# Predicting makespan in Flexible Job Shop Scheduling Problem using Machine Learning

S. Ehsan Hashemi-Petroodi, Simon Thevenin, Alexandre Dolgui

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-10 (2022) 1–6

# Predicting makespan in Flexible Job Shop Scheduling Problem using Machine Learning

**David Tremblet** * **Simon Thevenin** * **Alexandre Dolgui** *

* *IMT Atlantique, LS2N-CNRS, La Chantrerie, 4 Rue Alfred Kastler, B.P. 20722, 44307 Nantes, France (e-mail: david.tremblet@imt-atlantique.fr, simon.thevenin@imt-atlantique.fr, alexandre.dolgui@imt-atlantique.fr)*

**Abstract:** Machine learning tools have experienced a growing interest in the early 2010s, providing efficient predictive approaches for artificial intelligence and statistical analysis. These same prediction methods have also sparked interest in the operations research community for decision-making based on predictive analysis by exploiting massive histories and datasets. This study investigates the potential of machine learning tools to predict the feasibility of a production plan. Production schedules are often not able to adhere to the production plans because production plans are built without accounting for all the detailed requirements that arise at the scheduling level. We show that predicting the feasibility of a production plan with a decision tree yields a precision of around 90% versus 70% in the classical capacity constraints considered in planning tools.



*Keywords:* Production planning and scheduling, Artificial Intelligence

## 1. INTRODUCTION

Production planning is a crucial function in manufacturing operation management that aims to decide the production quantity per period and item. The resulting production plan is the basis to accept customer orders, quote the customer's delivery lead time, place raw material and components orders to suppliers, and adjust the production capacity. Production planning is typically a tactical decision, and it is the input for production scheduling at the operational level. Often, the planner discovers during the scheduling that the production capacity is too tight to implement the production plan. While some planning methods such as MRPII (Manufacturing Ressource Planning II) include capacity computation, the computation of the capacity should be accurate. If the capacity consumption is underestimated, the schedule cannot adhere to the plan. These infeasible schedules lead to firefighting, loss of the service level to customers when an order must be delayed, increased production costs when urgent orders must be placed to suppliers or urgent machine setups must be performed.

To ensure adherence to the plan, the capacity consumption has to be approximated for the period, and the planner must check whether the whole production can be handled within the available time or not. However, the consumption of the capacity remains difficult to determine, because it depends on the types of task to be carried out, the quantities to be produced for the period, and the uncertainties met during the production. In this work, we propose a machine learning approach to estimate the feasibility of a plan with given resources, job characteristics, and production lot sizes.

Machine learning approaches are growing in popularity since the early 2010s, and they have reached a good ability to predict values by relying on abstract and aggregated data. The machine learning model proposed in this work aims to be used during manual planning. While there is more and more software to automated the creation of the plan, the human planner remains in charge, and he often modifies the plan manually. The prediction of the available capacity is also valuable to accept production orders and to quote the lead times in make-to-order production systems. Finally, an accurate estimation of the available production capacity per period is valuable to deal with disruptions on the shop floor, which require to manually correct the plan. In this context, the present work provides a tool to determine in real-time if the plan created by the planner is implementable on the shop floor.

This work also aims to serve as a basis for future research, where we would like to investigate the use of the machine learning model as a surrogate model within a lot-sizing problem. The goal is to solve the integrated lot-sizing and scheduling problems where the complex scheduling model is replaced with a machine learning model (Dias and Ierapetritou, 2019). More precisely, we aim to train a machine learning method to predict the capacity consumption of a production period, and to translate this model into a mixed-integer linear program (Biggs and Hariss, 2017). To this end, we focus in this paper on the performance of a random forest classifier for predicting the feasibility of scheduling problem instances.

In this work, we consider that the scheduling problem corresponds to a flexible job shop with sequence-dependent setup times. Our motivation to consider this particular


10.1016/j.ifacol.2022.09.305

job shop environment is its wide popularity in industrial applications (Błażewicz et al., 1996; Shen et al., 2018). In a job shop environment, each job has its routing, where a routing corresponds to successive operations performed by different machine. Due to the rise of flexible manufacturing environments, modern machines have high flexibility. This leads to alternative routings for jobs, where some operations can be performed by more than one machine (Yang et al., 2016; Candan and Yazgan, 2014). Alternative routing offers major improvements to the quality of schedules, and they prevent the whole shop floor from general breakdowns typically induced by failures of one bottleneck machine. In addition, real applications of job-shop scheduling also have to deal with setup times between jobs, which depend on jobs and also on machines in which the changeover occurs. Besides its importance in practice, the flexible job-shop scheduling problem with sequence-dependent setup times (FJSP-SDST) is a generic model, and our work may remain valid for special cases (such as flow shop, simple job shop, ...).

This paper is organized as follows. Section 2 provides a literature review on machine learning approaches for scheduling. The problem treated in this paper and a formulation using constraint programming are defined in Section 3. A description of our approach based on machine learning is provided in Section 4. Section 5 provides the numerical experiments carried out to assess our approach, before concluding in Section 6.

## 2. LITERATURE REVIEW ON MACHINE LEARNING APPROACHES FOR SCHEDULING

There exist a wide variety of application of machine learning in the scheduling literature. The first work to use machine learning in scheduling (Nakasuka and Yoshida, 1992) seeks to predict the best dispatching rule for a given instance. This research area is still active nowadays (Jun et al., 2019; Li and Olafsson, 2005), and researchers aim to tackle more complex environments, such as job-shop scheduling problems, or they try to learn abstract dispatching rules from data obtained through simulation. These approaches can be seen as a pre-processing phase to improve the performance of a heuristic method. Machine learning may also be used as a post-processing phase (e.g., Li et al., 2020). When a feasible schedule is found, the machine learning model predicts if a rescheduling can improve the current solution.

Very few papers consider the use of machine learning models to predict either a value for the makespan of a scheduling problem or the feasibility of a schedule before given deadlines. In batch processing, machine learning based approaches have been the main focus of several papers trying to estimate the makespan. Raaymakers and Weijters (2003) investigate regression and neural networks to predict the makespan in a job-shop environment. Schneckenreither et al. (2020) also investigate this approach using neural networks, and the authors compare the neural network with other methods using simulation. Both papers highlighted the robustness of such models, and the potential of machine learning based approaches in predicting the lead time. In these papers on batch processing, the makespan prediction is used to provide quick answers for orders acceptance. Our study provides further work on machine learning models for the makespan estimation, by investigating this approach in a more complex production environment and using another machine learning model based on random forest. This type of machine learning model have also a prospective usage for the integration of scheduling and production planning problems.

Casazza and Ceselli (2019) and Dias and Ierapetritou (2019) propose approaches based on machine learning to integrate planning and scheduling activities. Casazza and Ceselli (2019) consider an integrated production planning and scheduling problem. The production planning part considers a set of jobs with release dates and due dates, which have to be performed within a given set of periods. Each job can also be split into two, to make the assignments easier. For each period, a scheduling problem with parallel machines is considered and the objective is to find a feasible assignment of jobs to periods that minimizes the total number of splits. Dias and Ierapetritou (2019) propose a similar approach using a discrete state-task network model for the scheduling part and a lot-sizing problem for the production planning. Both papers proposed a model based on machine learning methods to predict the feasibility of each schedule for each period given the corresponding jobs to be performed. The proposed machine learning model is then converted into a set of linear constraints and variables and added to MILP model to substitute the constraints related to the scheduling part. Dias and Ierapetritou (2019) has proven the efficiency of such an approach for solving large-scale instances.

## 3. PROBLEM DEFINITION

### *3.1 Description of the problem*

In this paper, we focus on a hierarchical production planning with a planning and a scheduling level. At the planning level, the planner handles the orders acceptance by assigning a set of orders to a planning horizon divided into $T$ time periods (e.g., weeks or months). The orders are assigned to time periods such that each order is entirely completed by the shop floor before the end of the time period. At the scheduling level, the orders are decomposed into a set of jobs, and these jobs are scheduled such that the makespan (i.e., the completion time of the last performed job) does not exceed the end of the time period. As a consequence, the scheduler has to construct a schedule for the jobs to be performed within each time period. The scheduling problem consider in this paper is a flexible job shop scheduling problem with sequence-dependent setup times (FJSP-SDST).

The FJSP-SDST is a variant of the well-known Job-Shop Scheduling Problem, where a set of $n$ jobs $J$ have to be performed following a routing on a set of $m$ machines $M$. Each job $i$ requires to perform $n_i$ consecutive operations, and we denote by $O_{ij}$ the $j^{\text{th}}$ operation of a job $i$. Each operation $O_{ij}$ can be performed on a subset of machines $M_{ij} \subseteq M$, and its processing time $p_{ijk}$ gives the number of time unit required to process $O_{ij}$ on machine $k \in M_{ij}$. In addition, a setup must be performed on the machine before to process an operation, and we consider a sequence

dependent setup time $s^k_{ii'}$ if job $i'$ is process after job $i$ on machine $k$. We also denote by $D^k$ the setup matrix for machine $k$, i.e., $D^k = (s^k_{ii'})_{1\leq i\leq n, 1\leq i'\leq n}$. The objective is to minimize the makespan $C_{max}$ (i.e., the completion time of the last performed operation).

Consequently, before accepting an order, the planner must determine if the shop floor can handle the production load associated with this order, which is a hard task. The orders acceptance requires to consider $T$ instances of the FJSP-SDST to check if the schedule can accommodate the additional load. Therefore, our paper presents a methodology based on machine learning to predict if all jobs in a FJSP-SDST can be completed within a planning time bucket.

### *3.2 Problem formulation*

Training the machine learning model requires positive and negative examples, and these examples correspond to solutions to the FJSP-SDST. To get these solutions, we use IBM ILOG CPLEX CP Optimizer. The formulation of the FJSP-SDST with the notations and global constraints provided by CP Optimizer (IBM, 2021) is as follow:

**Variables:**

- $I^{ops}_{\{ij\}}$ : Interval variables representing all possible operation choices, for each $j^{\text{th}}$ operation of job $i \in J$,
- $I^{mops}_{\{ijkp\}}$ : Interval variables for each operation, i.e., the $j^{\text{th}}$ operation of job $i \in J$, to be performed on machine $k$ with finite duration $p$,
- $S_{\{k\}}$ : Sequence variable for each machine $k \in M$ for all operation interval variable $I^{mops}_{ijkp}$,
- $C_{max}$ : Makespan to be minimised, i.e. the latest ending time among all operations $j \in n_i$ for all job $i \in J$.

$$\min\ C_{max} \tag{1}$$

s.t.

$$\text{endBeforeStart}(I^{ops}_{\{ij\}}, I^{ops}_{\{i(j+1)\}}) \qquad \begin{matrix}\forall i\in J,\\ \forall j\in 1..n_i-1\end{matrix} \tag{2}$$

$$\text{alternative}\left(I^{ops}_{\{ij\}}, \left\{I^{mops}_{\{ijkp\}} |\ \forall k \in M_{ij}\right\}\right) \qquad \begin{matrix}\forall i\in J,\\ \forall j\in 1..n_i\end{matrix} \tag{3}$$

$$\text{noOverlap}(S_{\{k\}}, D^k) \qquad \forall k\in M \tag{4}$$

$$\max_{\substack{i\in J\\ j\in 1..n_i}} (\text{endOf}(I^{ops}_{\{ij\}})) \leq C_{max} \tag{5}$$

The objective is to minimize the makespan (1). Constraints (2) states that the $j^{th}$ operation of each job $i$ must be performed before its successor $j+1$. Constraints (3) guaranty that each operation $j$ of job $i$ is performed by a machine in its set of possible alternative $M_{ij}$. Constraints (4) ensure that each machine processes at most one operation at a time, and that a setup given by matrix $D^k$ occurs between each pair of jobs. Note that $p$ corresponds to the size of interval variable $I^{mops}_{\{ijkp\}}$, and it is set to the processing time of operation $O_{ij}$ ($p = p_{ijk}$). Therefore, it should not be considered as an index for this variable. Inequality (5) sets the variable $C_{max}$ to the completion time of the last performed job.

## 4. DESCRIPTION OF THE MACHINE LEARNING BASED APPROACH

### *4.1 Tree based model*

In machine learning, a classifier predicts the class of an input described with a set of features $F$. A decision tree for classification is a classifier with a binary tree structure (Breiman et al., 1983). To classify a given input, one starts at the root node and follows the path until reaching a leaf node. Each node of the tree split the search space into the two following subtrees, and the path to follow depends on whether a linear inequality is respected or not. The leaf node provides the prediction. The training of the decision tree requires a dataset $X$ of $N$ samples, where each sample is described by a set of feature values and its outcome. The training algorithm usually finds the best possible splits by minimizing the errors between the targeted values and the prediction.

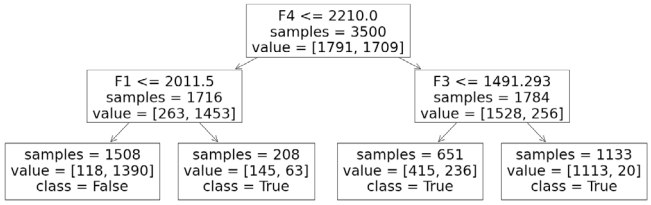


Fig. 1. Example of a single classification tree trained to predict a binary class

Figure 1 provides an example of a decision tree. For each node split, Figure 1 gives the linear condition based on one feature $f \in F$, the number of samples belonging to this node, the number of samples belonging to each class, and, for each leaf node, the class value corresponding to the value predicted by the tree (here binary values `True` or `False`) for any example arriving in this leaf node. To predict the class of a given input data $x$, the method starts at the root node and browses the tree until it reaches a leaf node. At each split node, the prediction method selects the branch according to the binary conditions computed with the features of $x$. The prediction corresponds to the class of the leaf node.

Decision trees have the benefits of being easily explainable compared to other machine learning methods (such as Neural Networks or Logistic Regression). As a result, the human planner can easily understand why the plan is not feasible, and he can correct it. In addition, these machine learning model can be converted into a set of linear constraints, and thus they can be incorporated into a mixed integer linear program (Biggs and Hariss, 2017). Therefore, our work open the way to further researches where the decision tree could approximate the capacity constraint in planning models.

To provide more accurate predictions, we considered a method based on decision tree called random forest classifier (Breiman, 2001). Random Forest model belongs to the ensemble methods, which use the performance of several estimators and return a prediction based on the average among all estimators. Since ensemble methods consider several estimators, they are less prone to overfitting, and thus they provide a better accuracy than a single esti-

mator. In a random forest classifier, the estimators are classification trees. Each tree is trained in differently, such that predictions may differ from one tree to another.

### *4.2 Features Selection*

Although the choice of the machine learning model is an important step, the prediction performance mainly relies on noticeable features extracted from a dataset. These numerical features have to be selected thoroughly since they play a crucial role in the predictive capability of a machine learning model. Several raw parameters, such as the processing time of each operation, directly impact the makespan of a production plan, and thus its feasibility over a production period. However, due to the large number of such parameters, the model may overfit and lead to poor quality of prediction. Similarly, approximated values provided by fast heuristics (such as scheduling by shortest processing times) appear to be promising features. However, these approximations are too hard to convert into linear constraints, which can lead to very complex decision trees for the perspective of a surrogate MILP model based on random forest.

The idea is to find linear combinations of parameters which improve the prediction performance of the machine learning model. After preliminary experiments, we highlighted several features having promising capabilities for predicting a suitable decision for the respect of the capacity. First, we define the following parameters:

- $o_{ijk}$ : The "operating ratio" for each operation $O_{ij}$ performed on machine $k$.
- $o_k$ : Estimated number of operation to be performed on machine $k \in M$.
- $s_k^{mean}$ : Average setup times between each pair of jobs that can be performed on machine $k$.
- $s_k^{min}$ : Minimum setup times between each pair of jobs that can be performed on machine $k$.

The operating ratio $o_{ijk}$ estimates the likelihood for operation $O_{ij}$ to be performed on machine $k \in M_{ij}$. Since flexible operations are more likely to be performed on machines where the processing time is the lowest, $o_{ijk}$ must be large when the processing time $p_{ijk}$ is small. In addition, the operating ratio must be computed such that:

$$\sum_{k \in M_{ij}} o_{ijk} = 1.$$

We compute the operating ratios of each operation $O_{ij}$ and each machine $k \in M$ as follow:

$$o_{ijk} = \begin{cases} 0 & \text{if } k \notin M_{ij} \\ 1 & \text{if } M_{ij} = \{k\} \\ \dfrac{1}{p_{ijk} \cdot \sum\limits_{k' \in M_{ij}} \frac{1}{p_{ijk'}}} & \text{otherwise} \end{cases}$$

The estimated number of operations to performed on machine $k \in M$ is computed based on the operating ratio as follows:

$$o_k = \sum_{i \in J} \sum_{j \in n_i} \sum_{k \in M} o_{ijk}.$$

Five features based on these parameters have been selected and computed for our dataset:

- $F_1 = \max\limits_{k \in M} \left\{ \left( \sum\limits_{i \in J} \sum\limits_{j=1}^{n_i} \sum\limits_{M_{ij} = \{k\}} p_{ijk} + s_k^{min} \right) - s_k^{min} \right\}$
- $F_2 = \max\limits_{k \in M} \left\{ \left( \sum\limits_{i \in J} \sum\limits_{j=1}^{n_i} o_{ijk} \cdot p_{ijk} \right) + (o_k - 1) \cdot s_k^{mean} \right\}$
- $F_3 = \frac{1}{m} \sum\limits_{k \in M} \left( \left( \sum\limits_{i \in J} \sum\limits_{j=1}^{n_i} o_{ijk} \cdot p_{ijk} \right) + (o_k - 1) \cdot s_k^{mean} \right)$
- $F_4 = \max\limits_{i \in J} \left\{ \sum\limits_{j=1}^{n_i} \min\limits_{k \in M_{ij}} \{p_{ijk}\} \right\}$
- $F_5 = \frac{1}{m} \sum\limits_{i \in J} \sum\limits_{j=1}^{n_i} \min\limits_{k \in M_{ij}} \{p_{ijk}\}$

$F_1$ estimates the largest workload among the machine $k \in M$. The computation of $F_1$ uses a lower bound on the workload on machine $k$, and this lower bound sums the processing time of non flexible operation and the minimum setup time on machine $k$. $F_2$ and $F_3$ represent respectively the maximum and the average estimated processing time per machine. This estimated processing time per machine is computed with the operating ratio of each operation. Finally, $F_4$ and $F_5$ correspond respectively to the maximum and the average minimum processing time per job. Note that preliminary experiments showed that using the estimated processing time per machine as a feature (rather than taking the average over all machine in $F_3$) did not yield good prediction.

$\mathrm{A}_1$ denotes an approximation of the makespan that we use to benchmark our approach. $\mathrm{A}_1$ is computed as the maximum between features the $F1$ and $F4$ approximations, since these two methods are classical lower bounds of the makespan in job shop scheduling problems (Raaymakers and Weijters, 2003). $F1$ and $F4$ also commonly used as approximations of the capacity consumption for production plannings models (Drexl and Kimms, 1997; Stefan Voß, 2006). Since approximation $\mathrm{A}_1$ is based on a theoretical lower bounds of the makespan, it should be noticed that $\mathrm{A}_1$ cannot results in false negatives, i.e., it is not possible to predict as infeasible a schedule which is labeled as feasible by using this approximation.

## 5. COMPUTATIONAL EXPERIMENTS

### *5.1 Instances generation*

We trained the machine learning model with a dataset of optimally solved FJSP-SDST instances. The benchmark instances from the literature on Flexible Job Shop Scheduling problem (Hurink et al., 1994; Brandimarte, 1993) contain around 250 instances with different sizes, which is not large enough to train the random forest model and may cause underfitting. Thus, we generate two datasets, $\mathrm{D}_1$ and $\mathrm{D}_2$, both composed of 5000 instances of FJSP-SDST. Each instance has been generated using the following rules:

(1) All instances in $D_1$ have 6 jobs, 6 machines and 6 operations per job, and 10 jobs, 10 machines and 10 operations per job for $\mathrm{D}_2$.
(2) We only considered sets $M_{ij}$ with a maximum length of 2. For each operation $|M_{ij}|$ is set to 1 or 2 with probability 0.5, and the machines in $M_{ij}$ are selected randomly (all machines have the same probability of

being selected). However, to avoid inconstancy, two consecutive operations of the same job cannot be performed on the same machine, excepted if one of the two operations is flexible.

(3) The processing times per unit of job $p^u_{ijk}$, the setup times $s^k_{ii'}$ and the number of lots $ls_i$ have been randomly generated using a uniform distribution with support $[1, 100]$, $[100, 150]$, and $[0, 10]$, respectively. We also set $s_{iik} = 0$ since no setup is required for operations of the same jobs performed on the same machine. To compute the processing times per lot for every job $i$, we multiply the processing time per unit of job by its lot-size (i.e., $p_{ijk} = p^u_{ijk} \cdot ls_i, \forall i \in J, \forall j \in 1..n_i, \forall k \in M$). If the lot size of a job $i$ is equal to 0, this jobs is not considered and any setup times and processing times involving this job $i$ are set to 0.

We consider the instance sizes of Fisher and Thompson (1963) with 6 jobs with 6 machines (small instances) and 10 jobs with 10 machine (medium instances), since these instances are considered as classical instances of the literature for the job-shop along with 20 jobs and 5 machines (Jain and Meeran, 1999). However, we were not able to solve enough data for the training phase with 20 jobs and 5 machines, this size is thus not considered in this study. The processing times and the setup times are generated similarly to Shen et al. (2018), but since we apply a batch-size to each job, the setup times were no longer significant compared to the resulting makespan, and we decided to consider larger setup times.

For each instances, the constraint programming model described in Section 3.2 was solved to optimality with `IBM Ilog CP Optimizer`. The experiments were conducted on a Windows 11 computer running with a Intel Core i7-1185G7 @ 3.00GHz processor with 32 Go of RAM. Dataset $D_1$ was generated in 3,502s, with an average computation time of 0.7s for each instance and $D_2$ was generated in 45,363s with an average of 9s per instance.

| dataset | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|
| $D_1$ | 2586 | 580 | 839 | 2188 | 2583 | 2980 | 4487 |
| $D_2$ | 4446 | 686 | 1988 | 3988 | 4461 | 4906 | 7391 |

Table 1. Dataset's descriptive statistics around the makespan

Table 1 provides descriptive statistics of the datasets. 25%, 50%, 75% represent respectively the $25^{\text{th}}$ percentile, the median, and $75^{\text{th}}$ percentile. Since the median makespan in our dataset $D_1$ is 2583, we trained our machine learning model to predict the feasibility of the schedule under the available capacity. In other words, we considered an available capacity of 2583 for our shop floor and half of the instances with a makespan lower than this capacity were labeled as `True`, while the remaining half was labeled as `False`. The same process was applied on dataset $D_2$. The random forest model was trained in `Python` using the `Scikit-Learn` methods `RandomForestClassifier` with default parameters (Pedregosa et al., 2011) and trained in less than 2s in average. We randomly sampled 70% of our dataset for training and the remaining 30% for the testing phase.

### *5.2 Numerical results*

| Model | RF | | $A_1$ | |
|---|---|---|---|---|
| Instances | $D_1$ | $D_2$ | $D_1$ | $D_2$ |
| Accuracy(%) | 91.93 | 91.00 | 79.06 | 73.80 |
| Precision(%) | 91.07 | 91.99 | 70.40 | 66.03 |
| Recall(%) | 92.90 | 90.18 | 100.0 | 100.0 |

Table 2. Prediction performance between random forest model and other approximation of capacity consumption

Table 2 report the performance of random forest model and of the approximation of capacity consumption $A_1$. The three rows (Accuracy, Precision, and Recall) are the common metrics used to evaluate a classifier model. The precision represents the ratio between the number of schedules correctly predicted as feasible and the number total number of schedules predicted as feasible. The recall gives the ratio between the number of schedules predicted as feasible and the actual number of feasible schedules in the test dataset. The accuracy provides the proportion of schedules correctly classified as feasible or infeasible. For the considered problem, a model with high precision is crucial to limit the number of instances wrongly predicted as feasible by our model. These infeasible schedules are undesirable since they lead to the unfeasible production plan, and this result in a lot of firefighting on the shop floor. At the operational level, planners employs various levers to deal with such infeasible plans, but they lead to larger production cost and lower service level. These levers includes rejection of some orders, express distribution modes, late deliveries to customer, express raw material deliveries, ... (Thevenin et al., 2017). On the contrary, an approach with low recall would remove a set of feasible solutions, but the resulting schedules would still be feasible. Therefore, a low recall may lead to sub-optimal planning. While a model with low recall results in larger costs at the planning level, it is preferred over a model with low precision that leads to infeasible plans.

Table 2 shows that the random forest model gives better overall results compared to method $A_1$. Although $A_1$ has better recall (100%) than the random forest, the random forest has a significantly better precision which is more important than the recall for our model. The Random Forest method provides a better approximation of feasibility, and its predictive performance is stable from dataset $D_1$ to $D_2$, while approximation $A_1$ seems to provide a more inaccurate approximation of the feasibility as instances become larger. Finally, the random forest gives a better trade-off between precision and recall, whereas $A_1$ favors the recall metrics over precision and accuracy.

## 6. CONCLUSION

This study investigates the use of machine learning to predict if a production plan is implementable or not. In practice, manufacturers often discover a production plan is not feasible when trying to create the production schedule. Our results show that machine learning tools such as random forest can provide accurate predictions on the feasibility of a production plan. We first defined a set of relevant features for the training of a random forest model. After training our model, we compared our approach with

an approximation of the capacity consumption commonly used in production planning tools. The computational experiments show that the random forest classifier outperforms classical capacity restriction in terms of precision and accuracy. These results confirm the effectiveness of machine learning approaches for the makespan estimation. Future works are twofold. First, since only five features have been selected in the current study, the model can be improved by refining the features selection by adding other features with a high potential for determining if the capacity is reached or not. Second, we will investigate the possibility to integrate the learned model into MILP model used for lot-sizing. The goal is to better approximate the capacity restriction and to automatically generate implementable production plans. Note that we pay attention to selecting only features that can be easily formulated as linear formulas, to be easily converted into linear inequalities.

## ACKNOWLEDGEMENTS

The present work was conducted within the project ASSISTANT (https://assistant-project.eu/) that is funded by the European Commission, under grant agreement number 101000165, H2020 – ICT-38-2020, Artificial intelligence for manufacturing. The authors would also like to thank the region Pays de la Loire for their financial support.

## REFERENCES

Beldiceanu, N., Dolgui, A., Gonnermann, C., Gonzalez-Castañé, G., Kousi, N., Meyers, B., Prud'homme, J., Thevenin, S., Vyhmeister, E., and Östberg, P.O. (2021). ASSISTANT: Learning and robust decision support system for agile manufacturing environments. *IFAC-PapersOnLine*, 54(1), 641–646.

Biggs, M. and Hariss, R. (2017). Optimizing objective functions determined from random forests. *SSRN Electronic Journal*.

Błażewicz, J., Domschke, W., and Pesch, E. (1996). The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research*, 93(1), 1–33.

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Ann Oper Res*, 41, 157–183.

Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. (1983). *Classification and Regression Trees*. Wadsworth International Group.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.

Candan, G. and Yazgan, H.R. (2014). Genetic algorithm parameter optimisation using taguchi method for a flexible manufacturing system scheduling problem. *International Journal of Production Research*, 53(3), 897–915.

Casazza, M. and Ceselli, A. (2019). *Heuristic Data-Driven Feasibility on Integrated Planning and Scheduling*, 115–125. Springer International Publishing, Cham.

Dias, L.S. and Ierapetritou, M.G. (2019). Data-driven feasibility analysis for the integration of planning and scheduling problems. *Optimization and Engineering*, 20(4), 1029–1066.

Drexl, A. and Kimms, A. (1997). Lot sizing and scheduling — survey and extensions. *European Journal of Operational Research*, 99(2), 221–235.

Fisher, H. and Thompson, G. (1963). Probabilistic learning combinations of local job-shop scheduling rules. *Industrial Scheduling, Muth and G. L. Thompson(Eds.)*, 45, 225–251.

Hurink, J., Jurisch, B., and Thole, M. (1994). Tabu search for the job-shop scheduling problem with multi-purpose machines. *OR Spektrum*, 15, 205–215.

IBM (2021). IBM ILOG CPLEX optimization studio 20.1.0: Online documentation. `https://www.ibm.com/docs/en/icos/20.1.0`.

Jain, A. and Meeran, S. (1999). Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research*, 113(2), 390–434.

Jun, S., Lee, S., and Chun, H. (2019). Learning dispatching rules using random forest in flexible job shop scheduling problems. *International Journal of Production Research*, 57(10), 3290–3310.

Li, X. and Olafsson, S. (2005). Discovering dispatching rules using data mining. *Journal of Scheduling*, 8(6), 515–527.

Li, Y., Carabelli, S., Fadda, E., Manerba, D., Tadei, R., and Terzo, O. (2020). Machine learning and optimization for production rescheduling in industry 4.0. *The International Journal of Advanced Manufacturing Technology*, 110(9-10), 2445–2463.

Nakasuka, S. and Yoshida, T. (1992). Dynamic scheduling system utilizing machine learning as a knowledge acquisition tool. *International Journal of Production Research*, 30(2), 411–431.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Raaymakers, W. and Weijters, A. (2003). Makespan estimation in batch process industries: A comparison between regression analysis and neural networks. *European Journal of Operational Research*, 145(1), 14–30.

Schneckenreither, M., Haeussler, S., and Gerhold, C. (2020). Order release planning with predictive lead times: a machine learning approach. *International Journal of Production Research*, 59(11), 3285–3303.

Shen, L., Dauzère-Pérès, S., and Neufeld, J.S. (2018). Solving the flexible job shop scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*, 265(2), 503–516.

Stefan Voß, D.L.W. (2006). *Introduction to Computational Optimization Models for Production Planning in a Supply Chain*. Springer Berlin Heidelberg.

Thevenin, S., Zufferey, N., and Glardon, R. (2017). Model and metaheuristics for a scheduling problem integrating procurement, sale and distribution decisions. *Annals of Operations Research*, 259(1), 437–460.

Yang, Y., Chen, Y., and Long, C. (2016). Flexible robotic manufacturing cell scheduling problem with multiple robots. *International Journal of Production Research*, 54(22), 6768–6781.