Abstract

The forest-products supply chain gives rise to a variety of interconnected problems. Addressing these problems is challenging, but could be simplified by rigorous data analysis through a machine learning approach. A large amount of data links these problems at various hierarchical levels (e.g., strategic, tactical, operational, online) which complicates the data computation phase required to model and solve industrial problem instances. In this study, we propose to use machine learning to generate models of the sawmills (converting logs into lumber) to simplify the data computation phase for solving optimization problems. Specifically, we show how to use these models to provide a recommendation for the allocation of cutblocks to sawmills for a wood allocation planning problem without needing extensive sawing simulations. Our experimental results on an industrial problem instance demonstrate that the generated models can be used to provide high-quality recommendations (sending the right wood to the right mill). Machine learning models of the sawmill transformation process from logs to lumber allows a better allocation exploiting the strengths of the mills to process the logs in our industrial case.

Keywords: Wood allocation planning, sawing simulation, machine learning application

1. Introduction

The industry faces many challenges that can be addressed with mathematical models using a large amount of data. For the forest-product industry, the wood allocation planning problem is one of those. Wood allocation consists of allocating the wood from cutblocks to sawmills, for a given company, over a given time period in order to maximize the profit from sales (lumber products, unprocessed logs, and by-products) minus the costs (harvesting,
transportation, production, and storage). Solving the wood allocation planning problem requires prior knowledge including the lumbers outputted when processing the wood from a given cutblock at a given sawmill. A sawing simulation can be performed for each individual log sample and sawmill pair to determine the outputs of the transformation. This process, however, is intensive in terms of both human and computational resources. A large forest company usually has not only one, but a network of sawmills and each log sample counts thousands of logs. In order to generate the mathematical model, every sawmill of the network first needs to be modeled in a format compatible with the simulator. Then, for each pair of sawmill in the network and individual log from the samples, the simulator evaluates thousands of feasible cutting patterns to select the one leading the highest value output which value is then used to build the mathematical model for wood allocation.

We propose machine learning as an approach to build approximate models of a sawmill network in order to facilitate the data computation phase for the wood allocation planning problem and simplify the overall process. We tested our models using them to provide input for a mixed-integer linear programming (MILP) formulation of the wood allocation planning problem (Jerbi et al., 2012) comparing them to the results obtained using the exact output of a conventional sawing simulator which is, in this context, considered as the ground truth.

The contributions of this research are threefold. First, we developed machine learning-based models of a sawing simulator. Second, we present a novel methodology to use these models for decision-making in the context of wood allocation. Finally, we evaluate the proposed methodology on an industrial wood allocation planning problem instance provided by an industrial partner.

On a broader perspective, this research could enable the use of the historical data on entry logs and output lumbers the companies possess to gain better knowledge of their sawing process without a need to generate detailed sawing simulations for all pairs of log and sawmill. This is especially interesting as not all companies have access to simulators. By demonstrating the performance of a machine learning approach for predicting the basket and therefore for decision-making, we add value to the historical data collected but not used by many companies. In other cases, where a simulator is available, it is not only computationally expensive to virtually transform the logs, it is also task-intensive in that human experts are needed to model the sawmill and to fine-tune the simulator. On the long term, a machine learning approach will provide automation opportunities for this process and simplify the human task in a context of labor shortage. Machine learning, by enabling the use of a log sample to estimate lumber, proves to be a useful data analysis tool for the forest-products industry. On the one hand, it facilitates the data
computation phase required to produce mathematical models for the forest-products industry problem solving and decision-making. On the other hand, it improves upon decisions taken using a procedure based on the historical production which is the current, less computationally and task intensive, alternative to simulation.

The paper is structured as follows. In Section 2, we provide an overview of the literature related to the wood allocation planning problem along with a review of important concepts in sawing simulation, and integration of simulation along with optimization. We touch upon metasimulation which can be viewed as a specific case of which machine learning is the general case. In Section 3, we describe the proposed machine learning approach to build models of the sawing process at a given sawmill. We present problem-specific metrics to evaluate the predictive (forecast) accuracy for these models. Experimental results are presented in Section 4. The experiment’s main purpose is to assess the decision-making suitability of the built models on an industrial case for wood allocation planning. We conclude in Section 5.

2. Preliminary Concepts

This section first presents important preliminary concepts related to wood allocation planning (Section 2.1). Then we discuss the log transformation process and we summarize the characteristics of actual sawing simulators (Section 2.2). We also present simulation and optimization as tools for solving wood allocation planning problems. Finally, we discuss background notions in metamodeling, a field of study of which machine learning can be seen as being the general case (Section 2.3).

2.1. Wood Allocation Planning

Forest-products supply chains are made of multiple interconnected and independent business units (D’Amours et al., 2008). Although we could see the various activities of the supply chain as independent problems, there are proven benefits from integrating the planning process (Rönnqvist, 2003; D’Amours et al., 2008). It is common to develop hierarchical decision plans in the forestry sector (Gunn, 2007; Church, 2007; Epstein et al., 2007). That is, the high-level decisions taken for a long period of time, e.g., investing in new sawmills or storage facilities (strategic level), become constraints and inputs to making lower-level decisions solved on a shorter time scale, e.g., wood allocation (tactical level). Planning problems in the forest supply chain are grouped in four levels, namely, strategic planning (long-term decision, more than 5 years), tactical planning (mid-term decision, 6 months to 5 years), operative planning (short-term decision, 1 day to a year), and online planning (less than a day) (Rönnqvist, 2003). From a broad decision-making perspective, the wood allocation planning problem is a tactical planning problem (Jerbi et al., 2012; Morneau-Pereira et al., 2014). In wood allocation, the capacities and the
location of the business units are assumed to be known from high-level decisions (Morneau-Pereira et al., 2014).

Wood allocation planning, of which specific examples and variants can be found in the work of Mendoza and Bare (1986), Maness and Norton (2002), Marinescu et al. (2005), Marinescu and Maness (2010), Ouhimmou et al. (2015), and Boukherroub et al. (2017), is the process of allocating the wood from cutblocks to sawmills for a given period of time (a year or several months). Wood from a single cutblock is usually allocated to a subset (one or more) of available sawmills. Practical allocation constraints include a maximum on the number of sawmills per cutblock which depends on the volume of the block (in cubic meters of wood). Other constraints include transport capacities, storage capacities, and production capacities. The objective is to maximize the profits of total sales from the produced lumbers, the unprocessed logs, and the by-products minus the costs of harvesting, transportation, production, and storage over the planning horizon. Wood allocation planning problems are often modeled as mixed-integer linear programs (MILP). The formulation of the problem we chose for our experiments was introduced by (Jerbi et al., 2012). Appendix A presents the model as generated in the context of our industrial application.

Wood allocation depends on data coming from at least two low-level processes: harvesting and sawing. It is influenced by harvesting since the available logs are unknown until the trees are harvested and the stems are truncated into logs. It is also influenced by sawing since we need to estimate the lumbers produced at each sawmill given the transformed logs in order to compute the potential benefits of allocating the wood from a given cutblock to a given sawmill. Figure 1 presents an overview of forest supply chain planning for wood allocation from tree harvesting to log transformation. The decision points for wood allocation are the quantities and the type of truck on the transport arcs from a cutblock (origin) to a mill (destination).

Figure 1: Forest supply chain planning for wood allocation

Complete literature reviews of the application of operations research to the forestry sector can be found in (Rönqvist, 2003), and (D’Amours et al., 2008).

2.2. Log Transformation and Sawing Simulation

North American softwood lumber products are normalized according to NLGA (National Lumber Grades Authority) grading rules. The type of lumber product is defined solely by its thickness, its width, and its length. The grade of a lumber prod-
uct is a category based on its quality with respect to the products of the same type. Even though the North American market allows for any produced lumber to be sold, prices vary in time (as other commodity products). By-products, such as sawdust and wood chips, are also produced by the mill and can be sold on the market, but at lower price than lumber.

Given a log to process, the sawing equipment decides in real time how that specific log will be transformed in lumber in order to maximize value. The transformation process of a log is a divergent complex physical process with co-production (see Figure 2). It is divergent since it results in multiple lumbers (outputs) with different characteristics, e.g., thickness and width. The lumbers are co-produced since their output is simultaneous for a single input (e.g., one $2'' \times 3''$, two $2'' \times 4''$, and two $2'' \times 6''$). The set of lumbers resulting from the transformation is called the basket (or mix) of products. The sawmill equipment is configured to select a feasible sawing pattern that maximizes produced value on the market. The feasible sawing patterns at a given sawmill depend on the available sawing equipment and of its configuration.

2.2.1. Sawing Simulation

Sawing simulation is one of the widely used techniques to obtain the data required to formulate the wood allocation planning problem when it comes to harvesting and sawing.

There exist multiple sawing simulators, e.g., AU-TOSAW (Todoroki et al., 1990), Innosim (Song et al., 2007), Optitek (FPInnovations, 2014), OPTSAW (Sandberg, 1997), Saw2003 (Nordmark, 2005), SAWSIM (HALCO, 2016), SEESAW (Todoroki, 1988), SIMSAW (Singmin, 1978), RAYSAW (Thomas, 2012), and WoodCIM (Useni and Heikkila, 2007). The input of most sawing simulators are virtual logs, i.e., models of actual logs represented, for instance, as three-dimensional scans or circular/elliptical cross-sections. Given the model of a sawmill describing the configuration of the equipment, the simulator virtually processes each virtual log and generates the basket of products resulting from its breakdown. Sophisticated simulators take into account the machinery used in the mill design along with their configuration. They also take into account the physical characteristics of the log and the physical constraints of the sawing process.

Sawmill design, a strategic level decision-making problem, is the typical application of sawing simulators. In fact, most sawing simulators are especially designed for this purpose. The simulation of multiple scenarios is run off-line and, differing from wood
allocation, the total running time of each simulation is not of great importance. Examples of such research are found in the work of Orbay (1984), Verret (1997), Tong and Zhang (2006), Lundahl (2007), Liu et al. (2007), Goulet (2007).

The sawing simulator we use in this research for comparison with the machine learning models is Optitek (FPInnovations, 2014). Optitek is a commercial sawing simulator based on the work of Sampson (1990), Renaudl (1992), Tremblay (1993), Légaré (1994), and Hébert et al. (2000). This simulator was chosen because of the availability of sawing models for all the sawmills involved in our experiments.

2.2.2. Simulation and optimization for Wood Allocation

In simulation and optimization planning systems, an optimization module is fed from the output of one or multiple simulations. The optimization models are solved to optimality to provide a recommended plan. In our wood allocation planning problem context, the Optitek simulator (FPInnovations, 2014) along with FPInterface (FPInnovations, 2016) can be used to obtain the low-level data required to formulate a MILP model (Jerbi et al., 2012; Morneau-Pereira et al., 2014). The optimization model is then solved to obtain a complete wood allocation solution. This is a particular way of combining simulation and optimization where simulation is used a priori to generate the data required for optimization (Ladier et al., 2014).

The LogiLab optimizer (Jerbi et al., 2012) we use in this research was developed in an effort to integrate the different planning problems of the supply chain in the forest-products industry. The output of two commercial simulators, FPInterface (FPInnovations, 2016) for the harvest and Optitek (FPInnovations, 2014) for the sawing, can be combined in the optimization model of LogiLab to obtain the data required for the wood allocation planning problem (Morneau-Pereira et al., 2014). Figure 3 presents an overview of the interactions between the simulators and LogiLab optimizer. Each cutblock is represented in the model by its own stand table (proportion per species and diameters). FPInterface is able to determine the type of logs resulting from the application of a specific bucking pattern at a given cutblock. The logs harvested from the forest are then grouped into categories based on their species and their size. The Optitek simulator, along with a suitable sawmill model, can be used to process the logs and to determine the result of the transformation at each sawmill, including the basket of products.
The data from the simulations provides the required parameters for the MILP model proposed by Jerbi et al. (2012) which is, in our case, the industrial wood allocation planning problem to solve. The sawing simulator is launched on all log and sawmill pairs in the definition of the problem in order to obtain the log and lumber pairs required as input of LogiLab. Problem-specific constraints have been added to the model to better reflect the operational constraints of an industrial partner. The specific MILP solved by the optimizer in the context of our industrial application is presented in Appendix A.

2.3. Metasimulation

Simulation of the sawing has tremendous computational costs with large volume of wood and logs especially when considering that large companies around the world can handle a volume of more than 5 million cubic meters of wood a year (Nylander and Friberg, 2007). Even when considering a single sawmill, the volume of wood processed per year can easily be close to a million cubic meters. Although sampling could be performed, the required computational time is still an issue for companies harvesting in multiple cutblocks and operating multiple sawmills. Metamodeling is often used in the literature to reduce the computational costs of simulation (Wang and Shan, 2007). Besides reducing computational costs, metamodels are used to gain insights on a phenomenon at a low cost (design space exploration), to enhance a problem formulation, and as optimization support (Wang and Shan, 2007).

In general, simulators can be considered as a black-box function $f$ from an input space $\Theta$ to a response space $\mathcal{Y}$ (Banks et al., 2010). The goal of a metamodel, which is a simpler mathematical relationship, is to approximate $f : \Theta \rightarrow \mathcal{Y}$ (Banks et al., 2010). Metamodels can be used for optimization. They are either global or local (Barton and Meckesheimer, 2006). A global metamodel is fitted once whereas a local metamodel is fitted as the optimization progresses. Regression analysis, Kriging interpolation, radial basis functions and neural networks are among the usual used metamodeling techniques (Barton and Meckesheimer, 2006; Kleijnen, 2015). We find, among other recently used metamodeling techniques, genetic programming (Can and Heavey, 2012), Bayesian networks (Poropudas et al., 2011), and least square regression (Salemi et al., 2012). These methods share common ground with statistical (machine) learning techniques. Complete surveys on metamodeling can be found in the work of Simpson et al. (2001), Shan and Wang (2010), and Razavi et al. (2012).

3. Building Models of Sawmills

The methodology we propose for our industrial wood allocation planning problem has two phases. First we build a model of each sawmill using machine learning. Second we use the models to gener-
ate the data required by the optimizer to feed the wood allocation MILP model and find an optimal solution. This solution to the MILP based on approximate data can be used to make decision.

This section is divided in four subsections. We first present an overview of important concepts in machine learning in Section 3.1. We present the machine learning algorithms we use in Section 3.2. We detail the data formatting and the training phases in Section 3.3. Finally, in Section 3.4, we discuss the metrics we use to evaluate the performance of a machine-learning based model in terms of accuracy before using it for decision-making.

3.1. Machine Learning Concepts

Machine learning algorithms aim at approximating a phenomenon from data observation (Bakır et al., 2007). Therefore, it can be seen as a general approach for which metamodeling is a specific application where the phenomenon of interest is the simulator. More precisely, in the supervised learning setting, the goal is to learn a predictor \( h : \mathcal{X} \rightarrow \mathcal{Y} \) that accurately predicts the output \( y \in \mathcal{Y} \) of an input \( x \in \mathcal{X} \). To do so, the learner (or learning algorithm) has access to a training set of \( m \) examples \( S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \in (\mathcal{X} \times \mathcal{Y})^m \), where \( x_i = [x_{i1}, x_{i2}, \ldots, x_{in}] \) is the feature representation (or feature vector) of the example \( i \), and \( y_i \) is the corresponding output. We assume the examples to be drawn iid (independently and identically distributed) from an unknown distribution \( D \), meaning they are generated using the same distribution (identically distributed) and the occurrence of any example does not affect the probability of encountering any other example (independently distributed). This hypothesis gives theoretical guarantees on the behavior of the learning algorithms, although it might be too strong in particular applications.

In our setting, the supervised learning problem corresponds to the prediction of the output basket \( y \) given an input \( x \). The input vector \( x \) is the vector of the main characteristics of a log that, when transformed at a particular plant, gives basket \( y \). The iid assumption is equivalent to assuming that the logs in our datasets are randomly selected from a specific forest and that the presence of a specific log in the dataset does not affect the probability of encountering any other given log.

In general, the goal of the learning algorithm is to find a predictor \( h \), among all predictors in \( \mathcal{H} \), achieving the lowest expected loss (minimum number of errors):

\[
r(h) = \mathbb{E}_{(x, y) \sim D} l(h(x), y),
\]

where \( l : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R} \) defines the loss incurred when the prediction \( h(x) \) differs from the observed value \( y \). Since the distribution \( D \) generating the examples is unknown, the expected loss can be estimated with the empirical loss \( r_{emp}(h) \) on the training set.
\[ r_{emp}(h) = \frac{1}{m} \sum_{i=1}^{m} l(h(x_i), y_i). \] (2)

It is important to point out that a learning algorithm that only minimizes the empirical loss can result in over-fitting. Over-fitting occurs when the predictor approximates the training set so perfectly and in such a complex fashion that generalization to new examples is no longer possible (Bishop, 2006). Over-fitted predictors usually have a poor performance on new data. For that reason, multiple learning algorithms minimize a regularized empirical loss that compromises between the performance of the predictor on the training data and its complexity. Moreover, the performances of a predictor are evaluated on an unseen testing set to ensure it can accurately predict the output value of new examples.

Since the output space \( Y \) is arbitrary, supervised learning can be applied to a wide variety of tasks, namely classification, regression and structured prediction. The output \( y \) can be a discrete value (e.g., a category of a finite set for classification tasks), a real value for regression tasks, or a structure like a sequence, a list or a tree for structured output problems. Thereby, the prediction of the basket \( y \) given a log \( x \) can be seen as a structured prediction problem, where the output basket is represented as a list of products. We can reduce the basket prediction problem to a classification task, where each class corresponds to a basket in the training set. In this setting, the predictor \( h \) would only be able to output one of the previously seen baskets. Note that this could be problematic since the correct output of a new log might not be in \( S \). For these reasons we present both classification and structured output algorithms for learning.

In our wood allocation planning problem context, the training phase consists in fitting the predictor whereas the prediction phase consists in approximating the sawmill’s output on a given input.

### 3.2. Learning Algorithms

We used four learning algorithms to train (fit) a predictor (model) of a given sawmill. Section 3.2.1 describes the \( k \)-nearest neighbors. Sections 3.2.2 and 3.2.3 describe tree-based classification algorithms (decision tree and random forest). Section 3.2.4 describes the kernel ridge regression algorithm for structured prediction.

#### 3.2.1. \( k \)-Nearest Neighbors Classifier

The \( k \)-nearest neighbors (\( k \)-NN) algorithm (Fix and Hodges Jr, 1951) predicts the output class of a new example by averaging the vote of the \( k \) closest examples in the training set. The training phase consists of storing the training examples in a search-efficient data structure and of selecting the number \( k \) of examples to consider for prediction. Thereby, the \( k \) closest training examples of a new example can be found using a distance function like the Euclidean distance over their feature vectors, i.e., over
the known inputs of the training set. The prediction of the output class $y$ of an unknown example $x$ is often done by taking the majority class among the $k$ nearest examples found. An alternative approach is to weight each of the $k$ nearest examples according to their distance from $x$. In our context, the known logs, i.e., the logs from the training set, are compared to an input log and a basket is outputted according to one of the techniques described above.

3.2.2. Decision Tree Classifier

The decision tree (DT) algorithm (Breiman et al., 1996) learns a predictor in the form of a tree structure. This tree consists of decision nodes and leaf nodes. A decision node contains a branching rule based on a single feature (feature $\leq$ value) which decides if an example $x$ should take the left or the right path of the tree. A leaf node contains the value to predict when $x$ reaches it.

In the training phase, the tree is constructed in a greedy fashion. The first branching rule is the one that gives the best partitioning of the training examples according to some metric. One of the most commonly used metrics is the gini impurity that computes the probability of an example to be misclassified given the branching rule. Next, the training examples take the left path of the tree if they respect the chosen rule, and the right path otherwise. A new decision node is then created on each path by using only the examples reaching that node. This greedy process stops, thus creating a leaf node, when all the examples reaching a node belong to the same class, or as soon as a specified condition to avoid over-fitting is met, e.g., maximum depth, maximum number of nodes. Finally, the value of a leaf node is set to the majority class of the training examples reaching it.

In the testing phase, the output class of an unseen example $x$ is predicted by passing this example through the tree until a leaf is reached. The value of the reached leaf is predicted.

3.2.3. Random Forest Classifier

The random forest (RF) algorithm (Breiman, 2001) is similar to the DT algorithm, but it creates $B$ trees instead of only one. In the training phase, $B$ new datasets of $m$ examples are created by sub-sampling the training examples uniformly with replacement. A tree is then learned on each of the $B$ datasets. In contrast with DT, only a random subset of features is used to select the branching rule of each decision node. The output class $y$ of an unseen example $x$ can be predicted by passing the example through each tree and returning the most frequently predicted class. This bagging approach (Bishop, 2006) has a tendency to reduce over-fitting.

3.2.4. Structured Kernel Ridge Regression

Given an input $x$, the structured kernel ridge regression (KRR) algorithm predicts a vectorial representation of the corresponding output $y$, and then transforms the predicted vector into a structure of
the output space \( \mathcal{Y} \) (Cortes et al., 2007). More formally, let \( \phi_X : \mathcal{X} \rightarrow \mathcal{H}_X \) be a function that maps an input \( x \) into a high-dimensional vector space \( \mathcal{H}_X \), and let \( \phi_Y : \mathcal{Y} \rightarrow \mathcal{H}_Y \) be a function that maps an output \( y \) into a high-dimensional vector space \( \mathcal{H}_Y \). The goal of the learning algorithm is to find a linear operator \( W \) that transforms a vector of \( \mathcal{H}_X \) into a vector in \( \mathcal{H}_Y \). Given the predicted vector \( W\phi_X(x) \in \mathcal{H}_Y \), the corresponding output in \( \mathcal{Y} \) is predicted by finding the closest \( y \in \mathcal{Y} \) once mapped into the \( \mathcal{H}_Y \) space:

\[
h(x) = \arg\min_{y \in \mathcal{Y}} ||\phi_Y(y) - W\phi_X(x)||. \tag{3}
\]

When predicting the output of sawing, \( W\phi_X(x) \in \mathbb{R}^p \) represents the frequency of each of the \( p \) products in the basket when the input log \( x \) is sent to a sawmill. However, since the predicted product counts are regression values, \( W\phi_X(x) \) is an unrealistic basket. The conversion of \( W\phi_X(x) \) into the \( \mathcal{Y} \) space is called the pre-image problem and is often computationally expensive. We first describe how to learn \( W \) and then show how we solve the pre-image problem in our case.

Let \( X \) and \( Y \) be the input and output matrix respectively, where \( X_i = \phi_X(x_i) \) and \( Y_i = \phi_Y(y_i) \). By minimizing a regularized \( l^2 \) loss, Cortes et al. (2007) shows that the predictor \( W \) is obtained in the following way:

\[
W = Y(X \cdot X^T + \lambda I)^{-1}X^T, \tag{4}
\]

where \( \lambda \) is a parameter controlling the penalty on the complexity of the predictor, \( I_{m \times m} \) is the identity matrix, \((-)^{-1}\) is the inverse matrix function, and \((-)^T\) is the transposed function. Note, however, that there is no need to consider the vectors \( \phi_X(x) \) and \( \phi_Y(y) \) (which can be of infinite dimensions), but only scalar products of the form \( \phi_X(x) \cdot \phi_X(x') \) and \( \phi_Y(y) \cdot \phi_Y(y') \). In many cases, there exists a function \( k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \) that emulates the scalar product \( \phi_X(x) \cdot \phi_X(x') \) by taking as input any \( x \) and \( x' \), and returning a real value representing how similar \( x \) and \( x' \) are in \( \mathcal{H}_X \). We call such function a kernel. Hence, we need two kernels, to emulate respectively the scalar product \( \mathcal{H}_X \) and \( \mathcal{H}_Y \). In our experiments, we considered the simplest kernel for the output by using \( \mathcal{H}_Y = \mathcal{Y} \), but an output kernel that encodes specific properties of the sawmill basket production problem could also be designed. For the input kernel, we used the radial basis function (RBF) (Bishop, 2006; Ben-Hur and Weston, 2010).

The vector \( W\phi_X(x) \) can now be predicted with:

\[
W\phi_X(x) = Y(K + \lambda I)^{-1}k_x^T, \tag{5}
\]

where \( K \) is an \( m \times m \) matrix with \( K_{ij} = k(x_i, x_j) \), and \( k_x = \{k(x_1, x), \ldots, k(x_m, x)\} \). Once the vector \( W\phi_X(x) \) is predicted, the corresponding output \( h(x) \) can be predicted by finding the basket, over all possible ones, that is the closest to \( W\phi_X(x) \). This could easily be done by rounding the values of \( W\phi_X(x) \) to the nearest integer.
3.3. Formatting the Data and Training the Models

Although we could use the complete scan of a log as input for training, the input space can be reduced by identifying the important characteristics of the logs. We chose the following characteristics which are commonly used by the forest-products industry to describe a log, as our input features representing the logs:

- species;
- volume (dm\(^3\));
- length (m);
- wide-end diameter (cm);
- narrow-end diameter (cm);
- curvature (cm/m); and
- shrinking (cm/m) (a measure of how fast the log shrinks from its wide end to its narrow end).

The input vector has a total of \( n = 9 \) features in our industrial case.\(^1\)

Given a log as input, the output is the basket of products represented as a vector \( y \in \mathbb{N}^p \) where \( p \) is the total number of lumber products that the sawmill can produce and where each entry corresponds to the amount of lumbers produced for a given lumber product (as defined by its type and its grade). As a consequence of the available and feasible virtual log and sawmill pairs and of the products that a given sawmill can produce, the size of the datasets used for training vary both in the number of rows and columns depending on the sawmill it corresponds to.

We train a different predictor for each sawmill. Therefore, in practice, we need to gather historical data for each sawmill involved in the wood allocation planning problem at hand. In absence of such data, sawing simulations can be performed on a set of virtual logs. Since not every sawmills can physically process all logs, simulations are performed only for feasible pairs of sawmill and log to obtain the lumbers. Note that once the predictor of a given sawmill is learned the simulator is no longer needed.

In our experimental context, the training examples were generated by simulating the sawing process using the Optitek simulator (FPInnovations, 2014) to build a predictor of the simulator’s black box function as this model was available for our case. More details on the experiment setting are given in Section 4.

3.4. Evaluating the Predictive Accuracy of a Learned Model

The predictive (forecast) accuracy of a predictor is to be maximized. There exist multiple metrics to quantify the accuracy of a predictor on a given example (or example set) for which we know the output(s). A common evaluation metric for the classification of a single example is the zero-one score.

---

\(^1\)Each species is modeled as a different 0/1 feature in the input vector for a total of three features in our specific industrial case. We avoid using a single integer value for the species since it would introduce an artificial order relation on the species.
which attributes a score of 1 if the predictor is right and a score of 0 if it is not. The zero-one score $s^z$ is a binary score rewarding 100% correct predictions only (Bakır et al., 2007). This is insufficient in our context since a wood-industry company needs to evaluate a prediction both in terms of over and underestimation of the real production. We thus propose two scores specific to sawing simulation to account for finer prediction errors. We call the first the production ratio $s^{pro}$ and the second the prediction ratio $s^{pre}$ (see Definitions 1 and 2).

**Definition 1 (Production).** Let $\epsilon$ be a small positive value. Given $\hat{\mathbf{y}} \in \mathbb{N}^p$ and $\mathbf{y} \in \mathbb{N}^p$ where $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_p]$ is the predicted output vector and $\mathbf{y} = [y_1, y_2, \ldots, y_p]$ is the real output vector, the production ratio $s^{pro}$ is the average bounded ratio of the real production on the predicted production:

$$s^{pro}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{p} \sum_{j=1}^{p} \min \left( 1, \frac{\max(y_j, \epsilon)}{\max(\hat{y}_j, \epsilon)} \right). \quad (6)$$

**Definition 2 (Prediction).** Let $\epsilon$ be a small positive value. Given $\hat{\mathbf{y}} \in \mathbb{N}^p$ and $\mathbf{y} \in \mathbb{N}^p$ where $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_p]$ is the predicted output vector and $\mathbf{y} = [y_1, y_2, \ldots, y_p]$ is the real output vector, the prediction ratio $s^{pre}$ is the average bounded ratio of the predicted production on the real production:

$$s^{pre}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{p} \sum_{j=1}^{p} \min \left( 1, \frac{\max(\hat{y}_j, \epsilon)}{\max(y_j, \epsilon)} \right). \quad (7)$$

The production ratio $s^{pro}$ corresponds to the percentage of predicted products that are effectively produced. The prediction ratio $s^{pre}$ corresponds to the percentage of the real production that is faithfully predicted.

We also propose an aggregated performance metric we call the production and prediction score (see Definition 3). This metric corresponds to the area of the region we obtain when plotting $s^{pre}$ against $s^{pro}$ for a given prediction. We use the expected $s^{pro \times pre}$ value over the sample as our principal quality indicator of a predictor $h$.

**Definition 3 (Production and prediction).** Given a prediction $\hat{\mathbf{y}} \in \mathbb{N}^p$ and an output $\mathbf{y} \in \mathbb{N}^p$ where $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_p]$ is the predicted output vector and $\mathbf{y} = [y_1, y_2, \ldots, y_p]$ is the real output vector, the production and prediction area $s^{pro \times pre}$ is:

$$s^{pro \times pre}(\hat{\mathbf{y}}, \mathbf{y}) = s^{pre}(\hat{\mathbf{y}}, \mathbf{y}) s^{pro}(\hat{\mathbf{y}}, \mathbf{y}). \quad (8)$$

The baskets of products obtained from the breakdown of a log are usually sparse meaning that they contain a small number of products. To avoid overestimating the quality of a prediction, it is convenient to filter the predicted output vector and the example output vector prior to evaluation by removing the products for which both equals zero.
4. Experiments

The goal of the experiment is to demonstrate the decision-making capabilities of the models (predictors) we built using a machine learning approach. In the experiments, we first select the most accurate predictors (Section 4.1) and then compare the wood allocation planning decisions made when using them to the best possible wood allocations (Section 4.2). For the purpose of the experiments, the best possible wood allocations are obtained when using the MILP built from the sawing simulation data instead of a MILP built using the machine learning data (the latter approximates the former).

4.1. Building Models of Sawmills and Evaluating Their Accuracy

We first build a set of predictors using the four machine learning algorithms we presented in Section 3.2: the k-NN algorithm, the DT algorithm, the RF algorithm, and the KRR algorithm. Our real industrial wood allocation case involves a total of five different sawmills (sawmills A to E) leading to a total of five different datasets for learning:

- sawmill A can produce 72 different products and 1754 logs from our training sets are compatible with this sawmill;
- sawmill B: 69 products and 1867 logs;
- sawmill C: 35 products and 3113 logs;
- sawmill D: 83 products and 1754 logs; and
- sawmill E: 38 products and 3113 logs.

Each sawmill was modeled using the Optitek simulator (FPIInnovations, 2014). The output of every sawmill for every log of every dataset was computed using the simulator leading to a total of five datasets for learning. In practice, one could use the historical production data of each of the five sawmills to build the five predictors needed to feed the optimizer with the predicted lumber data of unseen logs. In our experiments’ context, every log of every dataset is needed to reproduce the actual industrial problem instance. We thus build two predictors per dataset to reproduce the data while avoiding presenting seen input data to any given predictor while using them for output data generation for optimization purposes. That is, during the training phase, a predictor is trained using 50% of the data and the remaining 50% is used for testing its accuracy and to generate the unseen logs for optimization purposes. Since the entire dataset needs to be reproduced, we then switch the training and the test set to train a second predictor. This second predictor is used to predict the baskets for the first 50% of the data that we used to train the first predictor. Using this method, we are able to generate the data required by LogiLab without overestimating the efficiency of the machine learning algorithms, i.e., without presenting known (training) data to

---

2Predictors usually have a high accuracy on examples from training set, e.g., a k-NN-based predictor is likely to have 100% accuracy on these examples.
Table 1: All datasets average accuracy with a margin of error of the 95 % confidence interval; the best average results are highlighted in bold

<table>
<thead>
<tr>
<th></th>
<th>Base case</th>
<th>Learning algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hist. Average</td>
<td>DT</td>
</tr>
<tr>
<td>$s_{\text{pro}}$ (prediction)</td>
<td>.30 ± .028</td>
<td>.69 ± .008</td>
</tr>
<tr>
<td>$s_{\text{pro}}$ (production)</td>
<td>.82 ± .030</td>
<td>.73 ± .009</td>
</tr>
<tr>
<td>$s_{\text{pro} \times \text{pre}}$</td>
<td>.19 ± .010</td>
<td>.52 ± .011</td>
</tr>
</tbody>
</table>

Figure 4: 95 % confidence intervals around the average production and prediction area ($s_{\text{pro} \times \text{pre}}$) for all sawmills and each learning algorithm

We performed the training phase on 10 different partitions of the data as described above leading to a total of 20 predictors per algorithm per sawmill.

Table 1 shows the average accuracy and the 95 % confidence interval for all accuracy metrics and for all machine learning algorithms. The base case results (first column) are the results we obtain by using a predictor that always predicts the average basket of products in the training set. We can see that the base case results are the highest in terms of production ratio ($s_{\text{pro}}$) score. This is easily explained by the fact that the baskets of products are sparse leading to a low average value for most products. Considering the production ratio ($s_{\text{pro}}$) or the prediction ratio ($s_{\text{pre}}$) score alone only gives information whether a predictor has a tendency to underestimate or overestimate the production.

Figure 4 presents graphically the 95 % confidence intervals around the average production and prediction area ($s_{\text{pro} \times \text{pre}}$) for each sawmill and learning algorithm. By inspecting the figure we notice that two algorithms stand out: the RF algorithm and the DT algorithm. The confidence interval of any of these algorithms is disjoint from that of k-NN and KRR meaning that the difference in performance is statistically significant. There is, however, a slight overlap in between the confidence intervals of the RF algorithm and the DT algorithm.

Figure 5 presents the 95 % confidence intervals around the average production and prediction area ($s_{\text{pro} \times \text{pre}}$) for each sawmill and learning algorithm. We observe the tendency of the RF algorithm and of the DT algorithm to outperform the others on a per sawmill basis which confirm our previous observation on the aggregated case depicted on Figure 4. We also observe that, of all datasets, the datasets

---

Figure 5

---

\[ \text{Figure 4: 95 % confidence intervals around the average production and prediction area} \]
of sawmill C and E are the easiest for the learning algorithms and the dataset of sawmill D is the hardest. It supports our hypothesis that a larger number of products (output columns) and that a lower number of examples (rows) render the learning harder. We retained the models built using the RF algorithm and the DT algorithm for the second phase.

4.2. Performance of the Learned Models for Wood Allocation Planning

To evaluate the performance of the company when making decisions based on the predictors’ data, we compared such decisions to the ones that could have been made with the ground-truth model built using the simulated data from Optitek. In the previous section, we built a total of 10 pairs of predictors per learning algorithms each leading to a single MILP with approximate parameters. To a single optimal solution to one of those MILP corresponds wood allocation planning decisions. Reevaluating those decisions in the context of the ground-truth model basically means recomputing the objective value using the model ground-truth parameters.

A common practice in industry is to use the historical production data and to allocate the wood using the average basket of products. In our context, it means taking the average number of lumbers per product type for each sawmill. This is our baseline for comparison. The best that could be done with the data is to allow LogiLab to use the data generated by Optitek. Therefore, the objective function value of the optimal solution obtained when using this data corresponds to the maximal gain we could obtain with a perfect forecast of the production.

When using the data provided by the predictors for decision-making (i.e., when simulating those decisions in our context) the objective value we obtain should lie between those two extremes in order for machine learning to have a benefit over the histor-
Figure 6 presents a SMORE plot (Sturrock and Pegden, 2011) of the statistics of the relative performance gains in terms of the percentage of improvement of the objective value with respect to the historical average approach over 10 runs of the optimizer with the retained predictors. The average is represented as a diamond and the 95 % confidence interval around it as a blue-shaded box. The 1st and 3rd quartiles, and the median are represented by the rectangle and the middle line respectively. The whiskers are drawn from the minimum value (resp. 3rd quartile) to the 1st quartile (resp. maximum value). The baseline (historical average) has a gain of 0 %. The average gain we obtained with the RF-based predictors is 55 % (with a min of 37 % and a max of 68 %) whereas the average gain we obtained with the DT-based predictors is 83 % (with a min of 69 % and a max of 94 %). Table 2 presents the exact numerical values.

Although a high accuracy was hard to achieve on our sawing datasets since they have, by current machine learning standards, a small number of learning examples (the datasets have between 35 and 83 products on columns, the three of them with the most columns have under two thousand virtual logs on rows, see Section 4.1), our predictors achieve up to 94 % of the maximum theoretical gain which confirms their efficiency for wood allocation on our industrial instance. This is one situation where imperfect information about the situation nevertheless helps a lot in the decision-making phase. Such an improvement, when considered in terms of benefits for a company, translates into gains of tens of million of dollars when compared to a more traditional heuristic (i.e., our baseline). Indeed, achieving in average 83 % of the maximum gain (with DT) when considering only nine easy-to-gather log characteristics is of interest for the industry.

<table>
<thead>
<tr>
<th>Base case</th>
<th>Learning algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hist. Predictor</td>
<td>DT</td>
</tr>
<tr>
<td>CI at 95% around mean</td>
<td>1.96% ± 0.23%</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>1.16 %</td>
</tr>
<tr>
<td>Minimum</td>
<td>0.00 %</td>
</tr>
<tr>
<td>Maximum</td>
<td>3.90 %</td>
</tr>
<tr>
<td>Median</td>
<td>1.85 %</td>
</tr>
</tbody>
</table>
5. Conclusion

We first described the wood allocation planning problem and the context of its formulation. We discussed how the data required for the formulation of such a problem can be obtained by simulating the sawing process. We proposed a machine learning approach to build models of the sawmills in order to simplify the data computation phase of the wood allocation planning problem and proposed problemspecific accuracy metrics to help in choosing models for decision-making. To use a learned model for decision-making in the context of wood allocation, we first built models for each sawmill in our real industrial wood allocation planning problem instance and evaluated their accuracy. The models learned for our industrial instance achieved up to 94% of the maximum theoretical gain in a decision-making context. In the studied case, it is impossible to surpass the performance of the simulator’s models since they were used for learning. Still, the machine learning approach leading to the greatest gain, decision tree (DT), achieves an average gain of 83% with respect to a more usual method based on the historical production.

Despite the performance of the proposed approach with respect to our wood allocation planning problem, concerns on the applicability of a proposed solution in practice might arise. In this regard, the solution provided when feeding an optimization model using machine learning-based models is no different than the one provided when feeding the same optimization model using data computed by simulation. Both approaches are approximate. As a result, the recommendation provided by solving an optimization model for wood allocation is, in practice, used as a general guideline for the wood flow proportions allocated from cutblocks to sawmills (Jerbi et al., 2012).

It should also be mentioned that the approach, in its current form, is offline. This means that the learning and prediction phases, consisting respectively in generating the models and in using them to compute the data for optimization, are performed prior to optimization and in place of simulation. Further benefits could potentially be achieved by tying these phases more closely to the optimization and simulation processes. By integrating the models to an optimizer, for instance, we could perform only the necessary simulations. Similarly, the machine learning-based models could be used to improve the performance of the simulator.

The latter is one of the envisioned benefits of our method within a sawmill exploitation context. In this context, the learned models can be used as metamodels to predict the outcome of a simulation although it just started thus further improving the decision-making capabilities of the companies having access to a sawing simulator. Even for sawmill design, which is considered as long-term planning, such a capability could be used to provide meaningful insights to the engineers designing the plant leading to shortened decision time for companies in
the forest-products industry and increased benefits.

An additional research avenue that has not yet been covered by the current project includes the application of the approach to other problems from the forest-product industry. Exploiting the big data currently generated by the forest-products industry (e.g., three-dimensional log scans and historical production data) is perhaps, one of the greatest benefits of using machine learning-based models instead of a simulator. By simplifying decision-making for a variety of problems, the approach adds value to the historical data collected but not used by many companies. In the labor shortage context the forest-product industry is facing, an automated modeling approach not only simplifies the human task, it increases the competitiveness of the company by letting the decision-makers access information to generate and explore solutions they lack the workforce and time to produce. We studied the wood allocation problem, but our machine learning-based models are general. They could be used, just as a sawing simulator, to compute the data required to model other types of problems important to the decision-makers such as sawmill design and configuration. This generality is appealing as the benefits we observed for the wood allocation problem could be observed in other contexts at various decision-making levels (from online to strategic).

Acknowledgement

The authors would like to thank the FORAC Research Consortium and its partners, especially FPInnovations for the data related to the Optitek simulator. We would also like to thank Alexandre Morneau for his support with LogiLab. Our gratitude goes as well to the Natural Sciences and Engineering Research Council of Canada (NSERC) who provided funding for this research. This research has also been funded by Fonds de recherche du Québec – Nature et technologies (FRQ-NT).

Part of the computations were made on the supercomputer Colosse from Université Laval, managed by Calcul Québec and Compute Canada. The operation of this supercomputer is funded by the Canada Foundation for Innovation (CFI), the Ministère de l’économie, de la science et de l’innovation du Québec (MESI), and the Fonds de recherche du Québec – Nature et technologies (FRQ-NT).

References

Marinescu, M. V., Sowlati, T., Maness, T. C., Oct. 2005. The development of a timber allocation model using data en-


Appendix A. Wood Allocation Model

The MILP we used is the model generated by LogiLab for our industrial wood allocation planning problem instance as described in (Jerbi et al., 2012). Our industrial instance leads to a single period problem. The flexibility of LogiLab enabled us to adapt the model to the practical case to solve by providing additional constraints. The model we present further constraints the number of sawmills that can be allocated to a given cutblock based on the harvest volume at that block.

As a convention, we represent parameters and indices in lower case italic letters and variables or variables in uppercase capital letters. Sets are represented in calligraphic font.

Parameters and variables. We first define the following sets to clarify the definition of the model:

- the set $\mathcal{U}$ of business units: sorting yards or sawmills in our context;
- the set $\mathcal{P}$ of products, e.g., a specific type of lumber product;
- the set $\mathcal{W}$ of processes, e.g., the harvest at a cutblock, the inventory of a sorting yard, the transformation process at a sawmill;
- the set $\mathcal{K}$ of capacities of the processes, e.g., the inventory capacity of a sorting yard and the transformation capacity of a sawmill’s equipment;
- the set $\mathcal{E}$ of flow lines between the business units.

We use three sets of variables over real numbers to describe the wood flow in the model:

- The $Y_{uw}$ variables express the amount of process $w$ performed at unit $u$ ($\forall w \in \mathcal{W}, u \in \mathcal{U}$).
- The $D_{up}$ variables are the amount of product $p$ from business unit $u$ sold ($\forall u \in \mathcal{U}, p \in \mathcal{P}$).
- The $F_{ep}$ variables are the amount of product $p$ transported on the flow line $e$ ($\forall e \in \mathcal{E}, p \in \mathcal{P}$).

Wood allocation is primarily concerned by the flows from cutblocks to sawmills (quantities and transport mode). Additional binary variables, the $L_{uw}$ variables, are used to constrain the transportation of products from a business unit $u \in \mathcal{U}$ to a business unit $u'$ different than $u$, i.e., $u' \in \mathcal{U} - u$. 

Objective. Let $\mathcal{W}_u \subseteq \mathcal{W}$ be the set of processes performed at unit $u$. Let $d_{up}$ be the demand for product $p$ at unit $u$. Let $\rho_{up}$ be the value of product $p$
at unit $u$. Let $c_w$ and $c_{ep}$ be the cost of process $w$ and the transportation cost for product $p$ on flow line $e$ respectively. The objective is to maximize the benefits defined as

$$\max \sum_{u \in \mathcal{U}} \sum_{p \in \mathcal{P}} \rho_{up} D_{up} - \sum_{u \in \mathcal{U}} \sum_{w \in \mathcal{W}_u} c_w Y_{uw} - \sum_{e \in \mathcal{E}} \sum_{p \in \mathcal{P}} c_{ep} F_{ep} \quad (A.1)$$

subject to constraints (A.2) to (A.12). The first term of the objective function corresponds to the gross profit of the sales. The second and the third terms represent the production and the transportation costs respectively.

**Constraints.** Let $\gamma_{pw}$ be the amount of product $p$ produced by process $w$. Let $\alpha_{pw}$ be the amount of product $p$ consumed by process $w$. Let $\mathcal{E}_{in}^u \subseteq \mathcal{E}$ be the set of flow lines entering business unit $u$. Let $\mathcal{E}_{out}^u \subseteq \mathcal{E}$ be the set of flow lines leaving business unit $u$. We add the following flow conservation constraint:

$$\sum_{w \in \mathcal{W}_u} \lambda_{kuw} Y_{uw} \leq \tau_{kuw}, \quad \forall u \in \mathcal{U}, k \in \mathcal{K}. \quad (A.4)$$

$$\lambda_{e} \leq \sum_{p \in \mathcal{P}} F_{ep} \leq \tau_{e}, \quad \forall e \in \mathcal{E}. \quad (A.5)$$

$$\lambda_{e} \leq F_{ep} \leq \tau_{e}, \quad \forall p \in \mathcal{P}, e \in \mathcal{E}. \quad (A.6)$$

This completes the definition of the model without additional constraints.

**Additional constraints.** We now describe the additional constraints added to control the harvest method used on a cutblock and the allocation of the cutblocks to sawmills. There are three different harvest methods considered in the model. Cutblocks can be harvested tree-length or log-length depending on their volume. Harvested trees (stems or logs) are grouped into piles according to their species and harvest method. Each pile of a cutblock of a sufficient volume can be sent to up to two sawmills. The piles of the smaller cutblocks are sent to a single sawmill. Furthermore, the total volume harvested tree-length and log-length is subject to additional constraints.

Let $\mathcal{U}^{\text{block}} \subseteq \mathcal{U}$ be the set of business units that
are cutblocks, and \( \mathcal{I} \) be the set of piles. We denote by \( U_{\text{small}} \) the cutblocks with a low volume and \( U_{\text{large}} \) the cutblocks with a large volume. Let \( \mathcal{P}_i \) be the set of products in pile \( i \), and \( M \) be a sufficiently large number. Constraint (A.7) makes sure that the flow from business unit \( u \) to business unit \( u' \) is null when \( L_{uw'} \), i.e., the binary variables representing non null flow from business unit \( u \) to business unit \( u' \), is null.

\[
\sum_{u \in U} \sum_{u' \in U, u' \neq u} \sum_{p \in \mathcal{P}_i} F_{e,u} \leq ML_{uw'}, \quad \forall u \in U_{\text{small}}, u' \in U, i \in \mathcal{I}.
\]  

(A.7)

Constraints (A.8) to (A.9) enforce that the number of sawmills that can be allocated to a cutblock depends on the cutblock possible harvest volume and are expressed as follows.

\[
\sum_{w' \in W} L_{uw'} \leq 1, \quad \forall u \in U_{\text{small}}. \quad \text{(A.8)}
\]

\[
\sum_{w' \in W} L_{uw'} \leq 2, \quad \forall u \in U_{\text{large}}. \quad \text{(A.9)}
\]

Let \( W_{\text{tree}} \subseteq W \) be the set of processes using tree-length products. Let \( W_{\text{log}} \subseteq W \) be the set of processes using log-length products. Let \( U_{\text{block}} \) be the set of cutblocks that need to be harvested log-length due to their volume. Let \( \eta_{\text{tree}} \) and \( \eta_{\text{log}} \) be the maximal volume of wood that can be harvested tree-length and log-length respectively. Constraints (A.10) and (A.11) enforce the maximum volume (resp. minimum value) of wood that can be harvested tree-length (resp. log-length).

\[
\sum_{p \in \mathcal{P}} \sum_{w \in W_{\text{tree}}} \sum_{u \in U} \gamma_{pw} Y_{uw} \leq \eta_{\text{tree}}. \quad \text{(A.10)}
\]

\[
\sum_{p \in \mathcal{P}} \sum_{w \in W_{\text{log}}} \sum_{u \in U} \gamma_{pw} Y_{uw} \geq \eta_{\text{log}}. \quad \text{(A.11)}
\]

Finally, cutblocks that are harvested log-length cannot be harvested tree-length:

\[
Y_{uw} = 0, \quad \forall u \in U_{\text{log}}, w \in W_{\text{tree}}. \quad \text{(A.12)}
\]