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

David Tremblet, Simon Thevenin, Alexandre Dolgui

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

David Tremblet, Simon Thevenin, Alexandre Dolgui

[a] *IMT Atlantique, LS2N-CNRS, 4 rue Alfred Kastler, La Chantrerie, Nantes, 44307, , France*

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.

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

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 recomputing 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 (Liu et al., 2019; Filho et al., 2010). Many authors highlight the drawback of aggregated capacity constraints in lot-sizing models and the necessity to acquire further information at the planning level (Dauzère-Pérès and Lasserre, 2002; Almeder et al., 2015). This includes scheduling decisions or detailed capacity constraints, leading to impractical mathematical programs or constraints that are too complex to be integrated.

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 threefold: (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. For large-size instances, the proposed approach outperforms models based on integrated lot-sizing and scheduling and returns reliable production plans compared to standard lot-sizing model. In addition, we show the adaptability of our approach with an iterative lot-sizing and scheduling approach.

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 and Jo, 2018), decision trees and ensemble methods (Mišić, 2020; Biggs et al., 2022), among others (Fajemisin et al., 2023; Maragno et al., 2023).

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; Zhang et al., 2021; Şenyiğit et al., 2013; Yu et al., 2024). 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 (Dias and Ierapetritou, 2020; Badejo and Ierapetritou, 2022).

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 and Wang, 2004; Chod and 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 (Thevenin et al., 2017; Larroche et al., 2021).

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 and Hochbaum, 2001; Hwang, 2021), or the adjustment between different levels of capacity (Ou and 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, ...). 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 and 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 and 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 formu-

lations 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., Shinichi and Taketoshi, 1992; Lee et al., 1997) 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 and 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-know 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 and 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):

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

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

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

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

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

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

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

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 j^{th} 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., $s_{ii"}^k \leq s_{ii'}^k + s_{i'i"}^k$ for any jobs i, i' and $i" \in J$ 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 $D = \{\overline{\mathcal{X}^s}, \overline{\mathcal{Y}^s} \mid s = 1, \ldots, N\}$, 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 and 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.

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

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

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

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

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_k^{min} 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_k^{mean} 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:

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

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

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

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

Equations (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 and 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:

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

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

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

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

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

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

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 $j \in \mathcal{N}$. Equations (22) and (23) state that only one node is active at each stage of the regression tree. Equations (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 $\mathcal{Y}_s^{pred}$ for each sample $s \in D$.

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

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

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

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

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

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

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

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:

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

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. Figure 5.2.2 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 $iter_{max}$ of lot-sizing instances without finding periods where the capacity is violated. However, the approach remains time-consuming for complex instances,

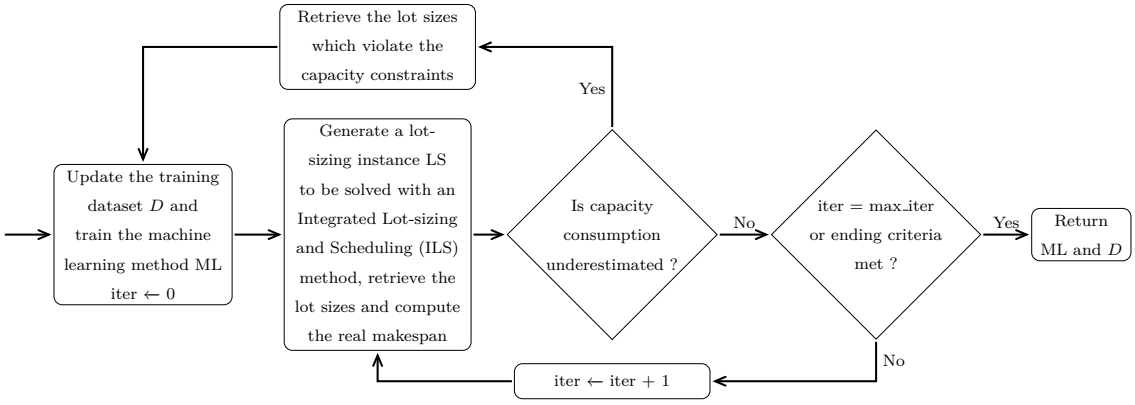

Figure 1: Flow chart for ILS-based

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:

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

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

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

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

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 on infeasible plans 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 subsection 6.1. Then, we compare the performance of all machine learning models in subsection 6.2. Subsections 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 subsection 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 @ 2,20GHz 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)
- 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 $N = 1,000$. All the lot-sizing models were solved by CPLEX with a time limit of 1 hour.

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,

	Size	6×6			10×10			20×5		
	T	5	30	50	5	30	50	5	30	50
	UB	1600	9368	15517	2659	15309	25746	5323	30423	51996
	LB	1600	9368	15517	2659	15309	25746	5323	30423	51996
ILS-LR	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
	UB	1603	9382	15537	2664	15361	25833	5332	30470	52034
	LB	1603	9382	15537	2664	15361	25819	5332	30470	52034
ILS-CLR	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
	UB	1600	9367	15518	2659	15308	25742	5324	30443	52153
	LB	1600	9367	15518	2659	15308	25739	5324	30424	51996
ILS-RT	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
	UB	1601	9371	$\times$	2659	15308	$\times$	5321	30420	$\times$
	LB	1601	9371	$\times$	2659	15308	$\times$	5321	30420	$\times$
ILS-PLR	Gap(%)	0	0	$\times$	0	0	$\times$	0	0	$\times$
	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
	UB	1602	9379	15527	2665	15342	25742	5333	30462	52023
	LB	1602	9379	15527	2665	15342	25718	5333	30462	52023
ILS-CPLR	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

Table 1: Comparison between constrained and standard machine learning 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.

	Metrics	Scheduling			Period			Setup costs		
		6×6	10×10	20×5	5	30	50	15	50	100
ILS-Exact	Gap^B(%)	0.3	1.51	9.17	1.19	3.69	0.45	2.31	0.03	2.51
	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(%)	0.41	0.39	0.4	0.68	0.29	0.22	0.4	×	×
	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(%)	1.85	0.31	0.09	0.99	0.74	0.21	0.19	1.41	1.49
	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(%)	0.52	1.61	4.66	2.67	1.73	2.39	0.27	2.06	4.46
	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(%)	3.59	1.48	1.44	3.2	1.75	1.56	0.29	1.86	4.36
	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(%)	3.31	1.86	1.2	2.54	1.79	2.02	0.21	1.74	4.41
	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(%)	4.38	1.23	0.64	3.06	1.7	1.49	0.31	1.79	4.15
	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	Gap^B(%)	3.94	1.66	0.25	2.46	1.61	1.79	0.23	1.53	4.09
	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

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

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.

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 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

	Metrics	Backlog costs			Setup times			Demand		
		Forbidden	1	3	None	×3	×5	[4,8]	[10,50]	[10,100]
ILS-Exact	Gap^B(%)	0.26	0.97	0.15	10.15	0.98	0.21	3.37	0.54	0.37
	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(%)	0.42	0.76	0.37	0.23	0.55	1.52	1.83	0.02	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(%)	0.2	0.72	0.19	0.17	0.93	0.26	0.98	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(%)	0.27	0.74	0.28	0.23	0.73	0.67	1.38	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(%)	0.3	0.68	0.29	0.3	0.79	0.57	1.42	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(%)	0.22	0.61	0.22	0.19	0.61	0.23	1.35	0.02	0.0
	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(%)	0.33	0.7	0.31	0.3	0.79	0.57	1.53	0.03	0.04
	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(%)	0.22	0.6	0.22	0.18	0.61	0.23	1.3	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: Results for lot-sizing models by varying backlog costs, setup times, and demand

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 schedule respects 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 seconds or if μ_t is smaller than 0.001 for any $t \in T$. To solve the lot-sizing problems, we considered both the CLSP and CLR95 models, denoted respectively by H-CLSP 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 high number of feasible plans, so we considered $\sigma_t = 1.0$ and $\mu_t = 0.15$ for ILS-CLR.

	Metrics	Scheduling			Period			Setup costs		
		6×6	10×10	20×5	5	30	50	15	50	100
H-CLSP	Gap^B(%)	0.12	0.63	2.27	0.39	0.33	2.29	0.0	0.16	2.84
	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-CLR95	Gap^B(%)	0.35	0.1	1.84	0.01	0.32	1.97	0.02	0.12	2.17
	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
CLR	Gap^B(%)	1.27	1.01	1.9	2.44	1.2	0.55	0.31	1.99	1.88
	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

Table 4: Aggregated results for heuristical approach

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. 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.

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. The resulting approach outperforms state-of-the-art lot-sizing models from the literature for large-scale instances, by providing 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 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. In the same way, 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.

Acknowledgements

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.

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, 727–738.

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

Axsäter, S., 1986. Technical note—on the feasibility of aggregate production plans. Operations Research 34, 796–800.

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

Balas, E., 1969. Machine sequencing via disjunctive graphs: An implicit enumeration algorithm. Operations Research 17, 941–957.

Begnaud, J., Benjaafar, S., Miller, L.A., 2009. The multi-level lot sizing problem with flexible production sequences. IIE Transactions 41, 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, 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, 954–964.

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

Breiman, L., Friedman, J.H., Olshen, R.A., 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. Springer International Publishing, Cham, pp. 115–125.

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, 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, 1–64.

Cozad, A., Sahinidis, N.V., Miller, D.C., 2014. Learning surrogate models for simulation-based optimization. AIChE Journal 60, 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, 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, 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, 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, 106714.

Drexl, A., Kimms, A., 1997. Lot sizing and scheduling — survey and extensions. European Journal of Operational Research 99, 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, 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, 296–309.

Fisher, H., 1963. Probabilistic learning combinations of local job-shop scheduling rules. Industrial scheduling , 225–251.

Fleischmann, B., Meyr, H., 1997. The general lotsizing and scheduling problem. Operations-Research-Spektrum 19, 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, 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, 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, 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, 6573–6596.

Lasserre, J.B., 1992. An integrated model for job-shop planning and scheduling. Management Science 38, 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, 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, c.I., 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, 55–61.

Meyr, H., 2002. Simultaneous lotsizing and scheduling on parallel machines. European Journal of Operational Research 139, 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, 1605–1624.

Ou, J., Feng, J., 2019. Production lot-sizing with dynamic capacity adjustment. European Journal of Operational Research 272, 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, 14–30.

Rebennack, S., Krasko, V., 2020. Piecewise linear function fitting via mixed-integer linear programming. INFORMS Journal on Computing 32, 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, 105676.

Rohaninejad, M., Kheirkhah, A., Fattahi, P., 2014. Simultaneous lot-sizing and scheduling in flexible job shop problems. The International Journal of Advanced Manufacturing Technology 78, 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, 3285–3303.

Seeanner, F., Meyr, H., 2012. Multi-stage simultaneous lot-sizing and scheduling for flow line production. OR Spectrum 35, 33–73.

Shang, C., Huang, X., You, F., 2017. Data-driven robust optimization based on kernel learning. Computers & Chemical Engineering 106, 464–479.

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

Shinichi, N., Taketoshi, Y., 1992. Dynamic scheduling system utilizing machine learning as a knowledge acquisition tool. International Journal of Production Research 30, 411–431.

Stadtler, H., 2005. Supply chain management and advanced planning—basics, overview and challenges. European Journal of Operational Research 163, 575–588.

Tenhiälä, A., 2010. Contingency theory of capacity planning: The link between process types and planning methods. Journal of Operations Management 29, 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, 437–460.

Tremblet, D., Thevenin, S., Dolgui, A., 2022. Predicting makespan in flexible job shop scheduling problem using machine learning. IFAC-PapersOnLine 55, 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, 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, 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, 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, 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, 105289.

Şenyiğit, E., Düğenci, M., Aydin, M.E., Zeydan, M., 2013. Heuristic-based neural networks for stochastic dynamic lot sizing problem. Applied Soft Computing 13, 1332–1339.

Appendix A. ILS-Exact

In the following, we present an MILP, adapted from Shen et al. (2018) and denoted here as ILS-Exact, which represents the scheduling decisions of the integrated lot-sizing and flexible job-shop model. First, let us define the following variables:

- ϕ_{ijt} : the starting time of operation O_{ij} in period $t \in T$
- $Z_{ijkt} = \begin{cases} 1 & \text{if operation } O_{ij} \text{ is assigned to machine } k \in M_{ij} \text{ in period } t \in T, \\ 0 & \text{otherwise} \end{cases}$
- $\beta_{iji'j't} = \begin{cases} 1 & \text{if operation } O_{ij} \text{ is performed before operation } O_{i'j'} \text{ in period } t \in T, \\ 0 & \text{otherwise} \end{cases}$

Thus, the flexible job-shop scheduling problem considered at each period of our capacitated lot-sizing problem can be implemented using the following equations:

$$\sum_{k \in M_{ij}} Z_{ijkt} = 1, \quad j \in n_i, i \in J, t \in T \tag{A.1}$$

$$\phi_{ijt} \geq \phi_{i(j-1)t} + p_{i(j-1)k} X_{it} - (1 - Z_{i(j-1)kt}) H \quad \begin{smallmatrix} \forall k \in M_{i(j-1)}, \\ j \in 2,\ldots,n_i, \\ i \in J, t \in T \end{smallmatrix} \tag{A.2}$$

$$\phi_{ijt} \geq \phi_{i'j't} + p_{i'j'k} X_{i't} + s_{i'ik} Y_{it} - (2 - Z_{ijkt} - Z_{i'j'kt} + \beta_{iji'j't}) H \quad \begin{smallmatrix} j \in 1,\ldots,n_i, j' \in 1,\ldots,n_{i'}, \\ (i,i') \in \{J \times J | O_{ij} \neq O_{i'j'}\}, \\ k \in M_{ij} \cap M_{i'j'}, t \in T \end{smallmatrix} \tag{A.3}$$

$$\phi_{i'j't} \geq \phi_{ijt} + p_{ijk} X_{it} + s_{ii'k} Y_{i't} - (3 - Z_{ijkt} - Z_{i'j'kt} - \beta_{iji'j't}) H \quad \begin{smallmatrix} j \in 1,\ldots,n_i, j' \in 1,\ldots,n_{i'}, \\ (i,i') \in \{J \times J | O_{ij} \neq O_{i'j'}\}, \\ k \in M_{ij} \cap M_{i'j'}, t \in T \end{smallmatrix} \tag{A.4}$$

$$\phi_{in_it} + p_{in_ik} X_{it} - (1 - Z_{ijkt}) H \leq C_t, \quad k \in M_{in_i}, i \in J, t \in T \tag{A.5}$$

$$Z_{ijkt} \in \{0,1\}, \quad j \in 1 \ldots n_i, i \in J, k \in M_{ij}, t \in T, \quad i \in J, t \in T$$

$$\beta_{iji'j't} \in \{0,1\}, \quad j \in 1,\ldots,n_i, j' \in 1,\ldots,n_{i'}, (i,i') \in \{J \times J | O_{ij} \neq O_{i'j'}\}, t \in T$$

$$\phi_{ijt} \geq 0, \quad j \in 1 \ldots n_i, i \in J, k \in M_{ij}, t \in T$$

These constraints define the scheduling decisions for each period $t \in T$ of the production plan. Constraints (A.1) ensure that each operation is performed by only one machine for each period. Constraints (A.2) force the operations of the same job to be performed

consecutively. Constraints (A.3) and (A.4) state that each pair of operations O_{ij} and $O_{i'j'}$ performed on a same machine $k \in M_{ij} \cap M_{i'j'}$ must not overlap. Finally, Constraints (A.5) define the capacity constraints for each period, i.e. the makespan of each flexible job-shop scheduling problem considered at each period $t \in T$ must respect the capacity.

Appendix B. Proof of Proposition 1

Suppose we have a job-shop scheduling environment with a set of jobs J, composed of n_i successive operations for each job $i \in J$, to be performed on a set of machines M. From any job-shop scheduling problem, we can derive its disjunctive graph, where nodes are associated with operations and arcs relate to the precedence constraints between operations (see Balas (1969) for more details). Assigning all the operations on the machines results in a sequence z, for which we can derive its conjunctive graph $\mathcal{C}(z)$. Each sequence z in the set of all possible sequences $\mathcal{S}$ provides the order in which the operations are performed on all the machines. For any sequence $z \in \mathcal{S}$, we can compute the completion time of any path $c \in \mathcal{C}(z)$ in the conjunctive graph. This completion time corresponds to the sum of processing times for all operations and all setups in this path, i.e.:

$$f(X, c) = \sum_{o \in c} \left(p_o \cdot X_{i(o)} + s_o \cdot Y_{i(o)} \right)$$

where:

$$Y_{i(o)} = \begin{cases} 1 & \text{if } X_{i(o)} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Here, $o \in c$ stands for one operation performed by the path c, $i(o)$ for its corresponding job $i \in J$, p_o for its processing time, and s_o for the setup time between operation o and its direct successor (s_o is equal to 0 if operation o is the last operation). For each sequence $z \in \mathcal{S}$, we can determine its critical path by finding the path c with the highest value, which is achieved by finding the maximum among all paths:

$$\max_{c \in \mathcal{C}(z)} \left\{ \sum_{o \in c} \left(p_o \cdot X_{i(o)} + s_o \cdot Y_{i(o)} \right) \right\}$$

By considering the set of all the possible sequences $\mathcal{S}$, the makespan of the scheduling problem is given by the sequence whose critical path is the shortest among all sequences. Then, we can define a function that returns the makespan for the lot sizes represented by

the vector X:

$$C_{max}(X) = \min_{z \in \mathcal{S}} \max_{c \in \mathcal{C}(z)} \left\{ \sum_{o \in c} \left(p_o \cdot X_{i(o)} + s_o \cdot Y_{i(o)} \right) \right\}$$

Function f is piecewise linear, as well as max and min functions. Thus, C_{max} being a composition of piecewise linear functions, C_{max} is also a piecewise linear function.

This proof can be extended to flexible job-shop scheduling problems since a flexible job-shop scheduling problem can be interpreted as a job-shop scheduling problem with more sequences (Brandimarte, 1993). It can also be extended to job-shop scheduling problems with sequence-dependent setup times.

Appendix C. Results

Scheduling size		6 × 6								
Setup costs		15			50			100		
Period		5	30	50	5	30	50	5	30	50
Exact	UB	1590.4	9382.8	15671.0	2193.6	12258.2	20463.2	2898.0	15762.3	26281.7
	LB	1590.4	9382.2	15669.8	2193.4	12107.7	20098.5	2897.8	14936.1	24746.5
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1
	Feasibility (%)	100.0	100.0	100.0	100.0	99.0	99.0	100.0	100.0	96.0
	Time (s.)	1.1	60.9	361.4	18.0	3600.0	3600.0	159.7	3600.0	3600.0
CLSP	UB	1590.2	9381.7	15669.1	2167.6	12112.3	20135.3	2810.3	15123.9	25076.4
	LB	1590.2	9381.4	15668.6	2167.6	12111.2	20133.6	2810.2	15122.5	25073.3
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Feasibility (%)	75.0	44.0	23.0	0.0	0.0	0.0	0.0	0.0	0.0
	Time (s.)	0.0	0.1	0.1	0.0	0.9	2.1	0.0	12.7	444.0
CLSP75	UB	1594.6	9411.0	15718.6	2235.7	12521.6	20813.7	3015.2	16329.0	27071.5
	LB	1594.6	9410.5	15717.7	2235.7	12520.4	20811.7	3015.1	16326.9	27017.5
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Feasibility (%)	100.0	93.0	95.0	69.0	2.0	0.0	26.0	0.0	0.0
	Time (s.)	0.0	0.1	0.1	0.0	3.3	41.0	0.0	323.9	3425.1
Fixed	UB	1593.6	9395.7	15691.2	2223.6	12301.3	20438.4	2969.8	15600.0	25871.9
	LB	1593.6	9394.9	15689.7	2223.5	12275.8	20293.6	2969.7	15417.0	25337.9
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Time (s.)	0.0	2.2	7.2	0.1	3002.9	3600.0	0.3	3601.3	3600.0
CLR	UB	1598.3	9424.3	15738.1	2274.1	12642.6	21018.9	3112.0	16681.4	27679.7
	LB	1598.3	9423.4	15736.6	2274.0	12504.4	20655.0	3111.9	16086.4	26388.8
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Time (s.)	0.0	3.7	45.8	0.1	3600.0	3600.0	0.2	3605.8	3602.5
CPLR	UB	1594.9	9410.1	15716.7	2252.0	12600.7	21040.7	3059.4	16671.6	27924.9
	LB	1594.8	9408.9	15702.0	2251.9	12138.0	20048.6	3059.2	14864.6	24338.4
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Time (s.)	0.1	839.9	3422.6	0.9	3600.0	3600.0	3.8	3600.0	3600.0
CLR95	UB	1600.3	9438.7	15763.3	2288.7	12748.0	21190.1	3146.5	16932.9	28071.2
	LB	1600.3	9437.8	15761.7	2288.7	12630.1	20869.5	3146.4	16472.2	27074.0
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Time (s.)	0.0	2.7	49.3	0.1	3600.0	3600.0	0.2	3600.0	3600.0
CPLR95	UB	1597.4	9422.6	15736.3	2266.0	12692.0	21176.3	3088.9	16855.5	28225.1
	LB	1597.4	9418.7	15707.9	2265.9	12159.8	20086.9	3088.6	14966.0	24522.5
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.1	0.1
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	99.0	100.0	100.0	99.0
	Time (s.)	0.1	1997.3	3581.6	1.1	3600.0	3600.0	4.4	3600.0	3600.0
H-CLSP	UB	1590.8	9384.3	15674.3	2217.2	12475.9	20797.5	3012.0	16528.2	27714.1
	LB	1590.8	9384.0	15673.7	2217.2	12474.7	20795.2	3012.0	16505.8	27580.1
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	88.0
	Time (s.)	8.0	48.2	100.0	14.6	134.4	859.6	10.3	2902.5	3600.0
H-CLR	UB	1591.5	9387.8	15679.6	2217.4	12442.3	20896.9	2992.8	16754.0	28019.4
	LB	1591.5	9387.3	15678.3	2217.3	12370.4	20616.3	2992.7	16235.9	26887.3
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Time (s.)	15.3	72.1	160.5	9.1	3348.2	3579.6	10.5	3600.0	3600.0

Table C.5: Results for 6 × 6

Scheduling size		10 × 10								
Setup costs		15			50			100		
Period		5	30	50	5	30	50	5	30	50
Exact	UB	2646.8	15670.1	26145.8	3702.0	×	×	5020.9	×	×
	LB	2645.8	15651.5	26075.0	3602.5	20012.0	33183.7	4633.6	24143.0	39879.6
	Gap (%)	0.0	0.0	0.0	0.0	×	×	0.1	×	×
	Feasibility (%)	100.0	100.0	100.0	100.0	0.0	0.0	100.0	0.0	0.0
	Time (s.)	1633.4	3573.9	3600.1	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0
CLSP	UB	2645.6	15651.5	26075.1	3594.4	20102.5	33352.8	4622.8	24931.5	41278.9
	LB	2645.6	15651.4	26074.7	3594.4	20100.8	33349.9	4622.7	24929.0	41262.3
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Feasibility (%)	63.0	47.0	31.0	0.0	0.0	0.0	0.0	0.0	0.0
	Time (s.)	0.0	0.1	0.2	0.0	1.6	3.2	0.0	58.9	2036.1
CLSP75	UB	2649.8	15677.4	26118.7	3680.7	20676.4	34310.6	4928.6	26742.8	44286.9
	LB	2649.8	15676.5	26117.2	3680.6	20674.3	34307.1	4928.5	26731.6	44166.6
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Feasibility (%)	98.0	96.0	98.0	8.0	0.0	0.0	1.0	0.0	0.0
	Time (s.)	0.0	0.1	0.2	0.0	7.7	157.0	0.0	1625.3	3600.0
Fixed	UB	2652.6	15677.7	26118.6	3742.4	20916.3	34850.9	5095.6	27557.7	46373.7
	LB	2652.5	15676.0	26110.0	3742.1	20209.0	33392.0	5093.4	24832.4	40656.4
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Time (s.)	0.1	428.9	3206.8	3.2	3600.0	3600.0	77.8	3600.0	3600.0
CLR	UB	2655.4	15690.2	26139.2	3761.9	20901.0	34682.6	5141.8	27431.5	45500.0
	LB	2655.4	15688.6	26136.2	3761.7	20550.9	33968.3	5141.4	26110.9	42904.9
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Time (s.)	0.0	21.4	710.9	0.5	3600.0	3600.0	2.7	3600.0	3600.0
CPLR	UB	2654.7	15688.6	26137.0	3756.0	20982.9	34919.3	5121.0	27701.2	46380.4
	LB	2654.6	15687.1	26126.8	3755.7	20353.3	33657.3	5120.5	25506.9	41936.4
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Time (s.)	0.1	244.6	3299.5	1.5	3600.0	3600.0	9.5	3600.0	3600.0
CLR95	UB	2654.3	15686.4	26133.1	3749.9	20862.0	34619.3	5112.3	27325.5	45300.6
	LB	2654.3	15684.9	26130.4	3749.6	20552.9	33983.4	5111.8	26144.7	42992.6
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Time (s.)	0.0	11.4	364.9	0.4	3600.0	3600.1	2.1	3600.0	3600.0
CPLR95	UB	2654.3	15686.4	26132.4	3751.2	20917.6	34844.2	5110.8	27593.1	46192.9
	LB	2654.2	15684.5	26119.0	3750.8	20303.3	33570.8	5110.3	25301.7	41658.8
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Time (s.)	0.1	609.2	3425.2	1.7	3601.7	3600.0	14.8	3600.0	3600.0
H-CLSP	UB	2646.7	15653.5	26078.9	3688.0	20778.6	34575.2	5025.0	27766.7	46126.8
	LB	2646.7	15653.2	26078.4	3687.9	20776.4	34559.1	5024.9	27672.8	45869.6
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	99.0	99.0	83.0	47.0
	Time (s.)	63.8	167.1	232.6	110.0	597.5	3130.2	51.4	3600.0	3600.0
H-CLR	UB	2646.3	15652.9	26077.4	3674.0	20633.6	34605.5	4965.4	27238.3	45881.8
	LB	2646.3	15652.3	26076.1	3673.8	20375.6	33850.2	4965.0	25889.5	42787.7
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Time (s.)	75.1	200.6	545.8	170.0	3600.0	3600.0	60.6	3600.0	3600.0

Table C.6: Results for 10 × 10

Scheduling size		20×5								
Setup costs		15			50			100		
Period		5	30	50	5	30	50	5	30	50
Exact	UB	5361.1	36705.6	×	×	×	×	×	×	×
	LB	5297.4	31237.9	51975.7	7064.0	39733.8	65866.9	8364.4	47332.3	78006.0
	Gap (%)	0.0	0.1	×	×	×	×	×	×	×
	Feasibility (%)	73.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Time (s.)	3600.0	3600.0	3600.0	3600.0	3600.0	3600.3	3600.0	3600.1	3600.1
CLSP	UB	5299.0	31239.2	51977.6	7199.7	39874.1	66010.0	9061.4	48002.4	79199.8
	LB	5298.8	31238.5	51976.3	7199.1	39870.4	66004.1	9060.5	47997.8	79192.0
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Feasibility (%)	6.0	9.0	7.0	0.0	0.0	0.0	0.0	0.0	0.0
	Time (s.)	0.0	0.2	0.4	0.1	4.4	8.5	1.0	105.6	642.3
CLSP75	UB	5323.4	31265.9	52008.0	7467.1	40187.3	66376.2	9718.5	49121.3	80695.6
	LB	5323.0	31263.1	52003.5	7466.3	40180.8	66348.1	9717.6	48984.7	80397.5
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Feasibility (%)	100.0	100.0	100.0	74.0	7.0	0.0	6.0	0.0	0.0
	Time (s.)	0.1	1.9	4.6	0.6	2094.7	3596.6	3.4	3600.0	3600.0
Fixed	UB	5339.5	31389.0	52229.5	7716.6	42545.3	71501.6	10445.6	56491.2	97473.1
	LB	5339.0	31318.7	52080.8	7715.9	40342.2	66561.4	10247.8	48907.0	80247.7
	Gap (%)	0.0	0.0	0.0	0.0	0.1	0.1	0.0	0.1	0.2
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Time (s.)	1.2	3600.0	3600.0	317.4	3600.0	3600.0	3592.8	3600.0	3600.0
CLR	UB	5324.0	31288.9	52046.4	7590.4	41205.4	68325.2	10215.1	53461.7	88587.3
	LB	5323.6	31285.8	52037.0	7589.7	40379.2	66589.3	10204.0	49389.1	80604.9
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Time (s.)	0.1	337.2	3099.9	10.5	3600.0	3600.0	1865.3	3600.0	3600.0
CPLR	UB	5317.7	31269.2	52015.4	7520.7	41104.1	68606.2	10095.1	53367.5	88814.7
	LB	5317.3	31266.1	52010.0	7520.0	40130.2	66234.0	10060.8	48392.4	79386.2
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Time (s.)	0.1	101.5	1103.8	16.8	3600.0	3600.0	2540.8	3600.0	3600.0
CLR95	UB	5319.8	31277.9	52028.9	7532.8	40916.5	67796.4	10080.4	52555.3	87117.9
	LB	5319.4	31274.8	52023.6	7532.1	40307.5	66507.4	10076.4	49188.8	80438.3
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	99.0	100.0	97.0	98.0
	Time (s.)	0.1	99.5	1145.1	7.2	3600.0	3600.0	1135.4	3600.0	3600.0
CPLR95	UB	5312.8	31256.9	51999.1	7461.2	40705.8	67634.9	9963.3	52363.6	87192.5
	LB	5312.5	31254.0	51994.3	7460.5	40101.4	66215.5	9949.2	48344.5	79366.9
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	99.0	100.0	94.0	93.0
	Time (s.)	0.1	8.4	23.3	9.5	3600.0	3600.0	1885.9	3600.0	3600.0
H-CLSP	UB	5303.6	31243.0	51981.4	7355.0	40169.4	66452.4	9848.1	51996.4	105529.9
	LB	5303.3	31241.7	51979.5	7354.3	40149.0	66379.0	9847.2	51114.0	82683.6
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2
	Feasibility (%)	100.0	100.0	100.0	100.0	98.0	82.0	100.0	84.0	38.0
	Time (s.)	217.9	1176.9	1677.7	212.4	2842.5	3600.0	120.5	3600.0	3600.0
H-CLR	UB	5305.0	31243.1	51982.0	7348.0	40161.8	66782.9	9735.0	52781.6	101475.1
	LB	5304.8	31241.4	51979.4	7347.3	40068.7	66274.4	9726.1	48837.8	80077.1
	Gap (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.2
	Feasibility (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Time (s.)	290.1	1003.8	1942.5	269.0	3540.7	3600.0	1596.0	3600.0	3600.0

Table C.7: Results for 20×5

Appendix D. Features description

This subsection describes the parameters used to describe features $\mathcal{X}_{(|J|+1)t}$, $\mathcal{X}_{(|J|+2)t}$, $\mathcal{X}_{(|J|+3)t}$, and $\mathcal{X}_{(|J|+4)t}$. Each of these features relates to a sum of processing times and uses the precomputed parameters described as follows:

o_{ijk}	:	The "operating ratio" for each operation O_{ij} performed on machine k.
o_k	:	Estimation of the total number of operations performed on machine $k \in M$.
s_k^{min}	:	Minimum setup times between all pairs of jobs that can be performed on machine k.
s_k^{mean}	:	Average setup times between all pairs of jobs that can be performed on machine k.

As mentioned in Section 4.1, the operation ratio o_{ijk} provides the likeliness of an operation being performed on machine k. Depending on the scheduling problem, operations that can be performed on more than one machine are more likely to be performed on machines that lead to a minimal makespan. Several factors can be considered to assess the likelihood of an operation being performed on one machine instead of another. One idea is to take the processing times into account when choosing on which machine an operation has to be performed. Thus, we compute the operation ratios as follows:

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

These ratios are computed such that operations that can be performed on only one machine has a ratio of 1, while the other operations have a ratio that depends on the processing times and the machine. Using these ratios, we can compute the estimated number o_k of operations on machine $k \in M$ as follows:

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

Appendix E. Instance Generation

Lot-sizing instances

To generate lot-sizing instances, we adapt the procedure given in Wolosewicz et al. (2015). The horizon length T takes values 5, 30, and 50 periods. We also vary the setup

costs by considering values of 15, 50, and 100. The default setup cost is 15. The production, holding, and backlogging costs remain identical between the instances, and we consider respective values of 4 for the production costs, 1 for the holding costs, and 5 for the backlogging costs. The demand d for each item is generated randomly in the interval $[5, 15]$. For capacity tightness, the required capacity at each period is obtained by summing the processing times and the mean setup time required to satisfy the whole demand and then dividing this sum by the number of machines. This capacity is then multiplied by an average utilization *cap* equals 0.55 for 6×6 instances. For 10×10 and 20×5 we select an average utilization of 0.35 instead of 0.55 as in Wolosewicz et al. (2015) since the capacity was large enough to solve all the instances optimally without exceeding the capacity for each period. We adapted the formula to the flexible job-shop case as follows:

$$C_l = cap \sum_{i \in |J|} \sum_{j \in n_i} \sum_{k \in M_{ij}} \frac{1}{|M_{ij}|} p_{ijk} D_{il} + s_k^{mean} \quad \forall t \in T$$

Scheduling instances

We considered the flexible job-shop scheduling instances mt06, mt10, and mt20 from Hurink et al. (1994), which are standard job-shop instances from (Fisher, 1963) modified to include flexible operations. There are three sizes of instances, namely, 6 jobs for 6 machines (6×6), 10 jobs for 10 machines (10×10), and 20 jobs for 5 machines (20×5), where each job is composed of a number of operations equal to the number of machines. We considered the set of benchmark instances *edata*, which includes an average number of machines per operation equal to 1.15. The setup times were generated following a uniform distribution with support $[1, 100]$, with valid triangular inequalities. To obtain the best makespan for each data sample, we compare the MILP (A.1)-(A.5) solved with CPLEX on a single period and a constraint programming model for the flexible job-shop scheduling problem with sequence-dependent setup times solved using IBM ILOG CP Optimizer, within a time limit of 60 seconds for each data sample. Although the two methods can solve sample data of size 6×6 and 10×10 to optimality, the constraint programming approach provided the best solutions for the large-scale instance 20×5, with an average gap that never exceeds 15%. Thus, we chose to solve each of our data samples with the constraint programming approach.

Appendix F. Capacity consumption prediction

This section investigates the performance of the machine learning models in predicting capacity consumption. We consider the following predictive models:

- *RT*: Regression Tree

- LR: Linear Regression
- CLR: Constrained Linear Regression
- $CPLR$: Constrained Piecewise Linear Regression
- $Fixed$: Capacity consumption computed as in the ILS-Fixed model
- $Approx$: Capacity consumption computed as in the ILS-CLSP model

$Approx$ represents the capacity consumption commonly used in the classical lot-sizing models. $Approx$ considers the maximum between the capacity consumption for each item and for each machine, i.e. $\max\{\mathcal{X}_{(|J|+1)t}, \mathcal{X}_{(|J|+2)t}\}$. $Fixed$ considers a given sequence of the scheduling problem, and it computes the makespan with the critical path method. For the piecewise linear regression, we considered breakpoints $[1, 2, 3, 4, 5, 6]$, $[1, 5]$ and $[1, 9]$ for respectively the 6×6, 10×10 and 20×5 scheduling sizes. For the 6×6 case, one region is considered for all the possible number of setups, which leads to a more precise approximation but a more complex embedding compared to the scheduling sizes 10×10 and 20×5 with 3 regions each.

The approaches are compared on three datasets of $10,000$ scheduling samples generated with the Lot-sizing based approach described in Section 5.2.2 with the lot-sizing model ILS-Fixed. To evaluate the performance of each model, we considered a binary classification problem where the machine model predicts if the given lot sizes respect the capacity or not. For each dataset, we consider the median makespan as the capacity limit. Each data sample was labeled as "Feasible" or "Infeasible" if the makespan found for each of these samples is lower than the capacity limit or not. The predictive performance of each model is evaluated on its ability to correctly predict whether an unseen data sample is Feasible or Infeasible. Therefore, each machine learning model was trained to predict the makespan of each data sample, and the prediction return for each testing sample is labeled afterward as Feasible or Infeasible depending on the value returned. For each dataset, we randomly sampled 80% of our dataset for training and 20% for testing.

Table F.8 reports the performance of the machine learning models and the approximation of capacity consumption. For each model and each scheduling size, we considered the Accuracy, Precision, and Recall, which are the common metrics used in classification evaluation. Accuracy corresponds to the proportion of schedules that were correctly classified as Feasible or Infeasible. Precision represents the percentage of schedules correctly predicted as Feasible over the total number of schedules predicted as Feasible, including false positives. Finally, Recall provides the percentage of schedules correctly predicted as feasible and the actual number of feasible schedules in the test dataset. The MAE reports the mean absolute error between the outcome of our methods and the real makespan.

Since *Approx* is based on the theoretical lower bounds of the makespan, it cannot classify as infeasible schedules that are feasible. Similarly, no false positive can result from the Fixed method since the given sequence provides an upper bound of the makespan.

Size	Metric	*RT*	*LR*	*CLR*	*CPLR*	*Approx*	*Fixed*
6×6	Accuracy (%)	97.2	96.2	87.7	87.1	93.9	94.7
	Recall (%)	97.6	96.2	75.5	74.3	100	89.6
	Precision (%)	96.8	96.2	100	100	88.4	100
	MAE	19.5	31.9	140.6	118.6	44.2	64.3
10×10	Accuracy (%)	95.9	97.0	86.1	86.5	82.9	71.2
	Recall (%)	96.0	97.2	72.2	73	100	43.5
	Precision (%)	95.8	96.8	100	100	74.5	100
	MAE	468.5	463.3	2364.7	1960.3	1529.4	4282.8
20×5	Accuracy (%)	94.6	95.2	82.6	87.2	60.2	64.0
	Recall (%)	95.6	94.2	65.2	74.4	100	30.2
	Precision (%)	93.7	96.1	100	100	55.6	100
	MAE	380.9	327.4	1679.0	1042.5	2452.5	5651.9

Table F.8: Prediction performance of machine learning and other approximations of capacity consumption

In the context of capacity consumption approximation, a model with high precision is crucial to limit the number of instances wrongly predicted as feasible by our model. These infeasible schedules are undesirable since they lead to unfeasible production plans, resulting in firefighting on the shop floor, delays in deliveries, and increasing costs of raw material ordering costs for express deliveries, etc (Thevenin et al., 2017). On the contrary, an approach with low recall removes feasible solutions, but the obtained production plan remains feasible. Therefore, a low recall may lead to sub-optimal planning. While a model with low recall results in larger costs at the planning level, it is preferred over a model with low precision which leads to infeasible plans.

Table F.8 shows that linear regression or regression trees have the highest accuracy and the smallest mean absolute error. However, these models did not yield a precision of 100%, which means that the capacity consumption of some testing samples was underestimated by both regression trees and linear regression. Although the precision remains really high (around 95%), these models are expected to provide solutions for the lot-sizing that are not feasible at the scheduling level. On the other hand, constrained linear and piecewise linear regression provide a precision of 100% for each testing dataset. However, there is no theoretical guarantee these methods don't underestimate capacity consumption. For small-size scheduling samples, Fixed outperforms constrained machine learning methods. However, these latter methods provide the best results for large-scale scheduling instances 10×10 and 20×5. This information is surprising since the dataset of 10,000 scheduling

examples was generated using the Lot-sizing based generation with ILS-Fixed, the lot sizes associated with each of these examples are supposed to be the best possible schedules found by ILS-Fixed. However, the machine learning models are able to predict a better capacity consumption than *Fixed* in these examples. Finally, while *Approx* has the best Recall value, this method has very bad precision, which decreases as the scheduling complexity increases.

Appendix G. Adversarial examples generation

This section presents a MILP that generates adversarial data samples for a given trained machine learning model. These adversarial data samples correspond to lot sizes where the prediction of capacity consumption is lower than its actual value. These points are potential solutions to the lot-sizing model that violates the capacity constraints.

We embed the adversarial data sample generator in an iterative training approach that successively trains the model and searches for an adversarial example. A mathematical model searches for examples that lead to an underprediction of capacity consumption. This MILP for adversarial data samples generation integrates the scheduling decisions of the flexible job-shop, the decision variables associated with the features (6)-(14), and the embedded machine learning model. The objective is to determine the lot sizes that maximize the distance between the machine learning prediction Y^{pred} and the real makespan of the scheduling.

Note that the full scheduling decision (A.1)-(A.5) cannot be integrated in a straightforward manner. As the model maximizes the error, it would maximize the makespan instead of minimizing it. We propose a formulation that relies on the set $\mathcal{S}$ of all possible sequences for the flexible job-shop scheduling problem. The flexible job-shop scheduling problem can be represented as a problem of finding the best sequence of operation among a set $\mathcal{S}$ of possible sequences (see Proposition 4.1). Sequence $z \in \mathcal{S}$ defines the production sequence on the machine, and it can be represented with a conjunctive graph $\mathcal{C}(z)$. This conjunctive graph models all the possible paths in the sequence z (see Wolosewicz et al., 2015, for further details). The makespan for a feasible schedule of a fixed sequence corresponds to the length of the longest path in the conjunctive graph. When more than one sequence is considered in set $\mathcal{S}$, the best sequence corresponds to the one whose longest path has the lowest makespan among all the sequences. This representation leads to a min-max formulation for the determination of the makespan.

The model maximizes the distance d^{-} between the machine learning prediction $\mathcal{Y}^{pred}$ and makespan C_{max}. For a fixed flexible job-shop scheduling problem and a given set of

sequences S, the adversarial samples generation method is formulated by a MILP as follows:

$$\begin{aligned}
\max \quad & d^- && \text{(G.1)}\\
\text{s. t.} \quad & (6)-(14) \\
& ML(\mathcal{X}) = \mathcal{Y}^{pred} && \text{(G.2)}\\
& \min_{z\in\mathcal{S}} \max_{c\in\mathcal{C}(z)} \left\{ \sum_{(o)\in c} p_o \cdot X_{i(o)} + s_o \cdot Y_{i(o)} \right\} \geq C_{max} && \text{(G.3)}\\
& C_{max} - \mathcal{Y}^{pred} \geq d^- && \text{(G.4)}\\
& 0 \leq d^- \\
& Y_i \in \{0,1\}, \quad UB_i \geq X_i \geq 0 \quad i \in J.
\end{aligned}$$

The objective function (G.1) maximizes the negative distance between the prediction and the makespan. Equations (6)-(14) compute the features. Constraint (G.2) gives the prediction returned by machine learning model ML, which refers to the translation of machine learning models as described in section 4. Constraints (G.3) set the makespan to the critical path of the sequence. In these constraints, c corresponds to a path of the set of paths in the conjunctive graph $\mathcal{C}(z)$ of the sequence z. Finally, the negative distance between the makespan and the prediction is determined through expression (G.4). We set maximum lot sizes for each product to avoid having an unbounded feasible region.

Model (G.1)-(G.4) with the entire set $\mathcal{S}$ of sequences returns the lot sizes that lead to the least accurate prediction of the makespan. However, the approach is not practical since the entire set $\mathcal{S}$ of sequences is too large. Therefore, we iteratively generate the set of sequences in a raw generation scheme. First, we start with a single sequence for set $\mathcal{S}$ with a machine learning model trained on an initial dataset D. The solution $\mathcal{X}$ of (G.1)-(G.4) gives the lot sizes X that lead to an underestimation of the makespan by machine learning model ML for the scheduling problem with fixed sequences. As the makespan in (G.1)-(G.4) is computed on a restricted set of sequences, we check that the lot sizes X underestimate the makespan when computed by solving the full flexible job-shop scheduling problem (A.1)-(A.5). If the makespan C^*_{max} of the optimal sequence z^* is greater than the prediction, we add the adversarial example $\mathcal{X}$ to D and retrain the model. Otherwise, if the prediction is greater than C^*_{max}, then the scheduling problem with fixed sequences provided a bad estimation of the makespan, and we add sequence z^* to $\mathcal{S}$ and resolve the adversarial program. This procedure is repeated until the solver declares the problem is unfeasible, or it finds an optimal solution with $d^- = 0$. Meeting these two cases for this problem means that no adversarial example can be found for the finite set of the sequence $\mathcal{S}$. Model (G.1)-(G.4) can be either solved to optimality or stopped when an incumbent solution is found.

When the method is trained to not underpredict the capacity, this iterative training approach converges to a model that does not underpredict the capacity. Proposition Appendix G.1 and Appendix G.2 show that coupling the adversarial approach with machine learning models providing perfect approximations of the capacity consumption yields the optimal solution to the integrated lot-sizing and scheduling problem. While these propositions require assumptions that would make the approach inefficient, they provide a theoretical basis for the approach considered.

Proposition Appendix G.1. *The adversarial example generation stops after a finite number of iterations when applied to piecewise linear machine learning models trained to never underestimate the makespan and whose breakpoints remain identical between each training procedure.*

Proof. Proof:

As explained in Proposition 4.1, since the makespan C_{max} of any FJSP is defined as a piecewise linear function, there are a finite set of regions $\mathcal{R}_1 = \left\{[a,b] \mid a,b \in \mathbb{R}^{|J|}, a_i \leq b_i \ \forall i \in J\right\}$ of lot sizes X where $C_{max}(X)$ is linear for all $X \in r$ and any $r \in \mathcal{R}_1$. Suppose we have a piecewise linear machine learning model ML trained on a dataset D to never underestimate its output, here the makespan of an FJSP. Similarly, we can define $\mathcal{R}_2$ as the finite set of intervals of lot sizes X where the prediction of ML is linear. Therefore, there are a finite set of intervals of lot sizes $\mathcal{R}$ where both C_{max} and ML are linear, i.e. $\mathcal{R} = \{r_1 \cap r_2 \mid r_1 \in \mathcal{R}_1, r_2 \in \mathcal{R}_2\}$. Applying the adversarial example method on machine learning model ML is equivalent to looking for the lot sizes $X^* \in \mathbb{R}^{|J|}$ where the absolute distance between prediction $ML(X^*)$ and $C_{max}(X^*)$ is maximal. Such a solution, if it exists, lies in a region $r^* \in \mathcal{R}$ where both function C_{max} and ML are linear. If an optimal solution X^* can be found, two cases can occur:

1. X^* is not unique and lies in a k-face of $r^*, k \in 1..|J|$
2. X^* is unique

In the first case, there is a k-face K of r^* where $C_{max}(X_1) - ML(X_1) = C_{max}(X_2) - ML(X_2), \forall X_1, X_2 \in K$. In this situation, machine learning model ML is parallel to C_{max} for all solutions lying in k-face K. For example, for a k-face K with $k = |J|$, finding optimal solutions X for all $X \in K$ means ML is parallel and strictly lower to C_{max} within region r^*. Training machine learning ML with the newly added X^* will prevent this case occuring for region r^* since the prediction of ML over k-face K will lie above C_{max}. In the second case, X^* is an extreme point (or a bound) of region r^*, and training ML by including this new extreme point automatically forbids the procedure to return this same point. For each region $r \in \mathcal{R}$, repeatedly applying the adversarial generation procedure to ML will, in the

worst case, generate one data point for each k-face of region r and one data point for all the extreme points of region r. Once all the extreme points of a region $r \in \mathcal{R}$ have been added to dataset D, machine learning model ML cannot underpredict the makespan for any lot size $X \in r$. $\mathcal{R}$ having a finite set of regions and each region being represented by a finite set of extreme points and facets, the adversarial generation procedure ends after a finite number of iterations.

□

Proposition Appendix G.2. *If a machine learning model always returns exact approximations of the capacity consumption in the training dataset and never overestimates the capacity consumption in any other data sample, applying the adversarial examples generation converges to a model that returns perfect estimations of the makespan.*

Proof. Proof: As the machine learning model does not overestimate the capacity consumption, the solution to the model provides a lower bound to the optimal solution of the integrated lot-sizing and scheduling problem. The solution is either optimal or it is infeasible because it violates the actual capacity constraints. If no adversarial example exists, the solution is optimal. □

An example of the perfect approximation of the capacity consumption is a machine learning model that integrates the makespan C_{max} as a feature through equation (G.3). Such a model will represent the capacity consumption perfectly, but its integration into the lot-sizing model leads to an inefficient program. As a consequence, machine learning models as well as the features used for the approximation should be chosen appropriately to balance the trade-off between accuracy and computational efficiency. We provide below additional practical implementation details for the approach.

For piecewise linear regression, we decompose the problem into $|R|$ subproblems, one for each linear regression associated with each region. This procedure may require a long time to find the first feasible solutions due to the complexity of the constraints (G.3), these constraints require a large set of binary variables and big-M constraints. To avoid generating unrealistic lot sizes, we set appropriate domains in the model for the value of $\mathcal{Y}^{pred}$, and C_{max}.

To speed up the solution process further, we decompose the problem into $|J|$ subproblems depending on the number of product setups. We first solve the MILP above with a constraint that sets the number of setups Y to 1, and we solve the procedure until unfeasibility is reached. When a single product is set up for the scheduling problem, only one sequence is required to represent the full scheduling problem. As a result, we solve or prove the infeasibility of the mathematical program (G.1)-(G.4) more rapidly. We then increase the

	Setup costs	15			50			100		
	T	5	30	50	5	30	50	5	30	50
	UB	1595	9409	15716	2252	12603	20970	3076	16605	27574
	LB	1595	9409	15716	2252	12401	20487	3076	15924	26057
ILS-CLR (D_{LHS})	Gap(%)	0	0	0	0	1.6	2.3	0	4.1	5.5
	Feasibility	100	100	100	100	99	97	98	91	86
	Time (s.)	0.02	5.0	79.9	0.12	3600	3600	0.48	3600	3600
	UB	1600	9435	15756	2289	12721	21162	3144	16867	27987
	LB	1600	9435	15756	2289	12530	20696	3144	16141	26419
ILS-CLR (D_{KP})	Gap(%)	0	0	0	0	1.5	2.2	0	4.3	5.6
	Feasibility	100	100	100	100	100	100	100	100	100
	Time (s.)	0.03	12.1	495.3	0.15	3600	3600	0.53	3600	3600
	UB	1600	9434	15754	2289	12704	21134	3141	16826	27919
	LB	1600	9434	15754	2289	12513	20647	3141	16085	26327
ILS-CLR (D_{ADV})	Gap(%)	0	0	0	0	1.5	2.3	0	4.4	5.7
	Feasibility	100	100	100	100	100	100	100	100	100
	Time (s.)	0.03	14.75	526.1	0.15	3600	3600	0.56	3600	3600

Table H.9: Comparison of the constrained linear regression trained on different datasets with scheduling size 6×6

number of setups by one and repeat the process until we reach unfeasibility for any number of products set up. Increasing the number of setups also increases the number of sequences required to represent the scheduling problem exhaustively, but it is faster than considering all the possible number of setups. Note that each time an adversarial example is found, all the $|J|$ subproblems are checked.

Appendix H. Evaluation of the training approach

The rest of this section evaluates the iterative training approaches given in Section 5. We consider three constrained linear regressions trained on three datasets, namely Latin Hypercube Sampling (D_{LHS}), the ILS-KP method (D_{KP}), and the adversarial example approach (D_{ADV}). We only consider instances with scheduling size 6×6 in these experiments since the adversarial approach does not scale well for large-scale instances. However, we vary the cost values since these parameters have a significant impact on the quantities produced at each period, and thus the capacity consumption. Dataset D_{LHS} was computed by generating $10,000$ data samples composed of different combination lot sizes. To generate datasets D_{KP} and D_{ADV}, the first 100 data samples from D_{LHS} are selected to compose the initial dataset and used to fit the constrained linear regression. The remaining data samples were generated as explained in Section 5.2.2 and Appendix G, and led to final datasets with sizes varying from 107 to 1134 samples.

Table H.9 reports the results of the embedded constrained linear regression trained using the three data generation procedures. Table H.9 shows that Latin Hypercube Sampling

yields solutions with lower objective values. However, this method leads to a low number of feasible solutions for instances with high setup costs. The adversarial example approach however can guarantee the feasibility for any lot-sizing instances, since the datasets are generated so that no constrained linear regressions trained with this dataset can underestimate the capacity consumption. The ILS-KP approach shows results that are relatively similar to the adversarial example one, with a feasibility of 100 for all the lot-sizing instances, despite finding solutions with slightly higher objective values. Nevertheless, the ILS-KP approach can scale to large-scale lot-sizing instances with large scheduling sizes. Therefore, for the next numerical experiments of this paper, we considered constrained linear regression and constrained piecewise linear regression trained using datasets generated with the ILS-KP approach.