R_C, R_C] \tag{7.5}$$

- thickness of the cut

$$t \in [0, R_C] \tag{7.6}$$

- denoting by N^P the maximum number of products allowed for the cut, position and width of products

$$y_i^P \in [-R_C, R_C],\ w_i^P \in [0, 2R_C],\ i = 1...N^P \tag{7.7}$$

- side product presence:

$$p_{ij}^S \in \{0, 1\},\ i = 1...N^P,\ j = 1...N^S \tag{7.8}$$

  equal to 1 if at position i there is the product j, 0 otherwise, and

$$p_i \in \{0, 1\},\ i = 1...N^P \tag{7.9}$$

  equal to 1 if at position i there is a product, 0 otherwise.

The following constraints must be satisfied:

- thickness of the cut:
$$x_0 + t = x_1 \tag{7.10}$$

- presence of the product:
$$\sum_{j=1}^{N^S} p_{ij}^S = p_i, \ i = 1...N^P \tag{7.11}$$
Since p_i is binary, this constraint also implies that at most one product can be active;

- thickness of the cut equal to that of the selected products:
$$\sum_{j=1}^{N^S} p_{ij}^S t_j^S \leq t + K(1 - p_i), \quad i = 1...N^P \tag{7.12}$$
$$\sum_{j=1}^{N^S} p_{ij}^S t_j^S \leq t + K(1 - p_i), \ \ i = 1...N^P \tag{7.13}$$
We consider the possibility that the cut is not present, in which case the constraint has no effect by choosing a K sufficiently large;

- product width (for convenience, we also add a saw blade thickness if the product is present):
$$\sum_{j=1}^{N^S} p_{ij}^S (w_j^S + s) = w_i^P, \ i = 1...N^P \tag{7.14}$$

- vertical position of products:
$$y_i^P = y_{i-1}^P + w_{i-1}^P, \ i = 2...N^P \tag{7.15}$$

- compulsory presence of the cut (needed for the innermost side cut (L0, B0, R0, T0) to force the presence of at least one cut:
$$p_1 = 1 \tag{7.16}$$

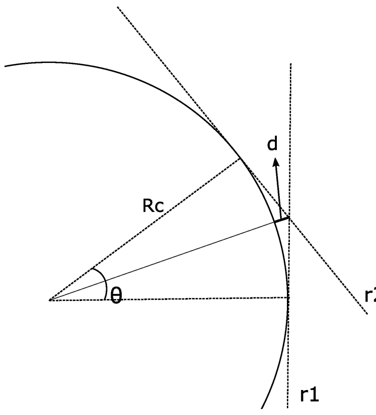


Figure 7.4: Shape constraints

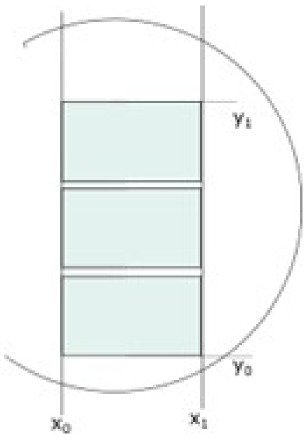


Figure 7.5: Main cut model

- continuous presence with respect to the index order:

$$p_i \leq p_{i-1},\ i = 2...N^P \tag{7.17}$$

- vertical cut limits:

$$y_0 = y_o^P,\ y_1 = y_0 - s + \sum_{t=1}^{N^P} w_i^P \tag{7.18}$$

- constraints related to the log:

  we must ensure that the products are inside the circle. It is sufficient to verify that the ends (x_1, y_0) and (x_1, y_1) are inside the circle of radius R_c. Linear constraints involving these variables are then added to approximate a circle. The chosen approximation is such that it extends the original domain of the variables, while keeping the maximum distance from the circle below 0.5 mm; for the variables (x_1, y_0), we consider positive slope constraints; for (x_1, y_1), negative slope constraints.

  Let d be the maximum allowed distance between the circumference and the polygon defined by the linear constraints to be inserted. (see Figure 7.4). It is easy to determine the angular step θ needed for a succession of these constraints, associated to lines $r1$, $r2$ etc., as the ones in Figure 7.4:

$$\theta = 2\cos^{-1}\left(\frac{R_c}{R_c + d}\right) \tag{7.19}$$

The two lines, $r1$ and $r2$ in Figure 7.4 represent an example of lines that define the polygon; each line, calculated with step θ in the range $[0, \pi/2[$ for the point (x_1, y_1), and in the range $]-\pi/2, 0]$ for the point (x_1, y_0), adds an inequality to the set of constraints. For each of the two points, N constraints will be needed, with N defined by $N = \lceil \frac{\pi}{2\theta} \rceil$, namely:

$$y_1 \leq -\frac{x_1}{\tan(\frac{\theta}{2} - k\theta)} + \frac{R_C}{\sin(\frac{\theta}{2} - k\theta)} + M(1 - p_1), \quad k \in 1...N, \tag{7.20}$$

$$y_0 \geq \frac{x_1}{\tan(\frac{\theta}{2} - k\theta)} - \frac{R_C}{\sin(\frac{\theta}{2} - k\theta)} - M(1 - p_1), \quad k \in 1...N. \tag{7.21}$$

In the event that the side cut does not appear, a large-enough constant M is added to make the constraint redundant if no products are selected in the cut.

Each side cut inserted into the model contributes to the objective function with the following value:

$$\sum_{i=1}^{N^P} \sum_{j=1}^{N^S} p_{ij}^S v_j^S \tag{7.22}$$

7.2.3 Model for each main cut

Let N^{MAX} be the maximum number of products that can be inserted in the cut (in standard conditions we can assume it to be 20). We define, for each of the main cuts, the following variables (see Figure 7.5):

- limits of the main cut:

$$x_0, x_1 \in [-R_C, R_C], y_0, y_1 \in [-R_C, R_C] \tag{7.23}$$

- selected product:

$$p_i \in \{0, 1\},\ i = 1...N^M \tag{7.24}$$

  equal to 1 if the product j is selected, 0 otherwise;

- quantity of products:

$$n_i^P \in [0, N^{MAX}],\ \text{integer},\ i = 1...N^M \tag{7.25}$$

We have to satisfy the following constraints:

- one and only one selected product:

$$\sum_{i=1}^{N^M} p_i = 1 \tag{7.26}$$

- width and height of the cut:

$$x_0 + \sum_{i=1}^{N^M} p_i w_i^M = x_1,\ \ y_0 - s + \sum_{i=1}^{N^M} n_i^P (t_i^M + s) = y_1 \tag{7.27}$$

- constraints relative to the log:

  they are expressed in an analogous way to the case of the side cut.

Each of the main cuts contributes to the objective function with the following value:

$$\sum_{i=1}^{N^M} n_i^P v_i^M. \tag{7.28}$$

7.2.4 Cuts-binding constraints

Additional constraints are added to create the sequences of cuts and join main and side cuts. With reference to Figure 7.2 for the names of the various cuts, we extend the notation to refer to the variables defined for each cut:

- limits to the number of products in side cuts:

$$N^{P,R0} = 3, \; N^{P,R1} = 2, \; N^{P,R2} = 1, \tag{7.29}$$

$$N^{P,L0} = 3, \; N^{P,L1} = 2, \; N^{P,L2} = 1, \tag{7.30}$$

$$N^{P,T0} = 2, \; N^{P,T1} = 1, \tag{7.31}$$

$$N^{P,B0} = 2, \; N^{P,B1} = 1; \tag{7.32}$$

- sequences of side cuts:

$$x_0^{R1} = x_1^{R0} + s, \quad x_1^{R1} = x_2^{R0} + s, \tag{7.33}$$

$$x_0^{L1} = x_1^{L0} + s, \quad x_1^{L1} = x_2^{L0} + s, \tag{7.34}$$

$$x_0^{T1} = x_1^{T0} + s, \tag{7.35}$$

$$x_0^{B1} = x_1^{B0} + s; \tag{7.36}$$

- joining main and side cuts:
  in order to join all the cuts, we have to consider them rotated according to their "direction" with respect to x-axis (see Figure 7.2). In particular, cut $T0$ is rotated by 90°, cuts $L0$ and $C0$ by 180°, cut $B0$ by 270°. Related constraints are the following:

$$-x_0^{C0} + s = x_0^{C1}, \tag{7.37}$$

$$x_1^{C1} + s = x_0^{R0}, \tag{7.38}$$

$$-x_1^{C0} - s = -x_0^{L0}, \tag{7.39}$$

$$x_0^{T0} \geq -y_0^{C0} + s, x_0^{T0} \geq y_1^{C1} + s, \tag{7.40}$$

$$x_0^{B0} \geq y_1^{C0} + s, x_0^{B0} \geq -y_0^{C1} + s; \tag{7.41}$$

- top and bottom side cuts internal with respect to main cuts:

$$-y_1^{T0} \geq -x_1^{C0},\ -y_0^{T0} \leq x_1^{C1}, \tag{7.42}$$

$$y_0^{B0} \geq -x_1^{C0},\ -y_1^{B0} \leq x_1^{C1}; \tag{7.43}$$

The objective function for the MILP problem is given by the sum of all contributions of the 12 cuts in the cutting pattern.

7.3 Application of ICPP in sawmill production planning

The objective of the ICPP is limited to maximizing the value of the products that can ideally be placed within a single log of a specified diameter. By varying the input data for this sub-problem (both the diameter and the product list), it is possible to obtain good associations between diameters and cutting patterns that can be used for an overall production planning problem.

Assuming a log yard where logs are sorted in piles by diameter, the optimization problem arises when there's a need to fulfill orders for a certain quantity of products with specific dimensions. The challenge lies in determining how many logs of each diameter to use from each pile and which cutting patterns to employ to achieve the desired products with the highest possible yield. By solving multiple instances of ICPP, we can generate feasible cutting patterns for each log diameter, based on the dimensions of the required products. This set of candidate cutting patterns can be thus used as part of the input of the overall production planning problem.

The ideality of the ICPP is motivated by the fact that the required cutting pattern optimization should not be tied to an individual log. Indeed, the cutting pattern generated through ICPP is taken as a reference for the evaluation of the products obtainable from a large number of logs of the same diameter when cut using that cutting pattern. We observe that for this type of production, the cutting line will be set up in a non-flexible manner, prioritizing speed.

Due to the large number of logs that will be cut over time using a given ideal cutting pattern, as output by ICPP, we can collect a large number of observations related to the actual value of the same pattern when actually applied to real logs. This information can be conveniently exploited to deal with the uncertainties related to the cutting process, due, e.g., to cut or measurement tolerances. The uncertainty in the log cutting process may lead to a difference between the set of products included in the ideal cutting pattern and the set of products that have actually been obtained from each log. For example, because of the real shape or the defects of each individual log, less products of a given size may be consistently obtained with respect to their ideal number. The collected observations can be used to infer some corrective factor that aligns the evaluation of the ideal cutting pattern according to ICPP to a more reliable evaluation to be used as an input to the overall production planning problem.

We also observe that information required by the approach described above can be obtained by simulating the application of an ideal cutting pattern to a significant dataset of logs of a given diameter acquired through tomography. By applying the specific cutting pattern to these datasets and generating virtual boards, as described in Section 2.3.3, one can assess the statistical production outcomes and adjust the evaluations before using them in production planning. However, further insight into the overall optimization problem is beyond the scope of this thesis and may be explored in future research.

7.4 Numerical Results

The model was implemented in C++ using the Gurobi API (version 9.11) [22] and ran on a PC with an Intel$^{(R)}$ Core$^{(TM)}$ i7-7820HQ CPU, clock 2.90GHz, 4 cores, 8 logical processors, 32GB RAM.

Scenario	N. Main	N. Side
S1	96	90
S2	48	90
S3	48	48
S4	48	24

Table 7.1: Scenarios

Four scenarios were considered with a different amount of available main and side products (see Table 7.1). The side products had thickness between 12 mm and 40 mm and width between 75 mm and 175 mm. The main products had thickness between 27 mm and 57 mm and width between 95 mm and 200 mm.

Each of the four scenarios has been tested with two different log diameters (400 mm and 500 mm). Figure 7.6 shows an example of the optimization result for scenario S1 and a log diameter of 500 mm.

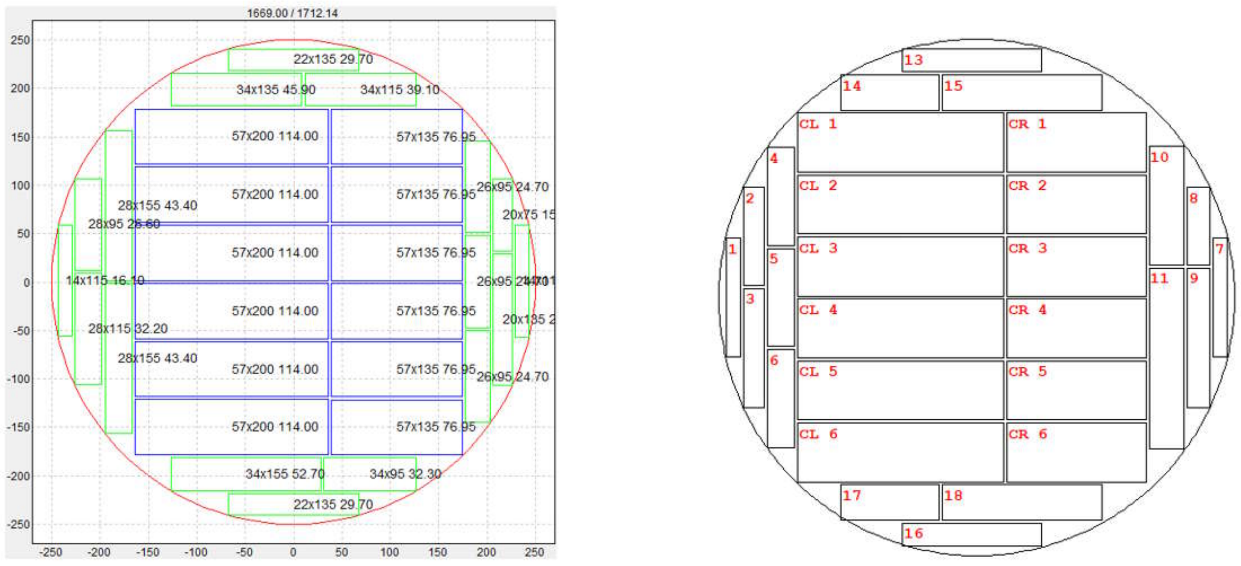
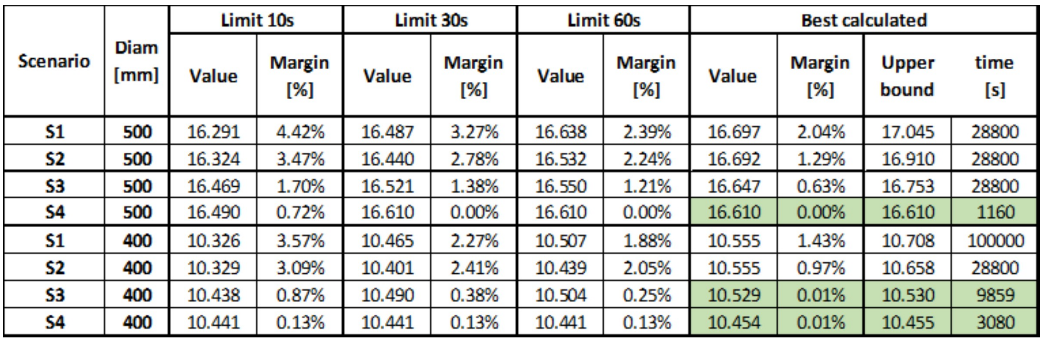
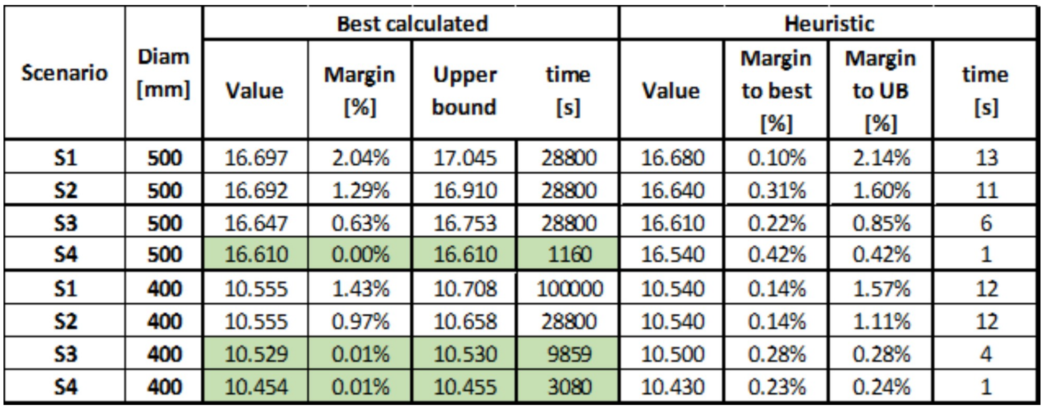


Figure 7.6: Solution obtained using MILP solver (left) or heuristic algorithm (right)



Scenario	Diam [mm]	Limit 10s		Limit 30s		Limit 60s		Best calculated			
		Value	Margin [%]	Value	Margin [%]	Value	Margin [%]	Value	Margin [%]	Upper bound	time [s]
S1	500	16.291	4.42%	16.487	3.27%	16.638	2.39%	16.697	2.04%	17.045	28800
S2	500	16.324	3.47%	16.440	2.78%	16.532	2.24%	16.692	1.29%	16.910	28800
S3	500	16.469	1.70%	16.521	1.38%	16.550	1.21%	16.647	0.63%	16.753	28800
S4	500	16.490	0.72%	16.610	0.00%	16.610	0.00%	16.610	0.00%	16.610	1160
S1	400	10.326	3.57%	10.465	2.27%	10.507	1.88%	10.555	1.43%	10.708	100000
S2	400	10.329	3.09%	10.401	2.41%	10.439	2.05%	10.555	0.97%	10.658	28800
S3	400	10.438	0.87%	10.490	0.38%	10.504	0.25%	10.529	0.01%	10.530	9859
S4	400	10.441	0.13%	10.441	0.13%	10.441	0.13%	10.454	0.01%	10.455	3080

Table 7.2: Results of MILP solver in different scenarios



Scenario	Diam [mm]	Best calculated				Heuristic			
		Value	Margin [%]	Upper bound	time [s]	Value	Margin to best [%]	Margin to UB [%]	time [s]
S1	500	16.697	2.04%	17.045	28800	16.680	0.10%	2.14%	13
S2	500	16.692	1.29%	16.910	28800	16.640	0.31%	1.60%	11
S3	500	16.647	0.63%	16.753	28800	16.610	0.22%	0.85%	6
S4	500	16.610	0.00%	16.610	1160	16.540	0.42%	0.42%	1
S1	400	10.555	1.43%	10.708	100000	10.540	0.14%	1.57%	12
S2	400	10.555	0.97%	10.658	28800	10.540	0.14%	1.11%	12
S3	400	10.529	0.01%	10.530	9859	10.500	0.28%	0.28%	4
S4	400	10.454	0.01%	10.455	3080	10.430	0.23%	0.24%	1

Table 7.3: Comparative results, MILP and heuristic

The computational results obtained by the MILP solver under the different conditions are reported in Table 7.2. The first two columns identify the scenario in terms of the product set used and the diameter of the circle. The next columns show the value obtained by the solver after 10, 30 and 60 seconds and an additional value (best) obtained at the end of the total run time. In addition, each value is also associated with the percentage gap calculated with respect to the best upper bound, shown in the second-to-last column of the table. The last column shows the total run time in seconds. A time limit of 28800 seconds (8 hours) was generally applied for all the test, but for one configuration (scenario S1, diameter 400) a longer time limit was considered (100000 seconds, about 28 hours). In the simplest three cases only, the solver reached an optimum (highlighted in green in the table). In more complex scenarios, finding the optimal solution can take several days.

The results obtained with the heuristic algorithm are reported in Table 7.3. For comparison, the left half of the table contains the best results of the MILP approach (exactly the same as Table 7.2). For the heuristic, the columns report the value obtained, the gap with respect to the best solution found by the MILP solver (in the total run time), the gap with respect to the upper bound calculated by the MILP solver and the time in seconds used by the heuristic to find its solution. Heuristic approach, in a really short time (13 seconds at most), always found a solution less than 0.42% far from the best found by the MILP solver, and less than 2.14% far from to the upper bound in the worst case.

The MILP approach considered here, although not directly usable in the specific application, certainly allowed us to verify the quality of the developed heuristics. A similar approach can be easily extended to other types of optimization based on cutting patterns in which the shape of the log can be still approximated with linear constraints (a convex shape). The approach finds its limitations in other applications in which it is necessary to consider the actual shape of the log and in which the value of the products is dependent on their position within the log itself.

Chapter 8

Conclusions and perspectives

The research undertaken in this thesis was motivated by a specific requirement: to enhance the utilization of log models acquired from tomographic scanners in sawmill cutting optimization processes. The existing methods, which relied solely on virtual board models and rule evaluations, proved inadequate to effectively optimizing cutting strategies in flexible production lines based on tomographic data.

The Flexible Log Optimization problem (FLO), as introduced in this thesis, formalizes the problem of determining the optimal cutting pattern while accommodating flexible cutting line conditions and considering the board's valorization relative to its position and orientation within the log.

Hence, the primary goal of this research was to devise a solution to expedite the board evaluation process, a fundamental component of optimization algorithms employed in this context. The initial proposed solution, the R-Maps-based valorization, has already been implemented in a sawmill facility in France. This approach addresses the needs for flexible optimization, matching the available degrees of freedom. However, empirical evidence indicates that the R-Maps solution exhibits notably inferior performance, both in terms of speed and achieved value, when compared to V-Maps, the alternative valorization approach proposed in this thesis. Despite its shortcomings, the R-Maps approach has the advantage of being relatively straightforward to implement. This allowed for its rapid deployment, given the absence of alternative algorithms tailored for flexible optimization based on tomographic log models. Nonetheless, it's worth noting that the R-Maps approach outlined in this thesis may benefit from further refinement to address defects and rules not currently managed. However, the level of rule approximation and the continuous adaptation required for managing new rules could potentially prevent further implementations of this approach.

The approach based on V-Maps, devised in this thesis, is certainly more promising. In particular, the thesis discusses how V-Maps can be computed using CNN and what performance can be expected in terms of optimization value improvement compared to the current solution using R-Maps. An estimate of the hardware needed for inline use and integration into an optimization procedure for FLO has also been made, showing that it may already be deployed in many cutting environments. The main stumbling block remains the design of the infrastructure needed to train and deploy these networks in general. In fact, the V-Maps approach requires specific training for each combination of products that one wants to achieve with a cutting section. In view of this, it is necessary to approach log-cutting optimization with a more standardized view in terms of product definition, and any changes in optimization parameters should be planned in advance. An interesting line of research will certainly be to engineer a solution for providing these CNN networks, perhaps keeping an up-to-date library not only of the logs but also of all the virtual boards needed for training, limiting the recalculation of information as much as possible, and exploiting transfer learning possibilities to speed up training. A centralized solution for computing and deploying CNN-based V-Maps, as, e.g., the one schematized in Figure 8.1, can define a more general and shared product standard, and may also be more sustainable. Once the V-Maps are implemented, their benefits extend beyond complex optimization scenarios like the one presented in this thesis. It can also enhance simpler solutions with fixed cutting patterns, allowing a more thorough exploration of optimal positioning within the pattern, even

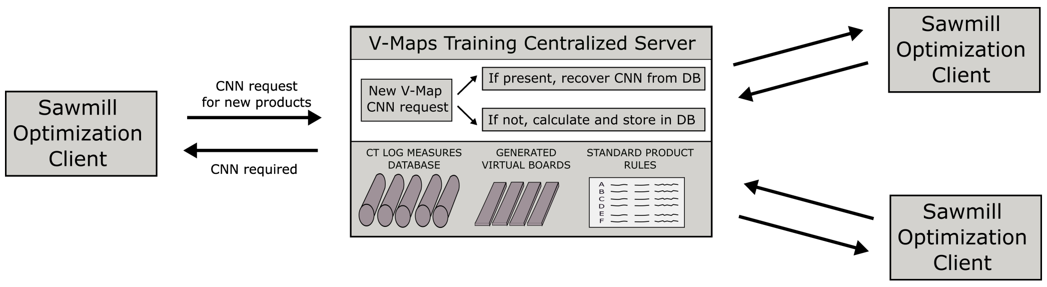

Sawmill Optimization Client
CNN request for new products
CNN required
V-Maps Training Centralized Server
New V-Map CNN request
If present, recover CNN from DB
If not, calculate and store in DB
CT LOG MEASURES DATABASE
GENERATED VIRTUAL BOARDS
STANDARD PRODUCT RULES
Sawmill Optimization Client
Sawmill Optimization Client

Figure 8.1: Centralized server concept for V-Maps CNN training

if it remains fixed, as in the current practice. This aspect will undoubtedly be further explored to improve both existing and future applications as this solution becomes more widespread.

The availability of a fast board evaluation has led to the definition of a solution approach to FLO based on three interacting phases. The first phase involves the board valorization, with its best implementation found in the V-Maps approach. The second phase, seed-constrained optimization, involves completing the optimization of a partially fixed solution (the seed) by positioning the boards starting from a seed selected by the third phase of the approach (seed selection). The seed consists of a partial cutting pattern where only the widths of the most important cuts are defined, together with the positioning of the pattern within the log in terms of horizontal alignment, slope, and rotation.

The thesis proposes an approach for the second phase of the optimization, the seed-constrained optimization, based on dynamic programming, that has proven well suited to address the specific problem, characterized by several technological constraints. It is not ruled out that there may exist an algorithmically optimal solution in relation to the problem's constraints, which could, for example, eliminate the need to try multiple combinations of center splits positions within limited ranges. However, this phase needs to be tailored very specifically to the conditions and constraints of the cutting line for which it is designed. The proposed algorithm efficiently handles the filling of the cuts fixed by the seed with a constrained sequence of boards. It may prove to be a very versatile tool for use in optimization with further specific constraints (such as optimization for a log carriage, see Section 2.1), thus representing a contribution not only as an integrated phase for FLO optimization, but also for other log optimization problems arising in more constrained cutting environments.

As for the third phase, the seed selection, this thesis has presented an experimental verification of how an original approach based on derivative-free algorithm can be used to address the FLO problem within the strict time limits that allow for an inline implementation, achieving better performance than a dedicated heuristic that, although simple, required a phase of study, implementation, and tuning. In particular, randomized multi-start proved to be essential in boosting the performance of the derivative-free algorithms. Further research may be devoted to improve the heuristic. However, the black-box approach proposed in this thesis remains more immediate and general. Further experiments can be conducted to extend and validate the approach for other cutting line configurations, e.g., for optimization of the saw carriage.

A fast evaluation of solutions to the FLO problem also paves the way for extending more precise optimization to bucking operations [12], which involve dividing longer logs, even up to 18 meters, into shorter logs. This preliminary optimization, typically based on external shape only or, when possible, on an aggregated internal quality measure, is constrained by the computational complexity of considering a thorough optimization of the short log as well. Existing solutions typically only sort the short logs into different bins based on a grade derived from computed tomography scans and virtual boards, once the bucking process is complete. New solution approaches to bucking optimization may thus benefit from improved procedures for FLO.

The thesis also addressed ICPP, a problem of potential interest as a tool for optimizing sawmill production at a higher level. The problem does not focus on individual log optimization, and is more suitable for fast cutting lines based on pre-selection of logs by diameter. Solving ICPP provides a list of possible standard cutting patterns according to a log diameter. The thesis contributes with a MILP approach to ICPP that has been useful to assess the quality of the solution obtained by the current heuristic procedure developed at Microtec.

The modeling and optimization opportunities enabled by the definition of V-Maps pave the way for other possible approaches to tackle the FLO. One particularly interesting research direction, which has already entered a preliminary study phase through an internship in collaboration with the University of Padua, seems to be the use of a reinforcement learning-based approach to address both phases two and three of the optimization.

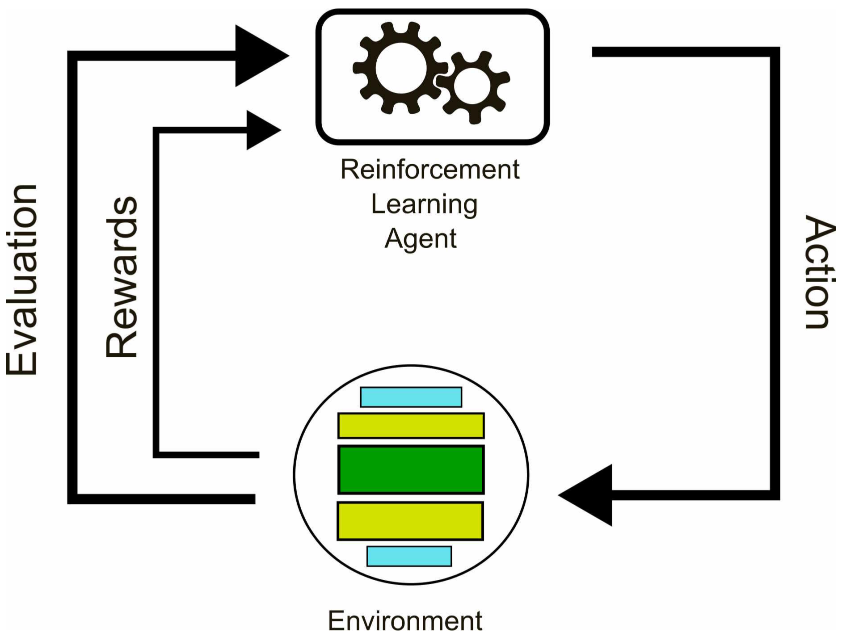


Figure 8.2: Basic representation of the Reinforcement Learning process

Reinforcement learning is a highly promising paradigm in machine learning, with applications ranging from autonomous driving to robotic manipulation, natural language processing, and gaming. In fact, the construction of the cutting pattern can be interpreted as a sequential decision-making process where the reinforcement learning algorithm can learn policies for inserting new cuts and boards (Figure 8.2). Certainly, the reward will be linked to the value of the inserted board at a certain position, and specific strategies (not trivial at first glance) should be adopted to consider the constraints of the cutting pattern construction and the utilization of V-Maps.

References

[1] A. De Almeida, M.B. Figueiredo, *A particular approach for the Three-dimensional Packing Problem with additional constraints*, Computers & Operations Research 37(11):1968-1976, 2010.

[2] C. Audet, J.E. Dennis, *Mesh adaptive direct search algorithms for constrained optimization*, SIAM Journal on Optimization, 17(1):188-217, 2006.

[3] C. Audet, S. Le Digabel, V. Rochon Montplaisir, C. Tribes. *NOMAD version 4: Nonlinear optimization with the MADS algorithm*, ACM Transactions on Mathematical Software, 48(3), 35:1-22, 2022.

[4] S.M. Bhandarkar, X. Luo, R.F. Daniels, E.W. Tollner, *Automated planning and optimization of lumber production using machine vision and computed tomography*, IEEE Transactions on Automation Science and Engineering, 5(4):677-695, 2008.

[5] A. Berglund, O. Broman, A. Grönlund, M. Fredriksson *Improved log rotation using information from a computed tomography scanner*, Computers and Electronics in Agriculture 90:152-158, 2013.

[6] E. Bettella, *A deep learning approach for log cutting pattern optimization in wood industry*, Master Thesis in Computer Engineering, University of Padua, 2022.

[7] M.C. Bouzid, S. Salhi, *Packing rectangles into a fixed size circular container: Constructive and metaheuristic search approaches*, European Journal of Operational Research, 285(3):865-883, 2020.

[8] L. Breinig, A. Berglund, A. Grönlund, F. Brüchert, U. Sauter, *Effect of Knot Detection Errors When Using a Computed Tomography Log Scanner for Sawing Control*, Forest Products Journal 63(7):263-274, 2013.

[9] K. Li, K.H. Cheng, *On Three-Dimensional Packing*, SIAM Journal on Computing, 19(5):847-867, 1990.

[10] C.A. Correa, M.R. Maldonado, D.M. Lozano, C. Carrasco, C. Aguilera, C. Acevedo, and D. Monsalve, *3D optimization of cutting patterns for logs of pinus radiata D.Don with cylindrical defective core*, in Proceedings of 10th Conférence Francophone de Modélisation, Optimisation et Simulation, pp. 1-8, 2015.

[11] W.B. Dowsland, *Three-dimensional packing—solution approaches and heuristic development*, International Journal of Production Research, 29(8):1673-1685, 1991.

[12] B. Faaland, D. Briggs, *Log bucking and lumber manufacturing using dynamic programming*, Management Science, 30(2):245-257, 1984.

[13] G. Fasano, *MIP-based heuristic for non-standard 3D-packing problems*, 4OR 6:291-310, 2008.

[14] M. Fredriksson, *Log sawing position optimization using computed tomography scanning*, Wood Material Science & Engineering, 9:110-119, 2014.

[15] J.L.A.P. Galvez, D. Borenstein, E. da Silveira Farias, *Application of optimization for solving a sawing stock problem with a cant sawing pattern*, Optimization Letters, 12(8):1755-1772, 2018.

[16] J. Geerts, *Mathematical solution for optimising the sawing pattern of a log given its dimensions and its defect core*, New Zealand Journal of Forestry Science, 14:124-134, 1984.

[17] P.C. Gilmore, R.E. Gomory, *Multistage cutting problems of two and more dimensions*, Operational Research, 13:94-119, 1965.

[18] S. Giovannini, D. Boschetto, E. Vicario, M. Cossi, A. Busatto, S. Ghidoni, E. Ursella, *Improving knot segmentation using Deep Learning techniques*, Proceedings of the 21st International Nondestructive Testing and Evaluation of Wood Symposium, Freiburg, Germany, 2019.

[19] F. Giudiceandrea, E. Ursella, E. Vicario, *A high speed CT scanner for the sawmill industry*, Proceedings of the 17th International Nondestructive Testing and Evaluation of Wood Symposium, 14-16, 2011.

[20] F. Grondin, *Improvements of the dynamic programming algorithm for tree bucking*, Wood and Fiber Science, 30(1):91-104, 1998.

[21] S. Grundberg, A. Grönlund, U. Grönlund, *The Swedish stem bank: a database for different silvicultural and wood properties*, Research report, Luleå Tekniska Universitet, 1995.

[22] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, https://www.gurobi.com, 2023.

[23] I. Hinostroza, L. Pradenas, V. Parada, *Board cutting from logs: Optimal and heuristic approaches for the problem of packing rectangles in a circle*, International Journal of Production Economics, 145(2):541-546, 2013.

[24] S.M. Hosseini, A. Peer, *Wood Products Manufacturing Optimization: A Survey*, IEEE Access 10:121653-121683, 2022.

[25] Y. LeCun, Y. Bengio, G. Hinton, *Deep learning*, Nature, 521:436-444, 2015.

[26] G. Liuzzi, S. Lucidi, F. Rinaldi, *DFLINT—an algorithm for black-box inequality and box constrained integer nonlinear programming problems*, https://doi.org/10.5281/zenodo.3653742, http://www.iasi.cnr.it/ liuzzi/DFL, 2020.

[27] G. Liuzzi, S. Lucidi, F. Rinaldi, *An algorithmic framework based on primitive directions and nonmonotone line searches for black-box optimization problems with integer variables*, Mathematical Programming Computation, 12:673-702, 2020.

[28] A. Lodi, S. Martello, M. Monaci, *Two-dimensional packing problems: A survey* European Journal of Operational Research, 141(2):241-252, 2002.

[29] F. Longuetaud, J. Leban; F. Mothe, E. Kerrien, M. Berger, *Automatic detection of pith on CT images of spruce logs*, Computers and electronics in agriculture, 44(2):107-119, 2004.

[30] C.G. Lundahl, A. Grönlund, *Increased yield in sawmills by applying alternate rotation and lateral positioning*, Forest Products Journal, 60(4):331-338, 2010.

[31] U. Nordmark, *Value recovery and production control in the forestry-wood chain using simulation technique*, PhD dissertation, Luleå tekniska universitet, Luleå, 2005.

[32] J. Oja, *Evaluation of knot parameters measured automatically in CT-images of Norway spruce (Picea abies (L.) Karst.)*, Holz als Rohund Werkst, 58:375-379, 2000.

[33] M. Pietikäinen, *Detection of knots in logs using x-ray imaging: Dissertation*, VTT Technical Research Centre of Finland, 1996.

[34] L. Pradenas, J. Garcés, V. Parada, and J. Ferland, *Genotype-phenotype heuristic approaches for a cutting stock problem with circular patterns*, Engineering Applications of Artificial Intelligence, 26(10):2349-2355, 2013.

[35] O. Ronneberger, P. Fischer, T. Brox, *U-net: convolutional networks for biomedical image segmentation*, International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 9351:234-241, 2015.

[36] A . Rais, E. Ursella, E. Vicario, F. Giudiceandrea, *The use of the first industrial X-ray CT scanner increases the lumber recovery value: case study on visually strength-graded Douglas-fir timber*, Annals of Forest Science, 74(2):1-9, 2017.

[37] J. Ren, Y. Wang, *Overview of Object Detection Algorithms Using Convolutional Neural Networks*, Journal of Computer and Communications, 10:115-132, 2022.

[38] S.M. Stängle, F. Brüchert, A. Heikkila, T. Usenius, A. Usenius, U.H. Sauter, *Potentially increased sawmill yield from hardwoods using X-ray computed tomography for knot detection*, Annals of Forest Science 72:57-65, 2015.

[39] F. Sultana, A. Sufian, P. Dutta, *Evolution of Image Segmentation using Deep Convolutional Neural Network: A Survey*, Knowledge-based systems, 201-202:106062, 2020.

[40] F.W. Taylor, F.G. Wagner, C.W. McMillin, I.L. Morgan and F.F. Hopkins, *Locating knots by industrial tomography - A feasibility study*, Forest Products Journal 34(5):42-46, 1984.

[41] E. Ursella, F. Giudiceandrea, M. Boschetti, *A Fast and Continuous CT scanner for the optimization of logs in a sawmill*, 8th Conference on Industrial Computed Tomography, 2018.

[42] E. Vicario, E. Ursella, *Patent: Computer-implemented method to provide a cutting pattern for a tree log to obtain wooden boards*, application n°: 812023000203964 filed at Ufficio Italiano Brevetti e Marchi (UIBM), December 2023.

[43] M. Wehrhausen, N. Laudon, F. Bruchert, U.H. Sauter, *Crack Detection in Computer Tomographic Scans of Softwood Tree Discs*, Forest Products Journal, 62(6):434-442, 2012.

[44] Q. Wei, B. Leblon and A. La Rocque, *On the use of X-ray computed tomography for determining wood properties: a review*, Canadian Journal of Forest Research 41:2120-2140, 2011.

[45] West Coast Lumber Inspection Bureau, *Standard No. 17 Grading Rules for West Coast Lumber*, 2018.

[46] Y. Wei, Z. Zhigang, L. Cheng, T. Huiming, *Pixel-wise regression using u-net and its application on pansharpening*, Neurocomputing, 312, 2018.