Learning-based scheduling of flexible manufacturing systems using ensemble methods

Paolo Priore[1*], Borja Ponte[2], Javier Puente[1], Alberto Gómez[1]

[1] Departamento de Administración de Empresas, Escuela Politécnica de Ingeniería de Gijón, Universidad de Oviedo, {priore, jpuente, albertogomez}@uniovi.es

[2] Logistics Systems Dynamic Group, Cardiff Business School, Cardiff University, ponteblancob1@cardiff.ac.uk

*corresponding author.

Abstract

Dispatching rules are commonly applied to schedule jobs in Flexible Manufacturing Systems (FMSs). However, the suitability of these rules relies heavily on the state of the system; hence, there is no single rule that always outperforms the others. In this scenario, machine learning techniques, such as support vector machines (SVMs), inductive learning-based decision trees (DTs), backpropagation neural networks (BPNs), and case based-reasoning (CBR), offer a powerful approach for dynamic scheduling, as they help managers identify the most appropriate rule in each moment. Nonetheless, different machine learning algorithms may provide different recommendations. In this research, we take the analysis one step further by employing ensemble methods, which are designed to select the most reliable recommendations over time. Specifically, we compare the behaviour of the *bagging*, *boosting*, and *stacking* methods. Building on the aforementioned machine learning algorithms, our results reveal that ensemble methods enhance the dynamic performance of the FMS. Through a simulation study, we show that this new approach results in an improvement of key performance metrics (namely, mean tardiness and mean flow time) over existing dispatching rules and the individual use of each machine learning algorithm.

Keywords

Machine learning; Knowledge-based systems; Ensemble methods; Scheduling; Simulation; Flexible Manufacturing System

1. Introduction

Scheduling represents an essential part of the control of Flexible Manufacturing Systems (FMSs). It refers to the process of allocating a limited and shared set of resources (e.g. plant and machinery resources) when manufacturing several products during the same time window. It is aimed at maximizing the efficiency of the operation and minimizing production costs; by means of determining when each job must be processed (Shaw, Park, & Raman, 1992). In this sense, scheduling significantly impacts on the firms' productivity and financial performance.

A scheduling problem may comprise two different decisions (Wang & Usher, 2005; Nouiri, Bekrar, Jemai, Niar, & Ammari, 2018). The first decision, which is known as *job sequencing* and is the root of the scheduling problem, entails calculating the sequence of the jobs awaiting their next operation in the machine queue. The second one, which is known as *job routing*, involves assigning the job operations to the different machines. This subproblem only appears when routing flexibility is allowed, and it makes the scheduling problem in FMSs significantly more complex than in traditional job shops, as both decisions strongly interact and impact on system performance (Chaudhry & Khan, 2016). Abedinnia, Glock, Grosse, and Schneider (2017), Chaudhry and Khan (2016), and Dios and Framinan (2016) offer recent reviews of the scheduling literature.

The literature includes four main methodological approaches to the scheduling problem: (1) exact methods; (2) heuristic; (3) simulation; and (4) artificial intelligence (Priore, De la Fuente, Gómez, & Puente, 2006; Priore, Gómez, Pino, & Rosillo, 2014). The first approach includes classical exact resolution methods, such as branch-and-bound and dynamic and integer programming, for which optimization packages like CPLEX and GLPK are usually employed. They provide the optimal solution of a scheduling optimization problem defined by an objective function and a set of constraints; see e.g. Azizoglu and Kirca (1998). However, this approach is only time-efficient for small-scale scheduling problems (Cho & Wysk, 1993), which are commonly built on assumptions that may often be understood as unrealistic simplifications. For large-scale problems, which are generally NP-complete problems (Garey & Johnson, 1979), these methods become extremely time consuming or even unfeasible, and other methodological solutions are required.

The complexity of many scheduling problems led research into heuristic methods. They generally translate into simple dispatching rules —although they also may take other, more complex, forms— for prioritizing all the jobs that are awaiting for processing in a dynamic, or reactive, manner (Ouelhadj & Petrovic, 2009). Their value lies in being easy-to-implement strategies that are able to provide high-quality solutions with a low computational effort (Xanthopoulos, Koulouriotis, Tourassis, & Emiris, 2013). However, their performance strongly depends on many factors, such as the selected optimization criteria, the system configuration, and the workload (Cho & Wysk, 1993). In this sense, a specific rule may work well in a certain state of the FMS, but may turn out to be inappropriate in a subsequent state.

For this reason, the design of systems capable of modifying the dispatching rule over time in response to the changes in the state of the system gained the attention of researchers. To do this, there are two main research streams in the literature. The first one is based on simulating a set of predefined rules and selecting at every moment that one which provides the best performance (see, for example, Ishii & Talavage, 1991; Jeong & Kim, 1998; Kim & Kim, 1994; Kutanoglu & Sabuncuoglu, 2001; Wu & Wysk, 1989).

The second one is based on the use of artificial intelligence techniques. This approach aims to gain knowledge of the FMS from a set of examples in order to determine the best rule for each possible system state. These examples —which may be obtained through simulation and/or from the operation of the real system— are used to train a machine learning algorithm (Michalski, Carbonell, & Mitchell, 1983), which generates the knowledge. These algorithms generally offer high-performance solutions to the scheduling problem in reasonable computation times. Thus, intelligent decisions can be made in real time (see, for instance, Azadeh, Maleki Shoja, Moghaddam, Asadzadeh, & Akbari, 2013; Azadeh, Negahban, & Moghaddam, 2014; Choi, Kim, & Lee, 2011; Guh, Shiue, & Tseng, 2011; Heger, Branke, Hildebrandt, & Scholz-Reiter, 2016; Mönch, Zimmermann, & Otto, 2006; Mouelhi-Chibani & Pierreval, 2010; Priore et al., 2006; Priore, Parreño, Pino, Gómez, & Puente, 2010; Shaw et al., 1992; Shiue & Guh, 2006; Shiue, Guh, & Lee, 2011). The reviews by Akyol and Bayhan (2007), Priore, De la Fuente, Gómez, and Puente, (2001), and Priore et al. (2014) provide further detail on machine learning applications to the scheduling problem.

In recent years, the sets of classifiers obtained through *ensemble methods* have been one of the research areas most explored within the machine learning field. A *set of classifiers* may be defined as a group of classifiers (that is, baseline machine learning algorithms) whose individual decisions are combined in some way to classify new examples (Dietterich, 1997). In this sense, various classifiers are employed at the same time with the aim of improving their individual accuracy.

Several ensemble methods can be found in the problem-specific literature. Three of the most widely used techniques are *bagging* (Breiman, 1996), *boosting* (Freund & Schapire, 1996; Schapire, 1990) and *stacking* (Wolpert, 1992). The first two methods generate homogeneous classifiers given the fact that a single learning algorithm is used (Dietterich, 2000). In contrast, the stacking method generates heterogeneous classifiers as a consequence of the use of different learning algorithms. Another noticeable difference is that this method, unlike the bagging and boosting methods, does not employ a voting mechanism. In this sense, it aims to avoid a final misclassification caused by most classifiers awarding wrong predictions.

In this article, we employ ensemble methods within the scheduling problem with the goal of improving the accuracy of the individual classifiers employed independently as well as the performance of the traditional use of dispatching rules. We apply and compare the bagging, boosting, and stacking mechanisms. It should be highlighted that to date sets of classifiers have been hardly employed in FMSs. One of the few studies is that by Shiue, Guh, and Lee (2012), who use the bagging method. These authors justify that this method fits better with the nature of the scheduling problem in real time than the boosting method, given that there is a substantial classification noise. Under this scenario, and to the best of our knowledge, ours is the first article employing and comparing different ensemble methods to solve the scheduling problem in FMSs.

The rest of this paper has been structured as follows. First, Section 2 describes the machine learning algorithms used in this article and Section 3 details the ensemble methods. Next, Section 4 presents our approach to scheduling jobs based on machine learning, while Section 5 develops the experimental study we conducted. The proposed scheduling system is compared both to the static use of dispatching rules and to the individual application of each machine learning algorithm. Finally, Section 6 concludes by revisiting the research goals of this article.

2. Machine learning algorithms

Ensemble methods use machine learning algorithms as baseline classifiers. To this end, we have selected four algorithms widely employed in practice: support vector machines (SVMs), inductive learning, backpropagation neural networks (BPNs) and case-based reasoning (CBR).

SVMs (Cortes & Vapnik, 1995) were originally conceived for binary classification. Later, this technique was adapted to problems with a high number of classes. This cutting-edge algorithm generates complex mathematical models, whose strength lies in its ability to model nonlinearities (Wang, Hao, Ma, & Jiang, 2011). In light of this, SVMs have proven to achieve high performance in a wide range of applications, such as credit scoring, financial time series, and scheduling.

Inductive learning algorithms sprang from the works of Hoveland and Hunt in the late 1950s, which culminated in the 1960s in the concept learning systems (Hunt, Marin, & Stone, 1966). These algorithms are able to generate a decision tree (DT) from a set of training examples —this set is recursively divided into subsets composed of single-class collections of examples. The C4.5 algorithm (Quinlan, 1993), which is the most well-known inductive learning algorithm, also produces a set of decision rules, whereby new problems can be solved by determining the class of these new cases.

Artificial neural networks are inspired by the structure and the operation principles of the nervous system of animals. BPNs, whose architecture is known as multilayer perceptron (Rumelhart, Hinton, & Williams, 1986), represent the most used type of neural networks for pattern classification and function approximation (Freeman & Skapura, 1991; Lippman, 1987). In the training process, this algorithm obtains the connection weights and thresholds that minimize the difference between the actual and the desired output. This allows the algorithm to classify new cases.

Finally, CBR looks at similar problems in the past to solve the current one. It can be formalized as a four-step process (Watson, 1997): (i) Retrieving similar cases to the target problem; (ii) Reusing the solutions from these cases to the problem; (iii) Revising the suggested solutions to better fit the problem; and (iv) Retaining the experience as a new case in memory. The nearest neighbour (k-NN) algorithm (Aha, 1992) is one of the most popular algorithms in CBR applications for retrieving and reusing past cases. To classify a new case, the k-NN first measures the distance of this case to each training

example (past case). Then, the predominant class in the k 'nearest', or more similar, examples is assigned to the new case.

3. Ensemble methods

Ensemble methods aim to compensate the errors made individually by the baseline classifiers in the different parts of the data. Thus, they are designed to obtain a better predictive performance than could be achieved from any baseline classifier alone. It should be noted that the diversity among the baseline classifiers is generally promoted, as it allows to enrich the model. This diversity can be achieved in different ways, e.g. employing different learning algorithms, calibrating the classifiers through different sampling methods, or projecting examples onto different features subsets (Kim, 2009). Next, we describe three popular ensemble methods, which are employed in this work: bagging, boosting, and stacking.

Bootstrap aggregating, commonly abbreviated as bagging, is one of the earliest ensemble learning methods (Breiman, 1996). In this method, each training subset is generated by randomly selecting *n* examples with replacement, where *n* is the size of the original training set. For this reason, some examples may be repeated in each training subset. From this point, each training subset is employed to obtain a baseline classifier of the same type. The classes of new examples are selected according to the majority of the baseline classifiers —that is, "one classifier, one vote" (Bramer, 2016).

On the contrary, boosting adjusts the weights of the original sample set. This adjustment increases the weight of examples that are misclassified by the learning algorithm, while it decrements those weights of the examples that are correctly classified. Therefore, the final model obtained by this method is a linear combination of the baseline classifiers weighted by their own behaviour. We use the AdaBoost (Freund & Schapire, 1996) algorithm, which is the most popular in this area, to implement the boosting method.

Stacking, or stacked generalization, was developed earlier, but it is less widespread since it is difficult to analyse from a theoretical perspective (Wang et al., 2011). Unlike bagging and boosting, stacking is not commonly employed to combine classifiers of the same type but it deals with classifiers of different nature, which are not expected to be correlated. Figure 1 provides an overview of this ensemble method, which uses the

concept of meta-learning, according to which a learning algorithm (level-2) is employed over a new dataset formed by the predictions made by the baseline learning algorithms (level-1) with the original data. As in cross-validation, the data used to develop the level-1 classifiers should not be employed to generate the data that will be used by the level-2 learning algorithm.

To implement this method, we employ different combinations of learning algorithms both in level-1 and in level-2. In addition, we use the Naïve Bayes algorithm (Mitchell, 1997) as a level-2 algorithm, given that simple algorithms commonly perform well at this level (Witten & Frank, 2005). Wang et al. (2011) provides further details, including the pseudocode, of the described ensemble methods.

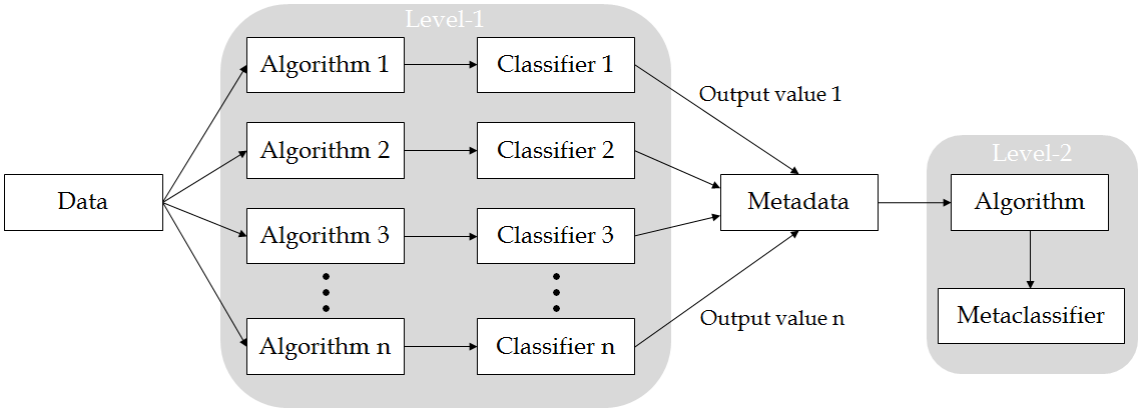

Figure 1. Structure of the stacking ensemble method.

4. Scheduling using machine learning and ensemble methods

According to Nakasuka and Yoshida (1992), a real-time scheduling system that dynamically modifies dispatching rules must verify two somewhat contrasting features in order to work properly. First, the selection of the best rule must take into consideration a wide variety of real-time information about the manufacturing system. Second, the selection process must be completed fast enough to avoid the delay of real operations.

To successfully meet both requirements, some kind of knowledge about the interdependencies between the FMS state and the optimal rule is required. Then, the key question becomes how this knowledge can be acquired. To cope with this problem, we develop a solution based on machine learning algorithms. Figure 2 shows the framework

we propose for the FMS scheduling system built on artificial intelligence-based techniques (Priore et al., 2006; Priore et al., 2010).

The operation of the learning-based system requires a vast number of training and test examples, which may be obtained from the past operation of the FMS and/or from simulating its performance in a wide range of scenarios. In our case, we employ a FMS simulation model as the generator of examples. To generate each example, the simulation model randomly generates a state of the FMS —defined by the rate of arrival of parts, the relative workload, the due date tightness, and so on— and calculates the best dispatching rule for each state.

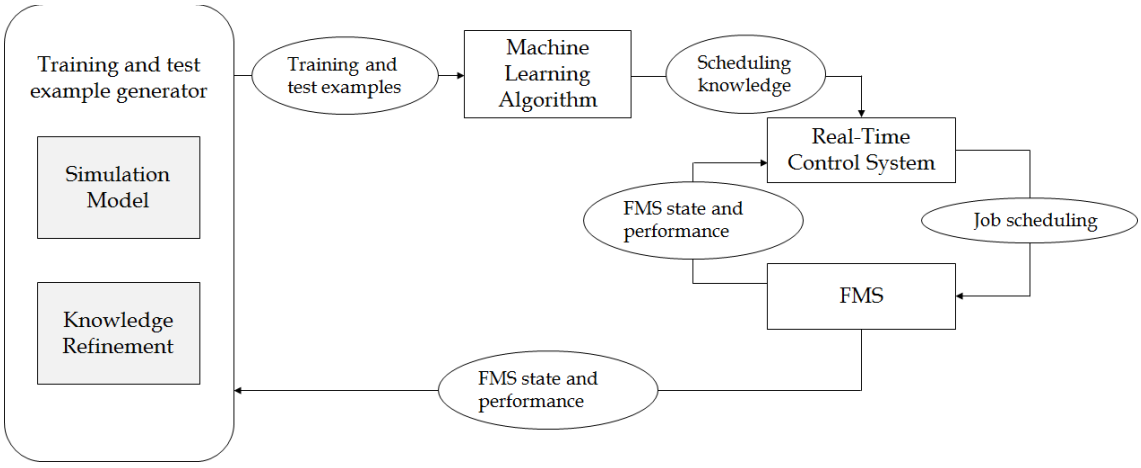

Figure 2. General overview of a learning-based scheduling system.

The knowledge required to regulate the FMS is generated by the different machine learning algorithms. It can be noted that each algorithm encapsulates the knowledge in a different manner; for instance, inductive learning algorithms construct a decision tree, which results in a set of decision rules. This knowledge would allow managers to make scheduling decisions in real time through a control system of the FMS. This real-time control system evaluates periodically, according to the monitoring period, the state of the system, and selects, employing the acquired knowledge, the best dispatching rule for scheduling jobs. In addition, the performance of the FMS is evaluated. It is relevant to highlight that this process results in a feedback loop emerging between the control system and the FMS, which can be seen in Figure 2. If at any time the performance of the system drops, further examples may be helpful to increase the accuracy of the algorithm by refining the knowledge about the manufacturing system. The mathematical formulation

of a learning-based scheduling system can be found in Park, Raman, and Shaw (1997) and Shiue et al. (2012).

Following the explanations in the previous section, the application of ensemble methods represents an evolution in the development of the control system for the FMS, as they make the control system capable of considering the knowledge obtained by different machine learning algorithms at the same time. In this sense, Figure 3 illustrates the conceptual transition from a FMS governed by a fixed combination of dispatching rules to the architecture we propose based on ensemble methods, understanding the control by means of machine learning algorithms as the intermediate step.

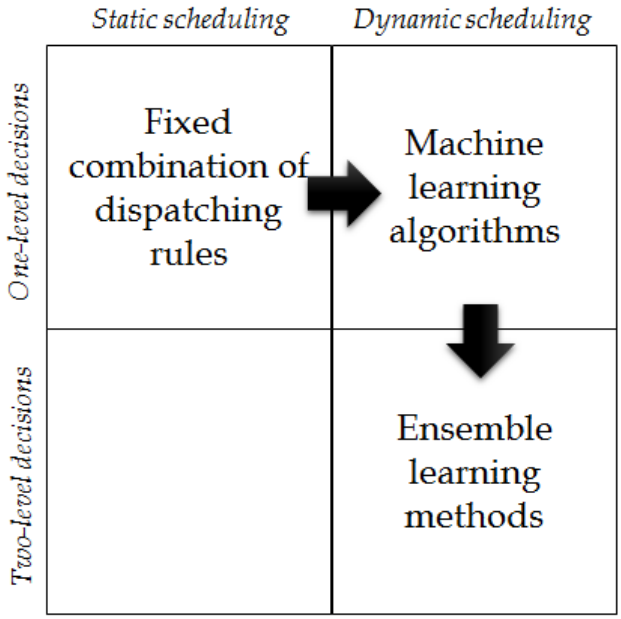

Figure 3. Conceptual evolution of the approaches for scheduling of FMS.

5. Experimental study

5.1. The proposed FMS

In this research, we employ the FMS model developed by Min, Yih, and Kim (1998) and Kim, Min, and Yih (1998). The baseline model is shown in Figure 4. It is formed of four machining centres, each one of them having each own input and output buffer. In addition, the system contains a washing machine, a crane as the material handling system, thirty-two storage racks for work-in-process, and a loading/unloading station. Although in practice each machine centre has different interchangeable tool magazines —which enables each of them to process various operations by mounting

different tool magazines—, we have assumed that the FMS works on the basis of a predefined policy for tooling arrangement.

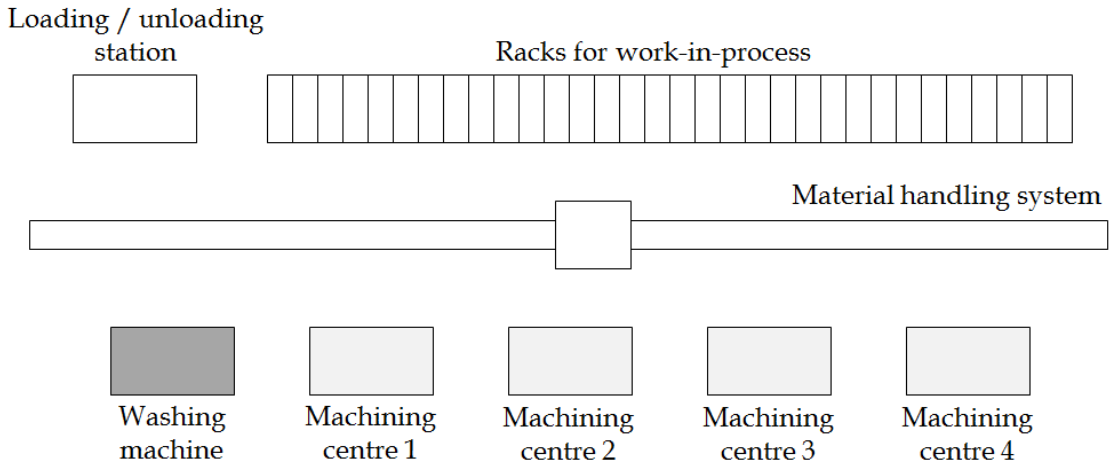

Figure 4. Configuration of the FMS.

Given that operations can be carried out on different machines, we consider two major decisions in the FMS: the job sequencing and the job routing problems (Wang & Usher, 2005; Nouiri et al., 2018). Firstly, the selection of the parts by the machines, i.e. the job sequencing problem. To prioritize the jobs that are competing for the use of each machine, we apply the following dispatching rules, which define various priority strategies: shortest processing time (SPT), earliest due date (EDD), modified job due date (MDD), and shortest remaining processing time (SRPT). In this sense, a priority index, x_i, is assigned to each job, and that with the lowest priority index will be executed first. The calculation of the priority index for each rule is shown below. For the SPT,

$$x_i = p_{ij}; \tag{1}$$

for the EDD,

$$x_i = d_i; \tag{2}$$

for the MDD,

$$x_i = \max\{t + P_{ij}, d_i\}; \tag{3}$$

and for the SRPT

$$x_i = P_{ij}; \tag{4}$$

where d_i is the due date of job i; p_{ij} is the processing time of operation j of job i; Pij is the remaining processing time for job i at the beginning of operation j, and t is the moment

when the scheduling decision is taken. The due date of job i (d_i) is calculated, following Baker (1984), by the following expression:

$$d_i = t_i + p_i \cdot F; \tag{5}$$

where F is the flow allowance factor which measures due date tightness; t_i is the moment when job i arrives at the system, and p_i is the total processing time of job i.

The second decision entails the selection of the machines by the parts i.e. the job routing problem. This involves assigning the job operations to the different machines, a problem which exist when routing flexibility is allowed, like in FMSs. The dispatching rules employed for this work are:

- shortest processing time (SPT), which selects the machine that requires less time to carry out the operation;
- number in queue (NINQ), which selects the machine with the lowest number of jobs in the buffer;
- work in queue (WINQ), which selects the machine whose input buffer contains the smallest amount of work; and
- lowest utilised station (LUS), which selects the machine with the lowest overall utilisation rate.

It should be underlined that the rules for both decisions have been chosen since they have shown to perform well in foundational works (Kim et al., 1998; Min et al., 1998; O'Keefe & Kasirajan, 1992; Shaw et al., 1992).

We have considered the following control attributes to describe the FMS state in every moment:

- flow allowance factor (F), measuring due date tightness (Baker, 1984);
- mean number of alternative machines for an operation (NAMO);
- mean utilisation of the FMS (*MU*);
- utilisation of each machine (U_n, where n refers to the machine);
- mean number of parts in the system, or work-in-process (*WIP*),
- ratio of the utilisation of the bottleneck machine to the mean utilisation of the FMS (RBM), that is,

$$RBM = \frac{\max\{U_1, U_2, U_3, U_4\}}{MU}, \tag{6}$$

- ratio of the standard deviation of the individual machine utilisations to the mean utilisation (RSDU), by

$$RSDU = \frac{\sqrt{\frac{(U_1 - MU)^2 + (U_2 - MU)^2 + (U_3 - MU)^2 + (U_4 - MU)^2}{4}}}{MU}. \quad (7)$$

Finally, we employ the mean tardiness and the mean flow time to measure the dynamic performance of the FMS in our experimental study, since both criteria are widely employed in the scheduling literature (Fernandes, Thürer, Silva, & Carmo-Silva, 2017; Fernandez-Viagas, Perez-Gonzalez, & Framinan, 2018). In this regard, it can be clarified that mean tardiness (*MT*) is defined as:

$$MT = \frac{\sum T_i}{N}. \quad (8)$$

where N is the number of finished jobs and T_i=max{0, L_i}, being L_i the difference between the date the job is finished and the agreed due date (d_i).

By way of summary, Figure 5 demarcates the scope of the FMS in this research by highlighting the system-in-focus and its state variables together with the control inputs and the key performance indicators.

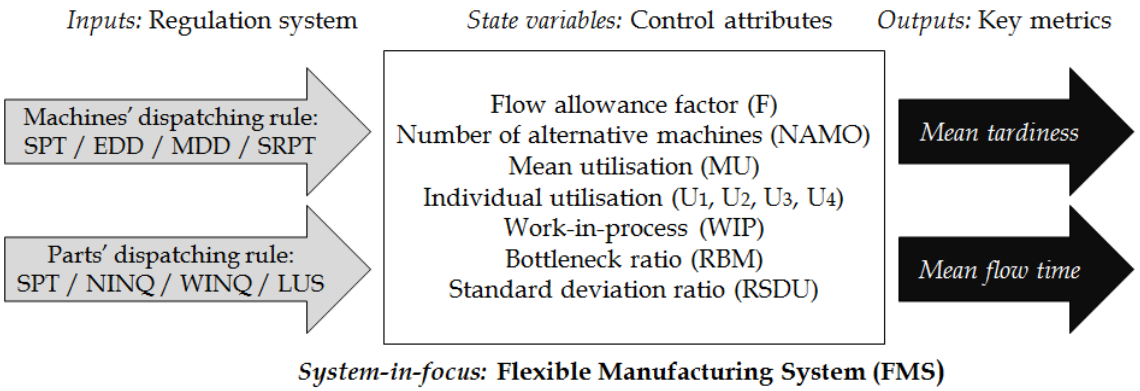

Figure 5. Modeling approach of the FMS considered in this research work.

5.2. Generating the examples

The FMS model has been implemented by means of the WITNESS simulation software (Witness, 2006). To carry out the simulation runs, we have considered the following assumptions:

- Jobs arrive at the system following a Poisson statistical distribution.
- Processing times for each operation are obtained from an exponential distribution with a mean of 1.

- The actual number of operations of each job is a random variable following a uniform discrete distribution from 1 to 4.
- The number of alternative machines for an operation varies between 1 and 2.
- The arrival rate varies so that the overall use of the FMS fluctuates between 55% and 95%.
- The factor F ranges between 1 and 10.
- To study the behaviour of the FMS in an unbalanced situation, we assume that the machining centres 1 and 2 have a greater workload.

To generate the examples that will be used in the subsequent sections, we have randomly generated 1,100 combinations of the seven control attributes as per the previously defined assumptions —each one of them will translate into one example per each criterion. We selected this size for the dataset given that prior works, such as Priore et al. (2006), have shown that over this number of examples the accuracy of the machine learning algorithms does not increase significantly. Each combination of control attributes was simulated for the 16 possible alternatives of combinations between the two different (machines' and parts') dispatching rules, and the system stores the best one according to each criterion. The combination of the seven control attributes and the class, which refers to the best combination of rules, define each example. By way of illustration, Table 1 presents a small sample of the training dataset for the criterion of mean tardiness.

Table 1. Extract of the training dataset for the criterion of mean tardiness.

	Control attributes									
No.	*F*	*NAMO*	*MU*	*RBM*	*RSDU*	*WIP*	U_1	U_2	U_3	U_4	*Class*
1	5	1	61.5	1.48	0.45	10.7	87	91.1	32.9	34.9	MDD+SPT
2	4	2	72.4	1.10	0.10	5.8	79.8	79.5	64.4	65.9	MDD+NINQ
3	7	1	58.6	1.48	0.45	8.2	83.2	86.8	31.1	33.3	MDD+SPT
4	7	2	74.1	1.12	0.12	6.2	82.9	82.6	64.6	66.3	EDD+NINQ
5	8	2	77	1.09	0.10	6.7	84.3	84.3	69	70.4	EDD+NINQ
6	7	1	58	1.48	0.45	7.7	82.6	85.9	30.8	32.9	MDD+SPT
7	2	2	84.6	1.03	0.03	8	87.1	87.1	82.4	81.6	SPT+NINQ
8	10	1	65.4	1.42	0.40	13.2	90.4	92.9	39.2	39.2	MDD+SPT
9	7	2	67.9	1.16	0.14	5.3	75.8	78.5	63	54.4	EDD+WINQ
10	2	2	85.8	1.04	0.05	8.4	89.1	89.3	86.8	78	SPT+WINQ

Looking at the overall training data set, it can be observed that three combinations predominate for the criterion of mean flow time (SPT+SPT, SPT+NINQ, SPT+WINQ).

However, most of the combinations are selected at some point for the criteria of mean tardiness. All in all, this fact clearly illustrates the need for modifying the dispatching rules in real time in response to the FMS state.

5.3. Application of the machine learning algorithms and ensemble methods

The experiments have been carried out by employing the data-mining software RapidMiner (Hofmann & Klinkenberg, 2013) with the Weka extension (Witten & Frank, 2005). For the machine learning algorithms, we use the same configuration both when they are employed individually and when they are employed within the ensemble methods —this allows us to ensure that the comparison has been carried out in a fair setting. In addition, the results have been validated through the cross-validation method, by which the example set is divided into ten different blocks —nine for obtaining the knowledge and the other for testing the classifier.

Table 2 shows the average accuracy for the baseline classifiers and for the bagging and boosting methods, when we consider the criterion of the mean tardiness. First of all, we observe that the BPNs achieve the highest average accuracy, followed respectively by the SVMs, the DT and the CBR. When the bagging technique is used, both the DT and the BPNs are able to increase their performance —especially in the case of the DT. On the contrary, neither the CBR nor the SVMs are capable of improving their response. Interestingly, similar results have been obtained for the boosting technique, although the increase in the accuracy of the DT is smaller in this case.

Table 2. Average accuracy of baseline classifiers, bagging, and boosting for the mean tardiness criterion.

Baseline classifier	*Average accuracy (%)*	*Bagging*	*Average accuracy (%)*	*Boosting*	*Average accuracy (%)*
DT	81.82	DT	83.36	DT	82.36
CBR	78.00	CBR	78.00	CBR	78.00
SVMs	85.45	SVMs	85.45	SVMs	85.45
BPNs	86.73	BPNs	87.45	BPNs	87.45

Table 3 includes the results of the stacking technique for different combinations of level-1 and level-2 classifiers. When comparing these results to those shown in Table 2, it can be seen that many of these combinations achieve a higher average accuracy than the top-performer algorithm when employed as a baseline classifier, i.e. the BPNs (86.73%). From inspection of Table 3, we observe that when the DT, the BPNs or the

CBR are employed in level-2, the SVMs and the BPNs provide the best results as baseline classifiers (level-1) —otherwise, the system will not outperform the individual results of the best baseline classifier. Note that, for example, when the BPNs are employed as level-2 algorithm and the DT, CBR, SVMs, and BPNs are employed as level-1 algorithms, the average accuracy (85.36%) is lower than the average accuracy obtained by the best baseline classifier, which in this case is the BPNs. Likewise, when the BPNs are used as level-2 algorithm and the DT, SVMs, and BPNs as level-1 algorithms, the average accuracy is 86.73%, which is the same as that of the best baseline classifier. From inspection of Table 3, the same observation can be made, as we discussed previously, when we use the DT or the CBR as level-2 algorithms.

However, when the SVMs are used in level-2, the three alternatives in level-1 (DT, CBR, SVMs and BPNs; DT, SVMs and BPNs; SVMs and BPNs) offer an accuracy of 86.91%, 87.18% and 87.64 %, respectively. Therefore, they outperform the results obtained by the top-performer algorithm (BPNs, whose accuracy is 86.73%). Similarly, it can be seen from Table 3 that, when the Naïve Bayes is used at level-2, the system is also capable of achieving higher accuracy than the top-performer algorithm, regardless of the combination of algorithms at level-1.

Finally, we would like to underline that among the 15 alternatives, the highest accuracy has been obtained for the stacking method when the Naïve Bayes is employed as level-2 algorithm and simultaneously the DT, the CBR, the SVMs, and the BPNs are employed as level-1 algorithms (88.70%). This combination improved the accuracy obtained by the best baseline classifier (86.73%), but also the best performance obtained by the bagging and boosting methods (87.45% in both cases).

Table 3. Average accuracy (%) of stacking for the mean tardiness criterion.

Baseline classifier (level-1)	*Algorithm (level-2)*				
	DT	CBR	SVMs	BPNs	Naïve Bayes
DT, CBR, SVMs, BPNs	86.45	84.36	86.91	85.36	88.70
DT, SVMs, BPNs	86.45	85.09	87.18	86.73	88.30
SVMs, BPNs	87.18	87.27	87.64	86.91	87.95

The same analysis can be conducted for the second criterion. As we highlighted in Section 5.2, a smaller number of combinations of dispatching rules are employed in this case. This explains why the errors are significantly lower for this criterion, and

consequently the average accuracy increases. Table 4 shows these results. In general terms, we observe that the CBR achieves the highest accuracy (98.45%). Paradoxically, this was the algorithm that achieved the lowest accuracy for the mean tardiness criterion. In this case, this algorithm is followed closely by the DT, while the performance of the BPNs and, especially, the SVMs is significantly lower.

Table 4. Average accuracy of baseline classifiers, bagging, and boosting for the mean flow time criterion.

Baseline classifier	*Average accuracy (%)*	*Bagging*	*Average accuracy (%)*	*Boosting*	*Average accuracy (%)*
DT	98.36	DT	98.27	DT	98.82
CBR	98.45	CBR	98.45	CBR	98.45
SVMs	94.91	SVMs	94.91	SVMs	94.91
BPNs	96.36	BPNs	97.09	BPNs	96.91

When we apply the bagging method, only the BPNs are capable of increasing the performance obtained by the baseline algorithms employed independently (97.09% and 96.36%, respectively). The CBR and the SVMs present the same accuracy, while the DT slightly decreases its accuracy. When we apply the boosting method, both the BPNs and the DT experience a slight enhancement (0.46% in the former and 0.55% in the latter), while again the results of the CBR and the SVMs do not modify.

Table 5. Average accuracy (%) of stacking for the mean flow time criterion.

Base classifier (level-1)	*Algorithm (level-2)*				
	DT	CBR	SVMs	BPNs	Naïve Bayes
DT, CBR, SVMs, BPNs	98.91	98.64	98.91	98.64	99.08
DT, CBR, BPNs	98.91	98.64	98.91	98.64	99.08
DT, CBR	99.00	98.82	98.91	98.18	99.00

Finally, Table 5 presents the results of the stacking method for this second criterion. To a greater or lesser extent, all the combinations achieve a higher accuracy than the employment of the CBR as the only baseline classifier (98.45%). In this case, the highest accuracy is again achieved when the Naïve Bayes is employed as level-2 algorithm, while the overall result does not seem to be very sensitive to the combination of algorithms employed in level-1. This combination (99.08%) also outperforms the best results obtained by the top-performer baseline classifier (98.45%) as well as the bagging (98.45%) and boosting (98.82%) methods. With the other level-2 algorithms (DT, CBR, SVMs or BPNs), the learning-based system is also capable of increasing the accuracy

over the best baseline classifier, with the exception of the BPNs being used as level-2 algorithms and the CBR and DT being used as level-1 algorithms. In this case, the accuracy (98.18%) is slightly lower than that achieved by the top-performer baseline classifier.

5.4. Performance of the learning-based scheduling

After evaluating the accuracy of the different knowledge-based mechanisms, we now quantify their operational performance. To this end, we implement the different scheduling systems in the FMS simulation model. We will compare the performance of: (a) the sixteen different combinations of dispatching rules employed statically; (b) the four machine learning algorithms controlling the FMS in real time according to the state of the system; and (c) the three ensemble methods controlling the FMS in real time according to the state of the system by considering the different results provided by the machine learning algorithms.

A core question in the evaluation process is the selection of the monitoring period, as the frequency used to measure the control attributes may significantly impact on the results. For this reason, it is advisable to assess the performance of the different strategies under a wide range of monitoring periods —we have selected 2.5, 5, 10, and 20 time units (see, for instance, Jeong & Kim, 1998; Kim & Kim, 1994; Wu & Wysk, 1989). After carrying out a number of preliminary tests, we chose 2.5 time units as the most appropriate monitoring period.

Under these circumstances, we have designed two different scenarios for the FMS. In the first one (*scenario I*), changes are generated in the FMS at given time periods, defined by an independent and identically distributed random variable following a uniform discrete distribution from 50 to 500 time units. In the second one (*scenario II*), the time periods for generating the change range between 2.5 and 250 time units —that is, the FMS is subject to a significantly higher number of changes here. We note that the changes of the control attributes have been generated within the intervals described in Section 5.2 for the training dataset. Finally, it should be highlighted that have obtained each numerical result shown below as the average of ten independent replications of 100,000 time units.

Having clarified all these important aspects, Table 6 presents the results that we have obtained. For the sake of readability, we show the values of the mean tardiness and mean flow time in each case referred to the lowest values obtained, which represents the optimal response of the FMS. In each column, we use italics to highlight the best static combination of dispatching rules, the best knowledge-based baseline classifier, and the best ensemble method.

All in all, Table 6 first shows that the best alternative is to employ a dynamic scheduling system based on machine learning techniques. Second, this table provides evidence that the scheduling approach based on ensemble methods clearly generates the best results. And third, it may be concluded from Table 6 that the stacking method outperforms the bagging and boosting methods.

Table 6. Mean tardiness and mean flow time, in relative terms, for the proposed strategies.

Strategy used	*Mean tardiness*		*Mean flow time*	
	Scenario I	*Scenario II*	*Scenario I*	*Scenario II*
SPT+SPT	4.046	5.356	2.109	2.405
SPT+NINQ	1.191	1.192	*1.039*	1.044
SPT+WINQ	1.184	1.171	1.041	*1.042*
SPT+LUS	2.469	2.528	1.511	1.519
EDD+SPT	3.466	4.604	2.206	2.611
EDD+NINQ	1.504	1.639	1.328	1.391
EDD+WINQ	1.499	1.647	1.327	1.395
EDD+LUS	2.834	3.215	1.865	2.050
MDD+SPT	3.478	4.645	2.299	2.676
MDD+NINQ	*1.115*	*1.117*	1.228	1.250
MDD+WINQ	1.122	1.128	1.231	1.255
MDD+LUS	2.351	2.470	1.773	1.854
SRPT+SPT	4.437	5.994	2.280	2.655
SRPT+NINQ	1.357	1.382	1.132	1.141
SRPT+WINQ	1.360	1.374	1.137	1.142
SRPT+LUS	2.792	2.950	1.671	1.710
CBR	1.058	1.068	*1.004*	*1.004*
BPNs	*1.011*	*1.012*	1.016	1.017
DT	1.039	1.044	1.004	1.005
SVMs	1.018	1.020	1.024	1.026
Bagging	1.007	1.008	1.004	1.004
Boosting	1.007	1.008	1.002	1.002
Stacking	*1.000*	*1.000*	*1.000*	*1.000*

For the mean tardiness criterion, the combination of MDD and NINQ is the best alternative from a static perspective, followed closely by the combination of MDD and WINQ. However, this solution is clearly sub-optimal, since the four dynamic schedules

controlled by machine learning algorithms perform significantly better. Specifically, the BPNs and the SVMs provide a relatively similar result, which outperforms the DT and the CBR. When the bagging and boosting techniques are employed in the BPNs, the FMS response improves in comparison with the alternative of employing exclusively the BPNs. In line with the discussion in the previous section, the optimal is achieved with the stacking method (employing Naïve Bayes in level-2 and simultaneously the DT, the CBR, the BPNs, and the SVMs in level-1), as a consequence of the significant increase in the average accuracy. In view of our results, we can conclude that the mean tardiness of the best static alternative is between 11.5% and 11.7% larger than the solution provided by the stacking method.

We now focus on the criterion of the mean flow time, according to which the combinations of SPT and NINQ, for the first scenario, and SPT and WINQ, for the second one, are the best static alternatives. Consistent with the findings in the previous section, the CBR is the algorithm that minimizes the mean flow time, followed respectively by the DT, the BPNs, and the SVMs. The FMS behavior does not improve with the bagging method in comparison with the CBR; however, the boosting method is capable of generating a lower mean flow time. Again, the best results are obtained when we employ the stacking technique (employing Naïve Bayes in level-2 and simultaneously the DT, the CBR, the BPNs, and the SVMs in level-1). In this case, when the stacking method is employed, the indicator decreases by between 3.9% and 4.2% over the best static alternative.

A one-way ANOVA was conducted to test the differences between the top nine scheduling strategies (according to the previous analysis) in both scenarios: the best two static strategies (MDD+NINQ and MDD+WINQ for the criterion of mean tardiness, and SPT+NINQ and SPT+WINQ for the criterion of mean flow time), the four baseline classifiers (CBR, BPNs, DT, and SVMs), and the three different ensemble methods (bagging, boosting, and stacking). For each strategy, we considered 10 different replications. Table 7 shows the results of this analysis, which reveal that that there are statistically significant differences (confidence level: 95%) in the performance of the nine strategies according to both criteria and for the two different scenarios.

Table 7. One-way ANOVA table under the two different criteria in scenarios I and II.

Performance criterion	*Source of Variation*	*SS*	*DF*	*MS*	*F-test*	*p-value*
MT - (I)	Between Groups	2.564	8	0.321	70.670	0.000
	Within Groups	0.367	81	0.005		
	Total	2.932	89			
MT - (II)	Between Groups	1.536	8	0.192	25.211	0.000
	Within Groups	0.617	81	0.008		
	Total	2.153	89			
MFT - (I)	Between Groups	2.151	8	0.269	19.815	0.000
	Within Groups	1.099	81	0.014		
	Total	3.250	89			
MFT - (II)	Between Groups	2.409	8	0.301	12.999	0.000
	Within Groups	1.876	81	0.023		
	Total	4.285	89			

Note: SS - sum of squares; DF - degree of freedom; MS - mean squares.

Table 8. Criterion of mean tardiness: *p*-values of comparisons among scheduling strategies.

Criterion (scenario)	*Scheduling strategy*	*MDD+ WINQ*	*CBR*	*BPNs*	*DT*	*SVMs*	*Bagging*	*Boosting*	*Stacking*
MT - (I)	MDD+NINQ	**0.496**	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	MDD+WINQ		0.000	0.000	0.000	0.000	0.000	0.000	0.000
	CBR			0.000	0.017	0º.000	0.000	0.000	0.000
	BPNs				0.004	**0.814**	**0.563**	**0.563**	0.009
	DT					0.008	0.001	0.001	0.000
	SVMs						**0.416**	**0.416**	0.005
	Bagging							**1.000**	0.039
	Boosting								0.039
MT - (II)	MDD+NINQ	**0.603**	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	MDD+WINQ		0.000	0.000	0.000	0.000	0.000	0.000	0.000
	CBR			0.000	**0.072**	0.000	0.000	0.000	0.000
	BPNs				**0.060**	**0.929**	**0.925**	**0.925**	0.035
	DT					**0.073**	0.049	0.049	0.000
	SVMs						**0.854**	**0.854**	0.029
	Bagging							**1.000**	0.044
	Boosting								0.044

Once the differences have been verified, we compare the individual performance of the various strategies by means of Fisher's least significant difference (LSD) test. Tables 8 and 9 provide this information via the relevant p-values. It should be highlighted that the stacking-based scheduling system stands out above the other strategies with a significance level of 0.05 for the criterion of mean tardiness in both scenarios. However, the bagging and boosting methods are not able to significantly outperform the best machine learning algorithms, which are BPNs and SVMs. As regards the criterion of

mean flow time, there are no statistically significant differences between the top five scheduling strategies in neither of the two scenarios. It means that the use of ensemble methods does not translate into a significantly improved performance over the independent use of the best machine learning algorithms.

Table 9. Criterion of mean flow time: *p*-values of comparisons among scheduling strategies.

Criterion (scenario)	*Scheduling strategy*	*SPT+ WINQ*	*CBR*	*BPNs*	*DT*	*SVMs*	*Bagging*	*Boosting*	*Stacking*
MFT - (I)	SPT+NINQ	**0.991**	0.000	0.000	0.000	0.004	0.000	0.000	0.000
	SPT+WINQ		0.000	0.000	0.000	0.004	0.000	0.000	0.000
	CBR			0.018	**1.000**	0.000	**1.000**	**0.691**	**0.426**
	BPNs				0.018	**0.112**	0.018	0.006	0.002
	DT					0.000	**1.000**	**0.691**	**0.426**
	SVMs						0.000	0.000	0.000
	Bagging							**0.691**	**0.426**
	Boosting								**0.689**
MFT - (II)	SPT+NINQ	**0.601**	0.000	0.000	0.000	0.004	0.000	0.000	0.000
	SPT+WINQ		0.000	0.000	0.000	0.017	0.000	0.000	0.000
	CBR			**0.076**	**0.889**	0.003	**1.000**	**0.783**	**0.580**
	BPNs				**0.101**	**0.216**	**0.076**	0.041	0.021
	DT					0.005	**0.889**	**0.679**	**0.489**
	SVMs						0.003	0.001	0.001
	Bagging							**0.783**	**0.580**
	Boosting								**0.781**

6. Conclusions

This paper suggests a new approach to scheduling based on the application of ensemble methods. Building on machine learning algorithms, which allow managers to deal with the scheduling problem from a dynamic perspective by selecting the most appropriate dispatching rule over time, the use of ensemble methods takes the solution mechanism one step further. These methods make the control system capable of considering the recommendations made by different machine learning algorithms in order to detect those most reliable at each particular moment. Thus, this represents a conceptual evolution in the design of control systems for FMSs.

The first step for practitioners wishing to implement this solution would be to replicate the real-world setting through a validated and verified simulation prototype. This would allow them to explore the optimal combination of dispatching rules in a wide range of scenarios. In light of this, different machine learning algorithms may be able to generate the required knowledge to control the system over time. From this perspective,

ensemble methods would be able to consider the solutions provided by the different machine learning algorithms and evaluate their reliability; eventually proposing a combination of dispatching rules as the solution of the scheduling problem. Overall, this approach would equip managers with a decision-making tool to optimize the control of manufacturing systems in highly complex and dynamic environments.

We have demonstrated that this approach results in a meaningful operational improvement in the FMS from the perspective of mean tardiness —however, the improvement was not statistically significant for the criterion of mean flow time since the average accuracy of the machine learning algorithms used independently was very high. The improvement in these key metrics is especially noticeable for the stacking method, which, unlike the bagging and boosting methods, does not employ a voting mechanism but uses the concept of a meta-classifier, i.e. a top-level algorithm which learns from the outputs of the set of classifiers obtained in the first phase.

Future research in this field might focus on employing more decision types for the proposed FMS and/or evaluating the performance of our proposal in other structures of FMSs. In this regard, the more decision types are used and/or the more complex the system is, the more simulation runs are required to generate the training and test examples. Another interesting avenue for future research would be based on increasing the understanding on the sets of classifiers —a major limitation of this technique is the fact that humans struggle to understand the knowledge learned. Finally, we may explore the addition of a knowledge base refinement module. This mechanism would be aimed at modifying the core of the knowledge acquired once the FMS faces major changes.

References

Abedinnia, H., Glock, C.H., Grosse, E.H., & Schneider, M. (2107). Machine scheduling problems in production: A tertiary study. *Computers & Industrial Engineering*, 111, 403–416

Aha, D.W. (1992). Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies,* 36, 267-287.

Akyol, D.E., & Bayhan, G.M. (2007). A review on evolution of production scheduling with neural networks. *Computers & Industrial Engineering,* 53, 95-122.

Azadeh, A., Maleki Shoja, B., Moghaddam, M., Asadzadeh, S.M., & Akbari, A. (2013). A neural network meta-model for identification of optimal combination of priority dispatching rules and makespan in a deterministic job shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 67, 1549-1561.

Azadeh, A., Negahban, A., & Moghaddam, M. (2014). A hybrid computer simulation-artificial neural network algorithm for optimisation of dispatching rule selection in stochastic job shop scheduling problems. *International Journal of Production Research, 50*, 551-566.

Azizoglu, M., & Kirca, O. (1998). Tardiness minimization on parallel machines. *International Journal of Production Economics*, 55(2), 163-168.

Baker, K.R. (1984). Sequencing rules and due-date assignments in a job shop. *Management Science*, 30, 1093-1104.

Bramer, M. (2016). *Ensemble classification. In Principles of Data Minin*g (pp. 209-220). London: Springer.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24, 123–140.

Chaudhry, I.A., & Khan, A.A. (2016). A research survey: review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, 23, 551-591.

Cho, H., & Wysk, R.A. (1993). A robust adaptive scheduler for an intelligent workstation controller. *International Journal of Production Research*, 31, 771-789.

Choi, H.-S., Kim, J.-S., & Lee, D.-H. (2011). Real-time scheduling for reentrant hybrid flow shops: A decision tree based mechanism and its application to TFT-LCD line. *Expert Systems with Applications*, 38, 3514-3521.

Cortes, C., & Vapnik, V. (1995). Support-vector network. *Machine Learning*, 20, 273-297.

Dietterich, T. G. (1997). Machine-learning research: four current directions. *AI Magazine*, 18, 97–136.

Dietterich, T. G. (2000). Ensemble methods in machine learning, in. Kittler, J., Roli, F. (Eds.), First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science, 1-15, New York: Springer Verlag.

Dios, M., & Framinan, JM. (2016). A review and classification of computer-based manufacturing scheduling tools. *Computers & Industrial Engineering*, 99, 229–249.

Fernandes, N. O., Thürer, M., Silva, C., & Carmo-Silva, S. (2017). Improving workload control order release: Incorporating a starvation avoidance trigger into continuous release. *International Journal of Production Economics*, 194, 181-189.

Fernandez-Viagas, V., Perez-Gonzalez, P., & Framinan, J. M. (2018). The distributed permutation flow shop to minimise the total flowtime. *Computers & Industrial Engineering*, 118, 464-477.

Freeman, J.A., & Skapura, D.M. (1991). *Neural Networks: Algorithms, Applications, and Programming Techniques.* Reading, MA: Addison Wesley.

Freund, Y., & Schapire, R.E. (1996). Experiment with a new boosting algorithm. In M. Kaufmann, editor, Proceedings of the Thirteenth International Conference on Machine Learning, pp. 148–156.

Garey, M., & Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman.

Guh, R.-S., Shiue, Y.-R., & Tseng, T.-Y. (2011). The study of real time scheduling by an intelligent multi-controller approach. *International Journal of Production Research*, 49, 2977-2997.

Heger, J., Branke, J., Hildebrandt, T., & Scholz-Reiter, B. (2016). Dynamic adjustment of dispatching rule parameters in flow shops with sequence-dependent set-up times. *International Journal of Production Research*, 54, 6812-6824.

Hofmann, M., & Klinkenberg, R. (2013). *RapidMiner: Data mining use cases and business analytics applications*. Boca Raton, FL: CRC Press.

Hunt, E.B., Marin, J., & Stone, P.J. (1966). *Experiments in Induction*. New York: Academic Press.

Ishii, N., & Talavage, J. (1991). A transient-based real-time scheduling algorithm in FMS. *International Journal of Production Research*, 29, 2501-2520.

Jeong, K.-C., & Kim, Y.-D. (1998). A real-time scheduling mechanism for a flexible manufacturing system: using simulation and dispatching rules. *International Journal of Production Research*, 36, 2609-2626.

Kim, C.-O., Min, H.-S., & Yih, Y. (1998). Integration of inductive learning and neural networks for multi-objective FMS scheduling. *International Journal of Production Research*, 36, 2497-2509.

Kim, M.H., & Kim, Y.-D. (1994). Simulation-based real-time scheduling in a flexible manufacturing system. *Journal of Manufacturing Systems*, 13, 85-93.

Kim, Y. (2009). Boosting and measuring the performance of ensembles for successful database marketing. *Expert Systems with Applications*, 36, 2161-2176.

Kutanoglu, E., & Sabuncuoglu, I. (2001). Experimental investigation of iterative simulation-based scheduling in a dynamic and stochastic job shop. *Journal of Manufacturing Systems*, 20, 264-279.

Lippman, R.P. (1987). An introduction to computing with Neural Networks. *IEEE ASSP Magazine*, 3, 4-22.

Michalski, R.S., Carbonell, J.G., & Mitchell, T.M. (1983). *Machine Learning. An Artificial Intelligence Approach*. Palo Alto, CA: Tioga Press.

Min, H.-S., Yih, Y., & Kim, C.-O. (1998). A competitive neural network approach to multi-objective FMS scheduling. *International Journal of Production Research*, 36, 1749-1765.

Mitchell, T.M. (1997). *Machine Learning*. New York: McGraw-Hill.

Mönch, L., Zimmermann, J., & Otto, P. (2006). Machine learning techniques for scheduling jobs with incompatible families and unequal ready times on parallel batch machines. *Engineering Applications of Artificial Intelligence*, 19, 235-245.

Mouelhi-Chibani, W., & Pierreval, H. (2010). Training a neural network to select dispatching rules in real time. *Computers & Industrial Engineering*, 58, 249-256.

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

Nouiri, M., Bekrar, A., Jemai, A., Niar, S., & Ammari, A.C. (2018). An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing*, 29, 603-615.

Ouelhadj, D., & Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling*, 12, 417-431.

O'keefe, R.M., & Kasirajan, T. (1992). Interaction between dispatching and next station selection rules in a dedicated flexible manufacturing system. *International Journal of Production Research,* 30, 1753-1772.

Park, S.C., Raman, N., & Shaw, M.J. (1997). Adaptive scheduling in dynamic flexible manufacturing systems: A dynamic rule selection approach. *IEEE Transactions on Robotics and Automation,* 13, 486-502.

Priore, P., De la Fuente, D., Gómez, A., & Puente, J. (2001). A review of machine learning in dynamic scheduling of flexible manufacturing systems. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 15, 251-263.

Priore, P., De la Fuente, D., Gómez, A., & Puente, J. (2006). A comparison of machine-learning algorithms for dynamic scheduling of flexible manufacturing systems. *Engineering Applications of Artificial Intelligence*, 19, 247-255.

Priore, P., Gómez, A., Pino, R., & Rosillo, R. (2014). Dynamic scheduling of manufacturing systems using machine learning: An updated review. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 28, 83-97.

Priore, P., Parreño, J., Pino, R., Gómez, A., & Puente, J. (2010). Learning-based scheduling of flexible manufacturing systems using support vector machines. *Applied Artificial Intelligence* 24, 194-209.

Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers.

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning representations by back-propagating errors. *Nature,* 323, 533-536.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5, 197–227.

Shaw, M.J., Park, S., & Raman, N. (1992). Intelligent scheduling with machine learning capabilities: the induction of scheduling knowledge. *IIE Transactions*, 24, 156-168.

Shiue, Y.-R., & Guh, R.-S. (2006). The optimization of attribute selection in decision tree-based production control systems. *International Journal of Advanced Manufacturing Technology*, 28, 737-746.

Shiue, Y.-R., Guh, R.-S., & Lee, K.-C. (2011). Study of SOM-based intelligent multi-controller for real-time scheduling. *Applied Soft Computing,* 11, 4569-4580.

Shiue, Y.-R., Guh, R.-S., & Lee, K.-C. (2012). Development of machine learning-based real time scheduling systems: using ensemble based on wrapper feature selection approach. *International Journal of Production Research,* 20, 5887-5905.

Wang, G., Hao, J., Ma, J., & Jiang, H. (2011). A comparative assessment of ensemble learning for credit scoring. *Expert Systems with Applications*, 38, pp. 223-230.

Wang, Y.C., & Usher, J.M. (2005). Application of reinforcement learning for agent-based production scheduling. *Engineering Applications of Artificial Intelligence*, 18(1), 73-82.

Watson, I. (1997). *Applying Case-Based Reasoning: Techniques for Enterprise Systems*. San Francisco, CA: Morgan Kaufmann Publishers.

Witness. (2006). User Manual. Release 8.0. Lanner Group Ltd.

Witten, I.H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques*. Boston: Morgan Kaufmann Publisher.

Wolpert, D. (1992). Stacked generalization. *Neural Networks*, 5, 241–259.

Wu, S.-Y.D., & Wysk, R.A. (1989). An application of discrete-event simulation to on-line control and scheduling in flexible manufacturing. *International Journal of Production Research*, 27, 1603-1623.

Xanthopoulos, A.S., Koulouriotis, D.E., Tourassis, V.D., & Emiris, D.M. (2013). Intelligent controllers for bi-objective dynamic scheduling on a single machine with sequence-dependent setups. *Applied Soft Computing*, 13, 4704-4717.