POLITECNICO DI TORINO

Dip. di Automatica e Informatica (DAUIN)
Dept. of Control and Conputer Engineering

Optimization and Operations Research Group

Integration of Machine Learning and Optimization Techniques for Flexible Job-Shop Rescheduling in Industry 4.0

Yuanyuan Li
Stefano Carabelli
Edoardo Fadda
Daniele Manerba
Roberto Tadei
Olivier Terzo

October 2019

DAUIN-ORO-2019-06

Integration of Machine Learning and Optimization Techniques for Flexible Job-Shop Rescheduling in Industry 4.0

Yuanyuan Li[3], Stefano Carabelli[1], Edoardo Fadda[1], Daniele Manerba[1,2], Roberto Tadei[1]*, Olivier Terzo[3]

[1]Dept. of Control and Computer Engineering, Politecnico di Torino - Turin (Italy)

[2]ICT for City Logistics and Enterprises Lab, Politecnico di Torino - Turin (Italy)

[3]LINKS Foundation - via Pier Carlo Boggio 61, 10138 Torino, Italy

Abstract

With the fourth industrial revolution, different tools coming from Optimization, Internet of Things, Data Science, and Artificial Intelligence techniques are applied together to the production management. This paper shows, with the information coming from the sensors embedded in production lines, how Job-Shop rescheduling can benefit from the integration of Optimization and Machine Learning approaches. With the fact that manufacturing processes are stochastic, rescheduling decisions must be taken under uncertainty. However, to judge if it is worthy of rescheduling is a complex task, which is often tackled on a greedy base in practice. Rescheduling too frequently leads to possible losses (mainly result from the time spent in reorganization), while rescheduling too rarely does not eradicate an increasing accumulation of delays. The problem is faced by proposing an innovative classification methodology based on optimization algorithms and machine learning techniques. Numerical simulations, inspired from a real world application, prove the effectiveness of the proposed rescheduling policy.

Keywords: Industry 4.0; Flexible Job-Shop Scheduling; Rescheduling; Machine Learning classification; Optimization algorithms.

1 Introduction

The fourth industrial revolution (I4.0) is inexorable. This new industrialization is improving the work environment by bringing new business and technical opportunities. In particular, I4.0 enables decision makers to get real-time information from different components of plants and creates the possibility for machines to communicate with each other. From this point of view, I4.0 can be seen as the application of Internet of Things (IoT) in industrial production.

The availability of new sources of data enhances the understanding of processes as well as its management (see Trstenjak and Cosic 2017). It is important to notice that the availability of new real-time information has already improved the management in other fields such as waste collecting management (Fadda et al. 2018), synchromodal routing and logistics (Giusti et al. 2019), and opportunistic data gathering (Fadda, Perboli, and Tadei 2018). In the scheduling field, the availability of data can trigger improvements on several aspects. The several most

*Corresponding author: roberto.tadei@polito.it

important ones are identified by McKinsey (2015), namely, reduction of dead times, reliability strengthening, shortening of set-up times, waste reduction, reduction of warehouse costs, real-time management of exceptions, and management of the fixing times. In this paper, it considers the real-time management of exceptions and, in particular, it focuses on the problem of deciding when it is worth rescheduling during the ongoing production.

Rescheduling problem represents an important branch of the scheduling literature (see, e.g., Brucker 2010 and Gupta, Maravelias, and Wassick 2016). The need for rescheduling comes from several factors, such as the accumulation of delays in processing, sudden arrival of urgent orders, fault of machines, or absence of operators. In order to fully implement a rescheduling process, it is necessary to define a rescheduling policy, which is a solution method for generating a new schedule, compromising the possible achievement with the time and effort to implement the changes. In a manufacturing plant, since the available time for calculating a complex rescheduling plan is generally short, fast heuristics such as local search fit in.

Given any optimization technique to generate a new schedule, the main problem remains in choosing the optimal rescheduling time. This decision is actually easy to make if the company is facing unexpected but urgent orders or machine fails, i.e., the rescheduling procedure should be run as soon as possible. More in general, within a manufacturing system, rescheduling is necessary whenever unexpected events occur and the planned schedule becomes unfeasible. On the contrary, without obviously serious disruptions, the production plan seems feasible (such as when we consider the accumulation of processing time variations), then the potential problem becomes rather tough to figure out. For example, different processing time variations in a set of machines may either lead to small or big change on the scheduling performance, and it is hard to infer from the combinations of numerous variables. For this reason, in this paper, it proposes a machine learning (ML) approach which is able to characterize a production system, and then decide whether it needs to be rescheduled or not.

This article presents an innovative rescheduling system inspired by the production system of the company *SIGIT S.p.A* (Italy), within the industrial project named *Plastic and Rubber 4.0* (P&R4.0)[1]. The project aims to be the Italian answer to I4.0 for companies in the plastic and rubber processing field. In this work, we provide the following contributions:

- we formalize a Flexible Job-Shop Scheduling problem with sequence-dependent setup time and limited dual resources;
- we develop, implement, and test a heuristic approach which is able to find good schedules in a reasonable amount of time;
- we propose the use of Machine Learning algorithms for deciding when to reschedule (i.e., when to use the heuristic), exploiting the information coming from technologies provided in the I4.0 framework;
- we demonstrate through processes simulation that the ML approach provide an automated decision tool more effective when compared to commonly other used rescheduling approaches.

The rest of the manuscript is organized as follows. In Section 2, it reviews the main literature on scheduling and rescheduling, highlighting their relations with I4.0. Section 3 presents the basic scheduling problem and shows its mathematical programming formulation. Moreover, a hybrid heuristic approach is proposed, which combines Genetic Algorithm and Tabu Search components, for finding good solutions in a reasonable amount of time. Section 4 describes the

[1]Plastic&Rubber 4.0. Piattaforma Tecnologica Fabbrica Intelligente, URL: https://www.openplast.it/

rescheduling problem and proposes an approach based on Machine Learning classification and Optimization techniques, and Section 5 shows the results of simulated numerical experiments. Finally, Section 6 draws conclusions and sketches future research lines.

2 Literature review

Scheduling is the process of assigning tasks to resources or allocation of resources to perform tasks over a period of time. Although it is highly complex to characterize the problems which are in vast variations, the research challenge is in general to obtain a good solution within a reasonable computational time. This work focuses on one of the most important scheduling problems, namely, the Job-Shop Problem (JSP) (see Zhang 2017). The paper written by Brucker (2010) is referred to readers for a deeper discussion on other scheduling problems.

As shown by Sellers (1996), there has been an intensive study on the approaches to JSP including heuristic rules, classical optimization, and artificial intelligence (AI). And it is pointed out by Đurasević and Jakobović 2018 that *priority rules* and *dispatching rules* are probably the mostly used heuristic rules embedded in metaheuristic methods for scheduling problems. However, through the reviews, it is found that many papers do not present a mathematical model for the problems. Thus possible misunderstandings with respect to the actual implementation may be generated. With this concern, this paper first formalizes the problem which is actually Flexible Job-Shop Scheduling problem with sequence-dependent setup time and limited resources. The limited resources include both general purpose machines and setup workers. The setup operation is usually operated by a worker, so only the presences of both machine and worker can guarantee a possible configuration. However, with the fact that the lack of availability on setup workers is a common phenomenon in factories, most research does not consider the worker availability. For example, the paper Azzouz, Ennigrou, and Ben Said (2017) introduces the problem concerning both selection of the machine and operation with sequence-dependent setups, without mentioning workers. Same limitation occurs to the works of Gao et al. (2006) and Roshanaei, Azab, and ElMaraghy (2013). On the other hand, the paper Costa, Cappadonna, and Fichera (2013) considers workers as critical resource but the production is single stage composed by a set of unrelated parallel machines, which does not fit into our problem. Another research carried out by Gong et al. (2018) proposes the solution related to the flexibility of both workers and machines and the precedence among operations, however, there is no constraint on setup. To our knowledge, this problem has neither been formalized in the literature, nor heuristics have been proposed as solutions, nor for rescheduling. Consequently, one contribution of the present paper is from filling in the blank.

In the context of I4.0, where potentially every piece of production data is accessible, new opportunities for the production planning are available. Nevertheless, the I4.0 approach requires an IT infrastructure capable to cope with various data flows. The system dealing with the problem is called Cyber-Physical System (CPS), i.e. a system integrating embedded computing technologies into the physical world (see Gunes et al. 2014). According to Lee, Bagheri, and Kao (2015), in order to transform today's factories into I4.0 smart factories, CPS is a necessity for every I4.0 application. Cyber Physical Production System (CPPS) is a manufacturing-centered version of a CPS, in Lee et al. (2018) the authors have implemented a CPPS architecture framework collaborating with IoT, AI, simulations, manufacturing execution systems and advanced planning and scheduling systems in a real factory. Optimization of production processes is a potential benefit of CPPS (Rudtsch et al. 2014). It is important to underline the great influence of CPS and CPPS in the I4.0 framework, but we do not discuss them in detail because they are out of the central topic.

In the literature, there are several applications concerning scheduling and rescheduling optimization under the context of I4.0, one of the most important ones is Rossit, Tohmé, and Frutos (2019). In the work, the authors introduce a new decision-making schema intended to yield flexible and efficient production schedules on the fly, taking advantage of the features in these new environments. Another interesting study is Larsen and Pranzo (2019). Based on a classic JSP, the authors introduce a general rescheduling framework to address problems aroused from the dynamic nature of production scheduling. The proposed solution is composed by a solver that assumes deterministic and static data, and by a controller dealing with uncertainty that triggers off a new solution from the solver if the scheduling performance drops down below a certain threshold. The work is similar to our approach in the ideas of capturing the dynamic properties of rescheduling, but we are solving the problems under different context. In their work, while the performance of decision maker controller is highly dependent on the time when relevant information are obtained, it does not mention the possible integration with real time data collection techniques.

To the best of our knowledge, the two most important surveys about scheduling and rescheduling in the context of I4.0 are Dolgui et al. (2019) and Zhang (2017). In Dolgui et al. (2019), the authors present an I4.0 survey on the applications of optimal control to scheduling in production and supply chain by focusing on the deterministic maximum principle. They have two main objectives. The first one is to derive major contributions, application areas, limitations, as well as research and application recommendations for the future research. The second one is to explain control models in terms of industrial engineering and production management. To achieve these objectives, optimal control models, qualitative methods for performance analysis and computational methods for optimal control are considered. Instead, in Zhang (2017) the authors describe several optimization problems applied in I4.0. In the survey, it emerges that one of the most important and active research fields is the application of distributed optimization algorithms. However, neither in Zhang (2017), in Dolgui et al. (2019) nor in other similar papers, the problem of deciding the best moment in which to reschedule is considered.

For this reason, in the rest of the sections we report the most important works in the context of rescheduling. One of the first work in the area is Bierwirth and Mattfeld (1999). In that work, a general model for JSP is described, and a Genetic Algorithm (GA) for solving the problem is presented. This algorithm is tested in a dynamic environment under different workload situations. The authors test this technique for scheduling and rescheduling in a non-deterministic environment. The main problem of this approach is that in the real field, GA must run with different time limits for scheduling and rescheduling purposes thus reducing the performance. The first work that presents definitions appropriately for most rescheduling manufacturing systems and describes a framework for understanding rescheduling strategies, policies, and methods is Vieira, Herrmann, and Lin (2003). After that, more rescheduling related papers appear. The two most important and recent reviews in the context of rescheduling are Narayanaswami and Rangaraj (2011) and Uhlmann and Frazzon (2018). The first article presents an extensive set of major railway rescheduling operations. Even though different algorithms are presented, most of them are problem specific and cannot be generalized into the smart industry framework. Instead, focusing on solutions involving integration among industries and real application cases, Uhlmann and Frazzon (2018) presents a systematic literature review to identify what have been studied about production rescheduling process. It mainly addresses the choice of the rescheduling heuristic rather than the decision of the reschedule timing. The lack is a common phenomenon in the rescheduling literature and it exists also in the small branch of the literature dealing with the machine learning applications to rescheduling. For example, in the article Šemrov et al. (2016), the authors present an algorithm that uses Q-learning principles to change the schedule of the train

on a single track railway and in Palombarini, Barsce, and Martínez (2014) the authors develop an artificial cognition control system for obtaining rescheduling knowledge in the form of decision rules. No works, at least to our knowledge, face the problem to decide when to reschedule in a direct way.

3 The mathematical model

The optimization problem being considered is the Flexible Job-Shop Scheduling problem with sequence-dependent setup time and limited resources (FJSP). Based on traditional JSP, our FJSP introduces:

- the flexibility in selecting machines since there can be more than one machine capable of same operations;
- the limited resources of setup workers;
- the sequence-dependent setup, which is under the control of both machine and setup worker.

The key assumptions are listed as follows:

- no preemption is permitted for each operation, operations among different jobs are independent;
- one machine and one worker can only work on one operation at a time;
- all jobs, machines and workers are known in the beginning.

Due to the unavailability of a mathematical programming formulation for the FJSP in the literature, we propose it in the following. It is important to note that the model is extended to the flexibility on the setup worker, which means there can be more than one worker to choose for setup configurations, and the workers are assumed to have equivalent skills.

Let us consider the following sets:

- $\mathcal{J} = \{1, 2, \ldots, j_{max}\}$ is the set of jobs;
- $\mathcal{T} = \{1, 2, \ldots, t_{max}\}$ is the set of time steps;
- $\mathcal{M} = \{1, 2, \ldots, m_{max}\}$ is the set of machines;
- $\mathcal{O} = \{1, 2, \ldots, o_{max}\}$ is the set of all operations that must be done, each operation belonging to a specific different job;
- $\mathcal{C} = \{1, 2, \ldots, c_{max}\}$ is the set of all configurations of each machine;
- $\mathcal{C}_m \subseteq \mathcal{C}$ is the subset of all the configurations that a machine $m \in \mathcal{M}$ can take;
- $\mathcal{C}_o \subseteq \mathcal{C}$ is the subset of configurations that a machine can take in order to process operation $o \in \mathcal{O}$.

Since each job is a predefined set of operations with a fixed precedence relation, we define a directed graph $\mathcal{G} = (\mathcal{O}, \mathcal{E} \subseteq \mathcal{O} \times \mathcal{O})$ that enforces the precedence relations of the operations in the same job (see, e.g., Balas 1969). More precisely, an arc from operation $\tilde{o}$ to operation o means that operation $\tilde{o}$ must be done before operation o.

We also define the following parameters:

- $T_{mt}^{c\tilde{c}}$ is the setup time needed to change from configuration c to configuration $\tilde{c}$ on machine m at time t;
- T_{om} is the processing time for operation o done on machine m;
- L_t is the number of set up workers available at time t.

Let us consider the following decision variables:

- C_{max} is the value of the total makespan for the set of jobs;
- C_o is the completion time of operation o;
- y_{omt} is a binary variable taking value 1 iff operation o is processed on machine m at time t;
- s_{omt} is a binary variable taking value 1 iff operation o starts to be processed on machine m at time t;
- z_{mt}^{c} is a binary variable taking value 1 iff machine m is in configuration c at time t;
- $w_{mt}^{c\tilde{c}}$ is a binary variable taking value 1 iff machine m changes from configuration c to configuration $\tilde{c}$ at time t.

Then, a Mixed Integer Linear Programming formulation for the FJSP is as follows:

$$\text{minimize } C_{max} \tag{1}$$

subject to:

$$C_{max} \geq C_o, \ \ \forall o \in \mathcal{O} \tag{2}$$

$$\sum_{m \in \mathcal{M}} \sum_{t \in \mathcal{T}} s_{omt} = 1, \ \ \forall o \in \mathcal{O} \tag{3}$$

$$\sum_{o \in \mathcal{O}} s_{omt} \leq 1, \ \ \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{4}$$

$$\sum_{t \in \mathcal{T}} y_{omt} = T_{om} \sum_{t \in \mathcal{T}} s_{omt}, \ \ \forall o \in \mathcal{O}, \ \forall t \in \mathcal{T} \tag{5}$$

$$s_{omt} \leq y_{om\tilde{t}}, \ \ \forall t \in \mathcal{T}, \forall \tilde{t} \in [t, t + T_{om}], \forall o \in \mathcal{O}, \forall m \in \mathcal{M} \tag{6}$$

$$s_{omt} \leq \sum_{\tilde{m} \in \mathcal{M}} \sum_{\tilde{t}=1}^{t} s_{\tilde{o}\tilde{m}\tilde{t}}, \ \ \forall (\tilde{o}, o) \in \mathcal{E}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{7}$$

$$T_{om} s_{omt} \leq \sum_{\tilde{m} \in \mathcal{M}} \sum_{\tilde{t}=1}^{t} y_{\tilde{o}\tilde{m}\tilde{t}}, \ \ \forall \ (\tilde{o}, o) \in \mathcal{E}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{8}$$

$$\sum_{c \in \mathcal{C}} z_{mt}^{c} = 1, \ \ \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{9}$$

$$z_{mt}^{c} = 0, \ \ \forall c \in \mathcal{C} \setminus \mathcal{C}_m, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{10}$$

$$\sum_{m \in \mathcal{M}} \sum_{c \in \mathcal{C}} \sum_{\tilde{c} \in \mathcal{C}} w_{mt}^{c\tilde{c}} \leq L_t, \ \forall t \in \mathcal{T} \tag{11}$$

$$s_{omt} \leq \sum_{c \in \mathcal{C}_o} z_{mt}^{c}, \ \ \forall m \in \mathcal{M}, \forall t \in \mathcal{T}, \forall o \in \mathcal{O} \tag{12}$$

$$C_o \geq t\, y_{omt}, \quad \forall o \in \mathcal{O}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{13}$$

$$s_{omt} + y_{\tilde{o}mt} \leq 1, \quad \forall o, \tilde{o} \in \mathcal{O}, o \neq \tilde{o}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{14}$$

$$1 - w_{mt}^{c\tilde{c}} \geq z_{mt}^{c} - z_{m,t+1}^{\tilde{c}}, \quad \forall c, \tilde{c} \in \mathcal{C}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{15}$$

$$1 - w_{mt}^{c\tilde{c}} \geq z_{m,t+1}^{\tilde{c}} - z_{mt}^{c}, \quad \forall c, \tilde{c} \in \mathcal{C}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{16}$$

$$1 - w_{mt}^{c\tilde{c}} \geq s_{om\tilde{t}}, \quad \forall o \in \mathcal{O}, \forall c, \tilde{c} \in \mathcal{C}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T}, \forall \tilde{t} \in [t, t + T_{mt}^{c\tilde{c}}] \tag{17}$$

$$s_{omt} \in \{0, 1\}, \quad \forall o \in \mathcal{O}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{18}$$

$$y_{omt} \in \{0, 1\}, \quad \forall o \in \mathcal{O}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{19}$$

$$z_{mt}^{c} \in \{0, 1\}, \quad \forall m \in \mathcal{M}, \forall t \in \mathcal{T}, \forall c \in \mathcal{C} \tag{20}$$

$$w_{mt}^{c\tilde{c}} \in \{0, 1\}, \quad \forall c, \tilde{c}, \in \mathcal{C}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T}. \tag{21}$$

The objective function (1) aims at minimizing the makespan of the production. Constraints (2) ensure the correctness of makespan value by defining it as the maximum of all the completion times. Constraints (3) impose that each operation must be executed while (4) enforce that each operation must start in only one time step on only one machine. Constraints (5) ensure that each operation must last for the right amount of time. Moreover, constraint (6) enforces that an operation can not be executed unless it starts. The constraints (7) and (8) enforce the precedence relation among the operations while constraints (9) enforce that each machine must take a configuration. Constraints (10) avoid a machine to take a forbidden configuration. Constraints (11) limit the number of configuration changes that can be done in a certain time step. Furthermore, constraints (12)–(17) add the links between the variables. In particular, constraints (12) impose that an operation cannot start if the machine is not in the right configuration, constraints (13) enforce that the completion time of one operation must be greater than the maximum processing time of that operation, and constraints (14) impose that if a machine is executing an operation, then no other operations can start during the processing time. Constraints (15) and (16) define the logic consistence on variables w and z. Constraints (17) impose that if a machine is changing configuration, no operations can start during the relative setup time. Finally, constraints (19)–(18) define the binarity of the variables.

3.1 A heuristic solution approach

Problem (1)–(21) becomes large, even for small-size instances. It has a number of variables of the order of $2^{\max[|\mathcal{O}||\mathcal{M}||\mathcal{T}|,\ |\mathcal{M}||\mathcal{T}||\mathcal{C}|^2]}$ where $|\cdot|$ is the cardinality of the set. Thus, even for relative small instances (e.g., for $|\mathcal{M}| = 7$, $|\mathcal{T}| = 100$, $|\mathcal{T}| = 30$ and $|\mathcal{C}| = 3$), exact solvers are not able to solve the problem in a reasonable amount of time. Since real applications needs very efficient scheduling procedures not affecting the overall makespan, a heuristic approach is proposed for computing the initial scheduling.

In order to find good solutions of model (1)–(21) in a reasonable amount of time, a hybrid algorithm (HA) is used. HA consists of Genetics Algorithm and Tabu Search as discussed in Meeran and Morshed (2012). Genetics Algorithm (GA) is a nature inspired evolutionary algorithm. It simulates the biological solution to find the fitter offsprings by combining good parent individuals (see Mitchell 1998). Through running repeatedly the generation process, a best individual which means the one achieving the lowest make-span is returned. The local search algorithm Tabu Search (TS) is a strategy good for exploiting the neighboring solution (see Glover and Laguna 1997). It utilizes a memory structure called Tabu List to keep track of the moves done to find recently visited solutions, thus allowing the acceptance of non-improving solutions to get out from local optima.

3.1.1 Workflow

The flow-char shown in Figure 1 describes in detail the HA procedure. It starts with the GA that provides a set of initial solutions as population, and then GA selects solutions to do crossover and mutation. On each of the newly combined solutions, TS performs a local search. Then GA uses the TS improved solutions to continue with new evolution. This hybrid framework can be converted into traditional GA by omitting the TS steps. Similarly, it can be converted into traditional TS by setting the population size to one and omitting the genetic operators.

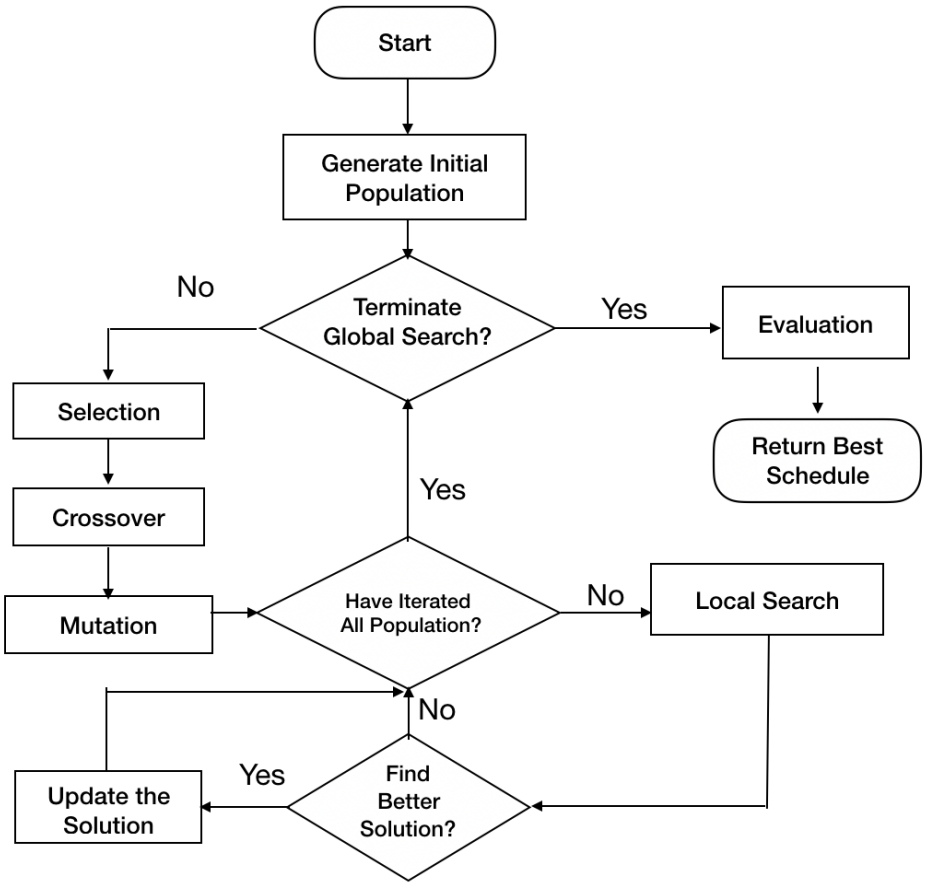

Figure 1: Flow chart of the heuristic approach.

3.1.2 Encoding and decoding

In GA, to describe a solution (i.e., a chromosome), it is essential to ensure all chromosomes, generated during the evolutionary process, guarantee feasibility. In the paper, it is demonstrated both from encoding and decoding aspects.

To encode an individual, job representation is chosen. A chromosome is a list of genes $[j_1, j_2, \ldots, j_{|\mathcal{O}|}]$, each gene corresponding to the job number of which a specific operation o belongs. More precisely, it means that the ith appearance of the job number j stands for the ith operation of job j.

Both the assignment of machines and the calculation of starting, ending time are done in decoding phase. The available machines for each operation can be more than one, therefore we use the modified greedy strategy to select only one randomly between the first and second earliest available machines. There are usually a few setup workers, so we set the preference on the first available setup worker. Afterwards, to calculate the starting and ending time of each operation, the availability of both machine and setup worker is considered. The objective is

to minimize the overall completion time, therefore the fitness is the value of makespan. The smaller the fitness is, the better the solution is.

An example is provided in Figure 2. We assume that each machine is in configuration 4, there is only one setup worker and we label the operations by jo, where $j \in \mathcal{J}, o \in \mathcal{O}$. All the machines and the worker are available from $t = 0$. The directed graph in Figure 3 shows the precedence relationships. X and Y are two dummy nodes denoting the source and sink, respectively, so that for each job there exist a path going from X to Y. The set of directed arcs states the ordered pairs of operations.

Job	Operation	Available Machines(pt)		Mold
J_1	11	M_1(3)	M_2(3)	2
	12	M_1(4)	-	1
	13	M_3(2)	-	3
J_2	21	M_1(2)	M_2(2)	1
	22	M_2(2)	-	3

Toy Instance - pt means processing time

Configuration		To		
		1	2	3
From	1	0	1	0.5
	2	0.5	0	0.5
	3	0.5	1	0
	4	0.5	1	0.5

Toy Instance - time for setup configuration

Figure 2: Example of one scheduling problem.

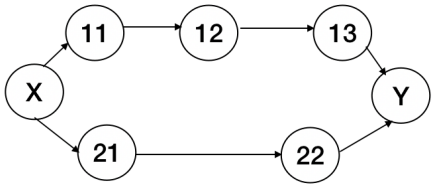

Figure 3: Directed graph representation of one feasible chromosome.

Given the above instance, a chromosome is composed of five operations, and could be equal to the list shown in Figure 4 (left side). In the middle, instead, we report a possible assignment of machines to operations. In particular, for an operation o, the earliest starting time on each machine is calculated based on the finish time of its predecessors and the available time of setup worker. Finally, on the right side we update the graph already shown in Figure 3 by adding the arcs modeling the operation precedence on the same machine (dotted lines).

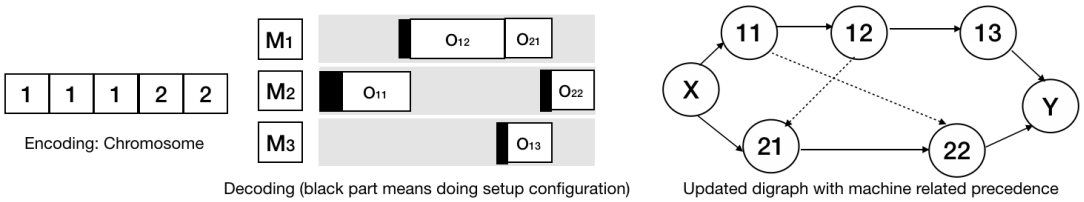

Figure 4: Encoding, decoding, and directed graph with machine precedence relationships.

3.1.3 Genetic operators

The initial population is composed of a set of those chromosomes with randomness in operation sequence (for operations in directed arcs). For each solution, genetic operators - selection, crossover and mutation come into place. To guarantee the feasibility, there is one check for each solution and one correction when it is discovered infeasible.

In each generation of the reproduction, selectors pick up individuals among the population to generate offsprings and the survivals, in general, depending on the fitness values. In our implementation, the tournament selector (selecting the best individual from random samples with replacement) is used for selecting survivors and roulette wheel selector (selecting according to the fitness proportion) is for selecting the individuals to create offsprings.

A general crossover operator of GA operates on two parents' strings at a time and generates offsprings by recombining both parent strings' features. This operator needs to preserve job sequence characteristics of the parent's string in the next generation. In our application we use two point crossover which randomly chooses two points in parents and swap the area between the two points. The infeasible children are generated from time to time, therefore each newborn is checked and fixed if it is infeasible.

The left side of the Figure 5 shows one crossover example, which generates two feasible children. On the right side, which is for the same problem shown in Figure 2, it shows one example of infeasible solution to illustrate the way to fix infeasibility.

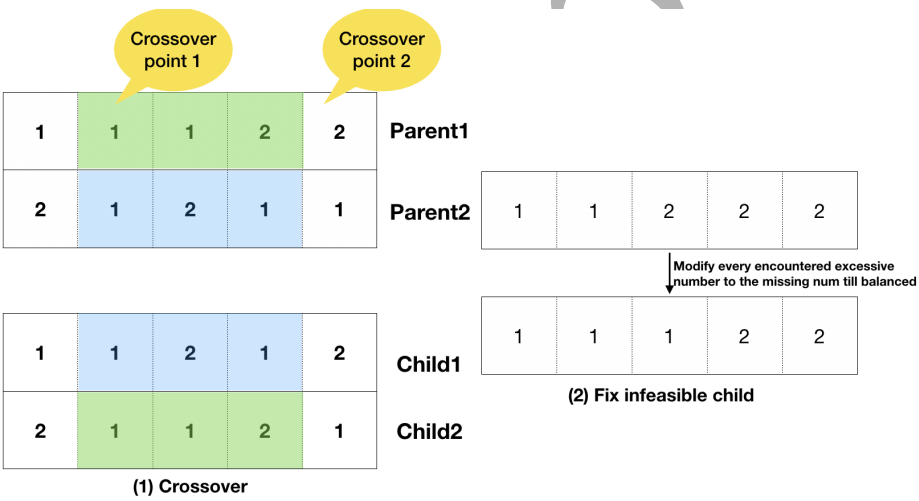

Figure 5: Crossover and infeasibility handling.

To avoid spending much time in managing feasibility rather than exploring for better solutions, a swapping mutation strategy is utilized. The approach is straightforward, randomly taking two positions in the recombined chromosome, then swapping genes on the two positions to obtain new offspring. Since we use job number to represent genes, the newborns are feasible on any swap.

3.1.4 Tabu Search

On the TS side, the main components include move, neighborhood structure, tabu list and aspiration criteria.

A neighborhood structure is a mapping from a solution to a set of neighbors (a neighbor is a solution with slight difference from the original one). In Van Laarhoven, Aarts, and Lenstra (1992), the authors propose the first successful neighborhood structure for JSP. The structure is constructed by reversing the order of two successive operations on the same machine. A move is the modification on a solution to get a neighbor. The reversing transition is a type of moves. The swapping move strategy exchanging any two operations on different jobs is adopted in this paper. With the intrinsic meaning of "tabu", forbidden, the tabu list is a memory structure recording the recent moves to avoid the solution cycle. In the presented work, the positions of the two operations in a move are recorded as an element in the tabu list. The list is cyclic with a fixed capacity, which means the oldest element will be removed when a new element

needs to be inserted but the full capacity is reached. As elaborated by Zäpfel, Braune, and Bögl (2010), TS excels at avoiding getting stuck in local optima with the usage of tabu list. However, it is inevitable to consider more for balancing intensification (exploring best neighbors) and diversification (disallowing the moves annotated as tabu) based on the length of tabu list. It is a common phenomenon that no fixed value is qualified for all problems.

While the advantage of tabu list is demonstrated, it exhibits the possibility on forbidding some solutions, which are discovered by applying the tabu move, being visited. To mitigate the risk, we accept the widely used aspiration criteria: accepting a tabu move which creates better solution than what is found so far.

If the system is allowed to run with more generations, it is likely to transform into better solutions especially for the difficult problems, however there should be a tradeoff between running time of TS and that of GA in HA. The best solution found by TS is recorded and encoded into a chromosome, then sent back to GA population.

4 Classification for rescheduling decisions

As stated above the decision of the time step to reschedule is very difficult. For this reason, we develop a classification algorithm that given information related to the topology and the present state of the production system, it returns the suggestion if it is better to reschedule or not.

Such an approach is useful for several reasons: first, the classifier needs short time to provide the result of the computation, second, it requires a small amount of computation power, third, since it will provide answer in a small amount of time, it can be run with high frequency and hence it can be really responsive. Finally, by using the proposed methodology, it is possible to know the characteristics of the plant that play heavily in deciding the necessity of a rescheduling. Thus, it is possible to modify the plant in order to improve its robustness, reduce bottleneck and etc.

Let us consider a set of scenarios $\Theta = \{1, 2, \ldots, \theta_{max}\}$. In each scenario $\theta \in \Theta$, the jobs that the plant has to fulfill, the associated operations and the number of machine are changed. In the following, we use the notation $u(\theta) = (u_1, \ldots, u_T)$ to indicate the schedules of the plant in scenario $\theta \in \Theta$. Given two different schedules $u(\theta)$ and $v(\theta)$, if the production follows $u(\theta)$ in $[0, t]$ and $v(\theta)$ in $[t+1, T]$, in order to indicate the concatenation of the two schedules, we use the notation $< u(\theta), v(\theta) >_t$. Given a schedule $u(\theta)$ and a scenario $\theta \in \Theta$, we call the operator $\mathcal{F}(u, \theta)$ the computed makespan.

For each scenario $\theta \in \Theta$, we define the processing time variations $\delta_{11}(\theta), \ldots, \delta_{|\mathcal{M}|1}(\theta), \ldots, \delta_{1|\mathcal{T}|}(\theta), \ldots, \delta_{|\mathcal{M}||\mathcal{T}|}(\theta)$. The interpretation of $\delta_{mt}(\theta)$ is the following: given an operation o that on machine m lasts for T_{om}, if scenario θ occurs, it lasts $T_{om}(1 + \delta_{mt}(\theta))$. These variations can be positive (under-estimation of processing time) with 70% possibility, negative (over-estimation of processing time) with 30% possibility. It is worth noting that the random variables $\delta_{mt}(\theta)$ are independent from machine to machine, from time step to time step, and they are independent from the scheduling (see Li, Shyu, and Adiga 1993). Furthermore, we assume that their expected value is zero. Please notice that this is not a restrictive hypothesis because if the decision maker knows that on average some process is longer than its expectation, then its expectation is updated.

In order to compute the data set to train classifier, for each scenario θ we compute the *actual schedule* $u(\theta)$ i.e. the schedule that the plant is supposed to follow by means of the heuristic described in Section 3.1. Then, for each time step t of scenario θ we run the rescheduling procedure to get a new schedule $u(\theta|t)$. The new schedule is computed by using the heuristic proposed in Section 3.1, with current schedule $u(\theta)$ as starting solution, over the following

optimization model:

$$\text{minimize } \lambda C_{max} + (1-\lambda) N_{var} \tag{22}$$

subject to (2)–(21). In the modified objective function (22), λ is the relative importance between the original C_{max} and the l_1 norm of the difference between the solution of the actual schedule and the rescheduled solution (N_{var}). The latter term is useful in order to limit the number of changes of the scheduling that do not improve the make-span.

Given schedule $u(\theta|t)$, threshold b, if

$$\mathcal{F}(< u(\theta), u(\theta|t) >_t, \theta) \leq (1-b)\mathcal{F}(u(\theta), \theta), \tag{23}$$

then it is better to reschedule (label 1) and $u(\theta)$ is updated with $< u(\theta), u(\theta|t) >_t$, otherwise no (label 0). We consider a threshold of 5% of make-span decrements in Eq. (23) in order to take into account for the time spent in re-organizing the production and to avoid rescheduling if only a little improvement can be achieved. Notice that, the threshold can be adjusted according to the actual requirement of each production manager. Under same conditions, the higher the threshold is, the less frequently being rescheduled. So for the productions which are difficult to reschedule often, a higher threshold can be set accordingly.

By using the aforementioned procedure, we get a set of plant states, one for each couple $(t, \theta) \in \mathcal{T} \times \Theta$, and for each of them we associate it with the label (reschedule or do not reschedule). From the dataset, we extract, for each state, a set of features taking into account the information about the processing time variations (PTV), the planned scheduling and the plant information (including the available resources for each operation, customer orders to be produced in the schedule and etc).

It is worth noting that all the simulations are assumed to have equal time horizon $\mathcal{T}$. This is not a restrictive hypothesis since we can always consider $\mathcal{T}$ as the ending time of the longest simulation.

Following the approach of Fadda et al. (2019), we do not consider automatic feature extraction. Instead, we exploited the experience of the involved company SIGIT to define the following set of features:

- t: the time step of the simulation,
- a set of features which are operation dependent:
  - OPT_o remaining processing time at time step t of operation o,
  - PTV_o processing time variation at time step t of the operation o,
  - ρ_o ratio of the available machines able to perform the operation o at time t.

Except for the first feature, all the remaining ones depend on the operation o. Since considering all operations may lead to over-fitting (the number of operations is greater than the number of scenarios), with the ratio defined as an indicator, only the operations with the high ratios which mean the operations with more flexibility in changing machines are considered. We call the number of considered operations OP_Num. The performance of the classifiers with respect to OP_Num will be analyzed in the section 5. When OP_Num is 2, the considered feature set will be $\{t,\ OPT\_0,\ PTV\_0,\ Ratio\_0,\ OPT\_1,\ PTV\_1,$
$Ratio\_1\}$.

After getting the aforementioned features, the resulting dataset is divided into training dataset (70%) and test dataset (30%). For data mining, there is no unique classification algorithm which is best for all types of data. Thus, we consider three commonly used techniques: random forest classifier which belongs to decision tree induction methods, multilayer perceptrons from neural networks and support vector machines (Kesavaraj and Sukumaran 2013):

1. Random Forest Classifier (RFC): a combination of decision tree classifiers and the ensemble of trees voting for the most popular class (Breiman 1999). RFC is easy to parametrize, not sensitive to over-fitting and it provides ancillary information like variable importance (Horning et al. 2010). However, a large size of data set can lead to high memory consumption (Santur, Karaköse, and Akin 2016).

2. Support Vector Machine (SVM): input vectors are mapped to high dimensional feature space and a linear decision surface is constructed in the space (Cortes and Vapnik 1995). It does not require any parameter tuning since it can find good parameter settings automatically (Joachims 1998). And it delivers a unique solution because the optimization problem is convex. However the feature of non-parametric brings convenience, it lacks the transparency of results (Auria and Moro 2008).

3. Multilayer Perceptrons (MLP): a type of neural network, which simulates human brains (Pal and Mitra 1992). It is a system of interconnected neurons, or nodes representing a nonlinear mapping between an input vector and an output vector. The algorithm works well for simple problems, but for difficult problems, several iterations are needed for the training convergence (see Singhal and Wu 1989). It shows one benefit that no need of the prior assumptions about distribution of training data nor the decision regarding the relative importance of input measurements. While the costs spent in deciding the number of layers and the number of nodes in those layers is not trivial and there is no single method for doing it (see Gardner and Dorling 1998).

We do not use more advanced techniques like deep learning, convolution neural network, etc. because in this work, our goal is to provide really simple techniques (we postpone the study of those techniques in future work). Moreover, we do not use data clustering techniques (such as the one used in Cuzzocrea et al. 2019) because in this setting they perform really poorly. Finally, we consider those techniques because they provide the user insights with respect to the features considered.

Additionally, as pointed out by Larson (1931) in the early 30s, using same data for training algorithm and evaluating performance leads to overoptimistic result. For selecting the validation model. we choose Cross-Validation (CV) technique which overcomes over fitting (see Arlot, Celisse et al. 2010).

5 Numerical Experiments

In this Section, we present the instance generation procedure and then discuss the experimental results carried out. The approach has been developed during the project P&R4.0.

5.1 Instances generation

The problem instances have been generated by using a general method to build all the sets, operators and parameters described in section 3. In particular, in all the section, we use it in order to model the plant of the company SIGIT. The plant consists of two product lines, one for molded rubber and the other one for plastic items. In the paper, only rubber line is considered. The line consists of 16 machines, all the jobs are composed by successive operations, i.e., there are no two operations of the same job that can be executed in parallel. Furthermore, just one worker is capable of doing the setup operation. Thus, each new setup operation required by a given operation needs to wait for completing setup of the previous one.

The empirical distribution of the PTV used is shown in Figure 6. As readers can notice,

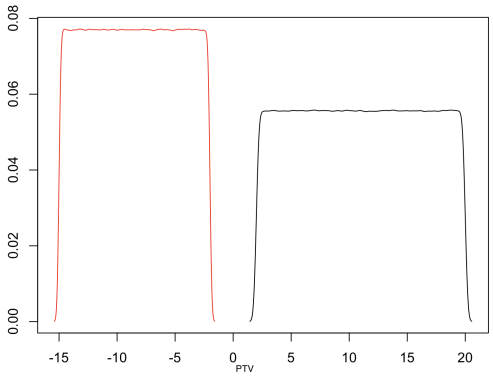

Figure 6: PTV distribution.

the maximum increment of processing time variation is 20% of the planned time while the maximum reduction is the 15%.

In the training phase we consider 23 scenarios. For each of them, the number of operations are simulated from 2 to 41, with processing time varied from 3 to 25 hours and available machine quantity from 1 to 14.

5.2 Implementation details

As described in Section 3.1, the first schedule is obtained by using a hybrid heuristic consisting of GA and TS. The GA part is built on the open source programming library *Jenetics* (see Wilhelmstötter 2016), while the TS part has been implemented based on the open source programming library *OpenTS* (see Robert 2019). By calibrating the parameters, the population size is set 200 for both GA only and HA. For the operators crossover and mutator, in HA, a two-point crossover with probability 0.86 and a swap mutator with possibility 0.3 are used; in GA only, two-point crossover with 0.76, swap mutator with 0.115 are adopted. And the tabu length is calibrated into 30 for TS only approach, and 20 for HA approach. For each individual in HA, TS is set to iterate 50 times as a stopping criterion to explore better solutions.

The machine learning procedure has been implemented by using the package *scikit-learn* (Pedregosa et al. 2011) in python 3.6. The machine used for the numerical experiments is equipped with an *Intel(R) Core(TM)* i5 CPU@2.3GHz, 8 GB RAM and running *macOS* v10.14.3. The MILP solver used in the numerical experiment is GUROBI Optimizer v8.1.0 (build v8.1.0rc1).

5.3 Results and discussion

The proposed approach is composed by several parts, therefore the experiments are divided into three parts. In subsection 5.3.1, it compares the performance of the heuristics implemented for solving (1)-(21), in subsection, 5.3.2 the characteristics of the classification problem being defined are analyzed and next, in subsection 5.3.3, the performance of the proposed approach compared with a periodic rescheduling is elaborated, proving its effectiveness.

5.3.1 Heuristic Performance

To test the efficacy of the method, the authors compare the percentage gaps of heuristics against the solutions provided by exact solver Gurobi (ES). The results are shown in Table 1, which includes two types of gap comparisons: the differences of makespans under same running time and the distinctions of heuristic makespans from the best achievable values of ES. Up to a certain extent, longer computation time results in better performance. To test the performance

with a wide computational time range, several time intervals are considered under the fact that running time is increased by the difficulty level of the problem. The first column shows the time ranges in seconds for running the instances. The second and third column provides the number of machines and of operations considered, respectively. Note that same machine number and operation number may represent different instances because of other varied feature values may exist (for example, the duration of each operation, the precedence and etc.). The next 3 columns: GA_gap, TS_gap, HA_gap respectively show their gaps (%) with the solutions supplied by ES for same instance under equivalent running time. Then $best\_T$ presents the running time in seconds for getting the best objective value by ES. Correspondingly GA_bgap, TS_bgap, HA_bgap separately show the differences of GA, TS and HA with same running time indicated in column $best\_T$. Note that $n.a.$ means not available, which appears when ES is not able to provide a solution for an instance under the corresponding computation time.

T(s)	$\|\mathcal{M}\|$	$\|\mathcal{O}\|$	**GA_gap**	**TS_gap**	**HA_gap**	**best_T(s)**	**GA_bgap**	**TS_bgap**	**HA_bgap**
[20, 40)	3	10	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
	5	10	-6.12	-6.12	-6.12	4002	2.22	2.22	2.22
	5	10	n.a.	n.a.	n.a.	4265	8.00	8.00	8.57
	5	10	n.a.	n.a.	n.a.	3905	1.00	1.00	1.49
	3	5	-10.11	-10.11	-10.11	3769	1.27	1.27	1.27
[40, 220)	4	20	-8.7	-8.70	-8.7	3656	0.00	0.00	0.00
	2	30	-3.85	-3.85	-3.85	3769	0.00	0.00	0.00
	4	30	n.a.	n.a.	n.a.	3324	2.00	2.00	2.56
	4	30	-46.94	-46.94	-46.94	3989	0.00	0.00	0.00
	4	30	-20.41	-22.45	-22.45	2765	8.33	5.56	5.56
[220, 400)	6	30	n.a.	n.a.	n.a.	4324	5.00	5.00	5.26
	6	30	-26.32	-26.32	-26.32	2965	0.00	0.00	0.00
	6	30	5.56	5.56	5.56	4297	11.76	11.76	11.76
	4	35	-34.69	-34.69	-34.69	2987	6.67	6.67	6.67
	4	30	n.a.	n.a.	n.a.	2658	3.00	3.00	0.00
[400, 2200)	4	40	n.a.	n.a.	n.a.	3456	0.00	0.00	0.00
	6	40	-48.59	-48.98	-48.98	3406	4.17	4.17	4.17
	6	50	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
	6	50	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
	6	60	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
AVG			-20.06	-20.26	-20.06		3.51	3.17	3.10
STD			17.76	17.77	17.76		3.57	3.36	3.48

Table 1: Comparing GA, TS and HA with ES

When running with same time, the recurrent appearance of negative values and $n.a.$ in columns GA_gap, TS_gap, HA_gap indicates all the three heuristics have significantly better performance than ES. The statistics in the last two rows AVG and STD strengthen the belief in the heuristics effectiveness.

With much bigger running time enabled in ES only, it gets improved results in most instances. For heuristics, keeping the running time shown in the first column, their performance is surpassed by Gurobi with small differences. It is observed that HA stays a bit beforehand comparing to other 2 heuristics. However, due to the limitation of ES, the makespans of the instances tested in Table 1 fall into the range from 17–120 time units. To compare in greater scales, the instances with longer operation durations (20–50 time units), bigger makespans (190–330 time units) are tested. In real and dynamic production, fast enough to get a feasible schedule is important, so the time values 20s, 40s, 95s is chosen for comparing the performance. Since exact solver is not able to provide solutions, HA is used as a

benchmark to calculate the gaps(%), as shown in Table 2. The bigger the gap is, the bigger the makespan got from either GA or TS.

$\|\mathcal{M}\|$	$\|\mathcal{O}\|$	**T(s)**	**GA_gap**	**TS_gap**
3	6	20	0.21	0.00
		40	0.00	0.00
		95	0.00	0.00
6	14	20	2.48	0.00
		40	1.65	0.00
		95	1.92	0.00
6	23	20	1.48	0.29
		40	1.00	0.04
		95	0.57	-0.12
6	34	20	5.66	2.77
		40	5.98	2.82
		95	3.25	1.19
AVG			2.02	0.58
STD			1.95	1.09

Table 2: Comparisons among GA, TS and HA

With most frequent appearance of non negative values shown in Table 2, HA demonstrates its good performance. HA outperforms both GA and TS, which is contributed by its mixture strategy in exploration and exploitation. As a result, HA is chosen for getting the initial schedule. TS achieves similar results compared to HA with a bit worsen quality. With the feature of neighbor exploration which means more discovery of similar solutions, and its satisfying quality, TS is chosen for rescheduling to get a better but are alike to initial one to avoid many production changes.

5.3.2 Classification Analysis

In this section, the performance of the proposed approach is analyzed from the results of different classification algorithms. As for performance estimator, the area under the receiver operating characteristic curve (AUC) is adopted due to the fact that it exhibits a set of more desirable properties comparing to overall accuracy (see Bradley 1997 and Wald and Bestwick 2014). The value of AUC ranges from 0.5 (useless test) to 1 (perfectly discriminated test). The higher the value is, the better the classifier is at distinguishing between the two classes.

The first test compares the performance of SVM, RFC and MLP with different amount of operations related information. OP_Num indicates the number of operations collected. The outcome is averaged by taking results from 10 different random seeds (different seed leads to the different RFC and MLP fitting behavior, which likely causes different scores). In Figure 7, it presents the average AUC values and standard deviations for a set of OP_Num ranging from 1 to 10. When OP_Num is 1, besides the time step, only the features of the operation with the highest ratio will be considered as discussed in section 4.

As readers can notice, RFC stays far ahead. For SVM, firstly AUC score grows but declines after reaching OP_Num 8. Instead, for MLP, the score keeps decreasing with some exceptions in the middle. As for RFC, its AUC values keep improving as OP_Num increases. Thus, the best AUC score 0.81 is achieved by the RFC considering operations with 10 highest ratios. For this reason, in the following subsection RFC and this setting are considered.

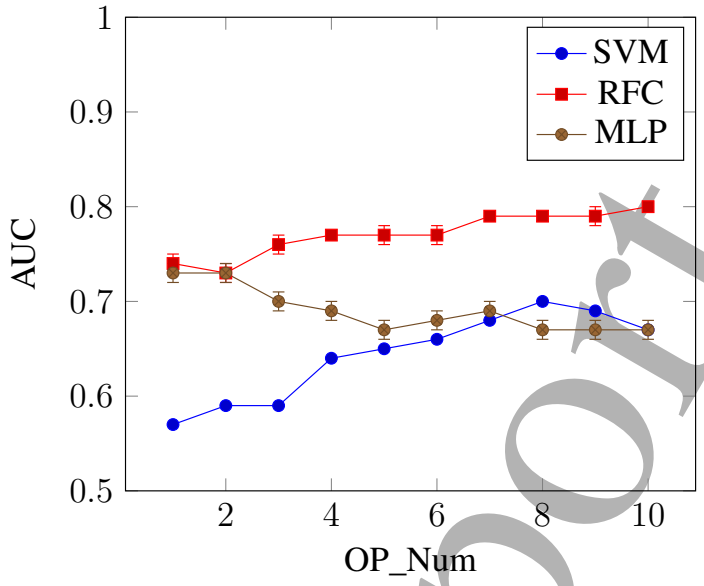

Figure 7: AUC values for different operation numbers.

5.3.3 Rescheduling Performance

This subsection compares the proposed approach (ML) with the different periodical rescheduling actions. The work flow of the rescheduling based on ML is pseudo-coded in Algorithm 1. As competitor, we consider rescheduling actions at every time interval (P-1), every 2 time intervals (P-2), every 4 time intervals (P-4), every 7 time intervals (P-7), and every 10 time intervals (P-10).

Algorithm 1 Rescheduling procedure

```
for θ ← 1 to θ_max do
    for t ← 1 to t_max do
        for m ← 1 to m_max do
            for o ← 1 to o_max do
                if o is in process by m then
                    T_om = T_om(1 + δ_mt(θ));
                end if
            end for
            collect feature values;
            send to ML classifier to get prediction;
            if prediction is 1 then
                reschedule;
            end if
        end for
    end for
end for
```

It is worth noting that due to their simplicity in rescheduling rules, the procedure to reschedule at fixed time intervals is often used in practice. Especially, according to our knowledge, several companies with 3 work shifts per day are tend to reschedule every 8 hours (i.e. at the beginning of each work shift). When I4.0 is not applied, with this policy, company gives to workers the mansions at the beginning of the shift and the workers stick with the plan until the end of their shift. In contrast, with I4.0 some companies provide workers wearable

devices which creates the possibility to communicate in a fast and effective way. Thus offering the potential to have real-time information to update the work without paying a real cost for reorganizing. For this reason, it is not considered to add a penalty when a rescheduling is considered.

The same oscillation values are added to the same scenario when recording the results of ML and periodical ones. Given the time interval of $T = 2$ time units and $\theta_{max} = 9$, the periodic approach with 1, 2, 4, 7 and 10 rescheduling time intervals and the ML rescheduling policies presented in Section 4 are tested. By implementing the procedures presented in Algorithm 1, the statistics at each time interval are collected till scheduling reaches the value of the originally planned makespan. For example, if a schedule is estimated to complete in 100 time units, the production will be simulated with 100 time units as time horizon.

Table 3 displays the statistics on the different approaches. Column $Approach$ shows the rescheduling policy, column N presents the rescheduling times, and column $avgTI$ indicates the average makespan improvement (i.e., how much production time is saved by averaging the improvements of all the rescheduling occurrences), and the last column, $stdTI$, presents the standard deviation of the makespan improvements. More precisely, let us define $n(\theta)$ as the times in which a rescheduling is performed in scenario θ. Then, the total rescheduling number is defined as

$$N = \sum_{\theta=1}^{\theta_{max}} n(\theta) \ \ \forall \, \theta \ \in \ \Theta. \tag{24}$$

Moreover, the total makespan improvement TI is defined as

$$TI = \sum_{r=1}^{N} I_r \tag{25}$$

where I_r is the makespan improvement achieved at each rescheduling occurrence.

As shown in the table, with the highest rescheduling frequency, P-1 gets a significantly small average makespan value. To notice with the assumption 1 time unit is 1 hour, this technique is equivalent to reschedule every 2 hours, thus it is not suitable to adopt in the real field (many operations can last longer than 2 hours). On the contrary, P-10 only reschedules 19 times. And it gets a highest average makespan improvement (18.63%). In fact, rescheduling less frequently creates a more extensive growing space than rescheduling with high frequency. Comparing ML and P-4, ML reschedules much less but it gets remarkably bigger makespan improvements. When considering ML and P-7, P-7 shows a small advantage in average value but with a bigger standard deviation.

Approach	N	$avgTI$	$stdTI$
ML	33	13.52	7.60
P-1	235	2.92	3.95
P-2	115	4.87	5.58
P-4	56	7.73	8.65
P-7	31	13.55	10.54
P-10	19	18.63	14.32

Table 3: Comparison in makespan improvements

We point out that, despite of the usage of modern technologies, the less rescheduling the better because any communication can fail for various reasons (workers might miss messages, misunderstand, lose time for understanding the message, and etc.).

The environment where the algorithm runs is dynamic. The ability to handle unexpected events plays an important role in stabilizing the production. Thinking about one example:

when new but urgent orders arrive, how to mitigate the disruptions to the ongoing schedule, which means guaranteeing maximally to deliver in time? It is essential to adopt a representative performance indicator - the makespan of the final time step (representing the effective ending of the planned scheduling, which shows how far the disturbed ones are from what have been expected).

We now investigate the differences in terms of makespan between the different approaches. For scenario $\theta \in \Theta$, C_θ is defined as the normalized remaining makespan (percentage value) at the last time step and D is the makespan difference with the ML approach, which is calculated in Eq. (26).

$$D_\theta = C_\theta - C_\theta^{ML} \ \forall \, \theta \ \in \Theta \tag{26}$$

Then for all the scenarios, TD is defined as the aggregated differences as in Eq. (27)

$$TD = \sum_{\theta=1}^{\theta_{max}} D_\theta \ \forall \, \theta \ \in \Theta \tag{27}$$

With the aforementioned methods on each scenario, both the average makespan difference (%) - $avg\,TD$ and the standard deviation (%) - $std\,TD$ are calculated. The results are shown in Table 4.

Approach	$avgTD$	$stdTD$
P-1	1.56	57.01
P-2	42.63	97.32
P-4	24.44	70.82
P-7	1.33	36.21
P-10	21.44	63.66

Table 4: Difference in final makespan

As shown in Table 4, the figures are all positive indicating all periodical solutions have bigger remaining makespans than those of ML approach. Therefore periodical ones mostly finish the schedules later than ML. P-1 and P-7 reach on average, the closest makespan values to ML. However, as stated before, P-1 is not a good approach in practice because its frequent rescheduling leads to unstable production and it is potentially wasting resources. The standard deviations are big because the tested instances are quite diversified in terms of operation quantity, machine quantity and processing time.

It can be found that although P-4 reschedules more frequently than P-10, it does not present advantage in reducing makespan values. Hence it is inferred more rescheduling is not certainly necessary in every scenario. Among all of them, P-7 is most competitive with ML approach.

Finally, we investigate the differences in terms of makespan not just at the end of time horizon but also during all the time interval. Thus, we collect and compare the normalized makespan difference [%] at each time step. Thus, we compute the difference by considering ML as the benchmark as the previous Table 4.

Similar to the calculation of D_θ, the difference is that each time step is counted in current case. Given scenario $\theta \in \Theta$ and a set of time steps $\mathcal{T}$, $M_{t\theta}$ is defined as the normalized remaining makespan (percentage) at the time step $t \in \mathcal{T}$, $D_{t\theta}$ is the makespan difference by comparing each approach with the ML approach at same time step t, which is calculated in Eq. (28):

$$D_{t\theta} = C_{t\theta} - C_{t\theta}^{ML} \ \forall \, \theta \ \in \Theta t \in \mathcal{T} \tag{28}$$

Then for all the time steps, MTD is defined as the aggregated differences:

$$MTD = \sum_{\theta=1}^{\theta_{max}} \sum_{t \in \mathcal{T}} D_{t\theta} \quad \forall\, \theta \ \in \Theta \tag{29}$$

After calculating TD^*, the averages and standard deviations follow the conventional calculation methods and the results are listed in Table 5.

Approach	$avgMTD$	$stdMTD[\%])$
P-1	14.08	42.44
P-2	13.69	52.45
P-4	8.61	38.85
P-7	17.36	65.96
P-10	8.99	40.35

Table 5: Difference in the makespan at each time step

Table 5 shows all the periodical approaches have greater makespans than ML, which proves the efficacy of ML in the ongoing production. Among periodical methods, P-4 stands out as having the smallest differences both in average and standard deviation.

Comparing the tables 3, 4 and 5, among periodical solutions, although P-7 shows a biggest makespan gap during the production (in Table 5), it does indicate a good tradeoff in rescheduling frequency and makespan controlling. In general, P-4 behaves fairly in all the aspects, which can be the underlying fact that it is commonly used in the factories. Considering the proposed approach - ML and periodical ones, it is clear to see while ML reschedules in a moderate number, it gets satisfying outcomes not only in saving overall production time, but also in the rescheduling effectiveness, which avoids wasting resources in managing machine and worker changes.

Further analysis is conducted on 2 scenarios. Figure 8 shows the makespan trend on the two scenarios. Table 6 shows the corresponding rescheduling times.

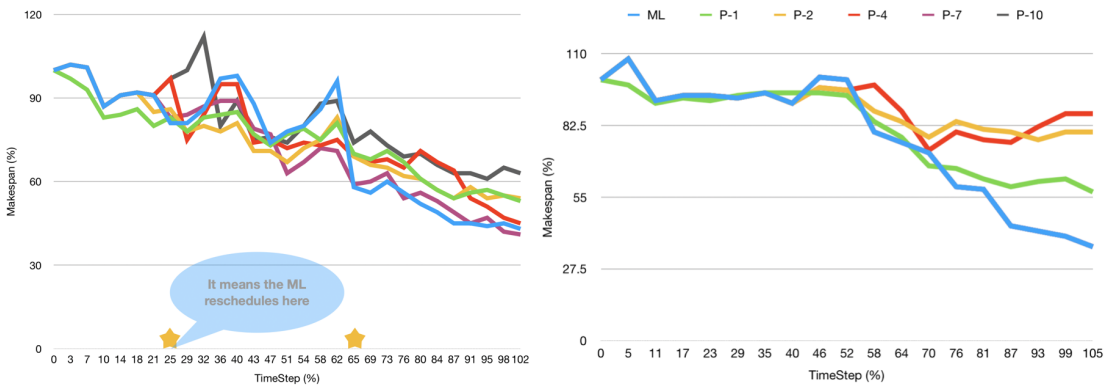

Figure 8: Comparison in makespan trend (left: instance1, right: instance2)

Instance	**ML**	**P-1**	**P-2**	**P-4**	**P-7**	**P-10**
1	2	28	14	7	4	2
2	0	18	9	4	2	1

Table 6: Rescheduling times in each scenario

To note that, both values of time steps and makespans are normalized by the actual makespans and presented in percentage terms.

On the left plot, ML suggests rescheduling twice at around time steps 25 and 65. In general, all the lines are shaking greatly from time step 18 to 69, which may result from the random oscillations added to the schedule. ML outperforms all others except for P-7 by the low makespan at time step 98.

On the right one, P-7, P-10 and ML overlap each other into one blue line. With no any rescheduling, ML gets equivalently best result in makespan. It can be inferred that for some schedules, even there are some disturbances, rescheduling is redundant.

It is necessary to point out that ML approach is adaptable to environmental settings in accordance with its performance, and the parameters including the rescheduling boundary proposed by operational manager.

In addition to that, the planning problem is $\mathcal{NP}-$hard, therefore adequate time to run metaheuristic algorithms is needed. In the continuous manufacturing process, the production status is changing with respect to the passing of time. With ML approach, a rescheduling decision can be made within seconds, and only if the positive decision is made, the actual rescheduling approach will be searched. Not only for the adaptability ML approach owns, but also for the time it saves, ML shows its potential in making better rescheduling decisions.

6 Conclusions and Further Research

In this paper, we have proposed a new way for dealing with rescheduling under the context of I4.0. This work represents the first approach to use machine learning and operations research together in the scheduling field. By means of some computational experiments we prove the efficiency and the effectiveness of these techniques in the real field. The main outcome is the definition of a first set of features that led to a good classifier and the general methodology. Furthermore, another important value for this problem is the formalization of the Flexible Job-Shop Scheduling problem with sequence-dependent setup time and limited resources and the definition of a good heuristic for obtaining qualified solutions.

Future study of this topic will consider more sophisticated machine learning algorithms (such as the one used in computer vision), as well as the definition of an enlarged set of features including deeper information related to the bottleneck of the scheduling as well as the property of the graph G. Finally, the work has proved that it is possible, for a machine learning technique, to learn the performance of a heuristic. This general aspect opens several new research lines about the effectiveness of this approach in other settings.

Acknowledgement

This research was partially supported by the Plastic and Rubber 4.0 (P&R4.0) Research Project, Piattaforma Tecnologica "Fabbrica Intelligente", POR FESR 2014-2020 - Azione I.1b.2.2, funded by Piedmont Region (Italy), Contract No. 319-31. The authors acknowledge all partners of the project for their contribution.

References

Arlot, Sylvain, Alain Celisse, et al. 2010. “A survey of cross-validation procedures for model selection.” *Statistics surveys* 4: 40–79.

Auria, Laura, and Rouslan A Moro. 2008. *Support vector machines (SVM) as a technique for solvency analysis*. DIW Berlin discussion paper.

Azzouz, Ameni, Meriem Ennigrou, and Lamjed Ben Said. 2017. "A hybrid algorithm for flexible job-shop scheduling problem with setup times." *International Journal of Production Management and Engineering* 5 (1): 23–30.

Balas, Egon. 1969. "Machine sequencing via disjunctive graphs: an implicit enumeration algorithm." *Operations research* 17 (6): 941–957.

Bierwirth, Christian, and Dirk C. Mattfeld. 1999. "Production Scheduling and Rescheduling with Genetic Algorithms." *Evolutionary Computation* 7 (1): 1–17.

Bradley, Andrew P. 1997. "The use of the area under the ROC curve in the evaluation of machine learning algorithms." *Pattern recognition* 30 (7): 1145–1159.

Breiman, Leo. 1999. "Random forests." *UC Berkeley TR567* .

Brucker, Peter. 2010. *Scheduling Algorithms*. 5th ed. Springer Publishing Company, Incorporated.

Cortes, Corinna, and Vladimir Vapnik. 1995. "Support-vector networks." *Machine learning* 20 (3): 273–297.

Costa, A, F. A. Cappadonna, and S. Fichera. 2013. "A hybrid genetic algorithm for job sequencing and worker allocation in parallel unrelated machines with sequence-dependent setup times." *The International Journal of Advanced Manufacturing Technology* .

Cuzzocrea, Alfredo, Mohamed Medhat Gaber, Edoardo Fadda, and Giorgio Mario Grasso. 2019. "An innovative framework for supporting big atmospheric data analytics via clustering-based spatio-temporal analysis." *J. Ambient Intelligence and Humanized Computing* 10 (9): 3383–3398. https://doi.org/10.1007/s12652-018-0966-1.

Dolgui, Alexandre, Dmitry Ivanov, Suresh P. Sethi, and Boris Sokolov. 2019. "Scheduling in production, supply chain and Industry 4.0 systems by optimal control: fundamentals, state-of-the-art and applications." *International Journal of Production Research* 57 (2): 411–432.

Šemrov, D., R. Marsetič, M. Žura, L. Todorovski, and A. Srdic. 2016. "Reinforcement learning approach for train rescheduling on a single-track railway." *Transportation Research Part B: Methodological* 86: 250 – 267. http://www.sciencedirect.com/science/article/pii/S0191261516000084.

Fadda, Edoardo, Pino Castrogiovanni, Guido Perboli, and Alessandro Rizzo. 2019. "Smartphone Data Classification Technique for Detecting the Usage of Public or Private Transportation Modes." *IEEE Access* –.

Fadda, Edoardo, Luca Gobbato, Guido Perboli, Mariangela Rosano, and Roberto Tadei. 2018. "Waste Collection in Urban Areas: A Case Study." *Interfaces* 48 (4): 307–322.

Fadda, Edoardo, Guido Perboli, and Roberto Tadei. 2018. "Customized multi-period stochastic assignment problem for social engagement and opportunistic IoT." *Computers & Operations Research* 93: 41–50.

Gao, Liang, CY Peng, C Zhou, and PG Li. 2006. "Solving flexible job shop scheduling problem using general particle swarm optimization." In *Proceedings of the 36th CIE conference on computers & industrial engineering*, 3018–3027.

Gardner, Matt W, and SR Dorling. 1998. "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences." *Atmospheric environment* 32 (14-15): 2627–2636.

Giusti, Riccardo, Daniele Manerba, Giorgio Bruno, and Roberto Tadei. 2019. "Synchromodal logistics: An overview of critical success factors, enabling technologies, and open research issues." *Transportation Research Part E: Logistics and Transportation Review* 129: 92–110. http://www.sciencedirect.com/science/article/pii/S1366554519303928.

Glover, Fred, and Manuel Laguna. 1997. *Tabu Search*. Norwell, MA, USA: Kluwer Academic Publishers.

Gong, Guiliang, Qianwang Deng, Xuran Gong, Wei Liu, and Qinghua Ren. 2018. "A new double flexible job-shop scheduling problem integrating processing time, green production, and human factor indicators." *Journal of cleaner production* 174: 560–576.

Gunes, Volkan, Steffen Peter, Tony Givargis, and Frank Vahid. 2014. "A survey on concepts, applications, and challenges in cyber-physical systems." *KSII Transactions on Internet & Information Systems* 8 (12).

Gupta, Dhruv, Christos T Maravelias, and John M Wassick. 2016. "From rescheduling to online scheduling." *Chemical Engineering Research and Design* 116: 83–97.

Horning, Ned, et al. 2010. "Random Forests: An algorithm for image classification and generation of continuous fields data sets." In *Proceedings of the International Conference on Geoinformatics for Spatial Infrastructure Development in Earth and Allied Sciences, Osaka, Japan*, Vol. 911.

Joachims, Thorsten. 1998. "Text categorization with support vector machines: Learning with many relevant features." In *European conference on machine learning*, 137–142. Springer.

Kesavaraj, Gopalan, and Sreekumar Sukumaran. 2013. "A study on classification techniques in data mining." In *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, 1–7. IEEE.

Larsen, Rune, and Marco Pranzo. 2019. "A framework for dynamic rescheduling problems." *International Journal of Production Research* 57 (1): 16–33.

Larson, Selmer C. 1931. "The shrinkage of the coefficient of multiple correlation." *Journal of Educational Psychology* 22 (1): 45.

Lee, Jay, Behrad Bagheri, and Hung-An Kao. 2015. "A cyber-physical systems architecture for industry 4.0-based manufacturing systems." *Manufacturing letters* 3: 18–23.

Lee, JuneHyuck, Sang Noh, Hyun-Jung Kim, and Yong-Shin Kang. 2018. "Implementation of cyber-physical production systems for quality prediction and operation control in metal casting." *Sensors* 18 (5): 1428.

Li, Rong-Kwei, Yu-Tang Shyu, and Sadashiv Adiga. 1993. "A heuristic rescheduling algorithm for computer-based production scheduling systems." *The International Journal of Production Research* 31 (8): 1815–1826.

McKinsey. 2015. "Industry 4.0 How to navigate digitization of the manufacturing sector." Accessed 2019-05-26. http://www.forschungsnetzwerk.at/downloadpub/mck_industry_40_report.pdf.

Meeran, S, and MS Morshed. 2012. "A hybrid genetic tabu search algorithm for solving job shop scheduling problems: a case study." *Journal of intelligent manufacturing* 23 (4): 1063–1078.

Mitchell, Melanie. 1998. *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press.

Narayanaswami, Sundaravalli, and Narayan Rangaraj. 2011. "Scheduling and Rescheduling of Railway Operations: A Review and Expository Analysis." *Technology Operation Management* 2 (2): 102–122.

Pal, Sankar K, and Sushmita Mitra. 1992. "Multilayer perceptron, fuzzy sets, and classification." *IEEE Transactions on neural networks* 3 (5): 683–697.

Palombarini, Jorge A., Juan Cruz Barsce, and Ernesto C. Martínez. 2014. "Generating Rescheduling Knowledge using Reinforcement Learning in a Cognitive Architecture." *CoRR* abs/1805.04752.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2011. "Scikit-learn: Machine learning in Python." *Journal of machine learning research* 12 (Oct): 2825–2830.

Robert, Harder. 2019. "OpenTS Tutorial." Accessed 2019-07-08. https://www.coin-or.org/Ots/docs/manual.html.

Roshanaei, V, Ahmed Azab, and H ElMaraghy. 2013. "Mathematical modelling and a meta-heuristic for flexible job shop scheduling." *International Journal of Production Research* 51 (20): 6247–6274.

Rossit, Daniel Alejandro, Fernando Tohmé, and Mariano Frutos. 2019. "Industry 4.0: smart scheduling." *International Journal of Production Research* 57 (12): 3802–3813.

Rudtsch, Vinzent, Jürgen Gausemeier, Judith Gesing, Tobias Mittag, and Stefan Peter. 2014. "Pattern-based business model development for cyber-physical production systems." *Procedia CIRP* 25: 313–319.

Santur, Yunus, Mehmet Karaköse, and Erhan Akin. 2016. "Random forest based diagnosis approach for rail fault inspection in railways." In *2016 National Conference on Electrical, Electronics and Biomedical Engineering (ELECO)*, 745–750. IEEE.

Sellers, David W. 1996. "A survey of approaches to the job shop scheduling problem." In *Proceedings of 28th Southeastern Symposium on System Theory*, 396–400. IEEE.

Singhal, Sharad, and Lance Wu. 1989. "Training multilayer perceptrons with the extended Kalman algorithm." In *Advances in neural information processing systems*, 133–140.

Trstenjak, Maja, and Predrag Cosic. 2017. "Process Planning in Industry 4.0 Environment." *Procedia Manufacturing* 11: 1744 – 1750. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy, http://www.sciencedirect.com/science/article/pii/S2351978917305115.

Uhlmann, Iracyanne Retto, and Enzo Morosini Frazzon. 2018. "Production rescheduling review: Opportunities for industrial integration and practical applications." *Journal of Manufacturing Systems* 49: 186 – 193. http://www.sciencedirect.com/science/article/pii/S0278612518304308.

Đurasević, Marko, and Domagoj Jakobović. 2018. "A survey of dispatching rules for the dynamic unrelated machines environment." *Expert Systems with Applications* 113: 555–569.

Van Laarhoven, Peter JM, Emile HL Aarts, and Jan Karel Lenstra. 1992. "Job shop scheduling by simulated annealing." *Operations research* 40 (1): 113–125.

Vieira, Guilherme E., Jeffrey W. Herrmann, and Edward Lin. 2003. "Rescheduling Manufacturing Systems: A Framework of Strategies, Policies, and Methods." *Journal of Scheduling* 6 (1): 39–62.

Wald, NJ, and JP Bestwick. 2014. "Is the area under an ROC curve a valid measure of the performance of a screening or diagnostic test?" *Journal of medical screening* 21 (1): 51–56.

Wilhelmstötter, FRANZ. 2016. "JENETICS Library user's manual." *Califórnia: Autor* .

Zäpfel, Günther, Roland Braune, and Michael Bögl. 2010. *Metaheuristic search concepts: A tutorial with applications to production and logistics*. Springer Science & Business Media.

Zhang, J. 2017. "Review of job shop scheduling research and its new perspectives under Industry 4.0." *Journal of Intelligent Manufacturing* .