Constraint learning approaches to improve the approximation of the capacity consumption function in lot-sizing models

David Tremblet, Simon Thevenin, Alexandre Dolgui

European Journal of Operational Research 322 (2025) 679–692

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/eor

Interfaces with Other Disciplines

Constraint learning approaches to improve the approximation of the capacity consumption function in lot-sizing models

David Tremblet *, Simon Thevenin, Alexandre Dolgui

IMT Atlantique, LS2N-CNRS, 4 rue Alfred Kastler, La Chantrerie, Nantes, 44307, France

ARTICLE INFO

Keywords:
Production planning
Lot-sizing
Scheduling
Machine learning
Data-driven methods

ABSTRACT

Classical capacitated lot-sizing models include capacity constraints relying on a rough estimation of capacity consumption. The plans resulting from these models are often not executable on the shop floor. This paper investigates the use of constraint learning approaches to replace the capacity constraints in lot-sizing models with machine learning models. Integrating machine learning models into optimization models is not straightforward since the optimizer tends to exploit constraint approximation errors to minimize the costs. To overcome this issue, we introduce a training procedure that guarantees overestimation in the training sample. In addition, we propose an iterative training example generation approach. We perform numerical experiments with standard lot-sizing instances, where we assume the shop floor is a flexible job-shop. Our results show that the proposed approach provides 100% feasible plans and yields lower costs compared to classical lot-sizing models. Our methodology is competitive with integrated lot-sizing and scheduling models on small instances, and it scales well to realistic size instances when compared to the integrated approach.

1. Introduction

Advanced Planning and Scheduling software is crucial for operation management in manufacturing industries. Such tools usually follow the hierarchical approach (Stadtler, 2005), where a production planning module provides the input for a scheduling model. Production planning gives weekly (or monthly) production quantity, adjusting the capacity, and placing orders with suppliers to meet the demand while minimizing inventories. At the operational level, the scheduling modules take as input the production quantities, and they assign the operations to machines, sequence the operations, and compute their starting times. To better integrate the limitation at the scheduling level, capacity consumption is computed at the production planning level.

This capacity consumption calculation has been included since the use of the MRPII planning system, but the resulting tools only roughly consider the time required on each resource, and they do not take into account the complexities of the scheduling environments. This computation plays a crucial role in production planning since underestimating capacity consumption leads to a plan that cannot be implemented on the shop floor. Such a situation often results in unmet demand, and a lot of actions must be engaged to produce the quantities on time. This situation is hard to manage for practitioners since it requires either re-computing the quantities for the whole plan, which is time-consuming or shifting the quantities, causing delay and a drop in customer confidence. In addition, the gap between an infeasible production plan and its repaired solution can be huge, and the cost associated with these initial plans becomes irrelevant. Overestimating capacity consumption leads to a loss of opportunity since it prevents the resources from being used at full capacity. As a result, despite the inclusion of capacity in complex optimization models provided by advanced planning systems (APS), this type of software keeps providing plans that are too tight, and often cannot be implemented in practice. For instance, Tenhiälä (2010) showed that APS with finite capacity do not fit well in job-shop-like environments because the user cannot provide accurate enough values for the required parameter (e.g., the capacity consumption per unit). As users are unsatisfied, they tend to turn towards simpler and less cost-efficient planning approaches (often relying on simple rules to apply by hand). As a result, a large proportion of manufacturers still rely on Excel software to plan their production (Filho et al., 2010; Liu et al., 2019). Many authors highlight the drawback of aggregated capacity constraints in lot-sizing models and the necessity to acquire further information at the planning level (Almeder et al., 2015; Dauzère-Pérès & Lasserre, 2002). This includes scheduling decisions or detailed capacity constraints, leading to impractical mathematical programs or constraints that are too complex to be integrated.

* Corresponding author.
E-mail addresses: david.tremblet@imt-atlantique.fr (D. Tremblet), simon.thevenin@imt-atlantique.fr (S. Thevenin), alexandre.dolgui@imt-atlantique.fr (A. Dolgui).

https://doi.org/10.1016/j.ejor.2024.11.039
Received 14 September 2023; Accepted 25 November 2024
Available online 10 December 2024

We assume that the capacity requirements are known and fixed, and we focus on finding approximations of the capacity consumption leading to feasible and cost-efficient production plans. This capacity consumption calculated at the lot-sizing level corresponds to an approximation of the scheduling problem from the next production level on the shop floor. The corresponding production environment is assumed to be immutable and deterministic, but the inclusion of uncertain parameters on the shop floor can be considered with our approach.

With the rising interest in machine learning, the operation research community recently provided several approaches to translate machine learning models into mathematical programs (e.g., Fajemisin et al., 2023). In this work, we propose to replace the basic capacity consumption function in lot-sizing models with an approximation built using machine learning algorithms. The capacity consumption is learned from examples that give the total amount of time required to complete all operations. While our experiments rely on production schedules optimized with linear and constraint programming, the methodology remains applicable when the examples for learning capacity are generated by other means. For instance, the examples can correspond to historical data obtained by reconciling Advance Planning System and Manufacturing Execution System data, or they can be generated from simulation models. Machine learning models lead to accurate approximations of capacity consumption, and this leads to integrated lot-sizing and machine learning models returning reliable and cost-efficient production plans. This reliability is also essential and time-saving since it prevents practitioners from recomputing the quantities in case of infeasibility.

To better understand the capacity consumption calculation and compare the value of each approach proposed in this work, we evaluated our approach in an integrated lot-sizing and scheduling problem. Hence, the capacity consumption at each period of the production plan is measured as the makespan of a flexible job shop scheduling problem. The violation of the capacity at the scheduling level is undesirable since this leads to plans that cannot be implemented in practice. As a result, we consider that a production plan respects the capacity if we can find a production schedule with a makespan lower than the number of working hours in the factory.

The contributions of this work are fivefold: (1) We propose several extensions of the lot-sizing problem (LSP) formulation where the capacity constraint is approximated with machine learning techniques. These formulations correspond to approximation with linear regressions, decision trees, and piecewise linear regressions. We study different sets of features to train machine learning models, and our results suggest that the most important features include the lot sizes and lower bounds on the makespan; (2) The optimal solution of a Mixed Integer Linear Program usually lies at the extremes of the feasible region. When a constraint is approximated by a machine learning model, approximation errors lead to undesirable solutions. Therefore, we propose a constrained training approach that prevents us from overestimating the capacity consumption in the training sample. This new training procedure increases the number of feasible plans of the proposed models, with a percentage of feasible plans increased by 17% up to 93%. In addition, we propose an iterative learning scheme that integrates machine learning training with an optimization approach. This learning procedure results in integrated lot-sizing and machine learning models returning 100% of feasible plans for all types of instances and can be adjusted to favor the cost of the plans over their feasibility. (3) We show that machine learning leads to good approximations of capacity constraints. A comparison with the exact (but unpractical) approach that integrates the lot-sizing and scheduling models shows that the proposed formulation yields close to optimal solutions. Our results show that the computational efforts required to solve the model depend on the complexity of the machine learning model. Simple approximations with linear regression do not impair the computational performance, while complex models such as deep decision trees lead models that are hard to solve. Our experimental results show that the proposed approach outperforms models based on integrated lot-sizing and scheduling, such as the approach proposed by Dauzère-Pérès and Lasserre (1994). In addition, the method we propose yields reliable production plans compared to standard lot-sizing models, including lot-sizing with fixed scheduling sequence proposed by Wolosewicz et al. (2015). (4) We incorporate the machine learning approximation approach into an iterative lot-sizing and scheduling procedure. The resulting methodology produces good quality solutions for large size instances. (5) The machine learning can be trained with scheduling examples that incorporate uncertainty, and the resulting method is efficient in dealing with uncertainty. We consider the case where the processing time is uncertain, and we show that the method returns feasible schedules even when the process duration is unknown in advance, only the probability distributions are known.

The paper is organized as follows. Section 2 gives a literature review of production planning and scheduling problems, as well as machine learning approaches to predict the makespan. Section 3 states the considered problem. Section 4 describes our data-driven approach and the different machine learning models used in this paper. Section 5 presents several approaches to generate relevant datasets related to the scheduling level considered. Finally, we compared our data-driven method with multiple integrated lot-sizing and scheduling models from the literature in the numerical experiments in Section 6, before concluding in Section 7.

2. Literature review

This section successively reviews the literature on the integration of machine learning models into mathematical programs, capacity consumption computation in lot-sizing models, and machine learning models in scheduling problems.

2.1. Constraint learning framework for production planning

Embedding machine learning models into mathematical programs is an increasingly popular area of research. This approach, referred to as *"constraint learning"* or *"surrogate modeling"*, leverages machine learning techniques to incorporate constraints or objective functions that are either computationally challenging or complex to formulate manually. Numerous studies have explored the translation of different machine learning models into linear programs, including neural networks (Fischetti & Jo, 2018), decision trees and ensemble methods (Biggs et al., 2022; Mišić, 2020), among others (Fajemisin et al., 2023; Maragno et al., 2023). This approach has led to multiple tools to facilitate the translation of machine learning models into mathematical programs, such as the `OptiCL` package (Maragno et al., 2023) or the *Gurobi Machine Learning* package.

There is a growing interest in the application of machine learning techniques for production planning. In this research field, machine learning is commonly used to either generate specific parameter values for the lot-sizing model or leveraged to solve the lot-sizing problem (e.g., Larroche et al., 2021; Şenyiğit et al., 2013; Yu et al., 2024; Zhang et al., 2021). For example, Rohaninejad et al. (2023) use neural networks to predict safety stocks and safety slacks in situations where processing times are uncertain. Similarly, Beykal et al. (2022) developed a data-driven optimization framework to solve bi-level production planning, with scheduling and lot-sizing under uncertain demand.

Machine learning approaches are also commonly used to learn the uncertainty set in robust optimization methods. For instance, Shang et al. (2017) used Support Vector Clustering (SVC) for a robust chemical planning problem. SVC computes the uncertainty set of several parameters, including demands and prices with piecewise linear kernels to ensure the resulting uncertainty set corresponds to a linear program.

The literature review on the application of machine learning for production planning is extensive. In the rest of this section, we focus on papers that consider a similar approach to the one we use in this

paper. Specifically, we concentrate only on papers related to constraint learning approaches for production planning programs. Casazza and Ceselli (2019) consider a data-driven model for the integration of production planning and scheduling, where the constraints related to the scheduling problem are replaced by a decision tree. At the scheduling level, a set of jobs has to be scheduled on a set of parallel machines while respecting release dates and due dates, and jobs can be split into two to make the assignment easier. The objective is to find a feasible assignment of jobs that minimizes the number of split jobs. Dias and Ierapetritou (2019) considered the integration of a lot-sizing model and scheduling decisions, where scheduling decisions correspond to a discrete state–task network. The authors incorporate classification models into lot-sizing to ensure the plan is feasible, and they consider different machine learning models such as neural networks, decision trees, and support vector machines. The authors show that the latter approach scales very well on high-dimensional instances when compared to methods integrating the whole scheduling decision. The resulting method also provides accurate approximations in the case of uncertain production capacity as by Hu et al. (2008). This latter study led to an increasing interest in surrogate modeling for the integration of production planning, scheduling, and control (Badejo & Ierapetritou, 2022; Dias & Ierapetritou, 2020).

These studies consider discrete scheduling problems (where the scheduling horizon is discretized in a set of discrete time periods) or parallel machine scheduling. To the best of our knowledge, this paper is the first to consider learning capacity consumption in a flexible job-shop environment with sequence-dependent setup times. Flexible resources are frequent in make-to-order industries (Bish & Wang, 2004; Chod & Zhou, 2014), and their popularity is increasing in the manufacturing industry (Begnaud et al., 2009). In addition, the flexible job-shops generalize many scheduling environments (job-shop, flexible flow shop, etc...), and our results remain valid in all these environments.

Setup times are also frequent in manufacturing systems and they have been considered early in the literature on capacitated lot-sizing problems (Trigeiro et al., 1989). With the inclusion of setup times, the problem of finding production plans respecting both the capacity and demand becomes NP-complete. Realistic scheduling applications often account for sequence-dependent setup time (Allahverdi et al. 1999), for instance in the food industry, chemistry, fast moving consumer goods (Larroche et al., 2021; Thevenin et al., 2017).

In addition, we investigate different approaches to improve the accuracy of machine learning models when used in optimization models. In particular, we propose methods to generate efficient datasets, including an approach that takes advantage of the scheduling problem structure to generate adversarial examples. In addition, we introduce additional features for our problem that improve the prediction of capacity consumption. Finally, we compared our approach with standard mathematical models for the integrated lot-sizing and scheduling problem, and show the potential of our data-driven approach for solving large-scale instances.

2.2. Approximation of capacity consumption in lot-sizing models

Lot-sizing models determine the optimal production quantities in each period of the horizon. Once the plan is available, the lots of each period become production jobs to schedule on the machine. The acquisition of capacity in lot-sizing problems also led to an increasing body of literature review. In these problems, the lot-sizing formulation incorporates decisions regarding the capacity, including subcontracting and capacity acquisition (Atamtürk & Hochbaum, 2001; Hwang, 2021), or the adjustment between different levels of capacity (Ou & Feng, 2019). In the classical hierarchical decision framework, scheduling decisions are made independently of production planning decisions (Axsäter, 1986). Lot-sizing models represent aggregated production planning problems, where the items correspond to aggregated product families rather than specific items produced on the shop floor. Consequently, the computation of the capacity consumption function in the lot-sizing model relies on the quantity per aggregated item family, which offers only a rough approximation of the actual resource consumption on the shop floor. The accurate computation of actual resource consumption takes place at the scheduling level, where product families are disaggregated to perform computations at a more detailed level. As scheduling has a smaller granularity, it often incorporates additional details that cannot be considered at the planning step, such as secondary equipment required for production, transportation time on the shop floor, blocking constraints, etc. Therefore, the feasibility of a production plan is only assessed at the production scheduling step. Several extensions (Copil et al., 2016) of the classical lot-sizing model integrate scheduling decisions into lot-sizing problems. For example, the continuous setup lot-sizing problem incorporates setup times into the lot-sizing model and determines if resource configurations change between periods (Drexl & Kimms, 1997). However, these models typically assume that machines can only perform one operation per period, and the capacity consumption remains a rough approximation of the actual complexity of the shop floor.

Other models introduced the concept of macro periods subdivided into several micro-periods, where each micro-period produces at most one item. This methodology has led to the general lot-sizing and scheduling problem (GLSP) presented in Fleischmann and Meyr (1997). Multiple versions of the GLSP have been proposed in the last decades, including versions with parallel machines (Meyr, 2002) or bills of materials with multiple levels (Seeanner & Meyr, 2012). For instance, Rohaninejad et al. (2014) propose a genetic algorithm and particle swarm optimization to solve the GLSP in a Flexible Job-Shop Scheduling environment. While these approaches provide better approximations of the capacity consumption, they remain aggregated models. The scheduling problems are not a detailed representation of the operations on the shop floor. For instance, such models cannot represent a job-shop environment precisely.

Some authors consider the integration of scheduling and lot-sizing (e.g., Lasserre, 1992). These approaches address situations where the sequencing of lots is crucial, such as when there are sequence-dependent setup times in the production process. Simultaneous lot-sizing and scheduling methods typically involve iterative procedures that determine lot sizes at the planning level and order operations on resources for fixed product quantities. Similarly, different mathematical models have been proposed to incorporate the scheduling decisions in each period of the production plan. Dauzère-Pérès and Lasserre (1994) propose a model that integrates a flexible job-shop scheduling problem with setup into a lot-sizing model. Dauzère-Pérès and Lasserre (2002) extend the model to the case of multi-level lot-sizing. Urrutia et al. (2014) propose an efficient solution method for this problem. Their method starts with an initial solution, and it creates this initial solution with the lot-sizing model with fixed sequences of operations proposed in Wolosewicz et al. (2015). Afterward, the approach iterates between a Lagrangian heuristic to solve the lot-sizing problem with a fixed sequence of operations and a Tabu-search to improve the sequence with fixed lot sizes.

Almeder et al. (2015) highlight the weakness of the classical capacitated lot-sizing formulations for the multi-level bill of materials. The classical models lead to lot-sizing solutions that are infeasible for the scheduling problem that considers each period separately. The authors propose an improved mathematical formulation for the batching and lot-streaming cases.

The integration of job-shop scheduling into lot-sizing models leads to accurate computation of the capacity consumption. However, solving the resulting model is hard, and no method exists for solving large-scale instances to optimality. In particular, for a flexible job-shop, alternative routings increase the number of operation sequences, and the integrated approach rapidly becomes impractical for large-scale instances. In addition, shop floors may involve complex structures and

constraints, including workers unavailabilities, machine breakdown during production, or the requirements of tools to perform certain operations. Such complexities are generally difficult to model as they would require a prohibitive number of binary variables and constraints. As a result, the final lot-sizing models discard these details, which leads to a less accurate computation of the capacity consumption. In our study, we aim to improve the approximation of the capacity consumption by training machine learning models with historical data from the scheduling problem encountered on the shop floor. The resulting approach yields a model that is computationally less expensive than the integration of scheduling decisions into lot-sizing models. In addition, since these machine learning models are trained directly from the historical data of the shop floor, they may incorporate all the complexity of the scheduling decision process, even the parts that are difficult to model mathematically.

2.3. Machine learning for scheduling applications

A wide variety of applications of machine learning exist in the scheduling literature. The first works to use machine learning in scheduling (e.g., Lee et al., 1997; Shinichi & Taketoshi, 1992) seek to predict the best dispatching rule for a given instance. Jun et al. (2019) show this methodology is relevant for flexible job-shop scheduling problems. These approaches can be seen as a pre-processing phase to improve the performance of heuristics. Very few papers study predictive models to approximate the value of makespan in job-shop scheduling problems.

Some works (e.g., Raaymakers & Weijters, 2003; Schneckenreither et al., 2020) propose regressive models to predict lead times of incoming orders in batch processing. The problem is to predict the lead time of incoming orders, to ensure that the shop floor can meet the demand on time. Predicting these lead times avoids computing the whole schedule, which saves precious time when urgent decisions have to be made in the case of incoming orders or unanticipated event on the shop floor. Raaymakers and Weijters (2003) introduced the use of regression analysis and neural networks to predict the makespan of scheduling problems in a job-shop environment. Schneckenreither et al. (2020) considered a similar approach by considering neural networks to predict the lead times in order release planning.

Recently, Tremblet et al. (2023) considered machine learning models to predict the makespan of flexible job-shop scheduling problems. These machine learning models have the advantage of instantly approximating the makespan without computing the scheduling decisions. The present study aims at integrating these powerful predictive models into capacitated lot-sizing models, in order to replace the well-known capacity constraints.

3. Problem description

This section presents the mathematical model of classical lot-sizing. The problem is to determine optimal lot sizes at a production planning level while satisfying capacity constraints at each period for the scheduling. In this study, we consider a flexible job-shop at the scheduling level, and this section provides a formal description of this scheduling problem.

3.1. Capacitated lot-sizing problem (CLSP)

The capacitated lot-sizing problem (Drexl & Kimms, 1997) sizes production lots to minimize holding costs, fixed setup costs, and unit production costs. The production plan accounts for customer demand, production capacity, and lead times.

The factory produces each item i in the set of items J in a batch of consecutive operations, since the processing of a batch of item $i \in J$ results in a setup time s_i, and a setup cost c_i^s. Each operation in the batch yields one unit of item i, and it has a cost c_i^p and a duration of p_{ik} units on machine $k \in M$. In each period $t \in T$ of the horizon, the production is limited by a given capacity of C_t units. The production plan must respect the demand d_{it} of item $i \in J$ in period $t \in T$. In our formulation, I_{it}^+ refers to the inventory level of item $i \in J$ at the end of period $t \in T$, and I_{it}^- refers to the backlog level of item i in period t. Inventory and backlog levels generate costs c_i^h and c_i^b, respectively. Therefore, the lot-sizing model involves decision variables for the lot sizes, setup, inventory level, and backlog level for each item $i \in J$ and each period $t \in T$, denoted respectively by X_{it}, Y_{it}, I_{it}^+, I_{it}^-. The CLSP corresponds to the following Mixed-Integer Linear Program (MILP):

$$\min \quad \sum_{t\in T}\sum_{i\in J} c_i^h I_{it}^+ + c_i^b I_{it}^- + c_i^s Y_{it} + c_i^p X_{it} \tag{1}$$

$$\text{s. t.} \quad I_{it-1}^+ - I_{it-1}^- + X_{it} - I_{it}^+ + I_{it}^- = d_{it}, \qquad i \in J, t \in T \tag{2}$$

$$X_{it} \le H \cdot Y_{it}, \qquad i \in J, t \in T \tag{3}$$

$$\sum_{i\in J} p_{ik} X_{it} + s_{ik} Y_{it} \le C_t, \qquad k \in M, t \in T \tag{4}$$

$$I_{i0}^+ = I_{iT}^- = I_{iT}^+ = 0, \qquad i \in J \tag{5}$$

$$X_{it} \ge 0,\ I_{it}^+ \ge 0,\ I_{it}^- \ge 0, \qquad i \in J, t \in T$$

$$Y_{it} \in \{0,1\}, \qquad i \in J, t \in T.$$

The objective function (1) minimizes the total cost, which includes holding costs c_i^h, backlogging costs c_i^b, setup costs c_i^s, and production costs c_i^p. Constraints (2) compute the inventory balance constraints for each product and period of the horizon. Constraints (3) force Y_{it} to be equal to 1 if a batch of items i is produced at a period t, using the well-known big M constraints, where $H = \sum_{t\in T} D_t$. Constraints (4) ensure that the capacity consumption does not exceed the capacity C_t for all periods $t \in T$. The basic formulation of the capacity constraint accounts for the process duration per production unit and for the setup time on each resource $k \in M$. Finally, constraints (5) ensure that there is no inventory level at the beginning of the period and that there is neither inventory nor backlogged items at the end of the planning horizon.

3.2. Flexible job-shop scheduling problem (FJSP)

At the scheduling level, each production lot becomes a job to schedule. As a result, the set J of items in the lot-sizing model corresponds to a set of J jobs in the scheduling problem. The flexible job-shop scheduling problem is an extension of the well-known job-shop scheduling problem, but a set of machines can perform each operation in the routing. A set J of n jobs have to be performed on a set M of m machines with respect to routing constraints. Each job $i \in J$ is subdivided into n_i successive operations, and O_{ij} denotes the jth operation of job i. Each operation O_{ij} performed on machine $k \in M_{ij}$ has a processing time p_{ijk}^u, where $M_{ij} \subseteq M$ denotes the set of machines that can perform operation O_{ij}. We also consider a sequence-dependent setup time $s_{ii'}^k$ occurs when job i' is processed after job i on machine k. To avoid inconsistency, we assume that setup times respect the triangular inequalities, i.e., $s_{ii''}^k \le s_{ii'}^k + s_{i'i''}^k$ for any jobs i, i' and $i\varepsilon \in J$ and any machine $k \in M$. This paper focuses on minimizing the makespan, i.e., the time required to complete all jobs $i \in J$. We consider a production plan period to be feasible if the makespan of the associated schedule for the production lot within that period is less than the total working hours in that period. A production plan is feasible if all its periods are feasible. We assume that, within a period of the production plan, each of the successive operations of each job has to be operated once and that none of these operations can be delayed or canceled if the inventory level is sufficient to perform the remaining operations of a job. We also assume that all the operations are available as soon as their predecessors have already been processed.

The Supplementary Materials of this paper provide the formulation of the integrated flexible job-shop scheduling and lot-sizing model. However, the latter approach leads to a complex mathematical model

that is not practical in large-scale instances. We use this integrated model to benchmark the proposed approaches that rely on constraint learning (Fajemisin et al., 2023) to replace the capacity constraints (4) by a machine learning model.

4. Machine learning based method

To improve the accuracy of the capacity constraint, we rely on machine learning models to predict the makespan of the schedule for the lots in each period. In our framework, the lot-sizing model integrates the translation of a fitted machine-learning model for each period t of the planning horizon. Given a production plan (X, Y, I^+, I^-), the linear program translation of each of the machine learning models predicts a value for capacity consumption. The model includes one machine learning model for each period. We restrict our study to the case where the machine learning models of each period are identical and trained on the same dataset.

Supervised learning models are predictive models trained with samples of past observations, and they return appropriate forecasts for the future. The training of a supervised learning model requires a training dataset $D = \{\overline{\mathcal{X}^s}, \overline{\mathcal{Y}^s} \mid s = 1, \dots, N\}$, where N is the number of samples, $\overline{\mathcal{X}^s}$ represents the value of the features for sample s, and $\overline{\mathcal{Y}^s}$ is a targeted value observed for this sample. The model is trained over this dataset by minimizing an error, typically the mean squared error, between the target and the output of the model.

Note that a classification model could predict if a plan is feasible or not. However, a regression model provides more flexibility. For instance, in scenarios where available capacity fluctuates across different time periods, the same regression model can predict capacity consumption. Conversely, addressing this variability with a classification model would necessitate training a distinct model for each period. In addition, a regression model integrates seamlessly into lot-sizing models that account for extra-capacity penalties. Finally, forecasting the capacity consumption rather than a binary class helps in evaluating the gap between the prediction and the actual makespan.

We assume that the flexible job-shop environment remains the same throughout the planning horizon, but the quantity associated with each job changes. Therefore, the training dataset corresponds to different processing durations in a single flexible job-shop scheduling problem. Each sample of the training dataset provides the makespan obtained when solving the flexible job-shop scheduling problem with the same resources and routing, but with different quantities X_i associated with each job $i \in J$. For each sample $s \in D$, the targeted value $\overline{\mathcal{Y}^s}$ represents the makespan, and the features $\overline{\mathcal{X}^s}$ are the quantities X_i of each job $i \in J$.

The rest of this section presents the input features of the machine learning model, before introducing three model, namely, linear regression, piecewise linear regression, and regression tree. The choice of these models is motivated by the following theoretical result, which that shows the capacity consumption function is a piecewise linear non-convex function.

Proposition 4.1. *Given any quantities $X \in \mathbb{R}^+_{|J|}$, the capacity consumption (i.e. the makespan of the resulting FJSP) is defined as a piecewise linear non-convex function.*

Proof. In the Supplementary Materials. □

4.1. Features selection

This section presents a set of relevant features $\mathcal{F}$ for our machine learning model that predicts the makespan. Besides the lot sizes X, additional features can be considered to improve the forecasting ability. Recent studies highlight important correlations between features of job-shop scheduling problems and the makespan (e.g., Mirshekarian & Šormaz, 2016; Schneckenreither et al., 2020). However, to translate the resulting machine learning model into a mathematical program, feature $f \in \mathcal{F}$ must be a linear combination of decision variables of the problem. Non-linear features cannot be used for the prediction. Also, as the production system remains the same over the entire planning horizon, features that do not depend on the decision variables will take the same values in all examples, and they are not relevant. In this sense, features based on scheduling decisions, such as assignment or sequencing decisions, are of little interest for capacity consumption prediction.

In addition, as we assumed that producing an item requires performing all its operations during the period, inventory and backlog level I^+, I^- have no impact on the capacity consumption, and there is no need to consider inventory level for work in progress.

For the sake of clarity, we make a distinction between a feature $f \in \mathcal{F}$ used to train a model, the value $\overline{\mathcal{X}^s_f}$ of this feature in a data sample $s \in D$, and the decision variables $\mathcal{X}_{ft}$ that represent the feature in each period $t \in T$ when embedded in a mathematical program.

The first features are the lot sizes X_i for each job $i \in J$, and they can be directly embedded into the lot-sizing since they correspond to decision variables X_{it}:

$$\mathcal{X}_{it} = X_{it} \quad \forall i \in J, t \in T.$$

In addition, we consider the four features introduced in Tremblet et al. (2022) that are linear combinations of lot sizes.

$$\mathcal{X}_{(|J|+1)t} = \max_{i \in J} \left\{ \sum_{j=1}^{n_i} \min_{k \in M_{ij}} \{p_{ijk} \cdot X_{it}\} \right\} \quad \forall t \in T \tag{6}$$

$$\mathcal{X}_{(|J|+2)t} = \max_{k \in M} \left\{ \sum_{i \in J} \sum_{j=1}^{n_i} \sum_{M_{ij}=\{k\}} p_{ijk} \cdot X_{it} + s^{min}_k \cdot Y_{it} \right\} \quad \forall t \in T \tag{7}$$

$$\mathcal{X}_{(|J|+3)t} = \max_{k \in M} \left\{ \sum_{i \in J} \sum_{j=1}^{n_i} o_{ijk} \cdot p_{ijk} \cdot X_{it} + (o_k - 1) s^{mean}_k \cdot Y_{it} \right\} \quad \forall t \in T \tag{8}$$

$$\mathcal{X}_{(|J|+4)t} = \frac{1}{m} \sum_{k \in M} \left(\sum_{i \in J} \sum_{j=1}^{n_i} o_{ijk} \cdot p_{ijk} \cdot X_{it} + (o_k - 1) s^{mean}_k \cdot Y_{it} \right) \quad \forall t \in T \tag{9}$$

Feature (6) and (7) are lower bounds of the makespan. Feature (6) is the maximum among all jobs $i \in J$ of the sums of processing times of the operations of job i. If an operation can be performed on more than one machine, the operation with the minimum processing time is selected. Feature (7) is the sum of the processing times of all operations processed on each machine. Since the machines are flexible, we only consider the operations that can be performed on a single machine and the minimum setup time s^{min}_k that occurs on the machine. Features (8) and (9) provide a more realistic estimate of the makespan and average sum of processing time for each machine. These features account for flexible machines by considering all possible operations O_{ij} that can be performed on each machine $k \in M$, where o_{ijk} represents the likelihood of operation O_{ij} being performed on machine $k \in M_{ij}$. o_k is an estimation of the number of operations performed on machine $k \in M$, and s^{mean}_k is the average setup time that occurs on this machine. The expressions used to compute o_{ijk} and o_k are described in Tremblet et al. (2022), and we summarize them in the Supplementary Materials.

Features (6)–(8) include max operator, and their representation in a MILP requires big-M formulations. To avoid the cumbersome big-M formulations, we can restrict the approximated capacity consumption function to be non-decreasing with features (6)–(8). For instance, in a linear regression, we can force the coefficient of these features to be positive. Since the value of the prediction should be as small as possible to respect the capacity constraints, the decision variables associated with these three features will automatically take the lowest values in

the mathematical program. Therefore, features (6)–(8) can be expressed as the following linear inequalities:

$$\mathcal{X}_{(|J|+1)t} \geq \sum_{j=1}^{n_i} \min_{k \in M_{ij}} \{p_{ijk}\} \cdot X_{it} \quad \forall i \in J, \forall t \in T \tag{10}$$

$$\mathcal{X}_{(|J|+2)t} \geq \sum_{i \in J} \sum_{j=1}^{n_i} \sum_{M_{ij}=\{k\}} p_{ijk} \cdot X_{it} + s_k^{min} \cdot Y_{it} \quad \forall k \in M, \forall t \in T \tag{11}$$

$$\mathcal{X}_{(|J|+3)t} \geq \sum_{i \in J} \sum_{j=1}^{n_i} o_{ijk} \cdot p_{ijk} \cdot X_{it} + (o_k - 1) s_k^{mean} \cdot Y_{it} \quad \forall k \in M, \forall t \in T \tag{12}$$

As setups are important in lot-sizing and scheduling problems, we considered another feature that computes the number of setups in each period $t \in T$:

$$\mathcal{X}_{(|J|+5)t} = \sum_{i \in J} Y_i \quad \forall t \in T \tag{13}$$

Note that a machine learning model may give different predictions when Y_{it} takes the value 1 or 0 when the value of X_{it} is equal to 0. In addition, the lot-sizing model (1)–(5) does not prevent Y_{it} taking the value 1 when X_i equals 0. If setting the value 1 to variable Y_{it} reduces the capacity consumption forecasted by the machine learning model, the solution may set a setup to 1 even if item i has a lot size of 0 during period t. Although this situation is unlikely to happen due to setup costs, this leads to a situation where the solution of the lot-sizing model is not consistent with reality. Constraints can be employed during the training of the machine learning model to force the prediction to increase when a setup is performed. However, this restriction can decrease the performance of the resulting model. Therefore, to avoid this situation, we impose a minimum lot size ϵ to each item $i \in J$ with a setup by adding the following constraints to the lot-sizing model (1)–(5):

$$\epsilon \cdot Y_{it} \leq X_{it} \quad \forall i \in J, t \in T \tag{14}$$

4.2. Model of capacity consumption with machine learning

We consider three machine learning models to predict capacity consumption, namely, linear regression, piecewise linear regression, and regression tree.

4.2.1. Linear regression

Linear regression is a simple choice when translating a machine learning model into a linear program. The model fitted associates coefficients $\overline{\alpha_f}$ for each feature $f \in \mathcal{F}$, as well as an intercept value $\overline{\alpha_0}$. Linear regression computes capacity consumption of vector $\mathcal{X}_j$ with the following formula:

$$\mathcal{Y} = \sum_{f \in \mathcal{F}} \overline{\alpha_f} \mathcal{X}_{ft} + \overline{\alpha_0}.$$

Therefore, the capacity constraints (4) are replaced by the following equations:

$$\sum_{f \in \mathcal{F}} \overline{\alpha_f} \mathcal{X}_{ft} + \overline{\alpha_0} \leq C_t \quad \forall t \in T, \tag{15}$$

where $\mathcal{X}_{ft}$ is the variable representing features $f \in \mathcal{F}$ in the dataset of period $t \in T$.

4.2.2. Piecewise linear regression

Piecewise linear regression divides the value of one feature $f^* \in \mathcal{F}$ into a discrete set of regions $\mathcal{R}$. Each region is delineated by two consecutive breakpoints in a set of breakpoints $\mathcal{B}$. Each sample of the data falls into one region $r \in \mathcal{R}$ depending on the value of the corresponding feature, and a linear regression is trained on the data points of each region. The vector α_{rf} represents the coefficients of each feature $f \in \mathcal{F}$ for each region $r \in R$.

To translate piecewise linear regressions into a linear program, we add some binary variables Z that determine the region the samples belong to. The resulting linear program is as follows:

$$\mathcal{X}_{f^*t} \leq b_r + H \cdot (1 - Z_{rt}) \quad \forall r \in 2..|\mathcal{R}|, \forall t \in T \tag{16}$$

$$\mathcal{X}_{f^*t} \geq b_r + H \cdot (1 - Z_{(r+1)t}) + \epsilon \quad \forall r \in 1..|\mathcal{R}|-1, \forall t \in T \tag{17}$$

$$\sum_{r \in \mathcal{R}} Z_{rt} = 1 \quad \forall t \in T \tag{18}$$

$$\sum_{f \in F} \overline{\alpha_{fr}} \mathcal{X}_{ft} + \overline{\alpha_{0r}} \leq C_t + H \cdot (1 - Z_{rt}) \quad \forall r \in \mathcal{R}, \forall t \in T \tag{19}$$

Eqs. (16) and (17) define the region to which the regression applies, depending on the value of the selected feature $f^* \in \mathcal{F}$. The sufficiently small value ϵ prevents a feature from being included in two regions. Constraints (18) ensure that only one region is selected for each period $t \in T$. Finally, the capacity constraints associated with this machine learning model are given by (19). For each period, only one capacity constraint is active, depending on the region where the regression occurs.

Finding the best feature that defines the regions requires testing each possible breakpoint. Some studies proposed mathematical models that compute the best feature and breakpoints for the fitting of a piecewise linear function (Rebennack & Krasko, 2020; Yang et al., 2016), but these approaches remain time-consuming and impractical for large models.

To delineate the regions, we select the feature corresponding to the number of setups $\mathcal{F}_{(|J|+5)}$ since the approximation of capacity consumption changes when the product mix changes. The number of breakpoints is a sensitive parameter since increasing the number of regions requires embedding more variables and constraints for the integrated lot-sizing and machine learning model, which increases the computing time significantly. In this work, we propose different sets of breakpoints depending on the size of the scheduling problem considered in the lot-sizing problem.

4.2.3. Regression tree

A regression tree (or decision tree regressor) is a machine learning model that iteratively splits the search space to provide the best prediction value according to the input data $\mathcal{X}$ (Breiman et al., 1983). A regression tree is composed of $\mathcal{N}$ nodes, which include a set of leaf nodes $\mathcal{L}$. Each splitting node works as a query prescribing the path to follow in the tree until falling into a leaf node $i \in L$, which returns the value to predict (here the capacity constraints). In the splitting nodes, the queries are conditions computed based on the features of input $\mathcal{X}$. Each query can be represented as a linear condition on the vector of features $\mathcal{X}$. For each node j in the set of nodes $\mathcal{N}$, these equations are represented as $\sum_{f \in F} A_{jf} \mathcal{X}_{ft} \leq b_j$, where parameter a_{jf} takes the value 1 if feature $f \in \mathcal{F}$ is involved in the splitting node j, and 0 otherwise, and parameter b_j represents the threshold of the splitting condition. If the condition is satisfied, the decision tree moves to the right child node, or to the left child node otherwise. After a number of queries, the tree arrives at a leaf where a score S lies, and this score corresponds to the outcome of the prediction. We adapt the mathematical formulation of Biggs et al. (2022) to embed random forest. In this formulation, binary variables q_{ij}^t indicate, for every node $j \in \mathcal{N}$, if the input $\mathcal{X}$ lies in a leaf node that is a descendant of node k. For each node $j \in \mathcal{N}$, the left and right child nodes are respectively given by l_j and r_j, and the parent node is provided as p_j. This formulation with binary variables represents the path followed in the tree for each data sample $\mathcal{X}$. The score of each leaf $j \in \mathcal{L}$ is provided by S_j. The model is described as follows:

$$\sum_{f \in \mathcal{F}} a_{jf} \mathcal{X}_{ft} - M(1 - q_{j,l_j}^t) \leq b_j, \quad \forall t \in T, j \in \mathcal{N} \tag{20}$$

$$\sum_{f \in \mathcal{F}} a_{jf} \mathcal{X}_{ft} + M(1 - q_{j,r_j}^t) \geq b_j + \epsilon, \quad \forall t \in T, j \in \mathcal{N} \tag{21}$$

$$q^t_{j,r_j} + q^t_{j,l_j} = q^t_{p_j,j}, \qquad \forall t \in T, j \in \mathcal{N} \tag{22}$$

$$\sum_{j \in \mathcal{L}} q^t_{p_j,j} = 1, \qquad \forall t \in T \tag{23}$$

$$\sum_{j \in L} S_j \cdot q^t_{p_j,j} \leq C_t \quad \forall t \in T \tag{24}$$

$$q^t_{j,l_j}, q^t_{j,r_j}, q^t_{p_j,j} \in \{0,1\}, \quad \forall t \in T, i \in \mathcal{N}$$

Constraints (20) state that variable q^t_{j,l_j} takes value 1 if the query at node $j \in \mathcal{N}$ is satisfied, so the predicted value lies in the left subtree of node j. Alternatively, constraints (21) ensure that variable q^t_{j,r_j} takes value 1 if the value predicted by the tree lies in the right subtree of $j \in \mathcal{N}$. Eqs. (22) and (23) state that only one node is active at each stage of the regression tree. Eqs. (24) compare the predicted value provided by the tree and the capacity.

5. Prediction improvement

The training procedure of a machine learning model is a crucial step, and datasets used to train a model have to be carefully selected. As the optimal solutions of mathematical programs often lie in the extreme rays of the feasible region, the optimization model embedding machine learning is prone to explore solutions that are not part of the training dataset (Goodfellow et al., 2014). Thus, the prediction of the machine learning models deteriorates when exploring solutions that are far from the data samples used for training. In particular, the lot-sizing solutions are likely to set the predicted capacity consumption equal to the available capacity. In such situations, a small underestimation of capacity consumption leads to an unfeasible solution. Recent papers addressed this issue by considering trust regions to limit the prediction to the convex hull of the training dataset (Maragno et al., 2023) or adaptative sampling to generate adversarial examples (Cozad et al., 2014). To alleviate these issues, we consider the latter paradigm, and this section presents a training procedure as well as methods to generate accurate data samples.

5.1. Training procedure to prevent infeasible solutions

Training procedures minimize the error between the prediction and the value observed in the dataset. If a regression model does not fit a training dataset perfectly, the prediction may underestimate the real capacity consumption for some training samples. As underestimating capacity consumption leads to infeasible plans, we propose a training approach that overestimates the prediction when fitting a linear model. In other words, the fitting procedure forbids underestimating the capacity consumption in the training dataset.

We propose a MILP to minimize the mean absolute error between the training dataset and the prediction of a linear regression model. This model relies on finding the best weight α_f associated with each feature $f \in \mathcal{F}$ of our regression model while minimizing an absolute error d_s between the actual capacity consumption and the prediction $\mathcal{Y}^{pred}_s$ for each sample $s \in D$.

A classical training model for a linear regression that minimizes the Mean Absolute Error (MAE) is as follows:

$$\min \quad \frac{1}{|D|} \sum_{s \in D} d_s \tag{25}$$

$$\text{s. t.} \quad \sum_{f \in \mathcal{F}} (\overline{\mathcal{X}^s_f} \alpha_f) + \alpha_0 = \mathcal{Y}^{pred}_s, \qquad \forall s \in D \tag{26}$$

$$\mathcal{Y}^{pred}_s - \mathcal{Y}^s \leq d_s, \qquad \forall s \in D \tag{27}$$

$$\mathcal{Y}^s - \mathcal{Y}^{pred}_s \leq d_s, \qquad \forall s \in D \tag{28}$$

$$\mathcal{Y}^{pred}_s \geq 0, d_s \geq 0, \qquad \forall s \in D$$

$$\alpha_f \in \mathbb{R}, \qquad \forall f \in \mathcal{F}.$$

The objective function (25) minimizes the mean absolute error between the output and the prediction. Constraints (26) link the weighted sum of features and the predicted value for each sample of the training dataset. Constraints (27) and (28) compute the absolute errors between the targeted output $\mathcal{Y}^s$ and the value predicted by the linear regression.

To ensure the fitted model overestimates the capacity consumption for all in-sample data points, we forbid negative errors during the training by replacing (28) with the following set of constraints:

$$\mathcal{Y}^{pred}_s \geq \mathcal{Y}^s, \quad \forall s \in D \tag{29}$$

The same process applies to piecewise linear regression. To train such models, we divide the dataset D into $|\mathcal{R}|$ smaller datasets depending on the region where each data point falls. We fit a linear regression to each of these datasets by using this new fitting procedure. We then consider these two machine learning models, named constrained linear regression (CLR) and constrained piecewise linear regression (CPLR) in the next section of this paper.

5.2. Data generation procedure

Fitting our machine learning models requires datasets that correspond to historical data from actual production schedules implemented on the shop floor. However, we may take advantage of available scheduling or simulation tools to generate data points that help train the machine learning model. For proper comparison with methods from the literature, we generate the dataset by solving flexible job-shop scheduling problems. Each dataset is associated with one scheduling instance, where each data sample corresponds to one vector of lot sizes X applied to each job. Generating data samples of a flexible job-shop scheduling instance requires both the quantities of each item and the associated makespan (or capacity consumption). The other features described in Section 4 are inferred from the lot sizes for each data sample. To build the training dataset, we generate some lot sizes X_i for each item $i \in J$, and associate each of them with the processing time of each operation of each job i of the corresponding flexible job-shop scheduling problems. We solve each sample with the MILP (see in the Supplementary Materials (A.1)–(A.5)) with a single period ($|T| = 1$). Note that the hardest instances were solved with constraint programming approaches. The rest of this section describes approaches to generate lot sizes examples.

5.2.1. Random procedure

One standard idea for data sampling is to generate the lot sizes X_i randomly. Advanced sampling methods such as Latin Hypercube Sampling (LHS) generate samples that cover the input space more evenly than simple Monte Carlo procedures (Mckay et al., 2000). This sampling method works by dividing the input space into $|\mathcal{F}|$ bins of identical sizes. The data samples are generated so that no two samples fall into the same bin. However, the samples generated using LHS may not represent the solutions that can be found by solving a lot-sizing model. For instance, randomly sampled lot sizes may have very small lots for some item $i \in J$, which is not coherent with the high setup costs that can be encountered in lot-sizing problems.

5.2.2. Iterative training procedure

This section suggests a practical enhancement where we look for adversarial examples by running a simulation. Fig. 1 summarizes the procedure. In each iteration, we solve a randomly generated instance of the lot-sizing problem, and we solve the associated scheduling problems to check if the capacity is violated in any period. Each sample that is underestimated by the machine learning method is added to the training dataset D, and the machine learning method is fitted into this new dataset. The method stops after solving a given number *max_iter* of lot-sizing instances without finding periods where the capacity is violated.

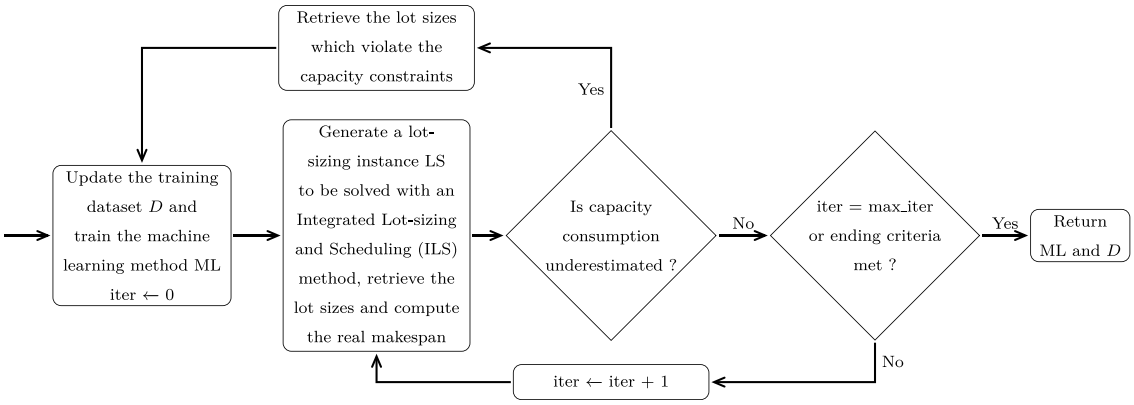

Fig. 1. Flow chart for ILS-based.

However, the approach remains time-consuming for complex instances, with large scheduling sizes or parameters such as setup costs. We proposed an approach (denoted as ILS-KP), that better identifies wrongly predicted samples better. The idea is to generate lot-sizing solutions with tight capacity and with different structures. To generate a wide variety of production plans, we randomly associate a profit λ_i with each item i, and we seek solutions that maximize the profit while respecting the capacity constraint. The capacity consumption is determined through the machine learning model ML translated into a mathematical program as described in Section 4. Solving the following MILP a large number of times with different weights yields various solutions with tight capacities:

$$\max \quad \sum_{i \in J} \lambda_i X_i \tag{30}$$

$$\text{s. t.} \quad (6)\text{–}(14) \tag{31}$$

$$ML(\mathcal{X}) \leq C \tag{32}$$

$$Y_i \in \{0,1\}, \qquad X_i \geq 0 \quad i \in J$$

The objective function (30) maximizes the profit of each item $i \in J$ while satisfying the capacity constraint (32). Constraints (6)–(14) compute the features. Constraint (32) approximates the capacity consumption with a machine learning model ML translated into a mathematical program for the input vector $\mathcal{X}$. Each iteration of this procedure generates a vector of profit λ as well as a capacity C, and we solve (30)–(32).

In the case of linear regression, this approach is close to a continuous knapsack formulation with a profit $\lambda_i \in \mathbb{R}$ for each item and a capacity C. In the numerical experiments, we run this procedure with the ILS-KP model, and the method stops when there is no more than ρ percent of underpredicted schedules in the last N iterations. To ensure that our machine learning models always overestimate the capacity consumption, the intercept of the constrained linear regression trained using this procedure is increased by the difference between the forecast and the real makespan of the last example that was underpredicted.

In addition to this iterative procedure, an exact method for finding adversarial examples is presented in Appendix G.

6. Numerical experiments

This section summarizes the results of the computational experiments. We define the lot-sizing instances in Section 6.1. Then, we compare the performance of all machine learning models in Section 6.2. Sections 6.3 and 6.4 provide the results of the machine learning models compared to standard mathematical models for integrated lot-sizing and scheduling. Finally, we propose an iterative lot-sizing and scheduling approach in Section 6.5. Experiments assessing both the prediction performance of models that approximate the capacity consumption and the data generation procedures are also proposed in Appendix F. All the experiments were conducted on computers with Intel Xeon Broadwell EP E5-2630v4 @ 2,20 GHz and 124 Go of RAM. The mathematical models were solved using IBM ILOG CPLEX 20.1.0.0 running with one thread. The linear regression and regression tree were trained using the Scikit-learn (Pedregosa et al., 2011) package from Python, with a maximum depth of 10 to limit the number of variables and constraints in the MILP formulation. The constrained linear regression and constrained piecewise linear regression were fitted using CPLEX to minimize the mean absolute error. Note that we also tried to fit these machine learning models using the same mathematical model but minimizing the mean squared error using a quadratic objective. However, we observed no significant improvement by considering the mean squared error instead of the mean absolute error.

6.1. Instance definition

To generate the lot-sizing instances we adopt the procedure given in Wolosewicz et al. (2015). At the scheduling level, the flexible job-shop scheduling instances mt06, mt10, and mt20 from Hurink et al. (1994) are considered. The Supplementary Materials details the instances generation procedure.

We compare the proposed approach with two methods from the literature, denoted by `ILS-Exact` and `ILS-Fixed`. `ILS-Exact` is similar to `ILS-CLSP` but it replaces constraints (4) with constraints (A.1)–(A.5). The resulting MILP solves the integrated lot-sizing and flexible job-shop problem. This model provides perfect information on capacity consumption at each period since it simultaneously finds the best quantities for each item and sequences the operations on the shop floor.

Wolosewicz et al. (2015) propose an approach that solves the lot-sizing problem with a fixed sequence of operations for the job-shop scheduling problem. The new lot-sizing model is less complex and solved with a heuristic based on Lagrangian relaxation. However, they only considered one possible sequence of operations for the scheduling problem. We denote by `ILS-Fixed` the lot-sizing model with a fixed sequence of operations for the scheduling problem as presented in Wolosewicz et al. (2015). We consider the sequence that is the solution to the flexible job-shop scheduling problem with lot sizes equal to 1 for each job.

We summarize below all the mathematical models used to solve the lot-sizing instances:

- `ILS-CLSP` : Capacitated Lot-sizing problem (1)–(5)
- `ILS-CLSP75` : Capacitated Lot-sizing problem with capacity reduced by 75%
- `ILS-Exact`: Integrated Lot-sizing and Flexible Job-shop Scheduling (A.1)-(A.5)

Table 1
Comparison between constrained and standard machine learning models.

	Size	6 × 6			10 × 10			20 × 5		
	T	5	30	50	5	30	50	5	30	50
ILS-LR	UB	1600	9368	15 517	2659	15 309	25 746	5323	30 423	51 996
	LB	1600	9368	15 517	2659	15 309	25 746	5323	30 423	51 996
	Gap (%)	0	0	0	0	0	0	0	0	0
	Feasibility	88	18	5	89	12	7	83	72	67
	Time (s.)	0.08	3.11	8.1	0.15	11.5	81.7	0.07	1.2	2.07
ILS-CLR	UB	1603	9382	15 537	2664	15 361	25 833	5332	30 470	52 034
	LB	1603	9382	15 537	2664	15 361	25 819	5332	30 470	52 034
	Gap (%)	0	0	0	0	0	0.05	0	0	0
	Feasibility	100	100	98	100	99	96	100	100	100
	Time (s.)	0.02	1.66	7.5	0.03	325.0	3409	0.09	19.1	142.1
ILS-RT	UB	1600	9367	15 518	2659	15 308	25 742	5324	30 443	52 153
	LB	1600	9367	15 518	2659	15 308	25 739	5324	30 424	51 996
	Gap (%)	0	0	0	0	0	0.004	0	0.06	0.3
	Feasibility	91	50	28	88	19	3	72	59	53
	Time (s.)	2.99	327.0	1154.5	10.8	1443.6	3263.6	111.1	3568.7	3600
ILS-PLR	UB	1601	9371	×	2659	15 308	×	5321	30 420	×
	LB	1601	9371	×	2659	15 308	×	5321	30 420	×
	Gap (%)	0	0	×	0	0	×	0	0	×
	Feasibility	64	5	0	64	3	0	7	1	0
	Time (s.)	0.02	0.2	0.4	0.02	0.3	0.43	0.07	0.7	1.45
ILS-CPLR	UB	1602	9379	15 527	2665	15 342	25 742	5333	30 462	52 023
	LB	1602	9379	15 527	2665	15 342	25 718	5333	30 462	52 023
	Gap (%)	0	0	0	0	0	0.09	0	0	0
	Feasibility	91	75	67	100	97	91	100	100	98
	Time (s.)	0.05	14.4	321.3	0.06	1645.7	3597.1	0.1	6.7	15.1

- `ILS-Fixed`: Integrated Lot-sizing and Scheduling with a fixed sequence
- ILS-LR: Integrated Lot-sizing and Scheduling with Linear Regression (15)
- ILS-PLR: Integrated Lot-sizing and Scheduling with Piecewise Linear Regression (16)–(19)
- ILS-RT: Integrated Lot-sizing and Scheduling with Regression Tree (20)–(24)
- ILS-CLR: Integrated Lot-sizing and Scheduling with Constrained Linear Regression
- ILS-CPLR: Integrated Lot-sizing and Scheduling with Constrained Piecewise Linear Regression

Both `ILS-CLR` and `ILS-CPLR` have been trained with $\rho = 100\%$ and $N = 10{,}000$ to ensure the feasibility of the production plans obtained. Since these models can be restrictive, we also considered two additional models, `ILS-CLR95` and `ILS-CPLR95`, trained with $\rho = 95\%$ and $N = 1000$. All the lot-sizing models were solved by CPLEX with a time limit of 1 h.

6.2. Machine learning models comparison

This section reports the performance of different lot-sizing models that embed machine learning methods to approximate the capacity constraint. We compare the performance of different machine learning models and different training methods. First, we analyze the solution quality and the feasibility of production plans obtained by embedding different machine learning models. For each scheduling size and each period horizon, we generate 100 lot-sizing instances. To check the feasibility of the solution returned by the models, the production quantities in each period are associated with a scheduling problem, and the solution of this scheduling problem gives the actual capacity consumption. A solution to the lot-sizing model is infeasible either if no feasible solution has been found by the solver after reaching the time limit or the solution returned by the solver includes at least one period where the capacity is exceeded.

Table 1 reports the performance of all the embedded machine learning and lot-sizing models presented in Section 6.1. The metrics used to compare the solutions are the upper bounds UB, lower bounds LB, and the relative gap found returned by CPLEX. Note that these metrics are provided only for the instances where all methods find a feasible solution.

Table 1 shows that most of the solutions returned by the linear regression and regression tree are infeasible, while the constrained approach leads to a large percentage of feasible solutions. When compared to linear regression, regression trees appear to perform badly, since this approach requires a significant computational time to find optimal solutions. The number of variables and constraints grows exponentially with the size of the tree. For example, a regression tree with a depth of 20 can include a total of 2^{20} nodes, which leads to at least 2^{20} variables and three times more constraints for each period in the horizon. The resulting mathematical model rapidly becomes impractical when the number of periods increases. Although models learned without the constraint that prevents underapproximation of the capacity consumption have high precision, they struggle to find solutions that respect capacity consumption. The importance of the constrained learning approach is clear, and we keep only the constrained machine learning models for the rest of the experiments.

6.3. Performance of the proposed approach

This section compares the state-of-the-art models `ILS-Exact`, `ILS-CLSP`, and `ILS-Fixed` with lot-sizing models that embed constrained regression, namely `ILS-CLR` and `ILS-CPLR`. We generate instances for this experiment by varying the scheduling size, horizon length, and setup costs. Machine learning models were trained using the ILS-KP method proposed in Section 5.2.2.

Table 2 reports the results on all the instances, aggregated per scheduling size, period, and setup costs. For each of these parameters, we considered the percentage gap (denoted by Gap^B) of each model over the best solution found, the percentage of plans that are feasible, the average absolute gap between the estimated capacity consumption and the real makespan (denoted by Gap^{MAE}), and the computational time in seconds. For a solution where the production plans resulted in impractical schedules, the percentage of violated capacity is provided in brackets.

The Supplementary Materials gives detailed tables with the results for each instance.

Table 2
Results for lot-sizing models aggregated by scheduling size, period, and setup costs.

	Metrics	Scheduling			Period			Setup costs		
		6 × 6	10 × 10	20 × 5	5	30	50	15	50	100
ILS-Exact	Gap^B (%)	0.31	0.07	1.31	0.07	0.45	0.47	0.18	0.07	0.66
	Feasibility (%)	99.3	55.6	8.6	74.8	44.8	43.9	75.2	44.2	44.0
	Gap^{MAE} (%)	11.7	26.5	43.5	25.0	13.2	13.3	28.7	11.6	8.5
	Time (s.)	1666.8	3378.6	3600.1	2201.4	3203.9	3240.2	2225.7	3202.0	3217.8
ILS-CLSP	Gap^B (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	×	×
	Feasibility (%)	15.8 (14.3)	15.7 (15.6)	2.4 (13.6)	16.0 (18.5)	11.1 (12.8)	6.8 (12.2)	33.9 (4.4)	0.0 (16.0)	0.0 (23.2)
	Gap^{MAE} (%)	8.3	11.1	18.7	14.2	12.1	11.9	11.2	12.9	14.0
	Time (s.)	51.1	233.3	84.7	0.1	20.5	348.6	0.1	2.3	366.7
ILS-CLSP75	Gap^B (%)	0.86	0.1	0.1	0.66	0.15	0.13	0.16	0.94	3.4
	Feasibility (%)	42.8 (3.0)	33.4 (5.9)	43.0 (4.7)	53.6 (4.5)	33.1 (4.8)	32.6 (4.4)	97.8 (1.8)	17.8 (4.3)	3.7 (6.3)
	Gap^{MAE} (%)	18.7	15.9	8.8	14.7	14.4	14.4	16.5	13.5	13.4
	Time (s.)	421.5	598.9	1433.5	0.5	850.8	1602.7	0.8	655.7	1797.5
ILS-Fixed	Gap^B (%)	0.53	0.82	4.47	1.78	1.64	2.4	0.25	1.98	3.59
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Gap^{MAE} (%)	6.2	12.8	30.0	17.5	15.6	16.0	22.6	14.4	12.1
	Time (s.)	1534.9	2013.0	2834.6	443.7	2781.7	3157.1	1205.2	2369.3	2808.0
ILS-CLR	Gap^B (%)	3.61	0.68	1.25	2.31	1.66	1.57	0.27	1.78	3.48
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Gap^{MAE} (%)	22.9	17.5	24.9	23.2	21.0	21.1	27.1	20.3	18.0
	Time (s.)	1606.5	1681.7	2190.3	208.8	2440.9	2828.8	468.8	2401.2	2608.5
ILS-CPLR	Gap^B (%)	3.22	1.05	1.01	1.63	1.62	1.93	0.18	1.62	3.42
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Gap^{MAE} (%)	22.7	21.8	22.9	25.0	21.1	21.3	25.1	22.1	20.2
	Time (s.)	2074.1	1995.0	2018.1	286.0	2531.8	3269.5	1001.4	2402.1	2683.8
ILS-CLR95	Gap^B (%)	4.4	0.44	0.45	2.18	1.61	1.5	0.3	1.72	3.29
	Feasibility (%)	100.0	100.0	99.3 (0.8)	100.0	99.7 (0.8)	99.7 (0.8)	100.0	99.9 (1.0)	99.4 (0.7)
	Gap^{MAE} (%)	27.7	16.5	22.9	24.0	21.6	21.6	27.8	20.9	18.5
	Time (s.)	1605.8	1642.1	1865.3	127.3	2412.6	2573.3	185.9	2400.9	2526.4
ILS-CPLR95	3.95	0.87	0.07	1.57	1.53	1.81	0.21	1.46	3.27	
	Feasibility (%)	99.8 (0.3)	100.0	98.4 (1.0)	100.0	99.3 (1.7)	98.9 (0.5)	100.0	99.8 (0.4)	98.4 (0.9)
	Gap^{MAE} (%)	26.3	18.4	18.8	23.0	20.2	20.3	24.2	20.8	18.5
	Time (s.)	2220.5	2050.3	1814.1	213.1	2690.7	3181.1	1071.7	2401.6	2611.7

For most instances of size 6 × 6, ILS-Exact finds at least one feasible production plan within the time limit, but it struggles to find feasible solutions when the size of instances increases. ILS-CLSP returns solutions within a reasonable computational time, and the feasible ones represent the best production plan. However, the large majority of solutions found by ILS-CLSP are not feasible at the scheduling level. Such solutions are undesirable, and the reliability of this model remains low when compared to the other approaches. Setup costs greatly impact the complexity of the instances since the plans resulting from their solutions include large lot sizes for some periods to avoid high setup costs. These solutions tighten the capacity constraints, leading to production plans that tend to be infeasible. For the infeasible solutions of ILS-CLSP, the capacity is highly violated, particularly in scenarios with large setup costs. The safety capacity feature prevents the model from utilizing more than 75% of the available capacity. While this parameter enhances the feasibility of solutions, a significant number of infeasible plans persist. This model still exceeds the capacity by 6% for instances with large scheduling sizes. To ensure feasibility in all instances, the model must account for a large safety capacity which would lead to poor quality solutions.

ILS-Fixed proposes the best trade-off between objective values and feasibility for small-size instances. However, ILS-CPLR outperforms the ILS-Fixed model when the instance size increases. Increasing the number of periods does not impact the overall performance of ILS-CLR and ILS-CPLR, whereas it decreases the quality of solutions for ILS-Fixed. However, increasing the setup costs has an impact on the solutions found by all the models, even if ILS-CLR and ILS-CPLR remain better on average. Our intuition is that lower setup costs imply small quantities of items for each period, which remain relatively easy to approximate for scheduling with a fixed sequence. Larger setup costs involve large lot sizes and multiple items produced at the same time, resulting in complex scheduling problems that are inadequately approximated with a single sequence. In this case, machine learning approaches forecast a more accurate capacity consumption on average, leading to better solutions for medium and large instance sizes. The models trained to reach 95% of feasibility provide the production plans with the lowest cost at the expense of a small decrease in feasibility ratio. Finally, machine learning based approaches are much less demanding in terms of computational efforts than lot-sizing models that integrate the full scheduling decisions.

The gap Gap^{MAE} between capacity consumption approximation and real makespan varies drastically between instance types and models. Most of the solutions of ILS-CLSP and ILS-CLSP75 models compute capacity consumptions that slightly underestimate or overestimate the real makespan, leading to low Gap^{MAE} when compared to other approaches. ILS-Exact and ILS-Fixed compute the capacity consumption by solving the scheduling problem and only feasible schedules are necessary. Therefore, these models may find optimal solutions at the lot-sizing level computed with nonoptimal schedules, resulting in a high Gap^{MAE} between the capacity consumption found at the lot-sizing level and the optimal makespan of the resulting scheduling problems. Machine learning models predict bad capacity consumptions for small scheduling sizes, but outperform the Fixed approach on large scheduling sizes.

6.4. Performance on different parameters

Lot-sizing parameters significantly impact the solution quality of the optimal plans. This subsection evaluates the performance of the proposed approach on different metrics. For this set of experiments, we vary three standard parameters of the lot-sizing problems: demand variations, setup times, and backlog costs as well as the non-allowance of backlog.

Table 3
Results for lot-sizing models by varying backlog costs, setup times, and demand.

	Metrics	Backlog costs			Setup times			Demand		
		Forbidden	1	3	None	×3	×5	[4,8]	[10,50]	[10,100]
ILS-Exact	Gap^B (%)	0.04	0.02	0.12	0.99	0.19	0.03	0.04	0.12	0.05
	Feasibility (%)	49.7	45.2	70.1	89.2	53.8	41.7	44.6	89.2	90.0
	Gap^{MAE} (%)	28.0	15.4	30.2	34.4	27.6	24.8	22.5	60.6	59.8
	Time (s.)	2232.4	2858.1	2272.7	2113.5	2627.6	2954.4	3173.3	987.0	945.3
ILS-CLSP	Gap^B (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Feasibility (%)	33.6 (4.2)	2.3 (9.6)	33.6 (4.4)	46.7 (3.6)	18.9 (5.2)	10.4 (5.8)	2.3 (7.6)	100.0	100.0
	Gap^{MAE} (%)	11.2	10.7	11.2	9.2	14.9	18.3	14.3	10.5	9.0
	Time (s.)	0.1	0.3	0.1	0.2	0.2	0.4	0.3	0.1	0.1
ILS-CLSP75	Gap^B (%)	0.16	0.14	0.14	0.15	0.44	0.18	1.11	0.02	0.01
	Feasibility (%)	97.9 (2.1)	49.0 (3.1)	96.8 (1.7)	99.0 (1.3)	91.8 (2.6)	49.0	58.7 (2.5)	100.0	100.0
	Gap^{MAE} (%)	16.5	15.2	16.5	19.2	12.4	15.0	13.0	19.5	21.4
	Time (s.)	0.6	3.0	0.7	0.4	800.1	0.1	440.2	0.1	0.1
ILS-Fixed	Gap^B (%)	0.25	0.37	0.24	0.22	0.32	0.67	1.29	0.02	0.01
	Feasibility (%)	99.9	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Gap^{MAE} (%)	22.8	23.0	22.8	24.3	19.0	15.5	25.4	26.3	27.5
	Time (s.)	1182.8	1776.6	1193.2	1150.6	1359.2	1510.8	1845.9	0.2	0.2
ILS-CLR	Gap^B (%)	0.28	0.31	0.26	0.29	0.38	0.58	1.33	0.02	0.02
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Gap^{MAE} (%)	27.0	23.0	27.0	29.9	26.6	22.6	23.6	31.2	32.2
	Time (s.)	449.8	1903.0	443.5	227.8	1217.1	1404.5	2112.1	0.1	0.2
ILS-CPLR	Gap^B (%)	0.19	0.24	0.17	0.17	0.21	0.23	1.24	0.02	0.01
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Gap^{MAE} (%)	25.0	22.5	25.0	26.1	23.1	20.2	24.9	22.9	22.8
	Time (s.)	970.0	2400.3	970.2	794.8	1890.8	2292.8	2400.2	0.2	0.2
ILS-CLR95	Gap^B (%)	0.29	0.33	0.28	0.29	0.38	0.58	1.44	0.03	0.03
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Gap^{MAE} (%)	27.8	23.8	27.7	29.9	26.5	22.5	25.3	30.4	31.1
	Time (s.)	165.3	1801.5	144.5	246.8	1225.6	1415.7	2138.7	0.2	0.2
ILS-CPLR95	Gap^B (%)	0.2	0.23	0.19	0.16	0.21	0.23	1.21	0.02	0.01
	Feasibility (%)	100.0	99.9 (0.6)	100.0	99.6 (2.1)	99.7 (1.5)	99.4 (1.6)	99.7 (1.0)	100.0	100.0
	Gap^{MAE} (%)	23.0	19.5	22.9	24.6	23.0	20.2	23.3	24.4	25.0
	Time (s.)	1059.1	2369.4	1074.6	847.2	1971.0	2306.6	2400.2	0.2	0.2

Table 3 reports the results aggregated on the three considered parameters. While the performance of classical models (ILS-CLSP, ILS-CLSP75, ILS-Exact) varies with the parameters, the machine learning-based approaches are robust, and their performances are competitive with the fixed scheduling approach for all types of instances. For example, reducing the backlog costs drastically reduces the number of feasible plans found by both ILS-CLSP and ILS-CLSP75 . When backlog cost is low, the lot-sizing solutions are likely to postpone production to reduce setups, and thus the estimated capacity consumption in a period is closer to capacity.

Similarly, when no setup times are considered, ILS-Exact can find a large number of feasible plans. However, the solutions found for large-size instances remain bad compared to the other approaches considered in this work. Note that the feasibility increases with the demand since the capacity calculation takes demands into account. For small demands, the capacity is reduced and it becomes difficult to respect the capacity constraints. Similarly, increasing the setup times leads to schedules with high makespans, which reduces the Gap between the optimal makespan and the approximated one.

6.5. Iterative lot-sizing and scheduling approach

In this subsection, we investigate an iterative procedure to solve the integrated lot-sizing and scheduling problem by repeatedly solving the scheduling problem obtained from the lot-sizing decisions. Each iteration of this algorithm consists of solving the lot-sizing problem with safety capacities for each period and computing the schedules of the resulting solution to adjust each safety capacity. If the capacity constraints are violated for at least one period, the available capacity is further reduced to limit the capacity consumption. Similarly, in the case of a feasible production plan, the capacity is increased in the hope of reaching better solutions. We describe the iterative procedure as follows:

1. Set a percentage of available capacity σ_t and decay μ_t for each period $t \in T$.
2. Solve the lot-sizing model for 300 seconds, with capacity $C_t = \sigma_t \cdot C_t, \quad \forall t \in T$.
3. If the problem is infeasible or no feasible solution is reached, increase $\sigma_t = \sigma_t + \mu_t$, decrease $\mu_t = 0.5\mu_t$ and go to Step 1.
4. If a feasible solution is obtained, solve the scheduling problems based on the lot sizes obtained for each period.
5. If the schedules respect the capacity in all periods, increase the available capacity $\sigma_t = \sigma_t + \mu_t$ and decrease $\mu_t = 0.5\mu_t$ and go to Step 1.
6. If the capacity is violated for a set of periods $\overline{T}$, reduce the available capacity $\sigma_t = \sigma_t - \mu_t$ and decrease $\mu_t = 0.5\mu_t$ for each period $t \in \overline{T}$ and go to Step 1.

In our experiments, the procedure stops after a time limit of 3600 s or if μ_t is smaller than 0.001 for any $t \in T$, and the best feasible plan obtained so far is returned as the final solution. To solve the lot-sizing problems, we considered ILS-CLSP, ILS-Fixed and ILS-CLR95 models, denoted respectively by H-CLSP, H-Fixed and H-CLR95 in the experiments. The latter model leads to the best trade-off between feasibility, solution quality, and computational time among all other machine learning models.

Parameters σ_t and μ_t were defined for each model based on preliminary experiments. For H-CLSP, we started with $\sigma_t = 0.8$ and $\mu_t = 0.1$, since most of the solution obtained with this model results in infeasible production plans. On the contrary, H-CLR95 achieved a large number of feasible plans, so we considered $\sigma_t = 1.0$ and $\mu_t = 0.15$ for ILS-CLR95. For ILS-Fixed, we considered a slightly different procedure similar to the one proposed by Dauzère-Pérès and Lasserre (1994). Instead of adjusting the available capacity in Step 5 and 6, the fixed sequence of operations in the lot-sizing model is replaced by the

Table 4
Aggregated results for heuristical approach.

	Metrics	Scheduling			Period			Setup costs		
		6 × 6	10 × 10	20 × 5	5	30	50	15	50	100
H-CLSP	Gap^B (%)	2.13	0.67	1.26	0.84	1.23	2.17	0.01	0.72	3.8
	Feasibility (%)	98.7	92.0	89.1	99.9	96.1	83.8	100.0	97.7	82.1
	Time (s.)	854.3	1266.8	1914.2	89.9	1664.8	2280.6	410.2	1278.6	2346.4
H-Fixed	Gap^B (%)	0.03	2.53	6.25	0.25	2.32	6.24	0.04	0.98	7.79
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Time (s.)	892.4	1274.9	2271.9	260.4	1945.6	2233.1	927.3	1887.0	1624.8
H-CLR95	Gap^B (%)	2.44	0.23	2.0	0.46	1.21	3.0	0.02	0.66	3.99
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Time (s.)	1601.1	1731.9	2208.8	277.3	2519.0	2745.5	478.4	2419.7	2643.8
ILS-CLR	Gap^B (%)	3.37	1.14	2.06	2.91	2.08	1.57	0.32	2.54	3.7
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Time (s.)	1606.5	1681.7	2190.3	208.8	2440.9	2828.8	468.8	2401.2	2608.5

solutions of the scheduling problems found in Step 4. Similarly, this procedure is stopped after a time limit of 3600 s or if the cost of the production plan has not improved after 3 iterations.

Table 4 provides the results of the iterative approach when the main parameters of the lot-sizing problems vary. The results show that even in the iterative framework, H-CLSP does not reach 100% feasibility for most instances. This finding shows that ensuring a feasible plan is a complex task, and the machine learning method provided in our work will strongly benefit the manufacturing industry. Similarly, H-Fixed performs well on small-size instances, but it leads to large costs on large-size instances. Although H-Fixed guarantees feasible production plans, this approach generally fall into weak local optima. On the contrary, embedding CLR95 in the iterative framework reaches solutions with 100% feasibility and lower costs than the standalone version. However, for large-size instances and for instances with large setup costs, H-CLSP outperforms H-CLR95. As the probability of violating the capacity for one period increases for large horizon instances, the procedure struggles to find feasible plans in the first iterations.

6.6. Consideration of uncertainty

This section evaluates the effectiveness of embedding machine learning models to deal with uncertainty on the shop floor. We consider the case where the duration of the operations is uncertain. More specifically, after solving the lot-sizing problem, the production schedule to be executed on the shop floor during each period corresponds to the solution π of the scheduling subproblems. This scheduling is optimized based on the estimated processing time p_{ijk} of each operation O_{ij} on machine k. However, variations in processing durations are common in practice. To account for this, we calculate the actual makespan based on the schedule π, but using actual processing times generated from a uniform distribution $\mathcal{U}\left(p_{ijk} - \omega \cdot p_{ijk},\ p_{ijk} + \omega \cdot p_{ijk}\right)$, where $\omega \in [0, 1]$ represents the level of uncertainty. This methodology is used to simulate the execution of processes that are generally uncertain and differ from their estimated processing time when executed on the shop floor. The gap between the expected makespan and the observed one often leads to an infeasible schedule in practice.

For this set of experiments, we compare ILS-Fixed and ILS-CLR, as these models yielded the highest percentage of feasible production plans in deterministic lot-sizing problems. For ILS-CLR, the linear model is trained using the ILS-KP method, following the same approach as previously described, but accounting for scheduling problems with uncertain processing times. The features are computed using the processing times p_{ijk} from deterministic scheduling problems, while the makespan in the training dataset is calculated based on actual processing times generated randomly with the procedure presented above. Note that we consider one trained model for each value of ω.

Table 5 shows the results of these two models for scheduling size 10 × 10 and setup costs of 15. For the experiments, we evaluated the models with parameter ω taking values 0.05, 0.15, and 0.3. The table shows that the ILS-Fixed model returns production plans with lower costs, but with very few feasible production plans, especially for $\omega = 0.3$. On the other hand, the ILS-CLR model returns 100% feasible production plans for $\omega = 0.05$ and $\omega = 0.15$, and finds a larger number of feasible plans compared to the ILS-Fixed approach. These results show that the capacity consumption calculation from embedded machine learning models deals efficiently with task time uncertainty in the scheduling part.

7. Conclusions and discussions

This paper presents innovative lot-sizing models that rely on machine learning to improve the approximation of capacity consumption. The resulting model is interesting for application in the manufacturing industry since it leads to lower production costs, and it ensures APS systems provide plans that are executable in the workshop. We have investigated machine learning models based on linear regressions, piecewise linear regressions, and decision trees to predict capacity consumption. As we incorporate these machine learning models into optimization approaches, they must be appropriately trained to avoid underestimating capacity consumption. Therefore, we constrain the learning process to avoid underestimating the capacity of training samples. In addition, we propose an iterative training sample generator that helps to train the machine learning model efficiently. For large-scale instances, the resulting approach outperforms the state-of-the-art lot-sizing models considered for comparison in this paper, including the integrated approach proposed by Dauzère-Pérès and Lasserre (1994) and the fixed scheduling model proposed by Wolosewicz et al. (2015). Our approach provides solutions with lower total costs, in short computational time, and these solutions are feasible for each period taking into account the scheduling constraints.

In addition, constrained machine learning models trained with the iterative procedure result in small final datasets, that are less than 1200 samples. Also, since the large-size scheduling instances used to train the models include a relative gap of around 15%, the models perform well even when trained with nonoptimal schedules or with few available data samples from scheduling. We observed that machine learning models underestimate capacity consumption when trained on large-size datasets (more than 100,000 samples) with classical training approaches. The iterative training approach we propose in our paper alleviates this issue. Therefore, we recommend practitioners use all the available data samples when training the models and, if possible, employ procedures to enhance the prediction, rather than relying on large datasets. While we only consider offline learning in our paper, in practice, the model could be retrained whenever an infeasible plan occurs. More generally, we can retrain the machine learning model regularly to account for the schedule implemented in the last periods.

Many extensions of lot-sizing problems include parameters such as setup carryover or overtime, and our formulation may be easily adapted

Table 5
Results for lot-sizing and scheduling with uncertain processing times.

ω		0.05			0.15			0.3		
Period		5	30	50	5	30	50	5	30	50
ILS-Fixed	UB	2652.6	15 677.7	26 118.6	2652.6	15 677.7	26 118.6	2652.6	15 677.7	26 118.6
	LB	2652.6	15 677.7	26 110.1	2652.6	15 677.7	26 110.1	2652.6	15 677.7	26 110.1
	Gap (%)	0.0	0.0	0.0003	0.0	0.0	0.0003	0.0	0.0	0.0003
	Feasibility (%)	98.0 (0.3)	91.0 (1.2)	80.0 (1.7)	93.0 (2.6)	56.0 (4.1)	36.0 (4.0)	71.0 (6.4)	4.0 (6.3)	0.0 (6.8)
	Time (s.)	0.1	425.5	3204.1	0.1	427.5	3202.1	0.1	424.5	3200.7
ILS-CLR	UB	2656.2	15 694.4	26 146.2	2658.3	15 708.3	26 169.1	2661.3	15 727.9	26 201.8
	LB	2656.2	15 694.4	26 146.2	2658.3	15 708.3	26 163.6	2661.3	15 727.9	26 189.0
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0002	0.0	0.0	0.0005
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	97.0 (2.1)	78.0 (2.1)	54.0 (2.2)
	Time (s.)	0.0	20.9	779.8	0.1	87.7	2262.5	0.1	271.7	3361.2

by involving the corresponding variables in the training process. As explained in Section 4.1, many features related to the lot-sizing parameter can be included in the training process, and the prediction may differ from one value to another. For example, setup carryover can be considered by adding binary variables indicating information about the sequencing of the resources, as well as additional constraints to allow the carryover of a setup if the conditions are met. This information can then be easily retrieved from production schedules and integrated as features in the training. As shown in the experiments, integrating machine learning models into lot-sizing can help to deal with uncertainty if the schedules used to train the models are built on uncertain environments, such as uncertain processing times or machine breakdown. Also, we restrict our work to single-level lot-sizing problems, where all the operations related to the same item have identical lot sizes. However, the consideration of a bill of materials for products, such as in multi-level lot-sizing problems, provides a more detailed representation of the operations on the shop floor and would be highly beneficial for our approach. In addition, the inventory level can be added as a feature for our machine learning models to express the consumption of components for the immediate production of successors, leading to a more accurate approximation of capacity consumption.

The proposed approach aims to complement MRP software by providing more accurate capacity consumption, but not to replace advanced planning and scheduling systems. The machine learning and lot-sizing models do not provide any information on the sequencing decisions. Our approaches only benefit at the lot-sizing level where quantities are determined, and not at the scheduling level to sequence the operations on the shop floor, although the approximation of the makespan of such approaches can help in approximating the scheduling problem. The proposed approach aims to complement MRP software by providing more accurate capacity consumption, but not to replace advanced planning and scheduling systems.

This initial work was conducted in a controlled environment, where we checked if the plans were feasible by solving a scheduling problem. Future work must investigate the possibility of learning capacity consumption from real data collected from manufacturing execution systems (MES), which is one of the objectives of our current European Project ASSISTANT (Castañé et al., 2022). An intermediate step might study the case where feasibility on the shop floor is checked in a detailed simulation. Such a detailed simulation will provide data for complex shop floors with many machines and jobs, and it may incorporate the instability commonly encountered in workshops, where a given production load may be feasible in one week but not in the next one (because of machine breakdown, or other uncertainties). Other interesting avenues for future research include the generalization of machine learning tools to other machine learning models such as neural networks. The models are also specifically trained to predict capacity consumption in a unique scheduling environment, and each model should be retrained when changing the configuration of the shop floor. The generalization of makespan prediction to each scheduling size would be highly beneficial.

CRediT authorship contribution statement

David Tremblet: Writing – review & editing, Validation, Software, Methodology. **Simon Thevenin:** Writing – review & editing, Validation, Supervision, Methodology. **Alexandre Dolgui:** Writing – review & editing, Validation, Supervision, Methodology.

Acknowledgments

The present work was conducted within the project ASSISTANT (https://assistant-project.eu/) 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.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ejor.2024.11.039.

References

Almeder, C., Klabjan, D., Traxler, R., & Almada-Lobo, B. (2015). Lead time considerations for the multi-level capacitated lot-sizing problem. *European Journal of Operational Research*, *241*(3), 727–738.

Atamtürk, A., & Hochbaum, D. S. (2001). Capacity acquisition, subcontracting, and lot sizing. *Management Science*, *47*(8), 1081–1100.

Axsäter, S. (1986). Technical Note—On the feasibility of aggregate production plans. *Operations Research*, *34*(5), 796–800.

Badejo, O., & Ierapetritou, M. (2022). Integrating tactical planning, operational planning and scheduling using data-driven feasibility analysis. *Computers & Chemical Engineering*, *161*, Article 107759.

Begnaud, J., Benjaafar, S., & Miller, L. A. (2009). The multi-level lot sizing problem with flexible production sequences. *IIE Transactions*, *41*(8), 702–715.

Beykal, B., Avraamidou, S., & Pistikopoulos, E. N. (2022). Data-driven optimization of mixed-integer bi-level multi-follower integrated planning and scheduling problems under demand uncertainty. *Computers & Chemical Engineering*, *156*, Article 107551.

Biggs, M., Hariss, R., & Perakis, G. (2022). Constrained optimization of objective functions determined from random forests. *Production and Operations Management*.

Bish, E. K., & Wang, Q. (2004). Optimal investment strategies for flexible resources, considering pricing and correlated demands. *Operations Research*, *52*(6), 954–964.

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

Casazza, M., & Ceselli, A. (2019). Heuristic data-driven feasibility on integrated planning and scheduling. In *Advances in optimization and decision science for society, services and enterprises: ODS, genoa, Italy, September 4-7, 2019* (pp. 115–125). Cham: Springer International Publishing.

Castañé, G., Dolgui, A., Kousi, N., Meyers, B., Thevenin, S., Vyhmeister, E., & Östberg, P.-O. (2022). The ASSISTANT project: AI for high level decisions in manufacturing. *International Journal of Production Research*, 1–19.

Chod, J., & Zhou, J. (2014). Resource flexibility and capital structure. *Management Science*, *60*(3), 708–729.

Copil, K., Wörbelauer, M., Meyr, H., & Tempelmeier, H. (2016). Simultaneous lotsizing and scheduling problems: a classification and review of models. *OR Spectrum*, *39*(1), 1–64.

Cozad, A., Sahinidis, N. V., & Miller, D. C. (2014). Learning surrogate models for simulation-based optimization. *AIChE Journal*, *60*(6), 2211–2227.

Dauzère-Pérès, S., & Lasserre, J.-B. (1994). Integration of lotsizing and scheduling decisions in a job-shop. *European Journal of Operational Research*, *75*(2), 413–426.
Dauzère-Pérès, S., & Lasserre, J.-B. (2002). On the importance of sequencing decisions in production planning and scheduling. *International Transactions in Operational Research*, *9*(6), 779–793.
Dias, L. S., & Ierapetritou, M. G. (2019). Data-driven feasibility analysis for the integration of planning and scheduling problems. *Optimization and Engineering*, *20*(4), 1029–1066.
Dias, L. S., & Ierapetritou, M. G. (2020). Integration of planning, scheduling and control problems using data-driven feasibility analysis and surrogate models. *Computers & Chemical Engineering*, *134*, Article 106714.
Drexl, A., & Kimms, A. (1997). Lot sizing and scheduling — Survey and extensions. *European Journal of Operational Research*, *99*(2), 221–235.
Fajemisin, A. O., Maragno, D., & den Hertog, D. (2023). Optimization with constraint learning: A framework and survey. *European Journal of Operational Research*.
Filho, O. S. S., Cezarino, W., & Ratto, J. (2010). Aggregate production planning: Modeling and solution via excel spreadsheet and solver. *IFAC Proceedings Volumes*, *43*(17), 89–94, 5th IFAC Conference on Management and Control of Production Logistics.
Fischetti, M., & Jo, J. (2018). Deep neural networks and mixed integer linear optimization. *Constraints*, *23*(3), 296–309.
Fleischmann, B., & Meyr, H. (1997). The general lotsizing and scheduling problem. *Operations-Research-Spektrum*, *19*(1), 11–21.
Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples.
Hu, X., Duenyas, I., & Kapuscinski, R. (2008). Optimal joint inventory and transshipment control under uncertain capacity. *Operations Research*, *56*(4), 881–897.
Hurink, J., Jurisch, B., & Thole, M. (1994). Tabu search for the job-shop scheduling problem with multi-purpose machines. *OR Spektrum*, *15*, 205–215.
Hwang, H.-C. (2021). Subcontracting and lot-sizing with constant capacities. *Mathematical Programming*, *193*(1), 271–314.
Jun, S., Lee, S., & Chun, H. (2019). Learning dispatching rules using random forest in flexible job shop scheduling problems. *International Journal of Production Research*, *57*(10), 3290–3310.
Larroche, F., Bellenguez, O., & Massonnet, G. (2021). Clustering-based solution approach for a capacitated lot-sizing problem on parallel machines with sequence-dependent setups. *International Journal of Production Research*, *60*(21), 6573–6596.
Lasserre, J. B. (1992). An integrated model for job-shop planning and scheduling. *Management Science*, *38*(8), 1201–1211.
Lee, C.-Y., Piramuthu, S., & Tsai, Y.-K. (1997). Job shop scheduling with a genetic algorithm and machine learning. *International Journal of Production Research*, *35*(4), 1171–1191.
Liu, J.-L., Wang, L.-C., & Chu, P.-C. (2019). Development of a cloud-based advanced planning and scheduling system for automotive parts manufacturing industry. *Procedia Manufacturing*, *38*, 1532–1539.
Maragno, D., Wiberg, H., Bertsimas, D., Birbil, Ş. İ., den Hertog, D., & Fajemisin, A. O. (2023). Mixed-integer optimization with constraint learning. *Operations Research*.
Mckay, M. D., Beckman, R. J., & Conover, W. J. (2000). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, *42*(1), 55–61.
Meyr, H. (2002). Simultaneous lotsizing and scheduling on parallel machines. *European Journal of Operational Research*, *139*(2), 277–292.
Mirshekarian, S., & Šormaz, D. N. (2016). Correlation of job-shop scheduling problem features with scheduling efficiency. *Expert Systems with Applications*, *62*, 131–147.
Mišić, V. V. (2020). Optimization of tree ensembles. *Operations Research*, *68*(5), 1605–1624.
Ou, J., & Feng, J. (2019). Production lot-sizing with dynamic capacity adjustment. *European Journal of Operational Research*, *272*(1), 261–269.
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., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.
Raaymakers, W., & 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.
Rebennack, S., & Krasko, V. (2020). Piecewise linear function fitting via mixed-integer linear programming. *INFORMS Journal on Computing*, *32*(2), 507–530.
Rohaninejad, M., Janota, M., & Hanzálek, Z. (2023). Integrated lot-sizing and scheduling: Mitigation of uncertainty in demand and processing time by machine learning. *Engineering Applications of Artificial Intelligence*, *118*, Article 105676.
Rohaninejad, M., Kheirkhah, A., & Fattahi, P. (2014). Simultaneous lot-sizing and scheduling in flexible job shop problems. *International Journal of Advanced Manufacturing Technology*, *78*(1–4), 1–18.
Schneckenreither, M., Haeussler, S., & Gerhold, C. (2020). Order release planning with predictive lead times: a machine learning approach. *International Journal of Production Research*, *59*(11), 3285–3303.
Seeanner, F., & Meyr, H. (2012). Multi-stage simultaneous lot-sizing and scheduling for flow line production. *OR Spectrum*, *35*(1), 33–73.
Şenyiğit, E., Düğenci, M., Aydin, M. E., & Zeydan, M. (2013). Heuristic-based neural networks for stochastic dynamic lot sizing problem. *Applied Soft Computing*, *13*(3), 1332–1339.
Shang, C., Huang, X., & You, F. (2017). Data-driven robust optimization based on kernel learning. *Computers & Chemical Engineering*, *106*, 464–479.
Shinichi, N., & Taketoshi, Y. (1992). Dynamic scheduling system utilizing machine learning as a knowledge acquisition tool. *International Journal of Production Research*, *30*(2), 411–431.
Stadtler, H. (2005). Supply chain management and advanced planning—-basics, overview and challenges. *European Journal of Operational Research*, *163*(3), 575–588.
Tenhiälä, A. (2010). Contingency theory of capacity planning: The link between process types and planning methods. *Journal of Operations Management*, *29*(1–2), 65–77.
Thevenin, S., Zufferey, N., & Glardon, R. (2017). Model and metaheuristics for a scheduling problem integrating procurement, sale and distribution decisions. *Annals of Operations Research*, *259*(1), 437–460.
Tremblet, D., Thevenin, S., & Dolgui, A. (2022). Predicting makespan in flexible job shop scheduling problem using machine learning. *IFAC-PapersOnLine*, *55*(10), 1–6, 10th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2022.
Tremblet, D., Thevenin, S., & Dolgui, A. (2023). Makespan estimation in a flexible job-shop scheduling environment using machine learning. *International Journal of Production Research*, 1–17.
Trigeiro, W. W., Thomas, L. J., & McClain, J. O. (1989). Capacitated lot sizing with setup times. *Management Science*, *35*(3), 353–366.
Urrutia, E. D. G., Aggoune, R., & Dauzère-Pérès, S. (2014). Solving the integrated lot-sizing and job-shop scheduling problem. *International Journal of Production Research*, *52*(17), 5236–5254.
Wolosewicz, C., Dauzère-Pérès, S., & Aggoune, R. (2015). A Lagrangian heuristic for an integrated lot-sizing and fixed scheduling problem. *European Journal of Operational Research*, *244*(1), 3–12.
Yang, L., Liu, S., Tsoka, S., & Papageorgiou, L. G. (2016). Mathematical programming for piecewise linear regression analysis. *Expert Systems with Applications*, *44*, 156–167.
Yu, K., Yan, P., Kong, X. T., Yang, L., & Levner, E. (2024). Sequential auction for cloud manufacturing resource trading: A deep reinforcement learning approach to the lot-sizing problem. *Computers & Industrial Engineering*, *188*, Article 109862.
Zhang, C., Zhang, D., & Wu, T. (2021). Data-driven branching and selection for lot-sizing and scheduling problems with sequence-dependent setups and setup carryover. *Computers & Operations Research*, *132*, Article 105289.