Computers & Industrial Engineering 196 (2024) 110507

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

A machine learning-based simulation metamodeling method for dynamic scheduling in smart manufacturing systems

Erfan Nejati [a,d], Ensieh Ghaedy-Heidary [a,b], Amir Ghasemi [c,*], S. Ali Torabi [a]

[a] School of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran
[b] Department of Management Sciences, University of Waterloo, Waterloo, Canada
[c] Institute AIFB, Karlsruhe Institute of Technology (KIT), Kaiserstr. 89, Karlsruhe, Germany
[d] Rotman School of Management, University of Toronto, Toronto, Canada

ARTICLE INFO

Keywords:
Smart manufacturing
Digital twin
Machine learning
Simulation
Metamodeling
Stochastic flexible job shop

ABSTRACT

Conventional Digital Twins (DTs) in smart manufacturing rely on complex and time-intensive simulation models, hindering real-time DT-based decision-making. However, the availability of big data in Manufacturing Execution Systems (MES) enables training different Machine Learning (ML) models for fast and accurate predictions and decision assessments. Accordingly, this paper proposes an ML-Based Simulation Metamodeling Method (MLBSM) to facilitate DT-based decision-making for dynamic production scheduling in complex Stochastic Flexible Job Shop (SFJS) environments. The proposed MLBSM integrates three modules: a novel data vectorizing method (SPBM), multi-output Adaptive Boosting Regressor (ABR) models, and a new statistical risk evaluation method. SPBM converts unstructured production log data into numerical vectors for ABR training by calculating numeric penalty scores for each job based on the position of operations in the schedule queue. Each trained ABR predicts mean job completion times for various dynamic scenarios based on shift schedules. The risk evaluation method estimates the standard deviation of job completion times and calculates the delay probability scores for each job, aiding DT in promptly evaluating production schedules. Working seamlessly together, MLBSM modules present a novel way of using production log data for ML training and ultimately bypassing several computationally intensive simulation replications. In this research, a simulation model generates the synthetic MES data, focusing on the machining process at a photolithography workstation in the semiconductor manufacturing. Experiments demonstrate the MLBSM's accuracy and efficiency, predicting high-risk jobs with over 80% recall and being at least 70 times faster than conventional simulation runs. Sensitivity analyses also confirm the MLBSM's consistency under different workstation conditions.

1. Introduction

The considerable digital transformation brought by Industry 4.0 (I4.0) has led to the advent of various information and communication technologies in the context of smart manufacturing environments, such as Cyber-Physical Systems (CPSs) (Ghasemi, Farajzadeh, Heavey, Fowler, & Papadopoulos, 2024). CPSs provide a network of cyber and physical counterparts with the ability to learn, detect changes, and make autonomous and real-time decisions (Lee, 2008; Leitão, Colombo, & Karnouskos, 2016). CPSs are used to create a virtual copy of the actual system, which is then monitored to collect data and make real-time decisions (Hermann, Pentek, Otto, et al., 2015). The aforementioned abilities of the CPSs have resulted in the advent of Digital Twins (DTs), which have been gaining attention in recent years.

More precisely, DTs are digital copies of physical systems hosted by the cyber side of CPSs. As Fig. 1 shows, by interacting with the shop floor through MESs, DTs are employed to plan and control complex manufacturing systems (García, Bregon, & Martínez-Prieto, 2022). To accomplish this, DTs require a competent tool to provide a valid and detailed representation of the physical system. This is usually provided by a detailed simulation model. Simulation models have long proven their potential in modeling complex and stochastic manufacturing environments (Pires, Cachada, Barbosa, Moreira, & Leitão, 2019). In other words, simulation models are dominantly used for evaluating system designs and planning decisions within manufacturing systems. Accordingly, DT-based decision-making is conducted by assessing the performance of the system under different scenarios using the simulation model. In this way, based on the performance measures calculated

* Corresponding author at: Institute AIFB, Karlsruhe Institute of Technology (KIT), Kaiserstr. 89, Karlsruhe, Germany.
E-mail addresses: e.nejati@mail.utoronto.ca (E. Nejati), eghaedyh@uwaterloo.ca (E. Ghaedy-Heidary), amir.ghasemi@kit.edu (A. Ghasemi), satorabi@ut.ac.ir (S.A. Torabi).

https://doi.org/10.1016/j.cie.2024.110507

Available online 20 August 2024

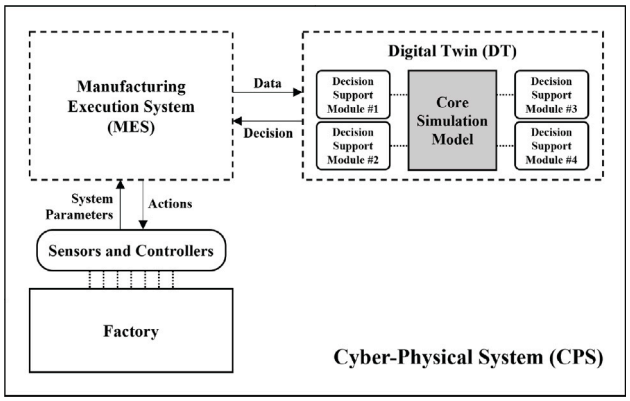


Fig. 1. A general framework for DT.

by the simulation model for each test scenario, the best decision can be made for the system under study.

Although simulation-based approaches are among the most powerful tools in modeling complex manufacturing systems, simulation runs can become highly time-intensive. This is mainly due to the fact that simulation models perform various numerical calculations for each entity or process involved in manufacturing systems (Pierreval, 1992). While in some cases, individual analysis of each entity or minor process might be required, in many others, simulation models are solely used to make an overall decision for the whole system based on performance measure values they calculate regarding each decision alternative. For instance, as it is shown for the case of DTs in Fig. 1, simulation runs are integrated with decision support modules that are usually some optimization methods (e.g., Genetic Algorithm) specifically designed to explore the solution space of different decision problems in the factory. In this collaboration, each optimization method passes one or a set of feasible solutions to the simulation model and waits until the simulation calculates and returns the corresponding system performance measure values for each feasible solution. Subsequently, based on the calculated performance measures, the optimization method compares the alternatives and makes the final decision, which could hopefully be an optimal one. However, it takes a significant amount of time for each performance measure to be calculated by the simulation model. Additionally, given the uncertain parameters involved in the simulation model, such as the stochastic nature of incoming entities, simulation runs must be replicated multiple times for each feasible solution to ensure reaching reliable calculated values. This computational demand can become even more substantial when considering the vast solution space that most decision problems encompass and the millions of feasible solutions that need exploration and evaluation in this process. In smart manufacturing environments (e.g., semiconductor manufacturing plants), online orders are placed every minute by customers, and several dynamic events, such as equipment status fluctuations, may happen at each moment; therefore, waiting about half an hour or even more to reschedule such a highly dynamic environment through a simulation–optimization collaboration makes no sense.

To overcome the computation complexity caused by simulation models, a potent substitution is needed to provide DT with the fast prediction of performance measures. Recently, the availability of big data collected via sensors of embedded Internet-of-Things (IoT) systems and stored at MES enables modern manufacturing systems to train Machine Learning (ML) models. These models can be employed to make time-efficient and accurate predictions about the system's performance. To be more specific, a subclass of ML methods, known as supervised ML methods, can be used to predict a target value (performance measure) based on independent variables (decision alternatives). Supervised methods can train accurate models to predict the value of performance measures (simulation outputs) based on each alternative decision (simulation inputs) (Monostori, 2002). Within the literature, such abstract models fitted on inputs and outputs of a simulation model are referred to as *Simulation Metamodels* (model of a model) (Can & Heavey, 2012). These ML-based metamodels can be considered a competent substitution for simulation models to provide fast and accurate predictions of system performance measures. This enhances the applications of DTs by providing required system metrics to them in real time without spending a huge computation capacity.

Considering the advantages mentioned above of ML-based metamodels, these methods are powerful tools to address complex problems within manufacturing systems in the era of I4.0. In this regard, this paper develops a Machine Learning-Based Simulation Metamodel (MLBSM) to address scheduling problem in the semiconductor manufacturing. As one of the early adaptors of I4.0, semiconductor manufacturing has one of the most capital intensive production systems (Mönch, 2018). Within this manufacturing system, the photolithography workstation is known as the bottleneck resource. Since the system's performance is highly dependent on the performance of its bottleneck, developing a schedule for the photolithography workstation can considerably improve the performance of the whole manufacturing system (Chen, Chen, & Hung, 2020). Thus, the proposed MLBSM focuses on the scheduling problem within the photolithography workstation while it can be tailored for different types of scheduling problems which can be seen in different manufacturing systems.

Furthermore, three special operational constraints differentiate the scheduling problem of the photolithography work area from other scheduling problems. These constraints, which are known as CAPPA constraints in the literature, refer to machine process capability, machine dedication, and the maximum number of times each reticle can be shared among different machines (Chen, Fathi, Khakifirooz, & Wu, 2022; Hu & Zhang, 2012). While simultaneous consideration of all these constraints is crucial to the development of DTs, it dramatically increases the simulation model's complexity and has a substantial impact on the amount of calculation required for a single simulation replication. In addition, highly flexible operations and stochastic setup and processing times are two inherent characteristics of the photolithography work area, which are further obstacles in the way of developing real-time DTs. In fact, these uncertainties add noise to the values calculated by the simulation model, and obtained results from a single simulation replication might not be reliable anymore. Hence, statistically, several replications are needed to ensure those calculated values are valid, which makes the simulation even more time-consuming.

On the other hand, dynamic events, such as job arrivals and machine breakdowns, are likely to happen in this workstation. These events prompt the need for real-time scheduling for this workstation. This is mainly due to the fact that whenever a dynamic event occurs, a new schedule needs to be developed (Chien & Lan, 2021; Xu, Shao, Yao, Zhou, & Pham, 2016).

In order to cope with the resource-intensive computations and achieve a real-time DT, the MLBSM is developed based on an ensemble supervised learning method known as Adapted Boosting Regressor (ABR), which predicts the performance measure (the risk of final product delays) of the photolithography workstation quickly and accurately. In addition to its accuracy and speed, ABR, as a simple tree-based ML algorithm, makes the MLBSM explicable, which is not always the case for other AI methods. For instance, while Deep Learning (DL) methods, which primarily rely on Neural Networks, may yield more accurate outputs, they are generally more time-intensive to train and are also not as explainable as tree-based approaches. The transparency inherent in all modules of the MLBSM, including ABR, can significantly enhance the likelihood of managers relying on this innovative tool. Employing the MLBSM as a replacement for the simulation model enables the developed DT to evaluate a vast number of alternative schedules quickly. In this way, using less computational resources, real-time decisions can be made dynamically in such a highly complex manufacturing

environment. It is important to note that simulation in a DT provides insights at a more detailed level, including individual jobs, machines, and even operations. This allows decision-makers, whether human or computer-based, to evaluate their alternatives with greater precision. Such detailed insights can also support decision-making in various areas beyond scheduling, such as inventory management, capacity planning, marketing, and delivery. To maintain this level of detail in DT, the MLBSM predicts results (delay risks) at the job level rather than for the workstation as a whole.

Training ML models for manufacturing environments is highly dependent on both the quality and size of the MES data (Dahmen et al., 2019). The data used for training needs to provide knowledge over all states (e.g., normal and abnormal operating conditions) that the model should consider. These all-inclusive data characteristics enable the ML model to provide DT with valid predictions over all possible operating conditions, even those that rarely happen on the shop floor (e.g., breakdowns or special order arrivals). As a result, depending solely on the collected data from the shop floor may not be sufficient for training a reliable ML model. This is mainly due to the fact that the data obtained from the sensors mostly represent normal and near-optimal working conditions, which the system was planned for and has operated based on that so far. In addition, considering supervised ML, another limitation is that a human manually labels the vast amount of input data. Manual labeling is prone to errors and is labor and time-intensive, especially in dynamic manufacturing environments with frequent changes in products and processes. These limitations may prohibit the development of successful ML applications in several manufacturing cases. In that direction, creating virtual (synthetic) datasets using simulation models could facilitate the creation of proper datasets for training ML models in a cost and time-efficient way (Alexopoulos, Nikolakis, & Chryssolouris, 2020).

Simulation models can be employed for creating valid training datasets, including a wide range of system states, as well as for automating the data labeling process. In this regard, this paper introduces a discrete-event simulation model representing the machining process in a photolithography workstation. The simulation model considers the existing assumptions of a real-world photolithography workstation and aims to generate virtual MES data for training ML models and validate the obtained ML predictions.

In summary, this paper tries to answer the following questions:

1. How to generate valid synthetic MES log data with the required quality and quantity, which embodies all aspects of the photolithography process?
2. How to preprocess raw MES data and train ML models such as the MLBSM which are capable of predicting the scheduling performance measures?
3. How can the developed the MLBSM be used for DT-based real-time decision-making in a dynamic environment?
4. How accurately does the MLBSM work in a DT to provide timely predictions regarding different operating conditions?

Accordingly, to address the above-mentioned research questions, the following content is provided in this paper.

- Considering the machine process capability, machine dedication, and maximum reticles sharing constraints (CAPPA constraints), simultaneously, as well as the stochastic nature of the photolithography workstation (stochastic processing times and sequence-dependent setup times) to design a simulation model. The model simulates the machining process of the workstation based on several random schedules and generates virtual MES log data which can be used to develop a DT (Question 1).
- Proposing new sequencing priority-based vectorizing method to convert different schedules to trainable numerical vectors. This vectorizing method extracts numerical penalty scores for each job based on the positions of their operations in the scheduling queue. The numerical vector representations of the schedules are then used to train an ABR method to predict the expected value of completion times for each job (i.e., the system's performance measure) (Question 2).
- Proposing a new empirical statistical method to convert the predicted completion times to a delay risk score for each job at the beginning of each production shift. The risk prediction method enables the MLBSM to be used in a set of multiple consecutive production shifts. In this way, the MLBSM can dynamically estimate the delay risk scores at the beginning of each production shift. The fast and accurate risk prediction helps to evaluate the shift schedule in a real-time decision-making process (Question 3).
- Finally, four sets of experiments are conducted to evaluate the validity of MLBSM predictions, accuracy and efficiency of ML models, and the performance of the MLBSM as a DT-based decision support tool, under either normal or abnormal scenarios in the workstation (Question 4).

The remainder of this paper is organized as follows. Section 2 provides a literature review on the application of ML methods for developing simulation metamodels in production systems. Section 3 presents the production scheduling problem within the photolithography workstation. Section 4 describes the proposed the MLBSM to address the production scheduling problem while Section 5 provides the application of the MLBSM in a dynamic environment. Section 6 describes the ML model's calibration procedure, and Section 7 reports the numerical results of our experiments. Finally, Sections 8 and 9 concludes the paper and looks at possible future directions.

2. Literature review

As stated previously, this paper aims to develop a simulation metamodel facilitating DT-based decision-making for production scheduling within smart manufacturing environments. Considering ML's great potential in forecasting, it can enhance DT by developing accurate ML-based metamodels. Hence, this section investigates the literature on training ML models to develop a simulation metamodel within the context of manufacturing systems. In this regard, firstly, this paper focuses on the simulation metamodeling concept. Subsequently, various metamodeling methods and the problem domains to which metamodels are applied are introduced. Finally, the main gaps in the literature and how this paper aims to address them are elaborated.

As it is described in Fig. 2, a simulation metamodel is a function, say f', that takes some design parameters of a simulation model (f) as inputs, represented here by a vector X, and generates an output $Y' = f'(X)$ (Barton, 2020). Examples of model design parameters include the probability distribution of the input parameters, such as arrival rate and mean service time, and system configuration parameters, such as the number of servers, service priority, operational protocols, and buffer capacity (Barton, 2020). In fact, the metamodel f' produces an approximation of some characteristic of a simulation output Y, e.g., the mean or standard deviation of Y. This output is usually an index indicating the performance measure of the system. Time in the system for a set of jobs or customers, utilization of a particular resource (e.g., operator, machine), and the net revenue over a specific time period are some examples of simulation outputs. Generally, these outputs are averaged over the length of a simulation run, but the averages vary randomly for each simulation run.

According to Barton (2020), Metamodels have superiority over simulation models mainly due to three reasons. Metamodels generally have explicit form, they have deterministic outputs, and finally, once fitted, they are computationally inexpensive to evaluate. Due to their explicit form, metamodels are considerably insightful in understanding the relationship between design parameters and performance measures related to simulation outputs. Also, fitting a metamodel allows for characterizing global variation of the stochastic outputs in a deterministic

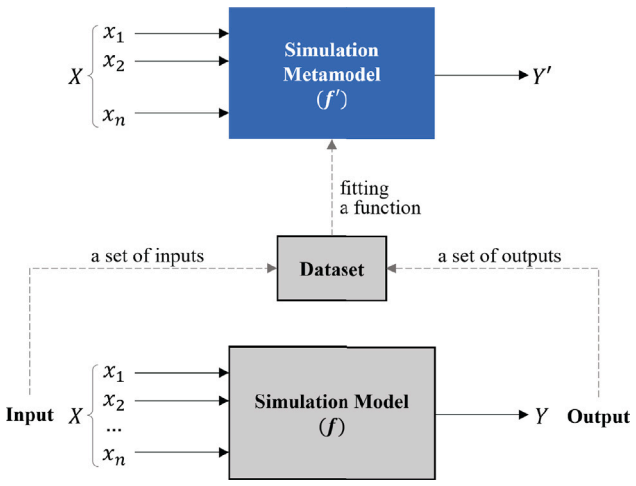


Fig. 2. The metamodeling framework.

value describing the output, e.g., the mean or standard deviation value. Finally, once fitted, metamodels can be used as a proxy to estimate performance measures instead of making computationally expensive and stochastic simulation replications. In this way, metamodels can be widely used to ease many computationally intensive operations, such as optimization, robust design, etc. (Amorim, Antunes, Ferreira, & Couto, 2019; Barton & Meckesheimer, 2006; Dellino, Kleijnen, & Meloni, 2009). In fact, the idea of developing simulation metamodels to facilitate DT-based decision-making is mainly based on their computationally inexpensive characteristic. Metamodels can help DT with fast evaluation and comparison of various feasible decisions to find the optimum one.

However, there are some undeniable challenges in developing metamodels (Barton, 1998), and in a comprehensive work, Shao and Kibira (2018) explore a wide range of them in the context of DT. The major issues in metamodeling are the selection of a set of input variables and justifying them to observe the output, as well as the selection of a functional form for the approximation function. Generally, metamodeling assumes that all input and output variables can take on continuously varying values (continuous problems). In contrast to the extensive usage of metamodeling techniques for continuous issue domains, discrete inputs and outputs (discrete problems) are rarely addressed by these techniques. Discrete problems deal with ordinal integers, categorical (qualitative) variables, permutations, and trees (Bartz-Beielstein & Zaefferer, 2017). They may be mixed among themselves or mixed with continuous variables. Ordinal integer variables can often be handled quite similarly to real-valued variables. Others, like permutations, are too complex to be easily represented by numeric vectors.

According to a survey conducted by Bartz-Beielstein and Zaefferer (2017), there is a set of six metamodeling strategies to deal with discrete problems:

(1) The naive approach: As long as the data can still be represented as a vector (binary variables, integers, categorical data, permutations) the modeling technique may simply ignore the discrete structure, and work as usual;
(2) Custom modeling: A specific modeling solution is tailored to fit the needs of a certain application;
(3) Inherently discrete models: Some models already are discrete in their own design. For instance, tree-based models, like regression trees or random forests;
(4) Mapping: Often, discrete variables or structures may be mapped to a representation which can be handled easier. Random key mapping for permutations or dummy variables for categorical variables are two instances of this type of approach. Similar to strategy 1, this approach may introduce redundancy or infeasibility into the data structure;
(5) Feature extraction: Instead of directly modeling the relation between an object (or its representation) and its quality, it is possible to calculate the real-valued features of the objects. For example, some features of a tree or graph can be extracted (path lengths, tree depths, etc.). These numeric features can then be modeled with standard techniques;
(6) Similarity-based modeling: Where available, measures of (dis)similarity may be used to replace continuous measures that are, e.g., employed in similarity-based models like k-nearest neighbor (k-NN), support vector machines (SVM), radial basis function networks (RBFN), or Kriging.

The above-mentioned strategies are not necessarily mutually exclusive. Depending on the point of view, a mapping approach can be interpreted as a similarity-based approach or vice versa. Moreover, none of these strategies can have superiority over the others, even the naive approach may be favorable if the problem is sufficiently simple (Bartz-Beielstein & Zaefferer, 2017). However, in general, one of the drawbacks of the more complex strategies (e.g., strategies 5 or 6) is extracting and adjusting proper features and measures to predict simulation outputs. Also, this may be problematic if these measures have to meet further requirements, like definiteness.

Another major issue in developing metamodels is the choice of a functional form for the approximation function. This selection depends on different factors such as the purpose of metamodeling, complexity of the system, type of input and output variables, deterministic or stochastic nature of outputs, and local or global approximation properties (Barton, 2009). While a wide range of functions is used within metamodels (mentioned in Table 1), metamodels can be categorized into three main groups based on their functional forms.

The first group consists of Regression Metamodels, such as those with any linear, polynomial, quadratic, or nonlinear functional forms. Regression models are simple and fast to fit on the data. Due to their simplicity, many direct insights into the simulation's behavior can be obtained from the fitted model. For example, the linear coefficients' magnitude indicates the simulation output's relative sensitivities to all design parameters (over the defined ranges of parameter values) (Barton, 2020). Similarly, quadratic coefficients can indicate nonlinearity and convexity/concavity. Kriging (Gaussian Process) Metamodels (also called Spatial Correlation), are the second group of metamodeling functions that have great flexibility. Thus, they can model more complex response function shapes compared with regression metamodels. If one requires a global approximation to a nonlinear response, kriging metamodels are an attractive alternative for regression. However, fitting kriging models on data is more challenging compared to regression metamodels. Furthermore, while fitted model coefficients give some indication of how sensitive is the response changes to input change, the detailed insight provided by a linear regression model cannot be obtained from a kriging model. In addition, these models can be sensitive to parameter choices (Gramacy & Lee, 2010).

Finally, the third group belongs to ML-based Metamodels. ML, a subset of AI, is a group of computer techniques that focuses on extracting useful knowledge from data and letting the ML component decide. Supervised ML models e.g., Neural Networks, Tree-based, and Support Vector Machine Methods, are widely acknowledged as promising tools for decision-making in manufacturing systems (Alexopoulos et al., 2020). These methods can develop models to provide fast and highly accurate predictions in the most complex systems (Min, Lu, Liu, Su, & Wang, 2019). These capabilities of ML models make them suitable for estimating proxy functions for the most complicated simulation models. However, MLs are generally viewed as black boxes. While regression coefficients or spatial correlation parameters are explainable, ML coefficients generally do not provide an explanation of the impact of independent variables on simulation output. Also, another challenge in developing ML-based metamodels is that they are highly sensitive to parameter selection and strategies dealing with discrete and mixed inputs (Barton, 2020).

Table 1
Summary of publications on metamodeling for a production scheduling problem with FS = Flow Shop; JS = Job Shop; FJS = Flexible Job Shop; and OS = Open Shop.

Reference	Production environment	Uncertainty in the input variables	Metamodeling strategy	Functional form	Response variable	Aim of metamodeling
Pierreval (1992)	FS	–	Naive approach	ML (Neural Network)	Mean tardiness	Dispatching rule selection
Pierreval and Huntsinger (1992)	JS	–	Naive approach	ML (Neural Network)	Mean tardiness	Dispatching rule selection
Weeks and Fryer (1977)	JS	–	Naive approach	Regression (Linear and nonlinear)	Earliness and tardiness cost	Providing insight
Koons and Perlic (1977)	FS	–	Naive approach	Regression (Linear)	Mean waiting time, total delay	Providing insight
Fonseca, Navaresse, and Moynihan (2003)	JS	Processing times	Mapping approach	ML (Artificial Neural Network)	Flow time	Providing insight
Ankenman, Nelson, and Staum (2010)	FS	Processing times	Similarity-based modeling	Stochastic Kriging	Mean cycle time	Providing insight
Azadeh, Moghaddam, Geranmayeh, and Naghavi (2010)	FS	Setup times and machine breakdown	Naive approach	ML (Artificial Neural Network)	Total tardiness	Dispatching rule selection
Yang, Liu, Nelson, Ankenman, and Tongarlak (2011)	JS	–	Naive approach	Regression (Nonlinear)	Mean cycle time	
Nasiri, Yazdanparast, and Jolai (2017)	OS	Job arrivals and processing times	Naive approach	ML (Artificial Neural Network Multilayer Perceptron)	Mean waiting time	Dispatching rule selection
Ghasemi, Ashoori, and Heavey (2021)	JS	Processing times and setup times	Feature extraction	ML (Genetic Programming)	Completion times	Developing a simulation optimization method
This study	FJS	Processing times and setup times	Feature extraction	ML (Adapted Boosting Regressor)	Completion times and jobs' risk	Providing real-time DT-based decision support

Regression models have been the popular tools for developing simulation metamodels in earlier studies. In this regard, Weeks and Fryer (1977) proposed a regression simulation metamodel for estimating minimum cost due dates in a Job Shop (JS) production system. In another study, Koons and Perlic (1977) analyzed a steel plant by regression simulation metamodel. For predicting product mix and cycle time as a function of throughput, Yang et al. (2011) developed a nonlinear regression metamodel. In a more complex manufacturing environment, for instance, semiconductor fabrication, Ankenman et al. (2010) extend the kriging methodology to design a metamodel. Their model represents the steady-state mean cycle time of a product.

ML models were also implemented in developing metamodels. In this regard, a neural network approach was proposed by Pierreval (1992) to select dispatching rules for a simplified flow shop manufacturing system. The obtained data from a simulation model was used to train the network. The trained network was able to select the best dispatching rule itself. In another study, Pierreval and Huntsinger (1992) applied an artificial neural network for simulation metamodeling of a JS with deterministic processing time, transportation time, and a constant rate of job arrival. Fonseca et al. (2003) also applied the same method to estimate the mean lead times for orders processed in a JS. Azadeh et al. (2010) proposed a trained multilayered neural network metamodel integrated with a fuzzy simulation with the use of the α-cuts method. Their algorithm, which was developed to solve a complex maldistributed combinatorial dispatching decision problem, is capable of modeling nonlinear and uncertain problems. In 2017, a simulation–optimization approach was presented by Nasiri et al. (2017) to address a stochastic open shop problem. Employing a multilayer perceptron artificial neural network, radial basis function, and data envelopment analysis, they tried to determine the most efficient dispatching rule for each machine. In a more recent study, Ghasemi et al. (2021) proposed a Genetic Programming (GP) metamodel to be replaced with the simulation replications in a stochastic environment. A novel training vector was developed to train GP, which was further integrated into an evolutionary optimization approach to sort feasible solutions of an SJSSP in a semiconductor manufacturing system. Table 1 presents a summary of the reviewed articles.

As illustrated in Table 1, although a number of researchers developed metamodels to address production scheduling problems, the models are still at rudimentary stages. On one hand, conventional metamodels are mostly developed within simple production environments such as FS and JS with small number of machines or operations which is not a competent representation of the real-world problems. On the other hand, these papers have mainly applied metamodels for providing insights by correlation study, selecting dispatching rules, or sorting different schedules. The common characteristic of all these applications is that they were looking for ratios and proportions of response variables but not an exact estimation of their values. In addition, none of the reviewed papers has reported any measurements to represent the accuracy of their metamodels. These simplifications, lack of accurate predictions, and weak functional form are the main drawbacks of using naive approaches to deal with simulation inputs. However, when it comes to developing a metamodel to be used within a DT, the least level of simplification is allowed, as DTs need to be as realistic as possible. In addition, providing fast and accurate predictions about the future of the system is one of the key roles that DTs have. Thus, despite the formerly developed metamodels, those applied to DTs need to accurately predict response variables, not just a relative comparison between them.

Considering the abovementioned, this paper introduces a novel and accurate metamodeling method, denoted as MLBSM, to facilitate DT-based decision-making for a large-scale production scheduling problem within one of the most complex production environments known as Stochastic Flexible Job Shop (SFJS). With all its real-world assumptions, the photolithography workstation was considered to develop the MLBSM. These assumptions include CAPPA constraints (special constraints belonging to the photolithography processes), stochastic processing times, and stochastic sequence-dependent setup times. Based on

the schedule(s) for each production shift (input variable), the MLBSM will be able to predict the mean completion times as well as tardiness risk scores for each job as response variables in a timely manner.

3. Problem description

AI is a well-known I4.0 tool preponderantly used to improve the performance level in various fields. As AI technology evolves to become a more affordable tool, its applications in production operation planning assist modern manufacturing systems with complex decision-making processes. ML, a subset of AI, is a set of computer techniques that uses learning or training to retrieve meaningful information from a large amount of data. The retrieved information can further be used to make real-time decisions for the business or factory that the data belongs to Ahuett-Garza and Kurfess (2018).

It is possible to group ML methods into different categories based on their outputs. Common algorithm types include (Monostori, 2002) (a) supervised learning in which a supervisor provides the correct response to each action, such as human or real-world data, (b) reinforcement learning, where less feedback about each response is given compared to the previous category (only an evaluation of each action is given by the supervisor), and (c) unsupervised learning in which no evaluation of actions is provided, assuming that there is no supervisor or known feedback. Focusing on supervised learning, in this method, the algorithm tries to find a pattern on a set of known actions and responses (i.e. the available data), which is called a training dataset. This function can be described as follows:

$$Q = q(\{z_1, \dots, z_n\}) \tag{1}$$

where $\{z_1, \dots, z_n\}$ is the vector of independent variables that describe possible actions and Q is an estimation of response(s) calculated using the ML fitted function q. Training ML methods on MES data to fit such functions within smart manufacturing systems can be beneficial. In other words, the trained ML models can provide fast and accurate predictions for a wide range of performance measures (responses) to evaluate operational decisions (actions) before implementing them on the shop floor. These real-time predictions allow decision makers (a human or DT-based decision support tool) to compare alternative decisions to either minimize or maximize the performance measure in the most complicated manufacturing environments.

SFJS environments are one of these complex manufacturing environments that represent many modern manufacturing systems as well as the photolithography (Zhang, Wang, Qiu, & Liu, 2023). Within SFJS, a certain number of jobs must be processed on a certain number of machines. Each job contains a certain number of operations. Moreover, at least one of the operations can be processed on a set of machines with a stochastic processing time for each. Within the photolithography workstation, an example of SFJS environment, a number of wafers (jobs) which consists of a number of layers (operations) are to be processed during each production shift. There are sets of machines to perform photolithography processes, and the processing time for each process is a stochastic parameter. Planning operations on machines in SFJS is commonly referred to as the Stochastic Flexible Job Shop Scheduling Problem (SFJSSP), a complex problem belonging to the NP-hard class of optimization problems due to the large solution space. In addition, particularly in photolithography workstations, the complexity of SFJSSP increases exponentially by considering practical assumptions of the workstation such as uncertainties, machine setup times, and photolithography operational constraints, machine capability, machine dedication, and maximum reticle sharing constraints. SFJSSPs mainly try to find an optimum schedule for the machines which process operations in order to complete and deliver jobs on time. The completion time for each job is an important indicator within SFJS as it directly impacts holding and penalty costs caused by early and tardy jobs, respectively. Consequently, calculating completion times as a performance measure is essential within SFJSs to compare alternative decisions (the production shift schedule).

Due to the complexity and uncertainties involved in SFJSSPs, accurate calculation of completion times is also a challenge. Consider a simple FJS environment (without stochastic parameters) with 10 jobs, 10 operations per job, and 10 machines capable of processing all operations. A Job Shop Queue which is a sequence of $10\times10 = 100$ operations (i.e., $X = [OP_1, OP_2, \dots, OP_{100}]$) can be considered as a feasible solution for this scheduling problem. Operations are assigned to machines based on this sequence. Due to the flexibility of the system, each operation in the queue can be assigned to a set of available machines in the workstation. As a result, $10\times10\times\cdots\times10 = 10^{100}$ different assignments are possible for this workstation based on a single job queue. Furthermore, when it comes to calculating completion times in an SFJS environment, considering stochastic processing and setup times, extend these discrete possible values to a continuous domain which can be described as $\infty \times \infty \times 10^{100}$. In this situation, the completion time for each job becomes a highly stochastic variable depending on both the decision and the stochastic system parameters.

To deal with the aforementioned challenge in evaluating each feasible solution (Job Shop Queue), typical DTs use several simulation replications to calculate the expected value of the completion times. Considering Table 2, the expected value of completion times for Job Shop Queue X calculated using a simulation-based DT, can be described as follows:

$$Y = \frac{1}{R}\sum_{r=1}^{R} f_r\left(X\right) \tag{2}$$

where f_r and R refer to the completion time values calculated in simulation replication r and the total number of simulation replications, respectively. Also, Y denotes the vector of expected completion times for all jobs. Although simulation models are capable of modeling complex and stochastic manufacturing environments, simulation replications are time and resource-intensive. As a result, simulation models lose their practicality when it comes to calculating system performance measures in SFJSSPs.

In contrast, supervised ML methods can provide fast and powerful DT-based decision support tools to estimate performance measures within smart manufacturing. In fact, these methods can be trained on a set of production data to predict performance measures such as completion times accurately substituting time-intensive simulation replications. In other words, the expected value of completion times for Job Shop Queue X calculated by an ML-based DT, can be described as follows:

$$Y' = f'\left(X\right); \quad f' \sim \frac{1}{R}\sum_{r=1}^{R} f_r\left(X\right) \tag{3}$$

where Y' is an estimation for the vector of expected completion times (Y), which is now predicted by a trained, supervised ML method, not the simulation model. The concept of supplanting a time-consuming simulation model with a trained ML model is called simulation metamodeling within the literature of AI applied to operation management problems. However, in this context, supervised learning approaches face two main difficulties: (1) the availability of appropriate datasets with sufficient quality and quantity and (2) adding labels to the dataset to train an ML model. According to Dahmen et al. (2019), three main types of data are used to train ML models: (a) in-vivo data that are captured from real-life situations that were not created or modified specifically for the purpose of data collection, (b) in-vitro data collected in a laboratory environment using physical sensors, and (c) synthetic data which are generated using computer simulation models. Although manufacturing companies typically have a large amount of in-vivo data, gathering and cleaning it takes time. Furthermore, training data must include knowledge of all states that the model should consider, such as normal and abnormal operating conditions. For instance, in the manufacturing environments, the majority of data typically reflects

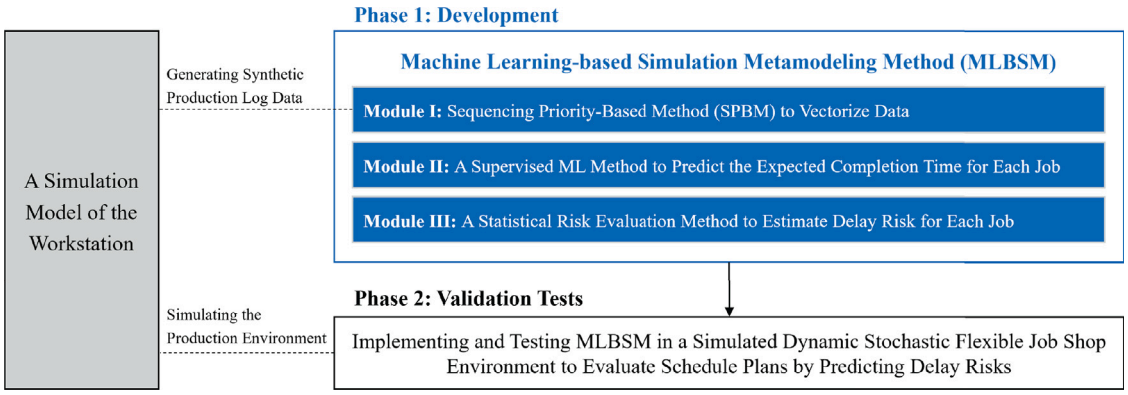


Fig. 3. The overall framework for developing, validating, and implementing the MLBSM.

undamaged parts, with actual defects being infrequent. Even defective parts often exhibit a varied distribution, where some types of defects are prevalent while others are exceedingly rare. Relevant situations requiring detection also tend to occur infrequently.

Moreover, another limitation of supervised ML is the human input required to label the vast amount of raw data manually. Manual labeling is prone to errors and is labor and time-intensive, especially in rapidly changing modern manufacturing environments where products and processes change frequently. For example, in many manufacturing environments workers visually inspect products on an assembly line, labeling items as defective or non-defective based on observed flaws. Workers also record repair and maintenance needs by manually logging information in production software or spreadsheets. Additionally, in many electronics manufacturing plants, new models of devices, such as smartphones, are released several times a year. Each new model may have different components, assembly processes, and quality control criteria. Manually updating and labeling data for each new model's assembly or inspection process can lead to significant errors, as the speed of change outpaces the ability of human workers to accurately label and categorize the data.

These limitations may make it difficult to develop successful AI applications in various manufacturing scenarios, as the time and cost required to build such models may be prohibitive. In this regard, using simulation tools to create virtual (synthetic) MES data could enable manufacturing systems to generate appropriate datasets for training ML models. Simulation models can be used to create valid training datasets and automate the data labeling process, which can then be fed into an ML algorithm. As a result, simulation can reduce the need for large amounts of high-quality data from the real world, as well as user participation in ML training and testing.

3.1. Overall framework

Taking into account the challenges mentioned above, this paper aims to develop the MLBSM as an ML-based metamodel to facilitate DT-based decision making. The overall framework of developing and implementing the MLBSM is presented in Fig. 3. The figure presents phases 1 and 2, the development and test implementation of the MLBSM, as well as the development of the simulation model for the photolithography workstation.

In order to address data availability and labeling, the initial step involves designing a simulation model of the photolithography workstation, which takes into account a comprehensive set of its specifications. This simulation model, described in Section 3.2, is further used to generate training datasets as input for the MLBSM. In the next step, several random schedules (SFJSSP's feasible solutions) are generated for all possible dynamic scenarios. All combinations of the jobs scheduled to be processed during each production shift and the tardy jobs from previous production shifts are considered in the aforementioned scenarios. Then, the developed simulation model, simulates the photolithography workstation for all of these randomly generated schedules. Subsequently, the simulation outputs are stored as virtual production log data in the MES (synthetic data).

The MES data is categorized based on possible dynamic scenarios of the fab, and it is used to create a number of training datasets for each scenario. Passing each dataset to the Sequencing Priority-Based Method (SPBM), a new effective vectorizing method, they get prepared to train ML methods. Further on, a number of multi-output Adapted-Boosting Regressors (ABRs) are trained on each vectorized dataset. These ABRs aim to predict the expected value of completion times for each job during each production shift. The trained ABRs are integrated with a novel empirical statistical method. The mentioned method empowers the MLBSM to predict delay risk scores for each job based on the predicted expected value of the completion times. Finally, the MLBSM is applied on a number of consecutive production shifts to evaluate the shift schedule by predicting the delay risk of the jobs at the beginning of each shift. To validate and evaluate the MLBSM, the predicted risk scores are compared with the simulated state of each job at the end of each production shift.

3.2. Simulation model for the photolithography workstation

As mentioned before, the simulation model developed in this paper aims to calculate the system performance measures (outputs) based on a Job Shop Queue (input). The simulation model is designed considering real-world assumptions of the photolithography workstation, such as stochastic parameters and all three CAPPA constraints, which are specialized assumptions related to this workstation. These constraints indicate that, first, certain machines within the photolithography tool must be qualified for different recipes (i.e. the machine process capability constraints). Secondly, for some critical layers (operations), specific machines within the toolset must be used to ensure the quality of the integrated circuit (i.e. the machine dedication constraints). Whether a layer is critical or non-critical, is an input parameter for the scheduling problem, which is related to the technical design of the wafer (job). Thirdly, the number of times a reticle is shared between different layers should be less than its maximum share limit (i.e. the maximum reticle sharing constraints) (Chung, Huang, & Lee, 2006).

Before presenting the simulation model formulation and algorithm, the following example is considered to illustrate the inputs of the simulation model. As shown in Fig. 4, this example consists of $N = 3$ jobs for the production shift s, which can be either the main jobs planned to be processed in the shift or uncompleted operations of a tardy job from the previous production shift. For each job $j = 1, 2, 3$, $NO_1 = 3$, $NO_2 = 3$, and $NO_3 = 1$ number of operations are considered, respectively. An operation ID $i \in \{1, \ldots, 7\}$ is assigned to each operation and Cr_i shows that if an operation is critical or not. Noting that all critical operations for job j must be processed on the same machine. Due to illustrative

Table 2
Table of notations.

Indices and Sets	
$s =$	Shift index, $s \in \{1, \dots, S\}$.
$j =$	Jobs index, $j \in \{1, \dots, N\}$.
$i, i', i'' =$	Operations ids, $i, i', i'' \in \{1, \dots, NO_s\}$.
$w =$	Operations indices for job j, $w \in \{1, \dots, NO_j\}$.
$m =$	Machine ids, $m \in \{1, \dots, NM\}$.
$k =$	Queuing position index, $k \in \{1, \dots, NO_s\}$.
$r =$	Simulation replication index, $r \in \{1, \dots, R\}$.
Parameters	
S	Number of shifts.
N	Total Number of jobs.
N_s^J	Number of jobs mainly planned to be processed at shift s.
N_s^U	Number of uncompleted jobs at the end of shift s.
$N_s = N_s^J + N_{s-1}^U$	Number of jobs to be processed at shift s.
NO_s	Total number of operations at shift s..
NO_j	Total number of operations for job j.
NM	Number of machines.
Π_{im}	Probability distribution of processing time of operation id i on machine m.
$\Psi_{ii'}$	Probability distribution of sequence-dependent setup time between operation ids i and i'.
$\tilde{p}_{im}$	Stochastic processing time of operation i on machine m.
$\tilde{q}_{ii'}$	Stochastic sequence-dependent setup time between operations i and i'.
J_s	Set of jobs mainly planned to be processed at shift s.
U_s	Set of uncompleted jobs at the end of shift s (each uncompleted job consist of undone operations of J_s).
A_i	Set of alternative machines capable to process operation i.
$\tilde{p}_{im}$	Stochastic processing time of operation id i on machine m.
$\tilde{q}_{ii'}$	Stochastic sequence-dependent setup time between operations i and i'.
M_{im}	Alternative machine, 1 if machine m is capable to process operation i, 0, otherwise.
Cr_i	Critical operation, 1 if operation i is critical, 0, otherwise.
k^S	Sequence-dependent setup occurrence rate.
k^D	Machine dedication ratio.
k^F	Flexibility ratio.
k^{Sq}	Constant of sequencing ($0.8 \le k^{Sq} \le 1.2$).
D	Shift duration.
R	Number of simulation replications.
NR	Number of rows in the dataset used to train the machine learning model.
M	A large number.
Decision variables	
$X_s = (x_{iks})_{NO_s}$	The vector of feasible solution (Job Shop Queue) for shift s where x_{iks} equals to 1, if operation id i is assigned to the kth position of the queue for shift s, 0, otherwise.
Functions	
$Y_s = (c_{js})_N$	The vector of completion times for shift s, where c_{js} denotes completion time of job j at shift s, $j \in \{1, \dots, N\}$.

purposes, here stochastic values of $\tilde{p}_{im}$ and $\tilde{q}_{ii'}$ are generated for all possible situations of processing operation i on the capable machine m as well as all possible sequences between operations i and i'. However, in the main simulation algorithm, the stochastic value for each situation is generated only if it happens. Also, the Job Shop Queue in Fig. 4 is an example of decision alternatives (feasible solutions) for SFJSSP, which is used to calculate the system performance measures. It is worth mentioning that the algorithm guarantees the feasibility of the generated Job Shop Queue before passing it to the simulation model. For example, operation five (note in this paper operations are indexed sequentially across starting from the lowest job index to the highest job index) is the second operation of job two. Therefore, based on precedence constraints, it must follow the first operation of job two, where $x_{i=5,k=3,s} = 1$ denotes that.

Considering the presented structure for the system parameters and the Job Shop queue as input to the simulation model, Algorithm 1 describes the simulation model to calculate the vector of performance measure Y_s for alternative decision X in one simulation replication. More precisely, the vector Y_s represents the completion times for all jobs processed during the shift s calculated regarding a specific Job Shop Queue X_s. The main steps in the algorithm can be summarized as follows:

1. Creating and setting initial values for temporary vectors, which aim to store and update machines and jobs' statuses during simulation execution.
2. Starting simulation procedure by iterating through operations on the Job Shop Queue.
    - 2-1. Assigning the operation to the earliest available machine considering machine capability and machine dedication constraints.
    - 2-2. Generating stochastic sequence-dependent setup time for the operation based on the latest operation processed on the selected machine.
    - 2-3. Generating stochastic processing time for the operation.
    - 2-4. Calculating start and finish times of setups and processes for the operation on the selected machine based on stochastic times generated in 2–2 and 2–3.
    - 2-5. Updating temporary vectors generated in 1.
3. For all jobs processed in the shift, set their latest operation's completion time as the job completion time and return all jobs' completion times as the output vector.

In addition to the parameters used in developing the simulation model, we considered k^S, k^D, and k^F as three ratios that generally describe some aspects of the operations processed during a production shift. These parameters are further used in the statistical risk evaluation method (Section 4.4) as well as in the validation and sensitivity analysis experiments (Sections 7.2 and 7.3). For better understanding, k^S is the average occurrence rate of sequence-dependent setups between operations. While the details of the setup for each operation are defined and simulated based on the $OpVec_i$ described in Algorithm 1, this ratio

indicates the general occurrence of setups for all operations scheduled during a production shift. Similarly, we defined two parameters to represent the machine dedication constraint and the system's flexibility. For machine dedication, k^D describes the dedication ratio, which means $k^D \times NO_j$ operations of a job are critical in a production shift, on average. Similarly, k^F is the flexibility ratio and shows the average percentage of machines capable of processing an operation.

Another important step in developing a simulation model is evaluating the model parameters. Due to a lack of access to real-world MES data, the main sources for parameters related to simulating the photolithography workstation in this research are two papers published by Ghaedy-Heidary, Nejati, Ghasemi, and Torabi (2024) and Ghasemi, Azzouz, Laipple, Kabak, and Heavey (2020). These papers have analyzed this workstation through simulation modeling and, together, they cover a wide range of real-world assumptions considered in this research. It is worth noting that, in order to validate both the simulation model and its outputs, we applied the developed model to simulate several test problems as presented by Ghasemi et al. (2020). These test problems align with the semiconductor fab investigated in this paper and encompass all CAPPA constraints relevant to it. Furthermore, we tested simplified versions of the model on other experimental problems mentioned in works by Ghasemi et al. (2021), Sharma and Jain (2017), Shen, Han, and Fu (2017) and Zhang et al. (2022).

The statistical compatibility observed between the results obtained from our developed simulation model and the values mentioned in the aforementioned papers gave us confidence. This confidence led us to utilize the simulation model for generating synthetic production log data, including job completion times.

4. Machine learning-based simulation method

Computer simulation models are among the most powerful tools for providing advice and making optimal control decisions in complex and uncertain environments. However, the advantages of this potentiality can only be realized when it is accompanied by high computational performance to provide real-time decision-making advice. As an example, a majority of research proposing Simulation-based Optimization methods to solve production scheduling problems deals with the computational challenges by reducing the number of simulation replications (Yang, Lv, Xia, Sun, & Wang, 2014). This entails forgoing some stochastic scenario explorations in order to solve the problem in a reasonable amount of computational time, with sometimes no insight into the ignored scenarios. In this regard, considering the DT environment, an efficient estimation method with accurate and robust objective value estimations would be advantageous. In this section, this paper focused on introducing an ML-based method, denoted as MLBSM, to decrease computational demand in dealing with large-scale SFJSSP by predicting job completion times and delay risks for a certain production shift.

4.1. Feasible solution structure

Similar to many evolutionary optimization algorithms developed for scheduling problems (Ghaedy-Heidary et al., 2024; Ghasemi et al., 2021), the input of the simulation model in this work is defined in a chromosome structure. The selection of this structure enables the simulation and, ultimately, the developed MLBSM to be integrated and implemented seamlessly with several existing optimization methods. In the simulation algorithm, a chromosome for the production shift s is represented as a sequenced vector X_s (the Job Shop Queue presented in Section 3) consisting of NO_s queue positions filled with NO_s operation IDs. It is worth mentioning that the algorithm guarantees the feasibility of the generated solution queues before passing them to the simulation model as input. As mentioned before, the simulation algorithm takes the generated solution queue (X_s) and operation vectors set to calculate the system performance measure (completion times) based on the procedure shown in Algorithm 1.

Algorithm 1: Simulation algorithm for production shift s.

Inputs : $X_s = (x_{iks})_{NO_s}$,
$OpVec_i = \left(j, Cr_i, M_i, \Pi_i, \Psi_i\right) \forall i \in \{1, ..., NO_s\}$ where
$M_i = \{m | M_{im} = 1\}, \Pi_i = \{\Pi_{i'm} | i' = i\}, \Psi_i = \{\Psi_{i'i''} | i'' = i\}$, N, NO, NO_j, NM

Output: $Y_s = (c_{js})_N$ The vector of completion times for jobs

begin
- **for** $j = 1$ **to** N **do**
  - $Jstatus_j = (LatestOpCompletionTime = 0, CriticalMachineId = NaN)$
- **end**
- **for** $m = 1$ **to** NM **do**
  - $Mstatus_m = (AvailableAt = 0, LatestProcessedOpId = NaN)$
- **end**
- **for** $k = 1$ **to** NO **do**
  - $i = find\left(i | x_{iks} = 1\right)$
  - $\tilde{j} = OpVec_i(1)$
  - **if** $Cr_i == 0$ **then**
    - $\tilde{m} = find\left(m | \min\left(Mstatus_m(1) \quad \forall m \in M_i\right)\right)$
  - **else**
    - **if** $Jstatus_{\tilde{j}}(2) = NaN$ **then**
      - $\tilde{m} = find\left(\min\left(Mstatus_m(1) \quad \forall m \in M_i\right)\right)$
    - **else**
      - $\tilde{m} = Jstatus_{\tilde{j}}(2)$
    - **end**
  - **end**
  - **if** $Mstatus_{\tilde{m}} == NaN$ **then**
    - $\tilde{q}_{i'i} = 0$
  - **else**
    - $\tilde{q}_{i'i} = Rand\left(\Psi_{i'i} | i' = Mstatus(2)\right)$
  - **end**
  - $\tilde{p}_{im} = Rand\left(\Pi_{i\tilde{m}}\right)$
  - $SetupStart_i = Mstatus_1$
  - $SetupFinish_i = SetupStart_i + \tilde{q}_{i'i}$
  - $ProcessStart_i = \max\left(Jstatus_{\tilde{j}}(1), SetupFinish_i\right)$
  - $ProcessFinish_i = ProcessStart_i + \tilde{p}_{im}$
  - $Mstatus_{\tilde{m}}(1) = ProcessFinish_i$
  - $Mstatus_{\tilde{m}}(2) = i$
  - $Jstatus_{\tilde{j}}(1) = ProcessFinish_i$
  - $Jstatus_{\tilde{j}}(2) = \tilde{m}$
- **end**
- $Y_s = \left(Jstatus_{\tilde{j}}(1) \quad \forall j \in \{1, ..., N\}\right)$

end

4.2. Generating vectorized dataset using the simulation model (Module I)

As discussed before, the ML model in this work is responsible for estimating the jobs' completion times (Y_s) for each production shift s regarding each Job Shop Queue X_s. The main question in this part of the research is how to extract and store knowledge from system parameters (the simulation model inputs) to generate a rich training dataset for ML, making it capable of predicting completion times.

To answer this question, this paper developed a multi-output training dataset consisting of NR rows with N_s features and N_s target columns. The row nr in this dataset represents a unique Job Shop Queue denoted as c (feature vector) as well as the calculated completion times (target vector) regarding c. For more detail, the jth element of the target vector in the row nr represents the average completion time for job j calculated in R simulation replications regarding Job Shop Queue c. However, designing the feature vector is quite complex as it needs to be a vector describing system parameters and differentiate between Job Shop Queues in the most reliable way. Ghasemi et al. (2021) address a Sequencing Priority-Based Method (SPBM) to extract

Operations Environment for production shift s (Inputs)

Job ID (j)	Operation ID (i)	IsCritical (Cr_i)	Stochastic processing times ($\tilde{p}_{im}$)			Stochastic sequence-dependent setup times ($\tilde{q}_{ii'}$)						
			Machine1	*Machine2*	*Machine3*	*Op1*	*Op2*	*Op3*	*Op4*	*Op5*	*Op6*	*Op7*
Job1	*Op1*	0	3	2	-	-	-	-	0.2	0	0.2	0
	Op2	0	4	6	5	0.1	-	-	0.1	0	0	0
	Op3	0	2	2	-	0	0.3	-	0.1	0.2	0	0.1
Job2	*Op4*	1	-	3	2	0.2	0	0.1	-	-	-	0
	Op5	0	4	4	-	0	0	0.3	0.1	-	-	0.2
	Op6	1	3	2	3	0.1	0	0.1	0.3	0	-	0
Job3	*Op7*	0	2	-	2	0.2	0	0.1	0	0.2	0.1	-

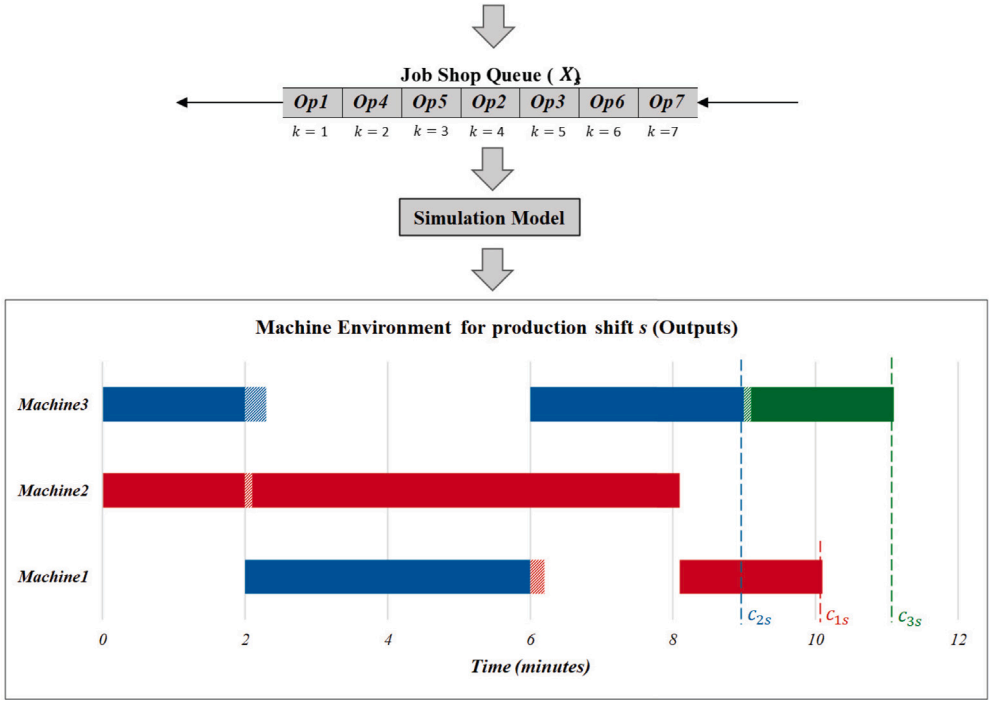


Fig. 4. Proposed SFJSSP example.

a set of trainable features from the Job Shop Queue for the SJSSP. As an important contribution, this study has tried to modify the SPBM method for FSJSSP and extract a vector of job penalty scores based on a Job Shop Queue. The modified SPBM method for FSJSSP is designed as follows.

Developing SPBM, initially, we defined the vector $solS_k(sol_c(k), A_{sol_c(k)})$ for each queue position, where $sol_c(k)$ describes the operation which is assigned to the kth position of Job Shop Queue X_s^c. Also, $A_{sol_c(k)}$ is a set of machine IDs that are capable of processing $sol_c(k)$. Within SPBM, in the first step, we need to define $MP^c_{m,sol_c(k)}$, which is the machine penalty caused by machine m on $sol_c(k)$ as follows:

$$MP^c_{m,sol_c(k)} = \max\left\{0, P^c_{sol_c(k')} | m \in A_{sol_c(k')}, 1 \le k' \le k-1\right\} \quad (4)$$

where $P_{sol_c(k')}$ denotes the penalty of earlier operations in the queue that will be explained further in this section . Due to the flexibility of the system, the sum of MPs for all machines capable of processing $sol_c(k)$ is considered as one of the penalty sources for the operation $sol_c(k)$. This summation is denoted as SMP and can be formulated as follows:

$$SMP^c_{sol_c(k)} = \sum_{m \in A_{sol_c(k)}} MP^c_{m,sol_c(k)} \quad (5)$$

By having $SMP_{sol_c(k)}$ formula, each operation in the Job Shop Queue will have a sequence penalty as follows. If $sol_c(k)$ is the first operation of any job j and $SMP_{sol_c(k)} = 0$:

$$P^c_{sol_c(k)} = 0 \quad (6)$$

If $sol_c(k)$ is not the first operation of any job j and $SMP_{sol_c(k)} = 0$:

$$P^c_{sol_c(k)} = P^c_{sol_c(k)-1} + 1 \quad (7)$$

where $P^c_{sol_c(k)-1}$ defines the penalty score for the operation $sol_c(k) - 1$ of job j (please note that $sol_c(k) - 1$ indicates the previous operation of the job j in the queue and not the previous operation in the queue $sol_c(k-1)$). If $sol_c(k)$ is the first operation of any job j and $SMP_{sol_c(k)} > 0$:

$$P^c_{sol_c(k)} = SMP^c_{sol_c(k)} + 1 \quad (8)$$

Finally, if $sol_c(k)$ is not the first operation of any job j and $SMP_{sol_c(k)} > 0$:

$$P^c_{sol_c(k)} = P^c_{sol_c(k)-1} + SMP^c_{sol_c(k)} + 2 \quad (9)$$

As result, the $SPBM^c_j$ score is defined as penalty score for job j in solution queue c and calculated as follow:

$$SPBM^c_j = \log_{10}\left(\sum_{sol_c(k)} P^c_{sol_c(k)}\right) \quad ; \quad \forall sol_c(k) \in \{1, \dots, NO_j\} \quad (10)$$

where $\{1, \dots, NO_j\}$ represents the set of operations for job j. Having $SPBM^c_j$ calculated for all jobs in a shift s, the training vector $Learning_c$, which is further used to train the ML method, is defined as follows:

$$Training^c_s = \left\{\{\cup_{j=1}^{N} SPBM^c_j\} \cup Y^c_s\right\} \quad (11)$$

where Y^c denotes the vector of completion times for all jobs, which is the average of completion times calculated in R simulation replications

	Calculated SPBM Scores for Each Job						Simulated Mean Completion Times for Each Job (Y_s)				
	$SPBM_1$	$SPBM_2$	$SPBM_3$	$SPBM_4$	$SPBM_5$		C_1	C_2	C_3	C_4	C_5
$Training_s^1$:	48.4	49.5	43.6	36.3	41.0		988.1	1165.9	822.5	757.8	803.8
$Training_s^2$:	44.9	43.7	45.5	33.9	39.7		1042.5	919.0	933.9	827.6	881.8
$Training_s^3$:	35.4	32.1	50.0	46.4	48.7		717.6	630.6	1063.3	1065.4	885.4
$Training_s^4$:	50.2	38.2	49.7	44.5	47.5	∪	1379.0	764.2	1057.0	829.2	1060.2
$Training_s^5$:	47.8	39.7	42.9	51.4	46.0		1062.3	722.4	816.5	1289.9	942.1
...	...	...	...	...	...		...	...	...	...	...
$Training_s^{NR}$:	43.0	36.8	46.0	43.4	40.3		1057.8	752.7	907.6	1079.0	819.5

Fig. 5. A sample structure of final vectorized dataset.

for Job Shop Queue c. The final dataset is generated for NR random feasible solutions and can be described as follows:

$$Training = \left\{\cup_{c=1}^{NR} Training_{s=-1}^{c}\right\} \tag{12}$$

Fig. 5 presents a sample structure of the final vectorized dataset ($Training$) for a production shift with five scheduled jobs. It worth mentioning the $Training$ dataset can be calculated on the data derived from any production shift s; however, as it is mentioned earlier, in this paper, the synthetic data obtained from the simulation is used for this purpose, which is not a real production shift. The value $s = -1$ in Eq. (12) represents this fact. The overall framework for generating the $Training$ dataset based on a sample problem of shift s is presented in Fig. 6.

To illustrate the training vector calculations, here we extend the example provided in Fig. 4 to ensure calculation validity, P scores for all operations are calculated based on the queue sequence (start with operation one, next four and etc.). Operation one is the first operation in the queue, and no earlier operations penalties ($P^1_{sol_c(k)}$) are calculated yet, so SMP^1_1 equals 0. As operation one is the first operation of job one, based on Eq. (6), P^1_1 equals 0. Operation four is the first operation of job two and can be processed on machines two and three. Using Eq. (4), both $MP^1_{2,4}$ and $MP^1_{3,4}$ equal 0 which results $SMP^1_4 = 0$ according to Eq. (5). Therefore, P^1_1 equals 0, based on Eq. (6). Operation five can be processed on machines one and two. Based on Eqs. (4) and (5), the SMP score for operation five equals to 0 ($SMP^1_5 = MP^1_{1,5} + MP^1_{2,5} = 0$). Therefore, mentioning that operation five is not the first operation of any job, $P^1_5 = P^1_4 + 1 = 1$, based on Eq. (7). Operation two can be processed on all three machines and using (4), $MP^1_{1,2}$, $MP^1_{2,2} = 0$, and $MP^1_{3,2}$ equal 1, 1, and 0 respectively which result in $SMP^1_2 = MP^1_{1,2} + MP^1_{2,2} + MP^1_{3,2} = 2$, based on Eq. (5). As a result, note that operation two is not the first operation of any job, Eq. (9) is used to calculate $P^1_2 = P^1_1 + SMP^1_2 + 2 = 4$. Likewise, the P score for operations three and six is 14 and 35, respectively. Finally, operation seven can be processed on machines one and three, so $SMP^1_7 = MP^1_{1,7} + MP^1_{3,7} = 70$. Since operation seven is the first operation of job three, using Eq. (8), the P score equals 71 ($P^1_7 = SMP^1_7 + 1 = 71$). Further, aggregation of P scores for all operations of a job based on Eq. (10) results $SPBM^1_1 = 1.2553$, $SPBM^1_2 = 1.5441$, and $SPBM^1_3 = 1.8513$. Considering processing and setup time values for all operations as described in Fig. 4 and letting $R = 1$, the value of the objective function for the current example solution is $F^1 = (10.1, 9, 11.1)$. Finally, using Eq. (11), the training vector of solution $c = 1$ is defined as follows:

$$Training_s^{c=1} = \{1.2553, 1.5441, 1.8513, 10.1, 9, 11.1\} \tag{13}$$

4.3. Machine learning method (Module II)

As stated previously, the ML model aims to predict average completion times regarding a Job Shop Queue. To do this, this study trained Adapted Boosting Regressor (ABR) on $Training$ to predict Y_s for each shift s based on Job Shop Queue X_s. ABR was first developed in a paper by Freund and Schapire (1997). However, in this study, the ABR class is implemented from Scikit Learn 1.0 ensemble-based methods, originally proposed by Drucker (1997). ABR's main idea is to fit a sequence of weak learners (i.e., models that are only marginally better than random guessing, such as small decision trees) to repeatedly modified versions of the data. The predictions from each of them are then combined to produce the final prediction via a weighted majority vote (or sum). Each so-called boosting iteration modifies the data by applying weights $w_1, w_2, \ldots, w_N$ to each of the training samples. At first, those weights are all set to $w_i = 1/N$, which means that the first step simply trains a weak learner on the original data. Each subsequent iteration modifies the sample weights individually and reapplies the learning algorithm to the reweighted data. At a given step, the weights of those training examples that were incorrectly predicted by the boosted model induced in the previous step are increased, while the weights of those that were correctly predicted are decreased. As iterations progress, difficult-to-predict examples gain increasing influence. The ABR algorithm can be summarized in the following steps:

1. Assigning equal weights to all the observations ($w_i = 1/N$ where N is number of records).
2. Classifying random samples using stumps. The base learner model (stumps) used in this study is a decision tree with maximum depth 3.
3. Calculating total error which is the sum of weights of misclassified records.
4. Calculating performance of the stump.
5. Updating Weights based on the performance of the stump.
6. Updating weights in iteration by making the second stump in the forest based on the normalized weight and repeating steps 2 to 5 again by updating the weights for a particular number of iterations.
7. Final predicting by obtaining the sign of the weighted sum of the final predicted value.

It is worth mentioning that, in this research, the obtained results from ABR is compared to three well-known tree-based ML algorithms to ensure its accuracy and performance. These algorithms include Random Forest Regression (RFR), Extra Trees Regression (ETR), and Gradient Boosting Regression (GBR), which are briefly described in Appendix A.

4.4. Delay risks evaluation (Module III)

In this study, the job risk value for each job is defined as the probability that the completion time of a job exceeds the production shift duration ($P\{\tilde{c}_{js} > D\}$). In a photolithography workstation, the stochastic setup and processing times are independent random variables for all operations in a shift. Also, the completion time of each job can be considered as a random variable which is the sum of setup and processing times. Thus, based on the Central Limit Theorem (CLT), as the number of operations increase, the probability distribution of job completion times tend towards a normal distribution (Ross, 2014). As

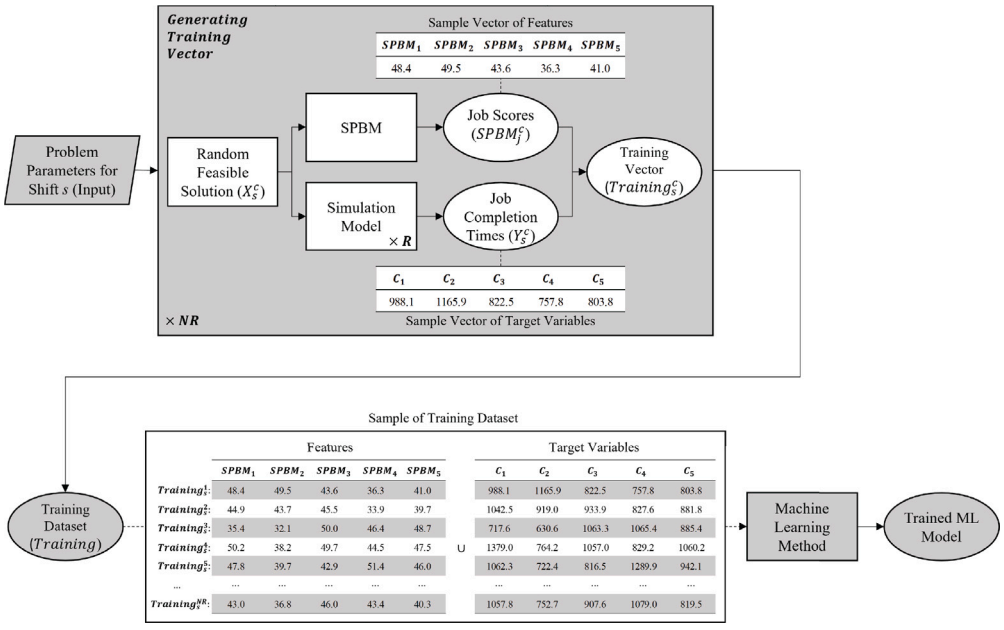


Fig. 6. Generating and vectorizing dataset for training the ML method.

a result, the risk value for job j processed at production shift s can be calculated as follows:

$$Risk_{js} = P\{\tilde{c}_{js} > D\}; \quad \tilde{c}_{js} \sim \mathcal{N}(\mu_{c_{js}}, \sigma_{c_{js}}) \qquad \forall s \in \{1, \dots, S\} \tag{14}$$

where D is the production shift duration and $\tilde{c}_{js}$ is the stochastic completion time of job j which is distributed normally with mean $\mu_{c_{js}}$ and standard deviation of $\sigma_{c_{js}}$. To calculate $Risk_{js}$ for each job, an estimation of these parameters is needed. The developed ML model is an accurate estimator for $\mu_{c_{js}}$ as its accuracy will be presented in Section 7.2. However, during several experiments, ML methods failed to estimate $\sigma_{c_{js}}$ accurately. It might be due to the fact that the main source of standard deviation in completion times is not laid in the sequencing (Job Shop Queue), which is used to design training datasets. As a result, in this section, an empirical statistical method is proposed for estimating $\sigma_{c_{js}}$. First, consider $\tilde{\sigma}_{js}^{op}$ as follow:

$$\tilde{\sigma}_{js}^{op} = \sqrt{(\bar{\sigma}_{js}^{P})^2 + k^S.(\bar{\sigma}_{js}^{S})^2} \tag{15}$$

which describes the estimated standard deviation of setup time and processing time for each operation in job j. In Eq. (15) $\bar{\sigma}_{js}^{P}$ and $\bar{\sigma}_{js}^{S}$ are defined as follows:

$\bar{\sigma}_{js}^{P}$: Average standard deviation of processing times for all operations in job j.

$\bar{\sigma}_{js}^{S}$: Average standard deviation of all possible sequence-dependent setup times defined for all operations in job j.

The above equation follows the additivity of variances of independent normal distributions. As Walpole, Myers, Myers, and Ye (1993) describes, the variance of the sum of two or more independent variables with normal distributions equals the sum of the variances of individual variables. In this case, for each operation, there is a variance stemming from its processing time and another from its setup time. Since not all operations require a setup, the expected value of a setup occurring (k^S) is considered as the number of times (in this case, less than 1) that a setup variance should be added to the processing time variance. By having $\tilde{\sigma}_j^{op}$ calculated, the estimated standard deviation of completion time for job j is defined as follows:

$$\tilde{\sigma}_{js} = \sqrt{k^{Sq}.k^F.NM.NO_j.(\tilde{\sigma}_{js}^{op})^2} \tag{16}$$

where k^F and k^{Sq} describe the flexibility ratio of the system (a system characteristic which is described in Section 7.2) and sequencing constant, respectively. To illustrate, Eq. (16) describes that the standard deviation of the completion time for each job j is mainly caused by the standard deviation in its operations and setup times ($NO_j.(\tilde{\sigma}_{js}^{op})^2$). In addition, in a fully flexible job shop, operations of job j can be processed on any machine before or after operations of job j', which causes an integration between operations of different jobs processed on the same machines. As a result, NM is multiplied to show the effect of this integration. However, multiplying NM models the integration effect for a fully flexible system, and as the flexibility ratio of the system decreases, the integration effect decreases, too. Thus, k^F is added to this equation, which is the system flexibility ratio and varies from 1 (for a fully flexible system) to 0 (non-flexible system). It adjusts the estimator for different levels of system flexibility. Finally, although the sequencing has a small effect on standard deviation, the sequencing constant ($0.8 \le k^{Sq} \le 1.2$) is added to adjust the estimator for unknown sequencing effects in case a dispatching rule is used.

5. Dynamic shifts scheduling

With the trained ML model and the risk evaluation method, we can estimate the mean completion time for each job and then estimate its delay risk score using the empirical statistical method within a certain production shift regarding the shift schedule (Job Shop Queue). Here, we extended the MLBSM to apply it on a series of production shifts and evaluate jobs' risk scores for all jobs planned to be processed. To become more clear, consider S number of consecutive shifts. J_s and U_{s-1} denote two sets of jobs planned to be processed at shift s. J_s defines a set of N_s^J number of jobs mainly planned to be processed at shift s. Also, U_{s-1} defines a set of N_{s-1}^U number of uncompleted jobs from shift $s-1$ where their remaining operations are considered as new jobs and planned to be processed at shift s. While J_s is known for all S production shifts at the beginning of the planning horizon ($s = 1$), the U_{s-1} for each shift is unknown until the shift $s-1$ ends. It is caused due to the uncertainty (stochastic setup and processing times) involved in the system. As a result, it is clear that different combinations of N_s^J and N_{s-1}^U are possible for each shift s. However, the developed ML model is only capable of estimating a fixed N-sized target vector based on a fixed N-size feature vector where N is the number of jobs scheduled to be processed in a production shift (Section 4.2). To deal with this, we trained a number of ML models for all possible combinations of $N_s^j + N_{s-1}^u$. As a result, we have $N_s^j \times N_{s-1}^u$ number of trained ML

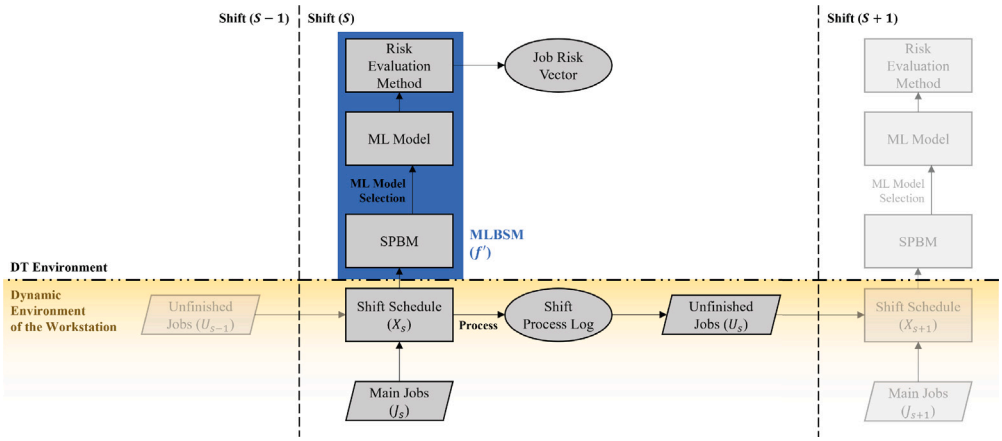


Fig. 7. Proposed framework of using the MLBSM as a DSS for scheduling in a dynamic environment.

models, supporting predictions for all possible dynamic scenarios for a production shift.

By having N_s^J, N_{s-1}^U, and Job Shop Queue X_s known at the beginning of shift s, the algorithm chooses the suitable ML model (which is trained on the $N_s^j + N_{s-1}^u$ dataset). Then, the chosen ML model is used to predict mean completion times for jobs in J_s and U_{s-1} regarding X_s. Finally, the predicted mean completion times are passed to the risk evaluation method, and delay risk scores are calculated for all jobs. Fig. 7 presents the application of the MLBSM in the described dynamic environment.

6. ML calibration

Setting parameters of ML models has a significant effect on their performance. Thus, in this section, we calibrate the parameter values for ML models using Taguchi's experimental design. This method is based on a special set of arrays called orthogonal arrays to conduct the minimum number of efficient experiments that could give insights into all factors that affect the performance measure.

In this study, we used the coefficient of determination (R^2 Score) to evaluate ML models, one of the most practical and known metrics in evaluating estimation methods. Considering $\hat{y}_i$ as the predicted value of the ith sample and y_i as the corresponding actual value, for total n samples, the general formulation of the R^2 Score is as follows:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{17}$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$. R^2 Score indicates the goodness of fit and, therefore, a measure of how well-unseen samples are likely to be predicted by the model. The best possible score is 1.0, and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y_i, disregarding the input features, would get an R^2 Score of 0.0.

To perform the Taguchi test, different levels of each affecting factor (parameters) must be selected. Based on related research in the literature, such as the work of Ghasemi et al. (2021), this study uses three parameter levels for testing. While adding more levels may result in more efficient values for calibrating ML models, the marginal improvements might not justify the exponentially increased computational time needed for training these complex ML algorithms and comparing the results. As this paper used four different ML methods for developing the MLBSM as well as experiments in this study, four independent Taguchi experiments with L9 orthogonal array are designed to calibrate critical parameters for each of these ML methods. Considering the time-consuming training phase of each of the four ML algorithms, the L9 orthogonal array can provides a good balance between the number of runs and the ability to estimate main effects. Table 3 describes three levels of parameter values for RFR, ETR, GBR, and ABR. In this table, NR, R, NEst, and LR denote the size of the training dataset, the number of simulation replications, the number of estimators and the learning rate, respectively, mentioning that the learning rate is only defined for GBR and ABR.

Table 3
Taguchi test parameters.

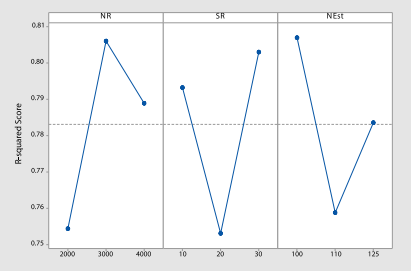
Method	Parameters	Level-1	Level-2	Level-3
RFR	NR	2000	3000	4000
	R	10	20	30
	NEst	100	110	125
ETR	NR	2000	3000	4000
	R	10	20	30
	NEst	100	110	125
GBR	NR	2000	3000	4000
	R	10	20	30
	NEst	100	110	125
	LR	0.08	0.09	0.10
ABR	NR	2000	3000	4000
	R	10	20	30
	NEst	100	110	125
	LR	0.08	0.09	0.10

It is important to note that parameter values at each level were chosen using basic statistical methods and tailored to the characteristics of each ML model. For instance, NR was set to its minimum value required to maintain the accuracy of the model and prevent overfitting (Freund & Schapire, 1997). R was determined using statistical sample size selection methods to ensure a sufficient number of simulation replicates for estimating the expected value of job completion time, which is a stochastic parameter. Additionally, NEst can significantly impact training times, and there is a trade-off between NEst and LR in terms of the accuracy of the ML model (Drucker, 1997); therefore, we carefully considered these aspects when selecting values for these two parameters.

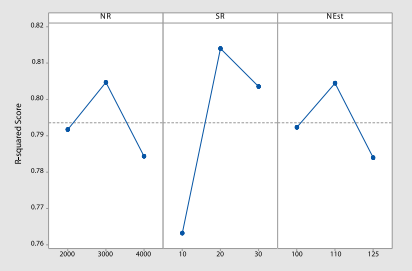
The Taguchi experiments were designed using Minitab 13. Then, 9 test runs were performed to train and evaluate each ML method (a total of 36 runs for all methods) using Scikit Learn 1.0 and Python 3.9. Table 4 describes selected parameter values for RFR, ETR, GBR, and ABR based on obtained Taguchi results presented in Fig. 8.

7. Experimental results and sensitivity analyses

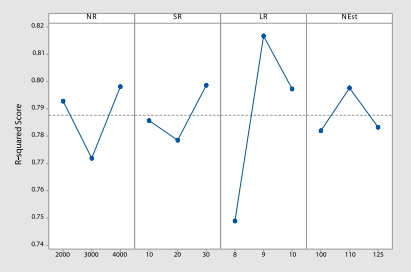
The main objective of this work is to present the MLBSM to predict job delay risks dynamically in a fast and accurate manner. Therefore, in this section, we have designed various experiments to evaluate the performance of the MLBSM. Then, the developed the MLBSM is applied to



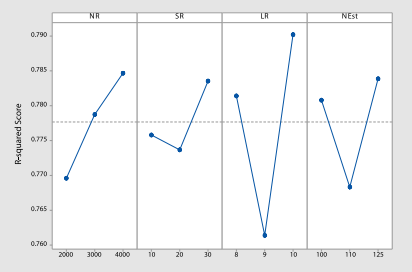
(a) Main effect plot for RFR.



(b) Main effect plot for ETR.



(c) Main effect plot for GBR.



(d) Main effect plot for ABR.

Fig. 8. Taguchi test results for RFR, ETR, GBR, and ABR.

Table 4
Selected parameter values for RFR, ETR, GBR, and ABR.

Method	Parameters			
	NR	R	NEst	LR
RFR	3000	30	100	–
ETR	3000	20	110	–
GBR	4000	30	110	0.09
ABR	4000	30	125	0.10

a dynamic environment to forecast the system behavior during several production shifts and analyze different sensitivity analysis scenarios.

7.1. Comparing ML methods

In this section, we compare ABR, which is used within the MLBSM, with three other ML methods. ABR is a powerful tree-based ML method used to forecast problems in different domains. Here, we compare ABR with RFR and ETR as two well-known and fast tree-based methods and also GBR as a powerful ensemble tree-based method.

To do this, we used the R^2 Score as a comparison metric presented in Section 6. Also, to get insights into the performance of each ML method in different dynamic scenarios, we generated nine different training datasets based on two sequential shifts considering the most probable dynamic scenarios. These scenarios vary in N_s^J and N_{s-1}^U as discussed in Section 5 and here we chose values 9, 10, 11 for N_s^J and 0, 1, 2 for N_{s-1}^U. All nine generated datasets had 4000 rows of data and were split to train (70%), and test (30%) sets used to train and evaluate R^2 Score for the selected ML methods (total 36 runs). Note that nine samples of 3000 rows were generated from these datasets based on the selected parameters for RFR and ETR in the Taguchi test results, and the mean completion times were calculated based on the selected values for R in Table 4.

Fig. 9 describes the average R^2 Score and total running times for training each ML model on different datasets (dynamic scenarios). Based on the obtained results, we conclude that the ABR method performs slightly better than other ML methods, and thus, it is employed in subsequent experiments in this work. Additionally, the ABR requires more training time; however, because obtaining ML's predictions for the mean completion times is significantly faster than simulation replications, we disregard this in this case. It is worth noting that other ML methods, e.g., Support Vector Regressor (SVR) and Multi-layer Perceptron (MLP), were also evaluated during our study. Still, they were not capable of being trained on the training vector (R^2 Score < 0.3).

7.2. Model validation

As the second part of the experiments, the developed the MLBSM was applied to a dynamic problem to validate the risk evaluation method. To do this, we designed S consecutive production shifts with a duration time D. Then N_s^J number of jobs were generated that are planned to be processed at shift s randomly. Each job has a random number of operations (NO_j). Also, each operation has a stochastic processing time distributed normally. The mean (μ_i^P) and the standard deviation (σ_i^P) of this distribution is known and evaluated based on distributions defined in Ghasemi et al. (2021). The setup times between operations are also normally distributed with known values of $\mu_{ii'}^S = 0.1 \times \mu_i^P$ and $\sigma_{ii'}^S = 0.1 \times \sigma_i^P$ for mean and standard deviation, respectively (Ghaedy-Heidary et al., 2024). We also used k^S, k^D, and k^F as described in Section 3.2 to consider some general aspects of the operations being schedules. Table 5 presents values for these parameters in more detail.

Before starting the validation procedure, we need to train a number of ML models supporting all possible dynamic scenarios in this numeric example. To be clear, they were trained on all combinations of N_s^J, and N_{s-1}^U, where N_s^J takes values 9, 10, and 11 and N_{s-1}^U can be 0 to 11 as it is almost impossible for the described system to face more than 11 tardy jobs at the end of a shift. It is worth mentioning, considering the procedure presented in Section 4.1, the training datasets for each scenario were generated on the second shift of a set of two consecutive shifts. These shifts' parameters are generated randomly based on the same parameters described in Table 5.

Fig. 10 presents the validation procedure. At the beginning of the procedure, we have ML models trained for different dynamic scenarios.

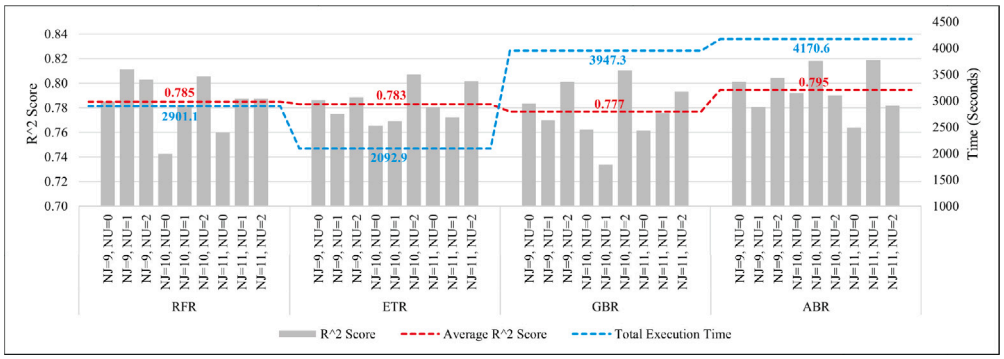


Fig. 9. Comparing ML models on different dynamic scenarios.

All J_s (set of main jobs for shift s) known. Also, we consider U_0 as an empty set which means that no tardy jobs come from shift $s = 0$ to $s = 1$; however, U_{s-1} is unknown for $s > 1$ at the beginning, and it will be calculated using the simulation model at the end of each shift $s - 1$. As a result, at the beginning of the shift s, J_s, and U_{s-1} are known, and a random Job Shop Queue is generated for the shift based on them. Having N_s^J and N_{s-1}^U, the suitable ML model is selected. The shift's Job Shop Queue is converted to a vectorized form using SBPM, and then mean completion times are predicted using the selected ML model. The predicted values for each job are then passed to the risk evaluation method. On the other side, we simulate the system based on the Job Shop Queue to find completed and delayed jobs. The obtained results from the simulation model are compared with evaluated jobs' risk values to validate the model. To ensure the performance of the developed model regarding the uncertainties and dynamic environment, first, we run 30 shifts as the warm-up phase, then the validation metrics are calculated for the rest of the shifts, excluding the warm-up phase.

Fig. 11 describes the system behavior and the completion time estimation model performance in the dynamic environment. The graph shows the number of processed jobs at each shift and their mean simulated completion times. The mean estimated completion times show that the estimated completion times using ABR follow the mean simulated completion time values during both the warm-up and validation phases. Also, the Root Mean Square Errors (RMSEs) of estimated completion times for jobs at each shift do not change meaningfully as shifts are passed. The average RMSE of estimating completion time for all jobs in the validation phase is 48.97 min. As a result, it seems that the ABR is performing well in training and estimating completion times for each job in the 480 min shifts.

To validate the risk evaluation method, we defined the shift tardiness ratio (number of delayed jobs at shift/total number of jobs processed in shift). The shift tardiness ratio was compared to the mean estimated job risk for each production shift. Fig. 12 shows the mean estimated job risks and tardiness ratio for the warm-up and validation phases and also indicates that there is a direct correlation between them. To conduct a more quantitative analysis of the validity of the MLBSM, we split jobs into low-risk (risk score $\leq$ 25%) and high-risk (risk score $\geq$ 25%) jobs based on their estimated risk score. Table 6 summarizes the overall confusion matrix for these job groups and their simulated state (completed or tardy) during the validation phase's 70 shifts. The matrix indicates that from 739 jobs processed in the validation phase, 54 of them did not finish during the shift they had started (tardy jobs). To be make it more clear, we took the advantage of recall rate to ensure the sensitivity of the MLBSM in detecting high-risk jobs. The recall rate can generally be defined as follows:

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{18}$$

Considering the confusion matrix presented in Table 6, as the MLBSM aims to predict tardy jobs as high-risk, tardy jobs predicted as High-risk and Low-risk are True Positives and False Negatives, respectively.

Table 5
Model validation parameters and values.

System parameters	
Learning model	ABR
D	480 min
S	100
Warm-up phase	$1 \leq s \leq 30$
Validation phase	$31 \leq s \leq 100$
N_s^J	$9 \leq N_s^J \leq 11$
N_{s-1}^U	$0 \leq N_{s-1}^U \leq 11$
Job parameters	
N_s^J	$\mathcal{U}(9, 11)$
NO_j	$\mathcal{U}(9, 11)$
Π_{im}	$\mathcal{N}(\mu_i^P = 62.49, \sigma_i^P = 11.20^2)$
$\Psi_{ii'}$	$\mathcal{N}(\mu_{ii'}^S = 6.25, \sigma_{ii'}^S = 1.12^2)$
k^S	0.1
k^F	0.5
k^D	0.3

Table 6
Confusion matrix for validation phase.

		Simulated state of the wokrstation		Total
		Completed job	Tardy job	
MLBSM's prediction	Low-risk	657	6	663
	High-risk	28	48	76
	Total	685	54	739

Consequently, the recall rate of the MLBSM for identifying tardy jobs as high-risk is:

$$Recall = \frac{48}{48 + 6} = 88.89 \tag{19}$$

In fact, the MLBSM estimated a risk score of greater than 25% for 48 out of 54 tardy jobs, resulting in an 88.89 percent recall rate for identifying tardy jobs as high-risk. While the model's precision (63.16%) is not particularly high, estimating risks in a way that detects all tardy jobs is much more important. It will help the shift scheduling procedure in rescheduling the shift plan to ensure critical jobs are completed on time.

7.3. Sensitivity analysis

The goal of this section is to investigate the performance of the MLBSM and also study the system behavior in the case of changing three important parameters of the photolithography workstation. We performed one-way sensitivity analysis on k^S, k^F, and k^D at three levels (50%, 100%, and 150% of the basic values) for each parameter value described in Table 7 (total 27 combinations). The experiment procedure and parameters are the same as the experiment presented in Section 7.2 and Table 5. Here, the mean tardiness ratio is used to show how the system will react to changing each parameter. The mean tardiness ratio is defined as the average tardiness ratio for all production shifts where

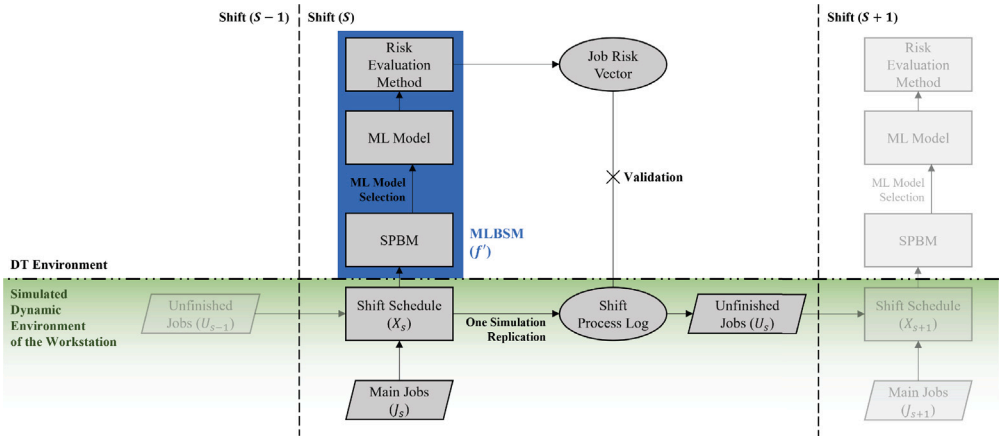


Fig. 10. Validating the MLBSM results in simulated dynamic environment of the workstation.

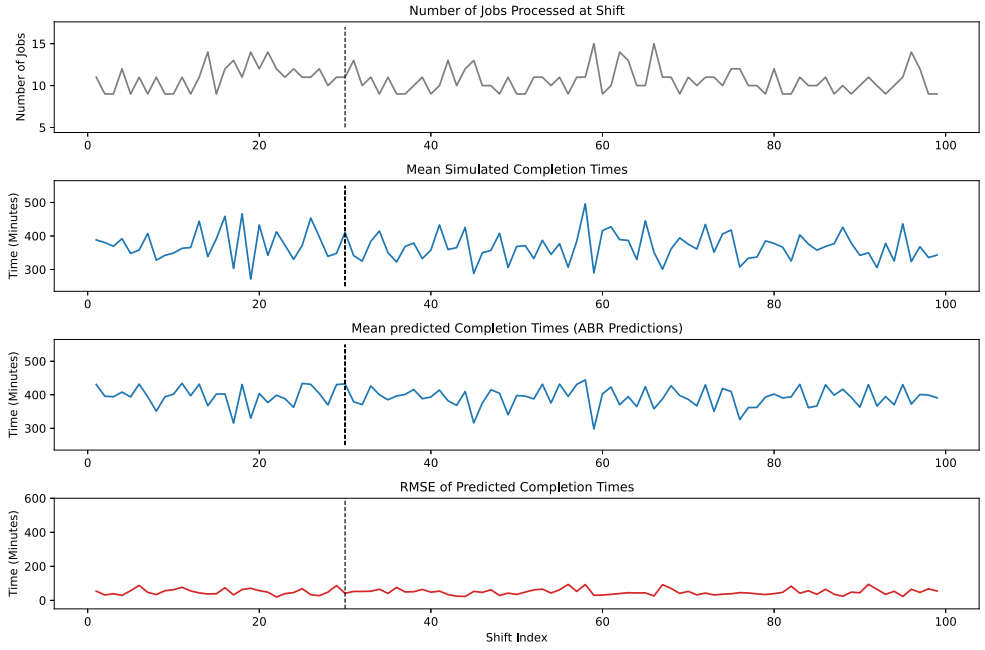


Fig. 11. System and the ABR performances during the warm-up and validation phases.

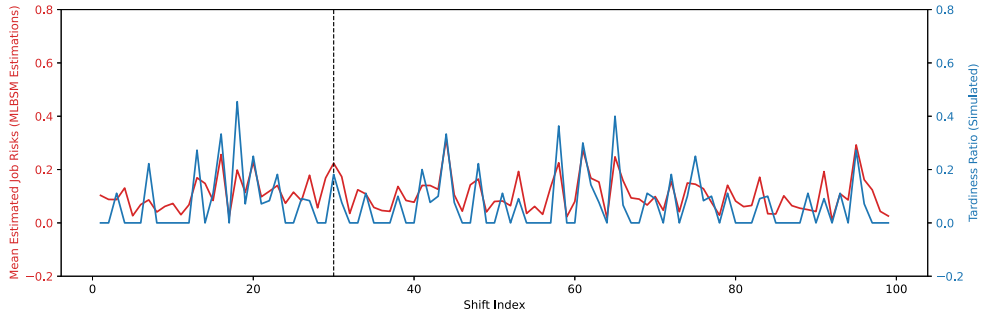


Fig. 12. Comparing the estimated mean job risks and tardiness rates during the warm-up and validation phases.

the warm-up shifts are excluded. Also, the mean estimated job risks are average for all estimated risks for jobs processed after the warm-up phase, which is used to evaluate the model performance within different parameter levels.

Fig. 13 describes increased flexibility and dedication constraints (decreasing k^F and increasing k^D) will lead to more tardy jobs. It also shows that the flexibility ratio has a more significant effect on the system, which is three times larger than the dedication ratio effect. However, increasing the setup ratio slightly affects the mean tardiness

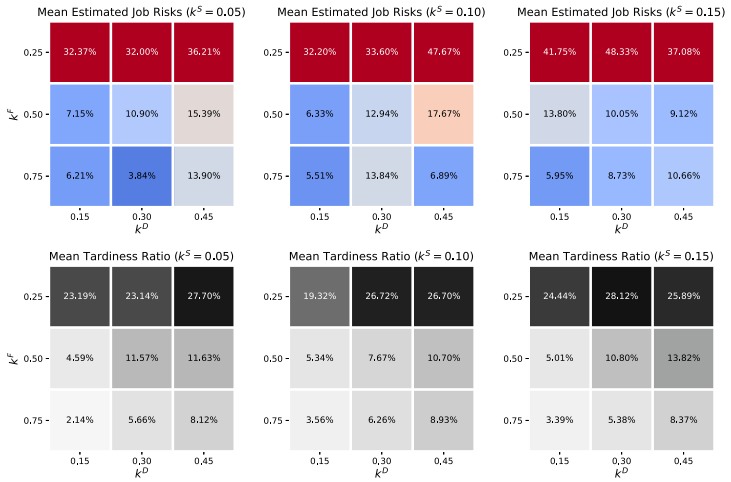


Fig. 13. Comparing mean job risks estimated by the MLBSM and mean tardiness ratios (simulated) for different k^S, k^F, and k^D levels.

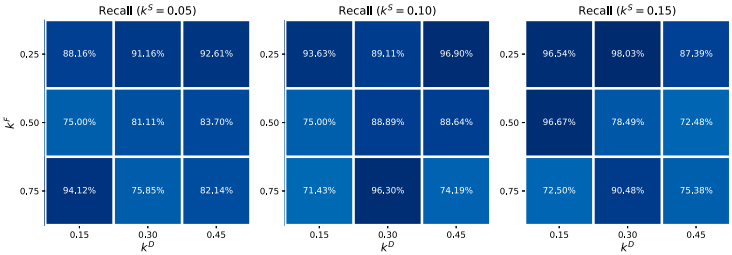


Fig. 14. Comparing recall score for different k^S, k^F, and k^D levels.

Table 7
Sensitivity analysis.

Parameter	Level-1	Level-2	Level-3
k^S	0.05	0.10	0.15
k^F	0.25	0.50	0.75
k^D	0.15	0.30	0.45

ratio (only a 1% increase in mean tardiness ratio when the setup ratio was increased from 0.05 to 0.15). Furthermore, the mean estimated job risks graphs indicate the same growth in job risks while flexibility and dedication constraints increase and a small effect from the setup ratio. The correlation between the mean tardiness ratio and the mean estimated job risks guarantees that the model can make valid decision supports for a wide range of jobs that vary in their parameters. It is worth mentioning that the mean estimated job risks are mostly greater than the simulated tardiness ratio of the system, which is caused as a result of the higher recall that the model has compared to its precision.

Moreover, Fig. 14 illustrates the model's recall in detecting high-risk jobs (risk score $\geq 25\%$) for different levels of k^S, k^F, and k^D. It is clear that the MLBSM always detects high-risk jobs with more than a 70 percent recall score. However, it seems that it performs slightly better with the main job parameters ($k^S = 0.10$, $k^F = 0.50$, and $k^D = 0.30$).

7.4. Time complexity analysis and trade-offs

One of the ultimate goals of the MLBSM is to accelerate DT-based decision-making by supplanting simulation models with ML models and, consequently, avoiding numerous time-consuming simulation replications. In this section, aiming to evaluate the performance of the MLBSM in terms of time efficiency, we have designed two experiments, one of which analyzes the MLBSM training and prediction times at different numbers of simulation replications, and the other examines these parameters when the MLBSM is applied to make predictions for large-scale problems.

In the first experiment, the environment and all parameters remain consistent with those outlined in Section 7.2 and Table 5. However, in this particular experiment, the developed simulation model also works parallel with the trained ML model to make estimations about the system's future status based on shifts' production schedules. Subsequently, a comparison is made between ML and simulation models at different levels of simulation replication numbers (R). This comparison is based on their computation times and the accuracy of the results they produce over 100 consecutive production shifts. It should be mentioned that the MLBSM recall score in this experiment is calculated using the same approach as described in Section 7.2, where ABR predictions of mean completion times are passed to the statistical risk assessment method. The MLBSM results are then compared with a single simulation replication for each production shift. For the recall score of the simulation, everything remains the same except that in this experiment, ABR is replaced with 30 simulation replications to calculate mean completion times and these values are passed to the statistical risk evaluation method. Subsequently, we calculated the recall score of the simulation with the obtained results compared with single simulation replication for each shift.

Fig. 15 presents the results of the first experiment. The foremost observation indicates the increasing trend of the simulation model execution time for large replication numbers. In contrast, the training or prediction times of the MLBSM remain unaffected by the number of simulation replications through which training datasets are generated as these results are independent of replication number. However, as the recall scores show, simulation replications are necessary to obtain reliable decisions based on the simulation model or to train an accurate ML-based tool using synthetic data. Considering this, for replication numbers above 25, where the recall score stabilizes, the MLBSM can perform approximately 20 times faster than a simulation model in

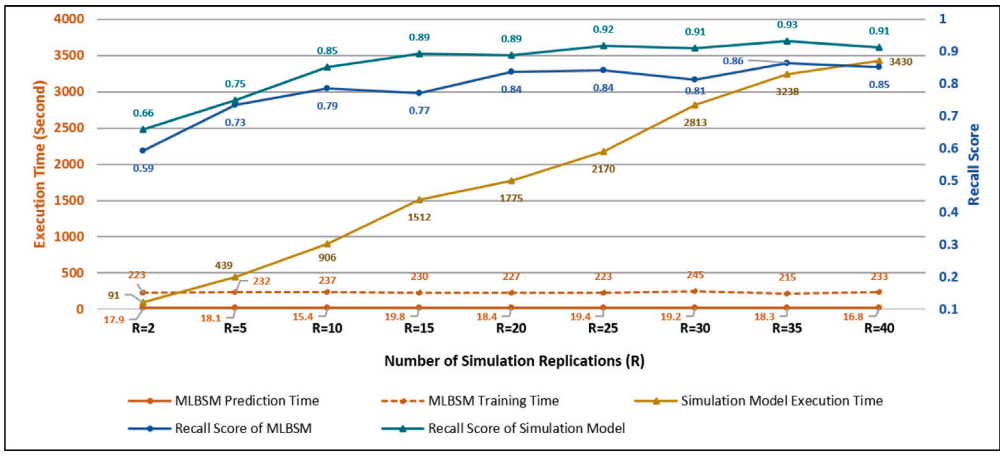


Fig. 15. Comparing time and recall score for the MLBSM and simulation model at different numbers of simulation replications.

Table 8
Probability distribution of the number of machines, jobs, and operations per job for different levels of problem size.

Parameter	Level-1	Level-2	Level-3	Level-4	Level-5	Level-6	Level-7
NM	5	10	15	20	25	30	35
N_s^J	$\mathcal{U}(4,6)$	$\mathcal{U}(9,11)$	$\mathcal{U}(14,16)$	$\mathcal{U}(19,21)$	$\mathcal{U}(24,26)$	$\mathcal{U}(29,31)$	$\mathcal{U}(34,36)$
NO_j	$\mathcal{U}(4,6)$	$\mathcal{U}(9,11)$	$\mathcal{U}(14,16)$	$\mathcal{U}(19,21)$	$\mathcal{U}(24,26)$	$\mathcal{U}(29,31)$	$\mathcal{U}(34,36)$

predicting the system's future. Even if an unprecedented event occurs in the factory, necessitating retraining of ML models, the MLBSM would still yield predictions over 10 times faster than simulation replications. It should be noted, however, that achieving these time-efficient predictions may require sacrificing 6 to 10 percent of the recall score.

The second experiment centers around the problem's size. When making production scheduling decisions, the size of the problem, denoting the number of machines, jobs, and operations per job involved in the system, has a profound impact on the computational demands needed to simulate the system. Furthermore, production scheduling problems within a flexible job shop environment belong to a category of optimization problems classified as NP-hard, which exhibit a solution space expanding exponentially as the problem's size grows. This fact underscores the importance of developing a time-efficient tool such as the MLBSM, which assists DTs in evaluating a significant portion of feasible solutions in a limited time. To examine the time and accuracy of the MLBSM compared to the simulation model, seven different levels of complexity are considered, which are described in Table 8. Other parameters and procedures of this experiment are the same as those of the first experiment, except for the simulation replication number, which is 30 and the same for all levels.

As Fig. 16 shows, when the size of the problem increases, the times for simulation replications and training the MLBSM increase exponentially, while there is no significant variation in the MLBSM prediction times. These increasing trends can be attributed to the larger number of entities and events that need to be simulated by the simulation model, as well as the greater number of items in the feature and target vectors for the ML model. However, the increase in training times for the MLBSM is relatively small compared to the simulation replication times. This fact makes the MLBSM suitable for evaluating feasible solutions for large-scale problems, even under unprecedented circumstances in the system that require training new ML models. Additionally, the MLBSM predictions are about 70 times faster than evaluating feasible solutions using simulation replications for large-scale problems. Furthermore, the obtained results from recall scores do not show a significant difference in the model's performance between the large-scale and small-sized problems, and approximately, there is an average gap of 6.42% between the MLBSM and simulation recall scores.

8. Discussion

As stated before, this paper aimed to develop an ML-based simulation metamodel to facilitate real-time DT-based decision-making for the production scheduling problem in complex manufacturing environments such as the photolithography workstation. The conducted experiments suggest that developing the MLBSM as a metamodel is a huge step forward to achieving this goal. According to the obtained results and regarding the research questions presented in Section 1, the contributions of this paper are summarized as follows:

- **(Question 1)** The first and most fundamental challenge in the way of developing ML-based tools in manufacturing systems is the lack of viable data. Although in smart manufacturing environments, the presence of sensors monitoring every second of the process can partly play a role in solving the data availability problem, this data-gathering process cannot provide sufficient data for analyzing many complex manufacturing problems. In fact, sensor-gathered data suffer from the lack of insight into abnormal operating conditions because the factory is often planned to operate at its optimum or near-optimum condition. Even the collected data is not enough for analyzing the normal conditions when it is compared with the huge solution space of a complex production scheduling problem (e.g. SFJSSP presented in Section 3).
  To deal with the data availability problem, in this work, we designed a detailed simulation model by considering all known aspects of the photolithography workstation, such as three special operational constraints (machine process capability, machine dedication, and the maximum number of times each reticle can be shared among different machine) known as CAPPA constraints, as well as, stochastic processing and sequence-dependent set-up times. The developed simulation model then simulated the factory in order to generate data similar to those collected by sensors and stored in MES. However, the simulation was not limited to the real operating condition of the workstation, and it could generate data based on millions of randomly generated operation conditions (either normal or abnormal). Having a massive amount of synthetic production log data solved both quality and quantity problems, paving the way for training advanced ML models and eventually developing the MLBSM. It should be mentioned that the similarity of simulated data with real-world MES data, in terms of data structure, allows seamless integration of the MLBSM with other MES and DT modules. This also enables the further use of real-world data collected and stored in MES for improving ML training and adjusting its predictions to changes in the process.

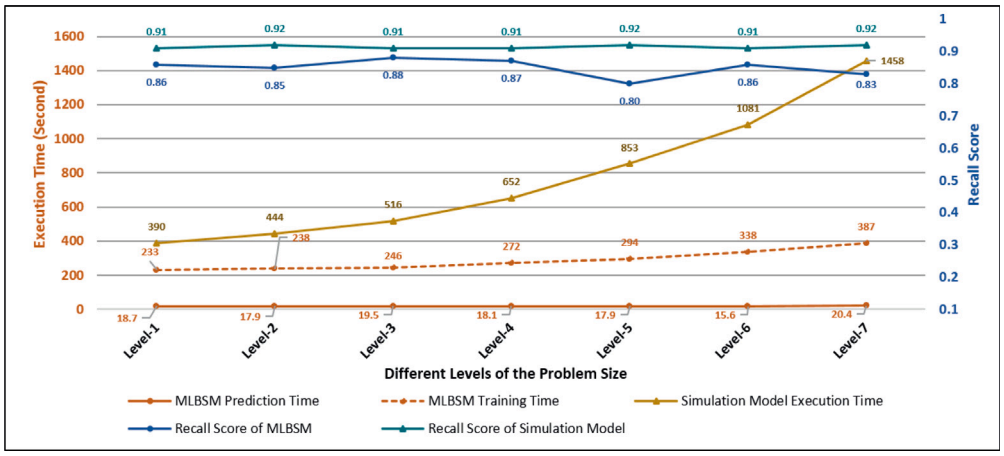


Fig. 16. Comparing time and recall score for the MLBSM and simulation model at different levels of problem size.

- **(Question 2)** The second problem was preprocessing and retrieving valuable information from raw production log data to address the scheduling problem. The retrieved information, in fact, must have formed a training dataset capable of training the ML method to predict expected values of completion times (performance measures) for each job. To deal with this problem, SPBM as a vectorizing method, was developed. SPBM transforms a Job Shop Queue (the structure of the shift schedule for each production shift) into a numerical vector to form the feature vector. Then, by joining the expected completion times, which were calculated by the simulation, with the feature vector, the training vector is generated. In fact, each training vector contains information about a shift schedule and the expected completion times calculated regarding that shift schedule. The developed SPBM and the huge amount of synthetic data available enabled us to design a number of training datasets to train ABR as an ML method on them. It is worth mentioning that we had a number of datasets, not only one, to address dynamic scenarios for each production shift. Each dataset was built on a set of filtered data to represent a combination of the number of the main jobs at that shift and the tardy jobs from previous shifts, ready to be processed at a shift.

  The obtained results from experiments conducted in Section 7.1 are quite promising. First of all, after training ABR, it was used to predict expected completion times regarding newly generated schedules (out-sample test) under different dynamic scenarios. When the ML-predicted values were compared with the simulated values (calculated based on 30 replications), the average R^2 Score of all scenarios was 0.795, and the value for each scenario did not fall below 0.760, which shows how reliable ABR works under different scenarios. Needless to say that the execution time for ABR is far smaller than 30 time-consuming simulation replications. Moreover, ABR was compared with three other tree-based algorithms; although the average R^2 Score of different scenarios was only slightly higher for ABR, this algorithm showed more stability over those scenarios, which convinced us to use this algorithm instead of others. All these results validate the performance of SPBM as a vectorizing method and also the accuracy of ABR as an ML method.
- **(Question 3)** The third challenge we faced was taking advantage of the developed ML-base tool for evaluating a large number of feasible shift schedules. While the predicted expected completion times seem to be a good performance measure for evaluating shift schedules and finding the optimum one, it suffers from ignoring the standard deviation, which is an important measure in such highly stochastic manufacturing environments. To cover this drawback, we defined a risk score as a performance measure that is calculated by integrating ABR and a new empirical statistical method. The statistical method enables the MLBSM to evaluate each shift schedule not only based on the predicted expected completion times but also by considering an estimation for the standard deviation of the completion times. Using those key modules, the MLBSM can provide DT with the delay risk scores for each job at the beginning of each production shift. Consequently, the DT can make a decision based on that prediction to choose either a shift schedule that minimizes the risk value for a critical job or one that minimizes the average delay for the whole system. It is worth mentioning that, in this paper, we proposed a framework in which the MLBSM was deployed to predict risk scores dynamically for a number of consecutive production shifts.

  In order to evaluate and validate the performance of the MLBSM as a whole package consisting of ABR and the risk evaluation modules, we designed an experiment, which is presented in Section 7.2. In this experiment, the MLBSM was used to predict delay risks for jobs at the beginning of a shift dynamically, and the obtained results were compared with the simulated status of each job after that production shift. The developed simulation model (one replication), in this experiment, was responsible for acting as the real manufacturing system during 100 consecutive shifts. The obtained results pointed out that during investigated shifts and among 739 processed jobs, 54 of them faced tardiness (were not completed during the shift). Accordingly, the MLBSM was capable of classifying 48 of those tardy jobs as high-risk jobs by predicting risk scores of higher than 25% for them, which resulted in an 88.89% recall ratio. As a result, considering the DT environment, the MLBSM seems to be quite a fast and reliable tool for comparing feasible shift schedules by taking uncertainties of a highly complex and dynamic production environment into account. It should also be noted that the MLBSM did not yield a high precision ratio (63.16%). Precision would be an important metric if decision-makers wanted to examine the exact completion status of each job at the end of each shift. However, the MLBSM primarily aims to detect high-risk jobs so that decision-makers can eliminate Job Shop Queues leading to high-risk states and choose less risky queues as an optimum decision. Therefore, the relatively low precision does not hinder the decision-making process for which the MLBSM is developed. Considering the high R^2 Scores of ABR predictions, the main source of this error is the statistical risk evaluation method. Thus, improving this method or replacing it with an ML model may significantly enhance precision for cases requiring more precise predictions.
- **(Question 4)** The last concern was about the performance of the MLBSM in terms of accuracy and time efficiency at different

operating conditions. In fact, some critical parameters of a manufacturing environment might change over time due to different technical setups (e.g., machine calibration) or types of orders being processed at the factory, and these changes might affect the performance of the MLBSM. In this regard, we aimed to ensure the quality of predictions by performing a three-stage sensitivity analysis on three critical parameters in the photolithography workstation: sequence-dependent setup occurrence rate, machine dedication ratio, and flexibility ratio. Additionally, a time complexity analysis was conducted to assess the computational advantages of the MLBSM compared to multiple simulation replications across various problem sizes. Furthermore, a time-accuracy trade-off analysis was provided in Section 7.4.

The obtained results in Section 7.3 illustrate that both ABR and risk evaluation modules in the MLBSM perform properly at different stages of each of these parameters. More precisely, not only was no remarkable trend detected in the performance of the MLBSM by increasing or decreasing a parameter value, but the recall ratio for the MLBSM always remained higher than 71.4%. In addition, by changing the value of critical parameters, the predicted risk values followed the same trend as the average number of tardy jobs at production shifts (simulated), approving that the MLBSM stays accurate in different conditions in the manufacturing environment. The time-complexity analysis presented in Section 7.4 also demonstrates that when the MLBSM is applied to evaluate feasible solutions for larger-sized problems, it can significantly reduce the time required compared to evaluation through simulation replications. To be more specific, when analyzing production schedules for 100 consecutive dynamically related production shifts in semiconductor manufacturing (representing real-world-sized problems), the MLBSM can identify high-risk jobs (the system performance metric) approximately 70 times faster than the simulation model. This time efficiency is achieved at the cost of a 6.42% reduction in the recall score. However, as long as the MLBSM offers quicker evaluations, it can assist the optimization process in exploring and assessing a larger portion of the problem's extensive solution space within a limited timeframe. This capability empowers the decision-making process by obtaining better feasible solutions in a shorter period, outweighing the relatively minor loss in prediction accuracy.

All these results made us confident in using the MLBSM as a fast, accurate, and reliable DT-based tool to provide a stochastic and complex manufacturing system with real-time decision support addressing production scheduling problems. Although the MLBSM is particularly tailored to make predictions in semiconductor fabs, it can be effortlessly adjusted to other flexible job shop environments by eliminating CAPPA constraints considered in designing the simulation model. Additionally, given the reliable performance of SPBM and the risk evaluation method in the complex and stochastic environment of the semiconductor fab, we are optimistic that these methods can produce equally or even more favorable results in other manufacturing settings with fewer complexities and uncertainties. For instance, SPBM, in its current configuration, can also be used to vectorize data in a job shop, or, by omitting machine penalty scores, it can be applied to parallel machine environments. Likewise, the risk evaluation method, owing to its reliance on general statistical concepts, can be directly applied to various manufacturing contexts such as the ones mentioned earlier. Nevertheless, we strongly recommend that researchers conduct thorough testing and validation of these the MLBSM applications in small-scale problems before implementing them in real-world factories. More generally, the MLBSM, as an ML-based metamodeling framework, may also prove beneficial in other fields and industries seeking real-time DT-based solutions for operational problems but deal with the lack of quality and quantity training data. Smart cities, healthcare systems, and retail delivery systems are some potential areas. However, it is essential to note that, in these contexts, all modules should be developed and rigorously tested from the ground up.

9. Conclusion

Despite several advantages, the computational time of simulation models obstructs the design and implementation of real-time DTs. However, the availability of big data in Manufacturing Execution Systems (MES) enables training different Machine Learning (ML) models for fast and accurate predictions and decision assessments. Accordingly, this paper have proposed an ML-Based Simulation Metamodeling Method (MLBSM), aiming to facilitate DT-based decision making for dynamic production scheduling in smart and highly complex manufacturing environments. The MLBSM consists of three seamlessly integrated modules: a novel vectorizing method (SPBM), trained multi-output ABR models, and a new statistical risk evaluation method. SPBM effectively converts unstructured production log data into numerical vectors, on which ABR algorithms are trained. Considering the dynamic nature of the workstation, each trained ABR predicts mean job completion times for different possible dynamic scenarios based on a shift schedule (Job Shop Queue). By having mean completion times, the risk evaluation method estimates the standard deviation of job completion times, and ultimately, a delay probability score for each job is calculated. These instant predictions can assist DT in promptly evaluating each production schedule. In this research, a simulation model is developed to generate synthetic MES data. The simulation model considers the special characteristics of a photolithography workstation at a semiconductor manufacturing front-end fab. Conducted experiments have demonstrated the performance and accuracy of each module and the MLBSM as a whole. Comparing the MLBSM results with the original simulation outputs shows that the MLBSM can be an accurate and efficient tool for developing real-time DTs in highly complex manufacturing systems, as it can predict high-risk jobs with a recall score of 88.89% and at least 70 times faster than simulation replications. Sensitivity analyses confirm that the MLBSM remains consistent under different operating conditions of the workstation.

This paper also recommends directions for future studies. As stated before, the lack of sufficient MES data necessitated the generation of synthetic MES data through a simulation model developed based on real-world assumptions and parameters. Although this approach has been used in several cases and has proven its potential to address data availability issues, it was still a limitation for this research. Therefore, future researchers are suggested to take the MLBSM a step closer to real-world data by training ML models on actual MES data or using such data for tuning simulation parameters and validating its results. Additionally, the pre-processing approaches, particularly the vectorizing phase, still need more investigation to achieve a higher recall ratio and, more importantly, to address the low precision ratio of the MLBSM. In this regard, one might focus on whether SPBM is the best method for extracting trainable vectors from the Job Shop Queue or not. Utilizing a more efficient and accurate vectorizing method can simplify the use of ML algorithms, resulting in more time-efficient training and prediction processes. Secondly, we considered a simple empirical statistical method for estimating the standard deviation. Although this method performs quite well, improving the statistical method or a metamodeling approach for estimating this parameter will lead to higher precision in the risk evaluation phase. Thirdly, replacing ABR with more advanced prediction methods like Deep Learning or employing Agent-based approaches to develop more detailed simulation methods can be advantageous for systems that prioritize higher modeling accuracy and have the flexibility to accommodate longer computational times. Additionally, there are many uncertainty sources and dynamic events present in manufacturing systems. In this paper, we considered the uncertainty in processing time and sequence-dependent setup times, as well as the number of jobs at each production shift (dynamic scenarios). However, other sources of uncertainty can be considered in future works. Finally, applying the presented the MLBSM to other real-world scheduling contexts is another research path worth exploring.

Table 9
Descriptive analysis of the synthetic dataset for ten jobs.

Columns (Features/Target variables)	Count (NR)	Average	Standard deviation	Minimum	Q1	Q2	Q3	Maximum
J1 SPBM Score	4000	42.87	6.99	14.41	38.27	44.40	48.79	52.18
J2 SPBM Score	4000	43.01	6.71	17.66	38.79	44.42	48.53	51.95
J3 SPBM Score	4000	42.47	6.72	22.43	38.28	44.01	47.95	51.80
J4 SPBM Score	4000	42.31	7.20	20.45	38.00	43.93	48.10	52.64
J5 SPBM Score	4000	43.08	6.70	17.02	39.66	44.57	48.29	51.97
J6 SPBM Score	4000	42.83	6.74	20.21	38.38	43.98	48.44	52.99
J7 SPBM Score	4000	43.02	6.30	19.14	39.50	44.47	48.10	51.94
J8 SPBM Score	4000	42.92	6.82	15.98	38.55	44.54	48.52	51.98
J9 SPBM Score	4000	42.71	6.83	17.55	38.44	44.49	47.94	51.88
J10 SPBM Score	4000	42.69	7.04	18.57	38.37	44.48	48.29	52.78
J1 Completion Time	4000	1018.14	188.14	526.52	871.00	1016.33	1162.53	1509.96
J2 Completion Time	4000	930.85	185.80	519.02	796.45	937.55	1060.88	1402.76
J3 Completion Time	4000	992.20	201.35	538.47	841.42	1000.29	1147.54	1583.93
J4 Completion Time	4000	1006.61	196.33	583.86	863.91	1012.40	1161.60	1510.62
J5 Completion Time	4000	928.45	183.50	565.81	809.28	924.63	1051.44	1383.94
J6 Completion Time	4000	972.10	184.36	518.20	844.11	990.90	1116.25	1364.35
J7 Completion Time	4000	993.84	199.47	580.53	843.96	1004.94	1132.08	1458.49
J8 Completion Time	4000	1027.19	183.31	608.46	885.23	1034.90	1169.35	1482.04
J9 Completion Time	4000	983.02	186.33	504.44	850.87	989.63	1119.55	1390.35
J10 Completion Time	4000	969.61	186.98	523.02	838.57	985.73	1105.82	1437.65

CRediT authorship contribution statement

Erfan Nejati: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Ensieh Ghaedy-Heidary:** Writing – original draft, Conceptualization. **Amir Ghasemi:** Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Methodology, Formal analysis. **S. Ali Torabi:** Validation, Supervision, Project administration.

Data availability

Data will be made available on request.

Appendix A. Descriptions of RFR, ETR, and GBR algorithm

- **RFR:** According to Breiman (2001), in RFR, each tree in the ensemble is constructed using a sample that is drawn with replacement, also known as a bootstrap sample, from the training dataset. Additionally, during the process of splitting each node while constructing a tree, the optimal split is determined by thoroughly searching the feature values of either all input features or a randomly chosen subset of the same size. These two sources of randomness aim to reduce the variance of the forest estimator. Individual decision trees usually have high variance and are prone to overfitting. The randomness introduced in forests results in decision trees with partially independent prediction errors. Averaging these predictions allows some errors to offset each other. By aggregating various trees, random forests lower the variance, albeit occasionally increasing the bias slightly. In practical applications, the reduction in variance is frequently substantial, resulting in a generally improved model.
  Unlike the original publication by Breiman (2001), the scikit-learn implementation (used in this research) aggregates classifiers by averaging their probabilistic predictions rather than having each classifier vote for a single class.
- **ETR:** In ETR, randomness is taken a step further in determining how splits are computed. Similar to random forests, a random subset of candidate features is used; however, instead of searching for the most discriminative thresholds, thresholds are randomly selected for each candidate feature. The best of these randomly-generated thresholds is then chosen as the splitting rule. This approach typically reduces the model's variance slightly more, albeit with a small increase in bias (Geurts, Ernst, & Wehenkel, 2006).
- **GBR:** GBR, as described by Friedman (2001), constructs an ensemble of weak learners, usually decision trees, in a sequential manner. The process begins with an initial model that typically predicts a constant value minimizing the loss function, such as the mean of the target variable for squared error loss. In each iteration, the algorithm computes pseudo-residuals, which are the gradients of the loss function concerning the current model's predictions. These pseudo-residuals indicate the errors the current model is making. A new weak learner is then fitted to these pseudo-residuals, focusing on correcting the errors of the current model.
  The current model is updated by adding the new weak learner's predictions, scaled by a learning rate to control the contribution of each learner and prevent overfitting. This iterative process continues, with each new learner improving the accuracy of the model by addressing the shortcomings of its predecessors. The final predictive model is a weighted sum of all the weak learners, combining their strengths to form a robust predictor. This method effectively reduces prediction error through gradient descent, minimizing the overall loss, and producing a powerful ensemble model.

Appendix B. Descriptive analysis and correlation study of the synthetic dataset

Table 9 presents a descriptive analysis of the features and target variables in the real-size dataset used in this research. It should be noted that, as described in Section 5, this research generates several synthetic datasets, each for training ABRs in a dynamic scenario. The dataset presented in this appendix is one of the largest ones in terms of the number of columns. Since the other datasets only differ from this one in their number of columns, the descriptive analysis presented in this appendix provides insights into all the others. This dataset consists of 4000 rows, each representing a randomly generated Job Shop Queue for a production shift. During each shift, ten jobs are scheduled (the dynamic scenario that this dataset is designed for), resulting in the dataset having 20 columns (two columns for each job). For each job, one column represents the calculated SPBM Score (feature), and the other column represents the simulated mean completion times (target variable). Other system and job parameters related to the generation of this data set are described in Table 5.

Due to the randomness of the 4000 Job Shop Queues, there is no significant gap between the descriptive values of the SPBM Score and the simulated mean completion times of different jobs when we

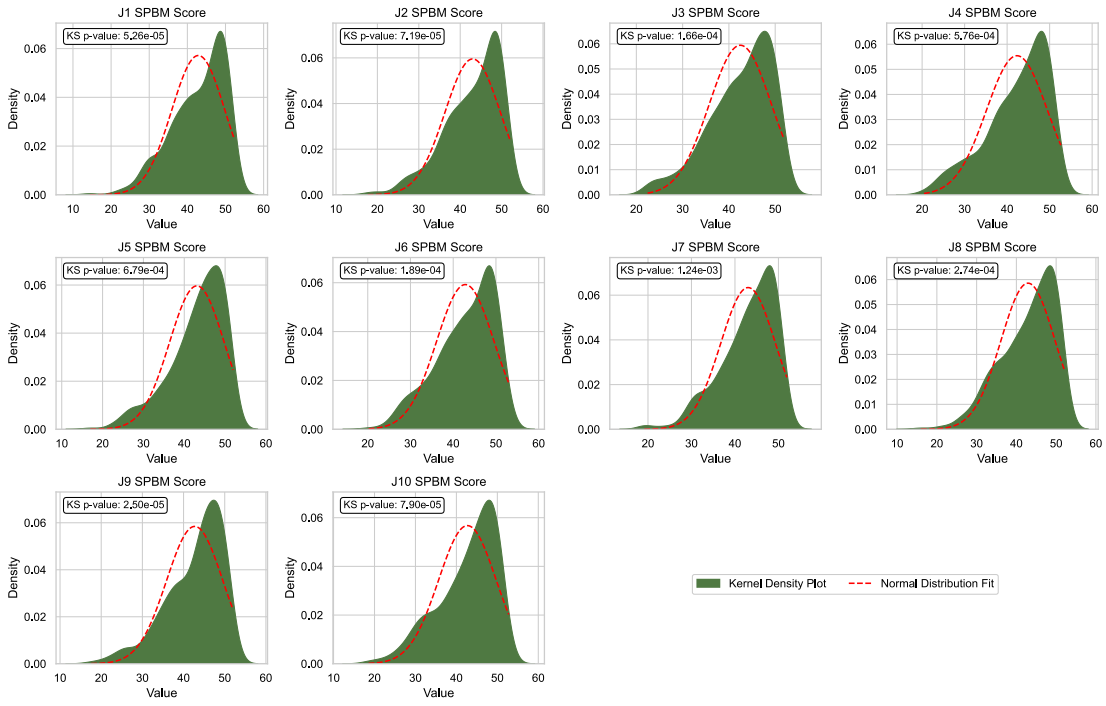


Fig. 17. Kernel density plots with Normal distribution fits and the *p*-value for Kolmogorov–Smirnov tests for the feature columns in the dataset (SPBM Scores).

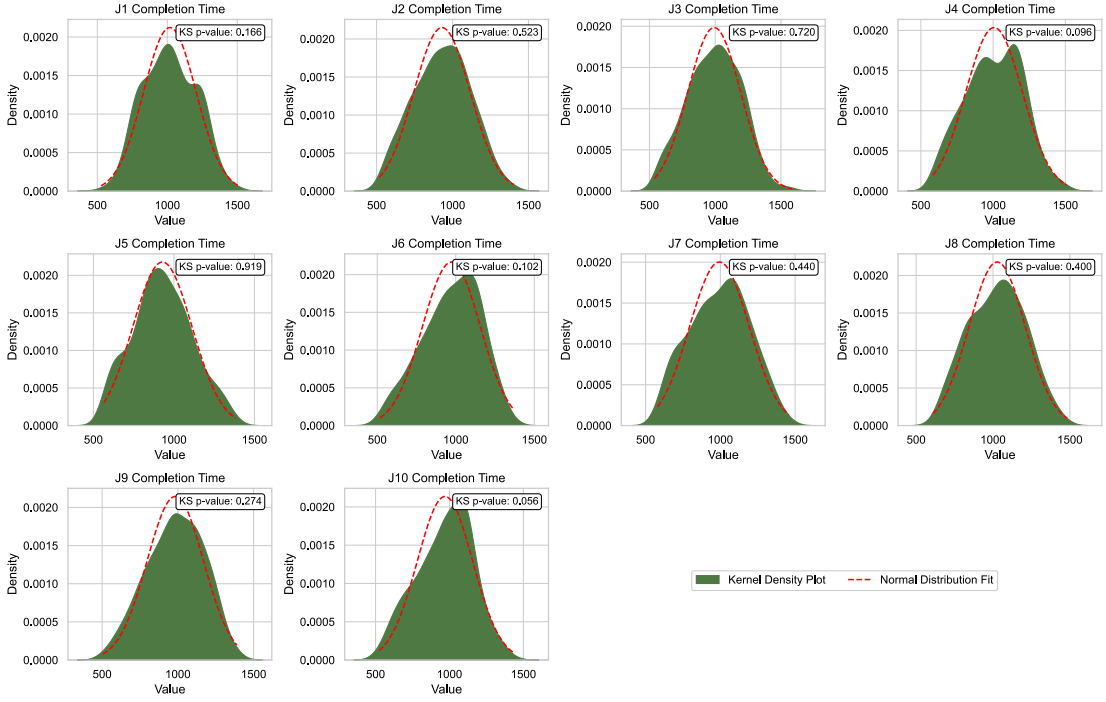


Fig. 18. Kernel density plots with Normal distribution fits and the *p*-value for Kolmogorov–Smirnov tests for the target variables columns in the dataset (Completion Times).

look at them as a whole. The slight differences stem from the nature of each job. For instance, J5, with the smallest average of simulated mean completion times, could have a smaller number of operations compared to J8, which has the largest average of simulated mean completion times. This gap is even smaller for SPBM Scores as these scores are entirely independent of the Job Shop Queue and the position of operations of each job in the queue. Therefore, since Job Shop Queues are generated randomly, the gap between the SPBM Scores of different jobs is minimized in this dataset.

Additionally, Figs. 17 and 18 illustrate the distribution of data within each column. Each subplot in these figures shows a Kernel density plot estimated for each column, alongside a Normal distribution fit. The comparison is further elucidated through a Kolmogorov–Smirnov goodness-of-fit test, with the corresponding *P*-value reported on each subplot. The obtained results suggest that at a 95% confidence interval, there is no evidence to reject the hypothesis that simulated mean completion times follow a normal distribution. This characteristic was previously discussed in Section 4.4 regarding the additivity property of independent normal distributions. However, the normality test for

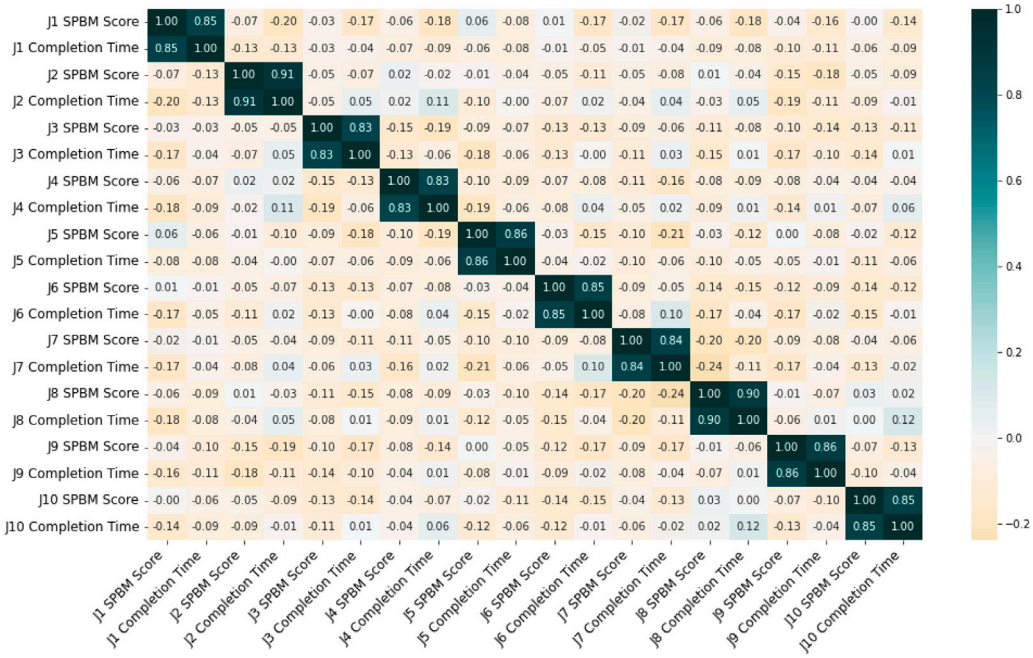


Fig. 19. Correlation matrix of features and target variables.

SPBM Scores is rejected at this confidence interval level, as the distribution of these variables is skewed to the right. The normal distribution of these variables was not a requirement, as ABRs can be trained and predict regardless of the distribution of features.

Finally, aiming to investigate the relationship between features and target variables, Fig. 19 illustrates the correlation matrix of the dataset columns. As expected, each SPBM column is highly correlated with its corresponding mean completion times column, demonstrating the validity of the SPBM method in vectorizing a Job Shop Queue into a numerical vector while minimizing the loss of information needed to predict mean completion times.

References

Ahuett-Garza, H., & Kurfess, T. (2018). A brief discussion on the trends of habilitating technologies for Industry 4.0 and Smart manufacturing. *Manufacturing Letters*, *15*, 60–63.

Alexopoulos, K., Nikolakis, N., & Chryssolouris, G. (2020). Digital twin-driven supervised machine learning for the development of artificial intelligence applications in manufacturing. *International Journal of Computer Integrated Manufacturing*, *33*(5), 429–439.

Amorim, M., Antunes, F., Ferreira, S., & Couto, A. (2019). An integrated approach for strategic and tactical decisions for the emergency medical service: Exploring optimization and metamodel-based simulation for vehicle location. *Computers & Industrial Engineering*, *137*, Article 106057.

Ankenman, B., Nelson, B. L., & Staum, J. (2010). Stochastic kriging for simulation metamodeling. *Operations Research*, *58*(2), 371–382, URL http://pubsonline.informs.org/doi/10.1287/opre.1090.0754.

Azadeh, A., Moghaddam, M., Geranmayeh, P., & Naghavi, A. (2010). A flexible artificial neural network–fuzzy simulation algorithm for scheduling a flow shop with multiple processors. *International Journal of Advanced Manufacturing Technology*, *50*(5–8), 699–715.

Barton, R. (1998). Simulation metamodels. *Vol. 1*, In *1998 winter simulation conference. proceedings (cat. no.98CH36274)* (pp. 167–174). Washington, DC, USA: IEEE, ISBN: 978-0-7803-5133-2, URL http://ieeexplore.ieee.org/document/744912/.

Barton, R. R. (2009). Simulation optimization using metamodels. In *Proceedings of the 2009 winter simulation conference* WSC, (pp. 230–238). Austin, TX, USA: IEEE, ISBN: 978-1-4244-5770-0, URL http://ieeexplore.ieee.org/document/5429328/.

Barton, R. R. (2020). Tutorial: Metamodeling for simulation. In *2020 winter simulation conference* WSC, (pp. 1102–1116). Orlando, FL, USA: IEEE, ISBN: 978-1-72819-499-8, URL https://ieeexplore.ieee.org/document/9384059/.

Barton, R. R., & Meckesheimer, M. (2006). Metamodel-based simulation optimization. *Vol. 13*, In *Handbooks in Operations Research and Management Science* (pp. 535–574). ISBN: 0927-0507 Publisher: Elsevier.

Bartz-Beielstein, T., & Zaefferer, M. (2017). Model-based methods for continuous and discrete global optimization. *Applied Soft Computing*, *55*, 154–167, URL https://linkinghub.elsevier.com/retrieve/pii/S1568494617300546.

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

Can, B., & Heavey, C. (2012). A comparison of genetic programming and artificial neural networks in metamodeling of discrete-event simulation models. *Computers & Operations Research*, *39*(2), 424–436.

Chen, J. C., Chen, T.-L., & Hung, H.-C. (2020). Capacity allocation with lot splitting in photolithography area using hybrid genetic algorithm based on self-tuning strategy. *Computers & Industrial Engineering*, *148*, Article 106656.

Chen, C., Fathi, M., Khakifirooz, M., & Wu, K. (2022). Hybrid tabu search algorithm for unrelated parallel machine scheduling in semiconductor fabs with setup times, job release, and expired times. *Computers & Industrial Engineering*, *165*, Article 107915.

Chien, C.-F., & Lan, Y.-B. (2021). Agent-based approach integrating deep reinforcement learning and hybrid genetic algorithm for dynamic scheduling for Industry 3.5 smart production. *Computers & Industrial Engineering*, *162*, Article 107782.

Chung, S., Huang, C.-Y., & Lee, A. (2006). Capacity allocation model for photolithography workstation with the constraints of process window and machine dedication. *Production Planning and Control*, *17*(7), 678–688.

Dahmen, T., Trampert, P., Boughorbel, F., Sprenger, J., Klusch, M., Fischer, K., et al. (2019). Digital reality: a model-based approach to supervised learning from synthetic data. *AI Perspectives*, *1*(1), 1–12.

Dellino, G., Kleijnen, J. P., & Meloni, C. (2009). Robust simulation-optimization using metamodels. In *Proceedings of the 2009 winter simulation conference* WSC, (pp. 540–550). IEEE, ISBN: 1-4244-5771-8.

Drucker, H. (1997). Improving regressors using boosting techniques. *Vol. 97*, In *ICML* (pp. 107–115). Citeseer.

Fonseca, D. J., Navaresse, D. O., & Moynihan, G. P. (2003). Simulation metamodeling through artificial neural networks. *Engineering Applications of Artificial Intelligence*, *16*(3), 177–183.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 1189–1232.

García, Á., Bregon, A., & Martínez-Prieto, M. A. (2022). Towards a connected digital twin learning ecosystem in manufacturing: Enablers and challenges. *Computers & Industrial Engineering*, *171*, Article 108463.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*, 3–42.

Ghaedy-Heidary, E., Nejati, E., Ghasemi, A., & Torabi, S. A. (2024). A simulation optimization framework to solve stochastic flexible job-shop scheduling problems—Case: Semiconductor manufacturing. *Computers & Operations Research*, *163*, Article 106508.

Ghasemi, A., Ashoori, A., & Heavey, C. (2021). Evolutionary learning based simulation optimization for stochastic job shop scheduling problems. *Applied Soft Computing*, *106*, Article 107309, URL https://linkinghub.elsevier.com/retrieve/pii/S1568494621002325.

Ghasemi, A., Azzouz, R., Laipple, G., Kabak, K. E., & Heavey, C. (2020). Optimizing capacity allocation in semiconductor manufacturing photolithography area – Case study: Robert bosch. *Journal of Manufacturing Systems*, *54*, 123–137, URL http://www.sciencedirect.com/science/article/pii/S0278612519301153.

Ghasemi, A., Farajzadeh, F., Heavey, C., Fowler, J., & Papadopoulos, C. T. (2024). Simulation optimization applied to production scheduling in the era of industry 4.0: A review and future roadmap. *Journal of Industrial Information Integration*, Article 100599.

Gramacy, R. B., & Lee, H. K. H. (2010). Cases for the nugget in modeling computer experiments. stat.

Hermann, M., Pentek, T., Otto, B., et al. (2015). *Vol. 45*, *Design principles for Industrie 4.0 scenarios: a literature review*. Dortmund: Technische Universität Dortmund.

Hu, H., & Zhang, H. (2012). A simulation-based two-stage scheduling methodology for controlling semiconductor wafer fabs. *Expert Systems with Applications*, *39*(14), 11677–11684.

Koons, G., & Perlic, B. (1977). *A study of rolling-mill productivity utilizing a statistically designed simulation experiment: Technical report*, Institute of Electrical and Electronics Engineers (IEEE).

Lee, E. A. (2008). Cyber physical systems: Design challenges. In *2008 11th IEEE international symposium on object and component-oriented real-time distributed computing* (pp. 363–369). IEEE.

Leitão, P., Colombo, A. W., & Karnouskos, S. (2016). Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges. *Computers in Industry*, *81*, 11–25.

Min, Q., Lu, Y., Liu, Z., Su, C., & Wang, B. (2019). Machine learning based digital twin framework for production optimization in petrochemical industry. *International Journal of Information Management*, *49*, 502–519, URL https://linkinghub.elsevier.com/retrieve/pii/S0268401218311484.

Mönch (2018). A survey of semiconductor supply chain models part III: master planning, production planning, and demand fulfilment. *International Journal of Production Research*, *56*(13), 4565–4584, URL https://www.tandfonline.com/doi/full/10.1080/00207543.2017.1401234.

Monostori, L. (2002). AI and machine learning techniques for managing complexity, changes and uncertainties in manufacturing. *IFAC Proceedings Volumes*, *35*(1), 119–130.

Nasiri, M. M., Yazdanparast, R., & Jolai, F. (2017). A simulation optimisation approach for real-time scheduling in an open shop environment using a composite dispatching rule. *International Journal of Computer Integrated Manufacturing*, *30*(12), 1239–1252.

Pierreval, H. (1992). Training a neural network by simulation for dispatching problems. In *The third international conference on computer integrated manufacturing* (pp. 332–333). IEEE Computer Society.

Pierreval, H., & Huntsinger, R. C. (1992). An investigation on neural network capabilities as simulation metamodels. In *Proc. of the summer computer simulation conference* (pp. 413–417).

Pires, F., Cachada, A., Barbosa, J., Moreira, A. P., & Leitão, P. (2019). Digital twin in industry 4.0: Technologies, applications and challenges. *Vol. 1*, In *2019 IEEE 17th international conference on industrial informatics* (pp. 721–726). IEEE.

Ross, S. M. (2014). *Introduction to probability models*. Academic Press.

Shao, G., & Kibira, D. (2018). Digital manufacturing: Requirements and challenges for implementing digital surrogates. In *2018 winter simulation conference* (pp. 1226–1237). IEEE.

Sharma, P., & Jain, A. (2017). Effect of routing flexibility and sequencing rules on performance of stochastic flexible job shop manufacturing system with setup times: Simulation approach. *Proceedings of the Institution of Mechanical Engineers, Part B (Management and Engineering Manufacture)*, *231*(2), 329–345.

Shen, X.-N., Han, Y., & Fu, J.-Z. (2017). Robustness measures and robust scheduling for multi-objective stochastic flexible job shop scheduling problems. *Soft Computing*, *21*(21), 6531–6554.

Walpole, R. E., Myers, R. H., Myers, S. L., & Ye, K. (1993). *Vol. 5*, *Probability and statistics for engineers and scientists*. Macmillan New York.

Weeks, J. K., & Fryer, J. S. (1977). A methodology for assigning minimum cost due-dates. *Management Science*, *23*(8), 872–881.

Xu, W., Shao, L., Yao, B., Zhou, Z., & Pham, D. T. (2016). Perception data-driven optimization of manufacturing equipment service scheduling in sustainable manufacturing. *Journal of Manufacturing Systems*, *41*, 86–101.

Yang, F., Liu, J., Nelson, B. L., Ankenman, B. E., & Tongarlak, M. (2011). Metamodelling for cycle time-throughput-product mix surfaces using progressive model fitting. *Production Planning and Control*, *22*(1), 50–68, URL https://www.tandfonline.com/doi/full/10.1080/09537287.2010.490026.

Yang, H., Lv, Y., Xia, C., Sun, S., & Wang, H. (2014). Optimal computing budget allocation for ordinal optimization in solving stochastic job shop scheduling problems. *Mathematical Problems in Engineering*, *2014*, 1–10, URL http://www.hindawi.com/journals/mpe/2014/619254/.

Zhang, G., Lu, X., Liu, X., Zhang, L., Wei, S., & Zhang, W. (2022). An effective two-stage algorithm based on convolutional neural network for the bi-objective flexible job shop scheduling problem with machine breakdown. *Expert Systems with Applications*, *203*, Article 117460.

Zhang, M., Wang, L., Qiu, F., & Liu, X. (2023). Dynamic scheduling for flexible job shop with insufficient transportation resources via graph neural network and deep reinforcement learning. *Computers & Industrial Engineering*, *186*, Article 109718.