Machine Learning and Optimization for Production Rescheduling in Industry 4.0

The International Journal of Advanced Manufacturing Technology (2020) 110:2445–2463
https://doi.org/10.1007/s00170-020-05850-5

ORIGINAL ARTICLE

Machine learning and optimization for production rescheduling in Industry 4.0

Yuanyuan Li[1] · Stefano Carabelli[2] · Edoardo Fadda[2,4] · Daniele Manerba[3] · Roberto Tadei[2] · Olivier Terzo[1]

Received: 2 February 2020 / Accepted: 30 July 2020 / Published online: 9 September 2020

Abstract
Along with the fourth industrial revolution, different tools coming from optimization, Internet of Things, data science, and artificial intelligence fields are creating new opportunities in production management. While manufacturing processes are stochastic and rescheduling decisions need to be made under uncertainty, it is still a complicated task to decide whether a rescheduling is worthwhile, which is often addressed in practice on a greedy basis. To find a tradeoff between rescheduling frequency and the growing accumulation of delays, we propose a rescheduling framework, which integrates machine learning (ML) techniques and optimization algorithms. To prove the effectiveness, we first model a flexible job-shop scheduling problem with sequence-dependent setup and limited dual resources (FJSP) inspired by an industrial application. Then, we solve the scheduling problem through a hybrid metaheuristic approach. We train the ML classification model for identifying rescheduling patterns. Finally, we compare its rescheduling performance with periodical rescheduling approaches. Through observing the simulation results, we find the integration of these techniques can provide a good compromise between rescheduling frequency and scheduling delays. The main contributions of the work are the formalization of the FJSP problem, the development of ad hoc solution methods, and the proposal/validation of an innovative ML and optimization-based framework for supporting rescheduling decisions.

Keywords Industry 4.0 · Flexible job-shop scheduling · Rescheduling · Machine learning classification · Optimization algorithms · Real-time data analysis

1 Introduction

The fourth industrial revolution, or Industry 4.0 (I4.0) for short, allows decision-makers to obtain real-time information from various plant components and machines to communicate with each other. I4.0 can, therefore, be viewed as the application of the Internet of Things (IoT) to industrial production (IIoT).

✉ Daniele Manerba
daniele.manerba@unibs.it

[1] LINKS Foundation, via Pier Carlo Boggio 61, 10138, Turin, Italy

[2] Department of Control and Computer Engineering, Politecnico di Torino, corso Duca degli Abruzzi 24, 10129, Turin, Italy

[3] Department of Information Engineering, Università degli Studi di Brescia, via Branze 38, 25123, Brescia, Italy

[4] ICT for City Logistics and Enterprises Lab, Politecnico di Torino, corso Duca degli Abruzzi 24, 10129, Turin, Italy

The exploitation of new data sources to improve system understanding, as well as its management, is a common trend in several fields (e.g., see [1] for an application in a generic industrial project, [2] in waste collection, [3] in fleet management, and [4] in the gig economy). This trend is even more promising if we observe the number of available smart manufacturing technologies on interconnected equipment, real-time monitoring, data collection with IIoT devices [5–8]. Furthermore, the technologies are still improving, e.g., the future Industrial 5G coverage will reduce latency times significantly [9].

In the scheduling sector, new data sources help to trigger improvements in several aspects: shortening wasted time, improving reliability, shortening setup times, reducing waste, handling exceptions in real-time, and controlling fixing times [10]. In this paper, we focus on the real-time exception management and, in particular, on the problem of determining when rescheduling is worthwhile during the ongoing production. The problem of rescheduling is an essential branch of the scheduling literature [11, 12]. The reasons for rescheduling come from several factors such as

accumulation of delays in production, the unexpected arrival of urgent orders, machine faults, or absence of the operator. To enforce a rescheduling, it is necessary to compute a new schedule balancing the possible time savings and efforts to implement the changes. Although rescheduling frequently helps manage unexpected disturbances, it needs additional working time in reorganization and affects the stability of shop flows. On the contrary, rescheduling too rarely does not eliminate enough a growing accumulation of the delays. Given an optimization technique to create a new schedule, determining the best rescheduling time remains the main problem. If the company receives unforeseen but urgent requests or machine fails, this decision is easy to make (i.e., the rescheduling process should be carried out as soon as possible). In general, rescheduling is required within a manufacturing process if unexpected events arise, leading to unfeasible schedules. However, in the continuous and complex production setting, deciding to reschedule or not quickly and effectively is not a trivial task. The problem is so complicated that in the real setting, many factories just reschedule periodically.

In the view of improving the rescheduling strategy, our paper proposes a new rescheduling framework by combining metaheuristic optimization algorithms and machine learning (ML) techniques. The proposed approach provides empirical evidence of efficiency and effectiveness in the production problems of some Italian companies, within the industrial project *Plastic and Rubber 4.0* (P&R4.0)[1]—a project aimed at being the Italian response to I4.0 for companies in the plastic and rubber processing field. It is essential to highlight that the paper goal is to describe the integration between ML and optimization and to show a comprehensively proof-of-work methodology, but not necessarily to exploit its full potential (which can be achieved only by tailoring the method to the studied setting). For this reason, both the chosen metaheuristics and ML algorithms are not the most advanced ones but selected among mature and popular methods, which have shown excellent performance in the past. This choice also shows the potential of getting better performance by adopting more advanced and tailored algorithms.

The paper is organized as follows. Section 2 reviews the literature on scheduling and rescheduling, highlighting their relationship with I4.0. Section 3 introduces the methodology of the integrated framework. Section 4 presents the scheduling problem and the mathematical model. Section 5 displays the adopted optimization approach. Section 6 demonstrates the creation of classification models. The results of the numerical experiments are shown in Section 7. Finally, we conclude and outline future research lines in Section 8.

2 Literature review

Scheduling is the process of assigning tasks to resources or allocating resources to perform tasks over time. This work focuses on a variation of the job-shop problem (JSP) [13]. Extensive research on JSP methods, including heuristic principles, classical optimization, and artificial intelligence (AI), is reported in [14]. [15] points out that *priority rules* and *dispatching rules* are probably the most frequently used heuristic policies embedded in metaheuristic methods for scheduling problems.

The scheduling problems exist in different manufacturing and service industries with their particularities. In plastic and rubber molding–related fields, the fabrication of injection molds supplies supports to other companies using injected components either as semi-finished products or as final products [16]. Their scheduling often involves complex production systems [17] owing in general to a large number of products, unrelated parallel machines, and sequence-dependent setup times. Such characteristics often match the job-shop scheduling problem ones [18]. Each order or aggregated orders can be seen as a job. For each job, there is a set of ordered activities, and each activity requires the exclusive use of a resource. Although a well-designed schedule is critical to get products delivered on time, the studies on the scheduling problems in plastic and rubber field are still limited. In [19], the authors develop mathematical models for the job-shop scheduling problem with sequence-dependent setup times and solve them through several search methods. In [16], a case on plastic injection molds is studied, and a flexible job-shop scheduling problem is addressed with Petri nets (PN) and genetic algorithms (GA). PN provides a formal representation of the complex system. GA creates a near-optimal schedule to minimize the total weighted tardiness based on the structure provided by PN. The work also provides a clear explanation of the characteristics of plastic injection molds. In [20], the authors describe why the production process in a Belgian rubber company is a job-shop scheduling problem. They solve such a problem through a hybrid shifting bottleneck procedure with a tabu search algorithm. Finally, in [21], a flexible JSP is transformed into a game, which is solved through game theory (GT) approaches. All the jobs and the manufacturer are players trying to maximize their profits. Moreover, each job also aims at minimizing its tardiness while the manufacturer also wants to minimize the makespan of all the jobs.

[1] Plastic&Rubber 4.0. Piattaforma Tecnologica Fabbrica Intelligente (Technological Platform for Intelligent Factory), https://www.regione.piemonte.it/web/temi/fondi-progetti-europei/fondo-europeo-sviluppo-regionale-fesr/ricerca-sviluppo- tecnologico-innovazione/piattaforma-tecnologica-fabbrica-intelligente

In this paper, we formalize the scheduling problem in P&R4.0, namely the flexible job-shop scheduling problem with sequence-dependent setup time and limited dual resources (FJSP), where dual resources mean general-purpose machines and setup workers. And we solve the problem with one of the possibilities—hybrid metaheuristics.

Although the lack of setup workers is a common phenomenon in factories, most researchers do not consider their availability. For example, [22] introduces the issue concerning both the selection of the machine and operation with sequence-dependent setups, without mentioning workers. In the papers [23] and [24], the authors have the same lack. In [25], setup workers are viewed as a critical resource in their single-stage production composed by a set of unrelated parallel machines. In [26], the authors consider setups in a dynamic environment. Specifically, the work deals with a scheduling problem managing a wide variety of products, and an implicit clustering is employed against the impractical building of a large-scaled setup matrix. However, the availability of setup workers is not mentioned. Another research [27] concerns the flexibility of both workers and machines as well as the precedence between operations, but there is no consideration of setup. To our knowledge, FJSP has not been formalized in the literature and no precise heuristics have been suggested as solution methods in the static view, nor for rescheduling in the dynamic setting. Consequently, to provide an integrated rescheduling framework is another contribution of the present paper.

As aforementioned, due to the I4.0 revolution, also scheduling optimization has the opportunity to develop new tools. In [28], the authors present an I4.0 survey on the implementation of optimum control to scheduling in production and supply chain by concentrating on the deterministic maximum principle. Not only do they derive major contributions, application areas, limitations, and research and application recommendations for future research but also they explain control models in industrial engineering and production management. In [13], the author reviews several JSP-related optimization problems applied in I4.0, which shows that one of the most important and active research fields is the application of distributed optimization algorithms. Especially, multi-agent-based systems have been proven to be very effective in several settings (see, e.g., [29]). Moreover, the technique is capable of generating effective schedules for both dynamic and static problem sets, as in [30]. Nonetheless, there is no research done in both articles on the problem of deciding the right rescheduling time. The above discussion testifies a lack in the literature if both I4.0 opportunities and rescheduling problems are considered. In the rest of the section, we focus on the papers considering the rescheduling problem. The work [31] is the first to describe a general model for providing schedules using JSP and GA. This algorithm is evaluated under different situations of workload in a dynamic environment. In [32], the authors are the first to provide well-defined concepts for most rescheduling production systems and to identify a framework for understanding rescheduling approaches, policies, and methods. After that, more rescheduling-related papers appear. [33] and [34] provide critical rescheduling analyses. The former concerns a broad set of operations for railway rescheduling. Even though different algorithms are presented, most of them are problem-specific and cannot be generalized into the context of the smart industry. Instead, by focusing on solutions involving the integration among industries and real application cases, [34] presents a systematic literature review of the studies on rescheduling production. Their paper mainly deals with the choice of the rescheduling heuristic rather than the decision of the rescheduling timing. The lack is common both in rescheduling literature and in the small branch of the literature dealing with rescheduling-specific ML applications. For example, [35] presents an algorithm that uses Q-learning principles to change the train schedules on a single-track railway and in [36] the authors develop an artificial cognition control system to acquire rescheduling knowledge in the form of decision rules. Another work [37] proposes a two-stage teaching-learning-based optimization approach, which avoids considerable modifications for ensuring robust and stable schedules after machine breaks unexpectedly.

In terms of scheduling and rescheduling framework, [38] introduces a general rescheduling framework to address issues arising from the dynamic nature of production scheduling for a classical JSP. The proposed solution consists of a solver that assumes deterministic and static data and a controller that handles uncertainty that triggers a new solution from the solver if the scheduling performance drops down below a certain threshold. Our research is close to their approach to capturing the complex rescheduling properties. However, while the decision-making controller's output depends on when relevant information is gathered in their method, they do not propose the possible integration of real-time data analysis. Another similar work has been proposed [39], which considers optimization scheduling and rescheduling under I4.0 and introduces a new decision-making scheme by using Tolerance Scheduling (identifying scenarios where a given schedule remains acceptable) to mitigate the rescheduling changes in the dynamic environment. They propose to start from defining the disruption events and designing tolerance for parameters, and then incorporate the scheme into a Cyber Physical Production System (CPPS, [40]), which can decide to reschedule only when the objective function value is significantly affected. However, the paper lacks a complete example or case study, which makes the

approach feasibility doubtful in terms of the complexity of implementation and calculation time. In our approach, we validate our rescheduling framework through a scheduling problem and numerical experiments. In [41], the authors propose an event-driven JSP mechanism under machine capacity constraints. The event-driven rescheduling strategy achieved better performance with respect to a periodic rescheduling approach. While our work and [41] both compare with the periodic approach and show superior performance, different rescheduling goals and actions can be identified. Concerning goals, in [41], the rescheduling is done once a dynamic event occurs while minimizing the objective values. Our work intends to balance the big deviation of objective value and energy spent on implementing rescheduling. Concerning actions, in [41] the rescheduling is applied after a disturbance occurs. Instead, our work focuses on combining real-time monitoring and prevention, depending on the various types of possible disturbances. While rescheduling must be done for some unexpected events (e.g., the arrival of urgent orders), some other disturbances that may occur more frequently (e.g., accumulated processing time variations) can be detected through real-time monitoring and integrated with ML to make rescheduling decision.

In [42], the authors propose a decision-making model based on minority game (MG) theory to organize and manage the resources and services provided by the autonomous participants of a cloud manufacturing system with private information. In the game, a set of classes is the agents competing for a group of workstations. Each agent chooses the workstation with maximum availability. The class i allocated on the workstation j wins if the workload of the workstation j is less than limited value. The game stops and gives the allocation when either all the agents win or the number of rounds equal to the limitation. Particularly, if a machine fails, MG reallocates the product classes adding the processing time due to recovery time. In the proposed model, each agent selects resources based on the best agents' score and not the best allocation of the workstations, the simulation results and the low computational complexity prove MG is adequate to solve the resource allocation problem in a system of sharing resources. However, the computation to compare the performance is only based on the workload of workstations. Regarding other objectives (e.g., minimization of makespans, maximum or total tardiness), the performance is not guaranteed. Also, it reschedules after a machine failure without the incorporation of early failure detection or prevention of rescheduling due to other disturbances. A similar lack occurs in other GT-based approaches, such as [43]. Also, in [44], a GT-based approach for self-optimization and learning of modular production units is presented. In the proposed method, each control parameter serves as a player. To avoid long training time and huge data set requirements, appropriate parameters are defined from the basic control level (BCL) to be learned by learning agents. In the learning algorithm, optimal actions for each player have to be inferred from interacting with the environment. However, the experiments focus on energy optimization. In the production scheduling applications, the ability to deliver customer orders in time is of primary importance. The applicability in time-related objectives is still to be validated. Besides, there are several limitations to GT including the fact that each player must know the cost functions of the other players and it is hard to choose when several Nash equilibria exist [45].

To fill in the blanks of the existing frameworks for rescheduling, we deepen the integration of ML and optimization under I4.0 and propose our methodology in detail in the next section.

3 Methodology

In this section, we present the general integration methodology and the specific one implemented in our case study. We recall the reader that the goal of the paper is not to implement the most advanced scheduling and rescheduling strategies or the most advanced ML techniques. Instead, we concentrate on the methodology to integrate various existing methods, which creates new possibilities in the area of rescheduling.

Figure 1 displays the proposed general methodology in a sequential workflow. The main steps are:

1. Analyze and classify the problem of scheduling. The problem may range from a classical JSP to a complex problem as the one proposed in Section 4.
2. Develop techniques for the specific problem, including:
   (a) Optimization algorithms for scheduling and rescheduling. They can be based on either mathematical programming algorithms or any appropriate metaheuristics. Usually, the scheduling algorithm should be used as a strategic plan, while the rescheduling algorithm should be used as a tactical adaptation of the original scheduling. Thus, the rescheduling optimization method has to be extremely fast.
   (b) The ML classification model for identifying rescheduling patterns. To enable the automation of rescheduling, meaning that the system knows when to reschedule without or with minimal human intervention, we use ML classification algorithms to learn from the historical data and

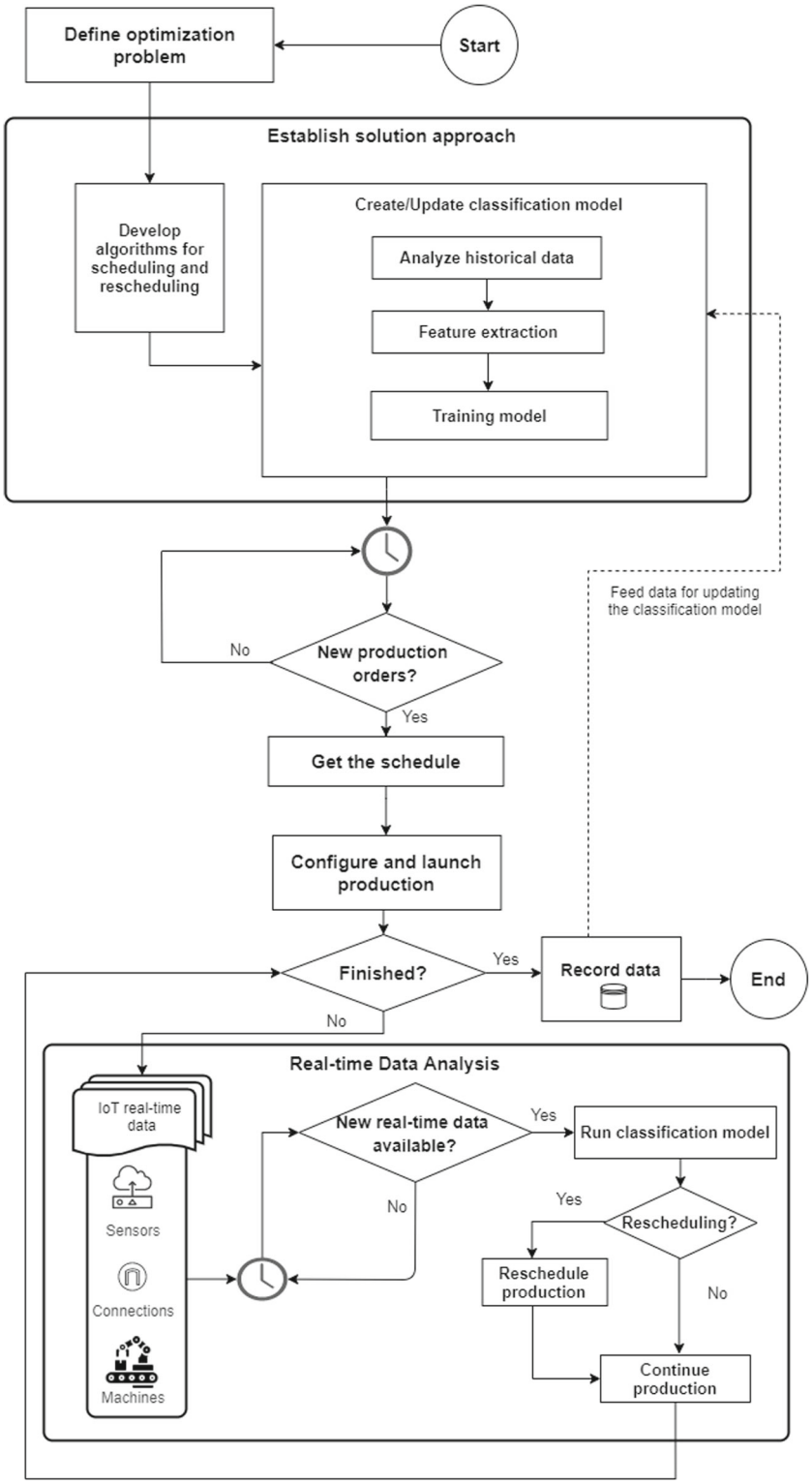

Fig. 1 Graphic representation of the data-driven rescheduling methodology

create a model for future prediction. The features describing operation-related status at each simulation time step are the inputs of the classification model. According to the classification algorithms chosen, each set of input features is mapped to output as the rescheduling decision. An implementation example is described in detail later (see Fig.7, Section 6). The ML strategy can rely on automatic feature extraction or more sophisticated methods.

3. Generate a new schedule periodically from a group of production orders with the predefined optimization algorithm. Then, each production schedule is started with interconnected systems and real-time monitoring. The real-time data are sent to the analytic data algorithms periodically for being translated into features that the classification model can recognize. Then, the model sends the output to the rescheduling controller, suggesting to reschedule or not. If the recommendation is to reschedule, the subsequent rescheduling operation will be taken. Otherwise, the output will be held to completion.
4. Record the data as feedback to update the classification model. Since this is a post-process executed only after finishing the scheduling, it is represented in the figure through a dashed line.

To elucidate the general methodology of the automated rescheduling framework, we implement the following steps for a case study within the P&R4.0 project:

1. Formalize the production scheduling problem.
2. Develop the solution approach for the scheduling and rescheduling problem.
3. Derive features and algorithms for creating a classification model.
4. Demonstrate the potential effectiveness, run the subsequent numerical experiments:
   (a) Implement and test a heuristic approach capable of finding good schedules in a reasonable amount of time;
   (b) Create data to simulate the information from technologies provided in the I4.0 framework;
   (c) Train ML model to learn the rescheduling patterns for deciding when to reschedule (i.e., when to trigger the heuristic for getting a new schedule and then to update the production schedule);
   (d) Compare the performance on the same scenarios followed by the proposed rescheduling framework and the commonly used periodical rescheduling that do not align with ML and real-time data analysis.

4 Problem definition and modeling phase

The optimization problem being considered is the flexible job-shop problem with sequence-dependent setup time and limited dual resources (FJSP). Based on conventional JSP, our FJSP introduces:

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

The key assumptions of the model are:

- No preemption is allowed for each operation, operations between 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 at the start.

The following sets are considered:

- $\mathcal{J} = \{1, 2, \ldots, j_{max}\}$ 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 operations to be done, each operation belonging to a specific job;
- $\mathcal{C} = \{1, 2, \ldots, c_{max}\}$ is the set of configurations;
- $\mathcal{C}_m \subseteq \mathcal{C}$ is the subset of the configurations that the 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 $o \in \mathcal{O}$.

Because each job is a predefined set of operations with a fixed precedence relationship, we define a directed graph $\mathcal{G} = (\mathcal{O}, \mathcal{E} \subseteq \mathcal{O} \times \mathcal{O})$, where $\mathcal{O}$ is the set of nodes and $\mathcal{E}$ is the set of arcs, which enforces the precedence relationships of the operations for the same job (for example, [46]). More specifically, an arc from operation $\tilde{o}$ to operation o means that prior to operation o, operation $\tilde{o}$ must be performed.

The following parameters are also specified:

- $T_{mt}^{c\tilde{c}}$ 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 setup 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 (MILP) formulation for the FJSP is as follows:

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

subject to:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The objective function Eq. 1 aims at minimizing the maximum production makespan. Constraints Eq. 2 ensure the correctness of makespan value by defining it as the maximum of all the completion times. Constraints Eq. 3 impose that each operation must be performed while Eq. 4 enforces that each operation must start in a single time step on only one machine. Constraints Eq. 5 ensure that the right amount of time for each operation is required. In addition, constraints Eq. 6 impose that an operation cannot be executed unless it starts. The constraints Eqs. 7 and 8 enforce the precedence relation between the operations, while constraints Eq. 9 enforce that each machine must have a configuration. Constraints Eq. 10 prohibit a machine from taking a configuration which is not in the set of configurations that it can achieve. Constraints Eq. 11 limit the number of configuration changes that can be made in a given time step. Furthermore, constraints Eqs. 12–17 add the relations between the variables. In particular, constraints Eq. 12 impose that an operation cannot start if the machine is not in the correct configuration, constraints Eq. 13 enforce that the completion time of one operation must be greater than the maximum time of that operation on the assigned machine, and constraints Eq. 14 impose that when a machine performs an operation, no other operations can start during the process. For variables w and z, the logic consistency is defined by constraints Eqs. 15 and 16. Constraints Eq. 17 impose that no operation should begin on it when a machine is changing the configuration. Finally, constraints Eqs. 19–21 define a binary condition on the variables.

5 Scheduling optimization phase

Problem Eqs. 1–21 become very difficult to solve, even for small-size instances. It has a number of variables of the order of $2^{\max\{|\mathcal{O}||\mathcal{M}||\mathcal{T}|,\ |\mathcal{M}||\mathcal{T}||\mathcal{C}|^2\}}$. Thus, even for relatively small instances (e.g., for $|\mathcal{O}| = 100$, $|\mathcal{M}| = 7$, $|\mathcal{T}| = 30$, and $|\mathcal{C}| = 3$), exact solvers cannot solve the problem in a reasonable amount of time. Since real applications need efficient scheduling procedures without affecting the overall makespan, we adopt a hybrid algorithm (HA) to calculate the initial schedule. HA consists of the genetic algorithm (GA) and tabu search (TS), as discussed in [47]. Note that there exist many other hybrid GA approaches dealing with flexible job-shop scheduling. For example, in [48], the authors design an approach for integrating GA with simulated annealing (SA). The introduction of SA is to overcome the premature convergence of GA, similar to the introduction of TS in our HA. however, since the focus of the paper is not to find the best scheduling algorithm, the comparison of different hybrid algorithms is considered out of the scope.

The flow chart (Fig. 2) describes the HA procedure by starting with GA to provide a set of initial solutions as a population and then selecting solutions to do crossover and mutation. TS performs a local search on each of the new solutions. GA then uses improved solutions from TS to start a new evolution. By omitting the TS steps, this hybrid framework can be converted into traditional GA. Similarly, by setting the population size to one and omitting the genetic operators, it can be converted into traditional TS. While HA is not new, we would like to provide interested readers with a clear view of how we adapt HA to solve FJSP in the following sections.

5.1 Encoding and decoding

In GA, it is essential to ensure that all solutions (i.e., chromosomes) generated during the evolutionary process are feasible. In the paper, we show aspects of both encoding and decoding.

The job representation is selected to encode an individual. A chromosome is an array of genes $[j_1, j_2, \ldots, j_{|\mathcal{O}|}]$, each gene corresponds to the job number of the operation.

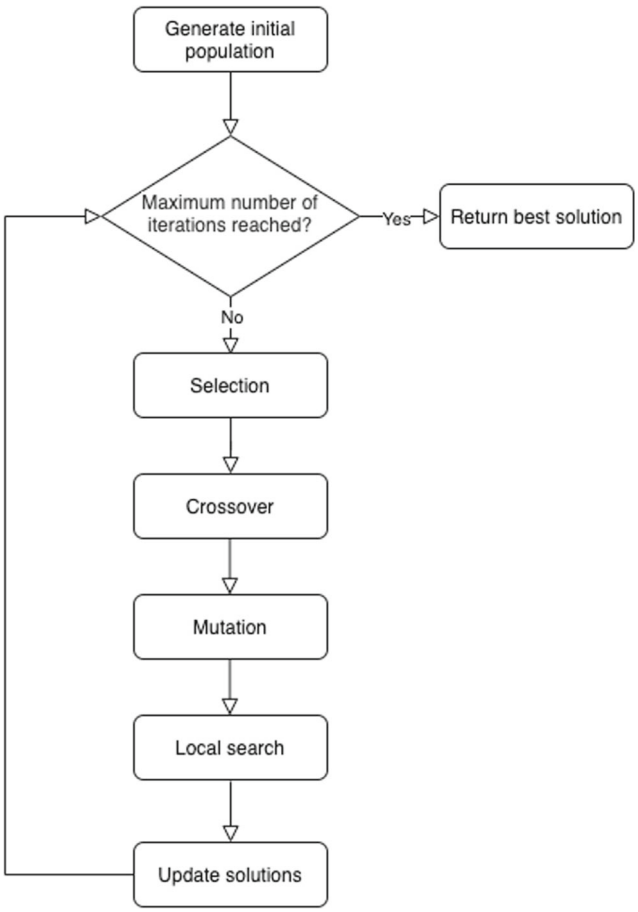

Fig. 2 Flow chart of the heuristic approach

More precisely, this means that the ith presence of the job number j is the ith operation of job j.

In the decoding phase, the assignment of machines and the calculation of start and end time are performed. There might be more than one machine available for each operation, so we use the modified greedy strategy to select randomly only one of the earliest available machines between the first two machines. Subsequently, the availability of both a machine and a setup worker is considered to calculate the start and end time of each operation. The objective is to minimize the overall completion time, so the value of makespan is fitness. The smaller the fitness, the better the solution is.

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

Figure 5 (left side) shows one example of chromosomes. The middle reports a possible machine assignment for operations. Notably, for an operation o, the earliest starting time on each machine is calculated based on the finishing time of its predecessors and the setup worker's available time. Finally, in Fig. 5 (right side), we update the graph by adding the arcs that model the operation precedence on the same machine (dotted lines).

5.2 Genetic operators

The initial population consists of the chromosomes with randomness in operation sequence (for operations in directed arcs). Genetic operators—selection, crossover, and mutation—are established for creating new solutions. To ensure the feasibility of each solution, once it is discovered that it is unfeasible, it will be corrected.

In each generation, we choose tournament selector (selecting the best individual from random samples with replacement) for selecting survivors and roulette wheel selector (selecting according to the fitness proportion) for selecting individuals to create offsprings.

A crossover operator acts on two strings of parents at a time and produces offsprings by recombining the characteristics of both parent strings. We use a so-called *two-point crossover*, which randomly chooses two points in parents and swap the area between the two points.

The left of Fig. 6 shows a crossover example, generating two feasible children. On the right, an infeasible solution to the same problem shown in Fig. 3 shows how to fix infeasibility.

Fig. 3 Example of one scheduling toy instance problem

j	o	$m(T_{om})$		c
1	1	m1(3)	m2(3)	2
	2	m1(4)	-	1
	3	m3(2)	-	3
2	1	m1(2)	m2(2)	1
	2	m2(2)	-	3

c		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

A strategy for swapping mutation is used to avoid spending more time in managing feasibility rather than exploring for better solutions. The solution is straightforward, taking two positions in the recombined chromosome randomly, then swapping genes on the positions to obtain new offsprings. The newborns are feasible on any swap since we use job numbers to represent genes.

5.3 Tabu Search

In TS, move, neighborhood structure, tabu list, and aspiration criteria are the main components.

A neighborhood structure is a mapping of a solution to a set of neighbors (a neighbor is a slightly different solution from the original). [49] proposes the first effective neighborhood structure for JSP by reversing the order of two successive operations on the same machine. A move is a modification on a solution to get a neighbor. The reversing transition is a type of move. This paper adopts the swap strategy exchanging any two operations on different jobs. With the intrinsic meaning of "tabu," forbidden, the tabu list is a memory structure recording the recent moves to avoid the solution cycle. In our 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 is removed when a new element needs to be inserted, but the full capacity is reached. As elaborated by [50], TS excels at avoiding getting stuck in local optima with the usage of the 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 the tabu list.

While the advantage of the tabu list is shown, it displays the possibility of 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 a better solution than what has been found so far.

TS records and encodes the best solution found in a chromosome, and then returns it to the GA population. Although it is likely to transform into better solutions by running for more generations, there should be a tradeoff between the running time of TS and that of GA in HA.

6 Machine learning–based classification phase

To overcome the difficulty of making the rescheduling decision, we create a classification model that returns the rescheduling suggestion, given the topology-related information and the current state of the production system. We highlight that the techniques presented in this section are one possible choice for the classification methods. We select those techniques because they are easy to implement, well known, and robust.

The approach is useful for the following reasons: first, the classifier returns the result of the calculation in a short amount of time, which satisfies the time criterion in dynamic production; second, it requires a small amount of computing power; third, as it provides answers in a short time, it can be run at high frequency and can, therefore, be responsive. Finally, it is possible to know the characteristics of the plant that play actively in deciding the need for rescheduling by using the proposed methodology. Therefore, the plant can be modified to improve its robustness, reduce the bottleneck, and so on.

Considering a set of scenarios $\Theta = \{1, 2, \ldots, \theta_{max}\}$, in each one $\theta \in \Theta$, the jobs that the plant has to fulfill, the related operations and the number of machines will be 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

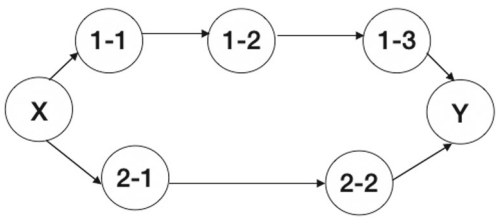

Fig. 4 Directed graph representation of one feasible chromosome

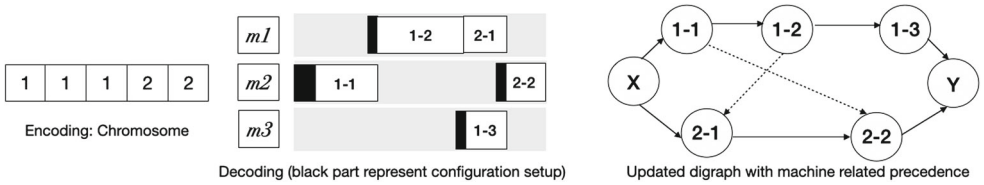

Fig. 5 Example of encoding, decoding, and directed graph with machine precedence relationships

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 as follows: given an operation o that on machine m lasting for T_{om} to process, if scenario θ occurs, it lasts $T_{om}(1+\delta_{mt}(\theta))$. These variations can be positive (under-estimated processing time) or negative (over-estimated processing time). It is worth noting that the random variables $\delta_{mt}(\theta)$ are independent with respect to machine and time step, and independent from the scheduling [51]. In addition, we assume that the expected value is zero. Please notice that this is not a restrictive hypothesis because if the decision-maker discovers that some processes last longer than expected on average, then the expectations change.

To compute the data set for training the classifier, we compute the *actual schedule* $u(\theta)$ in each scenario θ, i.e., the schedule to be followed by the plant by the heuristics described in Section 5. Then, we run the rescheduling procedure to get a new schedule $u(\theta|t)$ for each time step t of scenario θ. The new schedule is computed using the heuristic proposed in Section 5.3, with the current schedule being $u(\theta)$ the starting solution, over the following optimization model:

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

subject to Eqs. 2–21. In the modified objective function Eq. 22, $0 < \lambda < 1$ 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 to limit the number of scheduling changes while minimizing makespan.

Given schedule $u(\theta|t)$ and 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 b as decrements of the makespan in Eq. 23 and assign it 5% by taking into account the time spent in reorganizing the production and avoiding rescheduling if only a small improvement can be achieved. Note that the threshold can be adjusted to each production manager's actual requirement. A higher threshold can be set accordingly for productions that are difficult to reschedule frequently.

It is worth noting that the similar idea of Eq. 23 can be found in [39], where the authors call it *inertia factor*.

By using the procedure above, we get a set of plant states, one for each pair $(t, \theta) \in \mathcal{T} \times \Theta$, and we associate each of them with the label (reschedule or do not reschedule). From the dataset, we extract a set of features for each state, considering information on processing time variations (PTV), planned scheduling, and plant information (including the available resources for each operation and customer orders to be managed).

Besides, it is assumed that all simulations have the same time horizon $\mathcal{T}$. This assumption is not restrictive as we can always consider $\mathcal{T}$ as the longest simulation end time.

Following the approach of [52] and [53], we do not consider automatic feature extraction but we exploit the experience of the involved company to define the following features:

- t: the time step,
- at t, the features related to operation o:
  - OPT_o: remaining processing time,
  - PTV_o: processing time variation,
  - ρ_o: ratio of the available machines able to perform o.

Since considering all operations can lead to over-fitting (the number of operations is higher than the number of

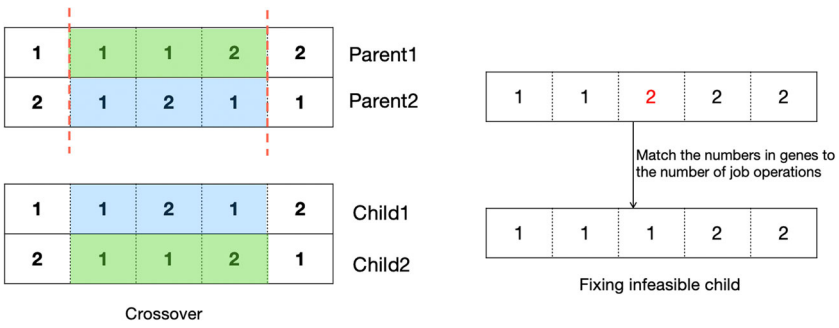

Fig. 6 Example of crossover and infeasibility handling. The red dashed lines represent crossover points

scenarios), with the ratio specified as a measure, only the operations with a high ratio, meaning those with more flexibility in changing machines, are considered. We call the number of considered operations OP_Num. The performance of the classifiers related to OP_Num is evaluated in Section 7. When OP_Num is 2, the considered feature set is $\{t,\ OPT_1,\ PTV_1,\ \rho_1,\ OPT_2,\ PTV_2,\ \rho_2\}$. A snippet of ML input and output example with OP_Num equal to 10 is shown in Fig. 7. The input is the same for all the algorithms compared (negative PTV values mean that the processing time is less than planned, and vice versa). The output is the rescheduling decision (1 means reschedule, 0 means not to reschedule), which is obtained according to the different ML algorithms.

After getting the features above, the resulting dataset is divided into a training dataset (70%) and a test dataset (30%). We consider three commonly used classification algorithms: Random Forest Classifier, which belongs to decision tree induction methods; Support Vector Machines; and Multilayer Perceptrons from neural networks:

1. **Random Forest Classifier (RFC)**: a combination of decision tree classifiers and the ensemble of trees voting for the most popular class [54]. RFC is easy to parametrize, not sensitive to over-fitting, and it provides ancillary information like variable importance [55]. However, a large size of data set can lead to high memory consumption [56].
2. **Support Vector Machine (SVM)**: input vectors are mapped to high dimensional feature space and a linear decision surface is constructed in the space [57]. It does not require any parameter tuning since it can find good parameter settings automatically [58]. It delivers a unique solution because the optimization problem is convex. However, while the feature of non-parametric brings convenience, it lacks the transparency of results [59].
3. **Multilayer Perceptrons (MLP)**: a type of neural network, which simulates human brains [60]. 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 [61]. It shows one benefit that it needs of neither the prior assumptions about the distribution of training data nor the decision regarding the relative importance of input measurements. The costs spent on deciding the number of layers and the number of nodes in those layers are not trivial, and there is no single method for doing it [62].

We have decided not to use more advanced techniques such as deep learning [63], convolution neural network [64], or clustering [65] since our aim in this work is to provide a proof of concept of the integration framework by starting from simple but widely used techniques. Eventually, these approaches are considered because they provide the user with insights on the features considered.

Furthermore, for selecting the validation model, we choose the cross validation (CV) technique, which overcomes over-fitting issues [66]. It is well known that using the same data for the training algorithm and evaluating performance leads to over-optimistic results [67].

7 Numerical experiments

In this section, we present the instance generation procedure, the implementation details, and then discuss the experimental results regarding the optimization and ML techniques separately and those regarding their integration in the rescheduling process.

7.1 Instance generation

The problem instances were created by using a general method to construct all the sets, operators, and parameters described in Section 4. Notably, we use it throughout the section to model a company's factory. The plant consists of two product lines, one for molded rubber and the other for plastic items. In the paper, only the rubber line is considered. The line is made up of 16 machines. All jobs are made up of successive operations, i.e., there are no two operations of the same job that can be carried out in parallel. Also, only one worker can perform the setup operation. Every new setup operation required by a particular operation must, therefore, wait for the setup to be completed.

TimeStep	PTV0	ratio0	OPT0	PTV1	ratio1	OPT1	PTV2	ratio2	OPT2	...	ratio7	OPT7	PTV8	ratio8	OPT8	PTV9	ratio9	OPT9	output
16	8	0.333333	5	19	0.333333	8	19	0.000000	13	...	0.0	9	-11	0.0	2	-11	0.0	6	1
18	15	0.333333	4	7	0.333333	4	7	0.333333	6	...	0.0	9	7	0.0	2	7	0.0	5	0
20	-9	0.333333	4	-6	0.333333	4	-6	0.333333	5	...	0.0	9	-6	0.0	2	-6	0.0	5	1
22	4	0.333333	4	4	0.333333	5	-7	0.333333	3	...	0.0	15	4	0.0	4	4	0.0	1	0
24	-7	0.333333	3	-7	0.333333	3	-7	0.333333	4	...	0.0	12	-7	0.0	4	-7	0.0	1	0

Fig. 7 An example of ML inputs and outputs

The empirical distribution of the PTV used is shown in Fig. 8.

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

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

7.2 Implementation details

As described in Section 5, the first schedule is obtained by HA, consisting of GA and TS. The GA part is built on the open-source programming library *Jenetics* [68], while the TS part has been implemented based on the open-source programming library *OpenTS* [69]. By calibrating the parameters, the population size is settled to 200 for both GA only and HA. For the crossover and mutation operators, in HA, a two-point crossover with probability 0.86 and a swap mutation 0.3 are used. In GA only, two-point crossover with 0.76 and swap mutation with 0.115 are adopted. The tabu length is calibrated into 30 for the TS-only approach and 20 for the HA approach. For each individual in HA, TS is set to iterate 50 times as a stopping criterion.

The ML procedure has been implemented by using the package *Scikit-learn* [70] in Python 3.6. The machine used for the numerical experiments is equipped with an *Intel(R) Core(TM)* i5 CPU@2.3GHz and 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).

The experiment results are shown in the next three subsections. In Section 7.3, we compare the performance of the heuristics implemented for solving Eqs. 1–21, in Section 7.4, the characteristics of the classification problem are analyzed, and next, in Section 7.5, the performance comparison between the proposed approach and periodic rescheduling is elaborated.

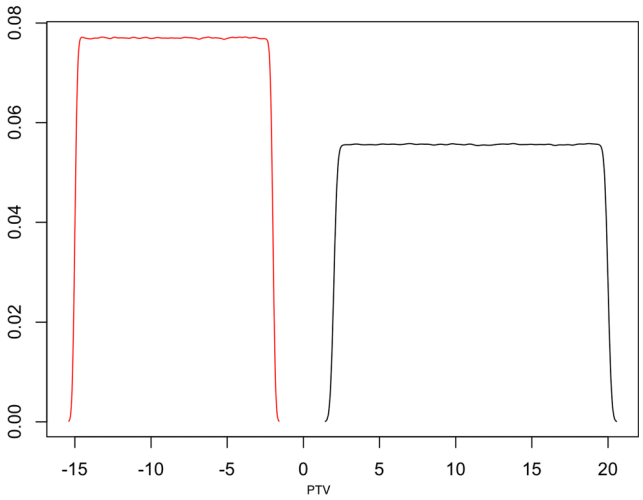

Fig. 8 PTV distribution

7.3 Heuristics performance

7.3.1 Comparing the results of heuristics against the exact solver

In Table 1, each row compares the heuristic gaps (%) against the solutions provided by the GUROBI exact solver (ES). All computation times reported are measured in seconds. The results include two types of comparisons:

- Under the same running time, the differences of makespan;
- The distinctions of heuristic makespan from the best possible values of ES.

Since running time is increased with the difficulty level of the problem, we encompass four intervals to cover a wide range of difficulty levels. The makespans of the instances tested in Table 1 fall within the range 17–120 time units due to the limitation of ES.

The columns under $|\mathcal{M}|$ and $|\mathcal{O}|$ provide the number of machines and operations used in each instance, respectively. The same machine number and operation number do represent distinct instances because other parameters are different (for example, the duration of each operation and the precedence). For the comparison under the same running time, GA_gap, TS_gap, and HA_gap respectively report their gaps (%) compared with the solutions supplied by ES. Moreover, $best\_T$ indicates the running time for Gurobi to find the optimal value for each instance. The columns under GA_bgap, TS_bgap, and HA_bgap separately show the differences of GA, TS, and HA compared with the optimal values found by ES. A dash (-) indicates for the given instance, the solving of solution approach ES exceeded the running memory of the computer; thus, no gap is quantifiable.

When comparing under the same running time, there is only one row of positive values (1 out of 20 instances) available in the first three gap columns, demonstrating that, in most cases, all the three heuristics perform significantly better than ES. The statistics in the last two rows (averages and standard deviations) support the effectiveness of the heuristics. We found that keeping the running time shown in the seventh column, ES slightly surpassed the performance of the heuristics. HA stayed a bit beforehand comparing with the other two heuristics.

7.3.2 Comparing the results of the heuristics

For comparing the three heuristics in larger scales, instances with longer operation period (20–50) and larger makespan

Table 1 Comparing GA, TS, and HA with ES

Time	$\|\mathcal{M}\|$	$\|\mathcal{O}\|$	GA_gap	TS_gap	HA_gap	$best\_T$	GA_bgap	TS_bgap	HA_bgap
(20, 40)	3	10	–	–	–	–	–	–	–
	5	10	−6.12	−6.12	−6.12	4002	2.22	2.22	2.22
	5	10	–	–	–	4265	8.00	8.00	8.57
	5	10	–	–	–	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	–	–	–	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	–	–	–	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	–	–	–	2658	3.00	3.00	0.00
(400, 2200)	4	40	–	–	–	3456	0.00	0.00	0.00
	6	40	−48.59	−48.98	−48.98	3406	4.17	4.17	4.17
	6	50	–	–	–	–	–	–	–
	6	50	–	–	–	–	–	–	–
	6	60	–	–	–	–	–	–	–
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

(190–330) have been evaluated. In real and dynamic production, it is critical to get a feasible schedule quickly enough, so to compare the results, the time values of 20s, 40s, and 95s were chosen. Because ES was unable to provide solutions, HA was used as a benchmark for gap calculation (%), as shown in Table 2. The bigger the gap shown under column GA_gap and TS_gap, the bigger the makespan they compared with HA.

HA has demonstrated its good performance with more frequent appearance of non-negative values in Table 2, which is contributed by its mixture strategy in exploration and exploitation. Consequently, HA is chosen to get the initial schedule. TS achieved similar results with a slightly worse quality compared with HA. With the feature of neighbor exploration tending to discover similar solutions and its satisfying quality, TS is chosen for rescheduling.

7.4 ML-based classification analysis

This section shows the performance of the proposed approach from the results of three classification algorithms. As for the performance estimator, the area under the receiver operating characteristic curve (AUC) is adopted since it exhibits more desirable properties comparing with overall accuracy [71, 72]. The value of AUC ranges from 0.5 (useless test) to 1 (correctly discriminated test).

The test compares the performance of SVM, RFC, and MLP and the number of operations considered. The X-axis

Table 2 Comparisons among GA, TS, and HA

$\|\mathcal{M}\|$	$\|\mathcal{O}\|$	$T(s)$	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

OP_Num indicates the number of collected operations ranging from 1 to 10. The outcome is averaged by taking results from 10 random seeds (different seed leads to different fitting behavior of RFC and MLP, which likely causes different scores). In Fig. 9, the average AUC values are presented.

As shown, RFC stayed far ahead. For SVM, firstly AUC score grew but declined after OP_Num 8 was reached. Instead, for MLP, the score kept decreasing, with some exceptions in the middle. Concerning RFC, as OP_Num increased, its AUC values kept improving. Thus, considering operations with the ten highest ratios, the RFC achieved the best AUC score of 0.81. For this reason, RFC and this setting are considered in the following subsection.

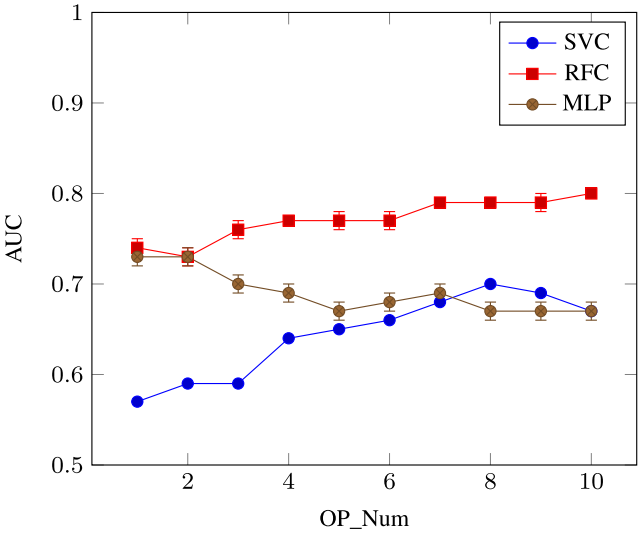

Fig. 9 AUC values for different operation numbers

7.5 Rescheduling performance

The following subsections will present the workflow of the rescheduling framework, its computational results on the comparisons of makespan improvements, remaining makespan, and detailed analysis of two examples.

7.5.1 Rescheduling simulation process

The pseudo-code in Algorithm 1 exploits the workflow of the simulation done in the rescheduling framework where ML and optimization techniques are integrated.

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
        end for
        Collect feature values
        Send to ML classifier to get prediction
        if prediction is 1 then
            Reschedule
        end if
    end for
end for
```

As for competitors, we consider rescheduling actions at every time interval (P-1), every 2 (P-2), 4 (P-4), 7 (P-7), and 10 (P-10) time intervals.

The method for rescheduling at fixed time intervals is often used in practice due to the simplicity of rescheduling rules. Especially, as we know, several companies with three work shifts per day tend to reschedule every 8 h (i.e., at the start of each work shift). With this policy, when the integration with I4.0 technologies is not applied, the company gives workers their mansions at the start of the shift and workers stick to the plan until the end of their shift. By contrast, some companies provide wearable devices for workers with I4.0 that create the opportunity to communicate quickly and efficiently—thus offering the possibility of having real-time information to update the work without paying a real cost for reorganizing. So it is not considered to add a penalty when implementing rescheduling.

7.5.2 Rescheduling computational results

Given the time interval of $T = 2$ time units and $\theta_{max} = 15$, we tested the periodic approach with 1, 2, 4, 7, and 10 rescheduling time intervals and the ML rescheduling policies outlined in Section 6. For the same scenario that ML and periodic approaches run, the same oscillation values are added. By implementing the procedures presented in Algorithm 1, the statistics were collected at each time interval until reaching the originally planned finishing time. For example, if a schedule is estimated to be completed in 100 time units, then in the simulation of production, 100 time units will be set as the time horizon.

All the values correlated with time are normalized and represented in percentages.

Comparison of makespan improvements The statistics on each approach are shown in Table 3. The column $Approach$ indicates the rescheduling mechanism, N represents the rescheduling times, and $avgI$ indicates the average improvement of the makespan (i.e., the average

Table 3 Comparison of makespan improvements

Approach	N	$avgI$	$stdI$
ML	38	12.76	7.71
P-1	306	3.20	5.20
P-2	150	4.00	5.37
P-4	71	6.28	8.23
P-7	39	11.87	10.60
P-10	24	16.00	13.92

saving of the production time, calculated by averaging the improvements of all the rescheduling occurrences). The last, $stdI$, presents the standard deviation of the makespan improvements. More precisely, by defining $n(\theta)$ as the times in which a rescheduling is performed in scenario θ, the rescheduling number taking into account all scenarios is defined as

$$N := \sum_{\theta=1}^{\theta_{max}} n(\theta). \tag{24}$$

As shown, P-1 had a negligible average improvement with the highest rescheduling frequency. We assume that a single time unit is 1 h, this strategy is equivalent to rescheduling every 2 h, so it is not ideal for real-world implementation (many operations can last longer than 2 h).

On the contrary, P-10 rescheduled just 24 times but achieved the highest average gain in makespan (16.00). Rescheduling less frequently creates a more extensive growing space. However, its considerable standard deviation value indicates the range of its improvement values is a bit too wide.

In general, ML performed best in terms of both average value and standard deviation by rescheduling a few times.

Comparison of remaining makespan For stabilizing the production, it is essential to manage unexpected events in a dynamic environment. Generally speaking, the less rescheduling, the better, despite the use of modern technologies, because any communication can fail for various reasons (workers may miss messages, misunderstand, lose time to understand the message, and so on).

To investigate the differences between the makespans achieved through each periodical solution and ML approach, C_θ is defined as the remaining makespan of scenario $\theta \in \Theta$ at the last measured time step (the time step is measured until the planned finish time) through periodical rescheduling, C_θ^{ML} is for ML approach and D_θ is the corresponding difference, calculated as in Eq. 25:

$$D_\theta = C_\theta - C_\theta^{ML}, \quad \forall\theta \in \Theta. \tag{25}$$

With the methods above applied in each scenario, both the average makespan difference $avgD$ and the standard deviation $stdD$ are calculated by considering all the scenarios. The results are shown in Table 4.

In Table 4, the figures excluding the row of P-7 are positive, which indicates that most periodical solutions had bigger remaining makespan than those of the ML approach. Therefore, the schedules were probably finished later than ML by the periodic ones. P-1 and P-7 reach, on average, the closest makespan values to ML. However, as stated before, P-1 is not the right approach in practice because its frequent rescheduling leads to unstable production and potential resource waste. The standard deviations were significant because the tested instances were quite diversified in operation quantity, machine quantity, and processing time.

Although P-4 rescheduled more frequently than P-10, there is no advantage in reducing makespan values; hence, we can infer rescheduling frequently was not indeed necessary in every scenario. P-2 rescheduled more often than P-4. However, it failed to reschedule at the most "profitable" time in general. Besides, we can see that compared with other periodical approaches, P-7 was most comparable with the ML approach.

Comparison of the makespan at each time step Ultimately, not only at the end of the time horizon but also during the time phases, we examine the discrepancies in makespan. At each time step, we compare the makespan difference and compute it by considering ML as the benchmark.

Similar to the calculation of D_θ, the difference at each time step is now counted. Given scenario $\theta \in \Theta$ and a set of time steps $\mathcal{T}$, $C_{t\theta}$ is defined as the remaining makespan at the time step $t \in \mathcal{T}$, $C_{t\theta}^{ML}$ is for ML, and D_θ^* is the makespan difference by comparing each approach with the ML approach at the same t, which is calculated in Eq. 26:

$$D_\theta^* = C_{t\theta} - C_{t\theta}^{ML}, \quad \forall\theta \in \Theta, \forall t \in \mathcal{T}. \tag{26}$$

After getting D_θ^*, the averages and standard deviations were calculated by following conventional methods. The results are listed in Table 5.

Table 5 shows that averagely all the periodical approaches had greater makespan than ML, which proves the effectiveness of ML in the ongoing production. Among

Table 4 Difference in remaining makespan

Approach	$avgD$	$stdD$
P-1	1.67	44.68
P-2	19.93	62.83
P-4	13.53	56.60
P-7	−0.33	28.34
P-10	6.8	31.82

Table 5 Difference in the makespan at each time step

Approach	$avgD^*$	$stdD^*$
P-1	1.16	44.67
P-2	10.18	46.18
P-4	4.78	36.44
P-7	11.88	61.81
P-10	6.32	36.69

periodical methods, P-4 stood out by having small values both on average and standard deviation.

Observing Tables 3, 4, and 5, while P-7 had a bigger average makespan considering all time steps compared with ML (in Table 5), it did indicate a good tradeoff between rescheduling frequency and schedule delays. In general, P-4 behaved fairly in all the aspects, which matches the fact that it is widely used in factories. Considering the proposed approaches—ML and periodical ones—we can see that by recommending to reschedule less frequently and at the right time, ML got satisfactory outcomes not only in saving overall production time but also in the rescheduling effectiveness, which avoided wasting resources in managing machine and worker changes.

Detailed analysis of two examples We take two scenarios for detailed analysis in Fig. 10 by showing the makespan trend. Table 6 shows the corresponding rescheduling frequency for each approach in the two scenarios.

On the left of Fig. 10, it shows that ML suggested rescheduling twice at around time steps 25 and 65. Generally, all lines shook greatly from time step 18 to 69, which might result from the random oscillations added to the schedule. ML outperformed all others except for P-7.

Table 6 Rescheduling times with each approach on the 2 scenarios

Scenario	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

On the right, P-7, P-10, and ML overlapped into one line. Without any rescheduling, ML got the best result equivalently in makespan. We can deduce that rescheduling is not necessary for every disturbance, and a periodical approach is rigid to fit in.

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 concerning the passage of time. A rescheduling decision can be made within seconds with an ML approach, and the actual rescheduling approach is searched only if the favorable decision is made. ML reveals the potential to make better rescheduling decisions not only for the adaptability it owns but also for the time it saves.

8 Conclusions and future research

In this paper, we have proposed a new framework for coping with rescheduling under the context of I4.0. This work represents the preliminary approach to use ML and optimization together in the rescheduling field by assuming the availability of real-time data analysis. We proved the potential of the integration of these techniques by conducting computational experiments. It is essential to notice that, despite the simplicity of the techniques used in the framework, we have been able to achieve good results.

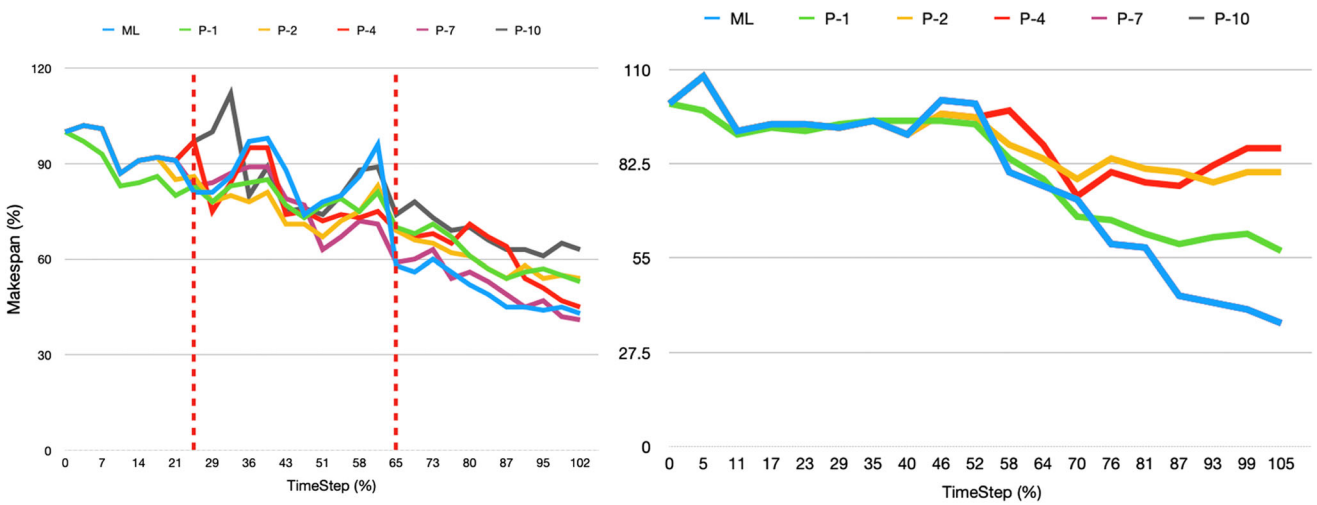

Fig. 10 Comparison of makespan trend for Instance 1, on the left, and Instance 2, on the right. The red dashed lines indicate when the ML approach reschedules

This is a promise of even better results if new and ad hoc techniques are used.

The main results of the paper are, therefore, the definition of the first set of features that led to a good classifier and the above general methodology. Furthermore, another contribution is the formalization of the FJSP through a mathematical programming model. While the case study is on plastic and rubber manufacturing, the proposed framework can also be tailored for other industries (such as printed circuit board, semiconductor, and metal), which often face the problem of making the rescheduling decisions. Specifically, the dedicated features should be derived for the new problem. Besides, we believe it is also possible to effectively adapt the approach out of the production industry, such as in personnel scheduling for hospitals, where daily fluctuations in emergencies, patient population, and levels of care occur frequently. For example, in [73], there is a list of available nurses, including floaters who are assigned to specific units in need, and casuals who have no employment contract and are typically called at the last minute. How to satisfy the patients' demand in time while avoiding excessive workload of nurses remains a big challenge. In this particular case, our rescheduling framework may help to balance the service. The average waiting time of patients, the number of patients, and the number of available nurses can be exciting features to be included within the ML approach.

Several future developments on this topic could be considered:

- Improving the performance of the heuristics by reducing the number of machines capable of carrying out each task [74]
- Expanding the simulation by differentiating the stochastic oscillations under different disturbance factors because the current distribution of PTV is too general and it may result in the rapid increase of processing time
- More advanced machine learning algorithms, as well as the definition of an enlarged set of features including the deeper knowledge related to the bottleneck of the scheduling and the property of the graph G. In particular, we are interested in exploring the research with graph theory and neural networks of scheduling and rescheduling patterns [75]. There is also a need for more detailed instructions on the methodology of data analysis.

Finally, the work has shown that the performance of a heuristic is possible to be learned for a machine learning technique. This general aspect could be applied in several other contexts and opens several general research lines as, for example, the possibility to use the ML techniques not just to classify the application of a heuristic but to guide and calibrate it on the ongoing setting.

Acknowledgments The authors acknowledge all the Plastic and Rubber 4.0 research project partners for their contribution.

Funding Open access funding provided by Università degli Studi di Brescia within the CRUI-CARE Agreement. This research was partially supported by the Plastic and Rubber 4.0 (P&R4.0) Research Project, POR FESR 2014–2020—Action I.1b.2.2, funded by Piedmont Region (Italy), Contract No. 319-31.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Trstenjak M, Cosic P (2017) Process planning in industry 4.0 environment. Procedia Manufacturing 11:1744–1750. https://doi.org/10.1016/j.promfg.2017.07.303, 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy
2. Fadda E, Gobbato L, Perboli G, Rosano M, Tadei R (2018) Waste collection in urban areas: a case study. Interfaces 48(4):307–322
3. Giusti R, Manerba D, Bruno G, Tadei R (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. https://doi.org/10.1016/j.tre.2019.07.009
4. Fadda E, Perboli G, Tadei R (2018) Customized multi-period stochastic assignment problem for social engagement and opportunistic IoT. Computers & Operations Research 93:41–50
5. Frank AG, Dalenogare LS, Ayala NF (2019) Industry 4.0 technologies: implementation patterns in manufacturing companies. Int J Prod Econ 210:15–26
6. Cohen Y, Faccio M, Pilati F, Yao X (2019) Design and management of digital manufacturing and assembly systems in the Industry 4.0 era. The International Journal of Advanced Manufacturing Technology 105(9):3565–3577. https://doi.org/10.1007/s00170-019-04595-0
7. Farahani S, Brown N, Loftis J, Krick C, Pichl F, Vaculik R, Pilla S (2019) Evaluation of in-mold sensors and machine data towards enhancing product quality and process monitoring via Industry 4.0. The International Journal of Advanced Manufacturing Technology 105(1-4):1371–1389. https://doi.org/10.1007/s00170-019-04323-8

8. Zhang Y, Cheng Y, Wang XV, Zhong RY, Zhang Y, Tao F (2019) Data-driven smart production line and its common factors. The International Journal of Advanced Manufacturing Technology 103(1-4):1211–1223. https://doi.org/10.1007/s00170-019-03469-9
9. Handelsblatt (2019) How 5G revolutionizes the industry. https://www.handelsblatt.com/adv/siemens-digital/schnell-vernetzt-stabil-gehalten-wie-5g-die-industrie-revolutioniert/24093034.html?ticket=ST-19056066-l6TWhVJaFBvxSAXdbkcQ-ap2
10. McKinsey (2015) Industry 4.0 how to navigate digitization of the manufacturing sector. https://www.mckinsey.com/business-functions/operations/our-in sights/industry-four-point-o-how-to-navigae-the-digitization-of-the-manufactur ing-sector
11. Brucker P (2010) Scheduling algorithms. 5th edn, Springer Publishing Company, Incorporated
12. Gupta D, Maravelias CT, Wassick JM (2016) From rescheduling to online scheduling. Chem Eng Res Des 116:83–97
13. Zhang J (2017) Review of job shop scheduling research and its new perspectives under Industry 4.0. J Intell Manuf 30:1809–1830
14. Sellers DW (1996) A survey of approaches to the job shop scheduling problem. In: Proceedings of 28th Southeastern Symposium on System Theory, IEEE, pp 396–400
15. Đurasević M, Jakobović D (2018) A survey of dispatching rules for the dynamic unrelated machines environment. Expert Syst Appl 113:555–569
16. Caballero-Villalobos JP, Mejía-delgadillo GE, García-Cáceres RG (2013) Scheduling of complex manufacturing systems with Petri nets and genetic algorithms: a case on plastic injection moulds. The International Journal of Advanced Manufacturing Technology 69(9-12):2773–2786
17. Mönch L (2007) Simulation-based benchmarking of production control schemes for complex manufacturing systems. Control Eng Pract 15(11):1381–1393
18. Graham RL, Lawler EL, Lenstra JK, Kan AR (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. In: Annals of discrete mathematics, vol 5, Elsevier, pp 287–326
19. Tamaki H, Hasegawa Y, Kozasa J, Araki M (1993) Application of search methods to scheduling problem in plastics forming plant: a binary representation approach. In: Proceedings of 32nd IEEE Conference on Decision and Control, IEEE, pp 3845–3850
20. Sels V, Steen F, Vanhoucke M (2011) Applying a hybrid job shop procedure to a Belgian manufacturing company producing industrial wheels and castors in rubber. Computers & Industrial Engineering 61(3):697–708
21. Nie L, Wang X, Pan F (2019) A game-theory approach based on genetic algorithm for flexible job shop scheduling problem. In: Journal of Physics: Conference Series, IOP Publishing, vol 1187, pp 032095
22. Azzouz A, Ennigrou M, Ben Said L (2017) A hybrid algorithm for flexible job-shop scheduling problem with setup times. International Journal of Production Management and Engineering 5(1):23–30
23. Gao L, Peng C, Zhou C, Li P (2006) Solving flexible job shop scheduling problem using general particle swarm optimization. In: Proceedings of the 36th CIE Conference on Computers & Industrial Engineering, pp 3018–3027
24. Roshanaei V, Azab A, ElMaraghy H (2013) Mathematical modelling and a meta-heuristic for flexible job shop scheduling. Int J Prod Res 51(20):6247–6274
25. Costa A, Cappadonna FA, Fichera S (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 69:2799–2817. https://doi.org/10.1007/s00170-013-5221-5
26. Baykasoğlu A, Ozsoydan FB (2018) Dynamic scheduling of parallel heat treatment furnaces: a case study at a manufacturing system. Journal of Manufacturing Systems 46:152–162
27. Gong G, Deng Q, Gong X, Liu W, Ren Q (2018) A new double flexible job-shop scheduling problem integrating processing time, green production, and human factor indicators. J Clean Prod 174:560–576
28. Dolgui A, Ivanov D, Sethi SP, Sokolov B (2019) Scheduling in production, supply chain and Industry 4.0 systems by optimal control: fundamentals, state-of-the-art and applications. Int J Prod Res 57(2):411–432. https://doi.org/10.1080/00207543.2018.1442948
29. Fadda E, Perboli G, Squillero G (2017) Adaptive batteries exploiting on-line steady-state evolution strategy. In: Squillero G, Sim K (eds) Applications of evolutionary computation. Springer International Publishing, Cham, pp 329-341
30. Sahin C, Demirtas M, Erol R, Baykasoğlu A, Kaplanoğlu V (2017) A multi-agent based approach to dynamic scheduling with flexible processing capabilities. J Intell Manuf 28(8):1827–1845
31. Bierwirth C, Mattfeld DC (1999) Production scheduling and rescheduling with genetic algorithms. Evol Comput 7(1):1–17
32. Vieira GE, Herrmann JW, Lin E (2003) Rescheduling manufacturing systems: a framework of strategies, policies, and methods. J Sched 6(1):39–62
33. Narayanaswami S, Rangaraj N (2011) Scheduling and rescheduling of railway operations: a review and expository analysis. Technology Operation Management 2(2):102–122
34. Uhlmann IR, Frazzon EM (2018) Production rescheduling review: opportunities for industrial integration and practical applications. J Manuf Syst 49:186–193. https://doi.org/10.1016/j.jmsy.2018.10.004
35. Šemrov D, Marsetič R, Žura M, Todorovski L, Srdic A (2016) Reinforcement learning approach for train rescheduling on a single-track railway. Transportation Research Part B: Methodological 86:250–267. https://doi.org/10.1016/j.trb.2016.01.004
36. Palombarini JA, Barsce JC, Martínez EC (2014) Generating rescheduling knowledge using reinforcement learning in a cognitive architecture. arXiv:abs/1805.04752
37. Buddala R, Mahapatra SS (2019) Two-stage teaching-learning-based optimization method for flexible job-shop scheduling under machine breakdown. Int J Adv Manuf Technol 100(5-8):1419–1432. https://doi.org/10.1007/s00170-018-2805-0
38. Larsen R, Pranzo M (2019) A framework for dynamic rescheduling problems. Int J Prod Res 57(1):16–33. https://doi.org/10.1080/00207543.2018.1456700
39. Rossit DA, Tohmé F, Frutos M (2019) Industry 4.0: smart scheduling. Int J Prod Res 57(12):3802–3813
40. Rudtsch V, Gausemeier J, Gesing J, Mittag T, Peter S (2014) Pattern-based business model development for cyber-physical production systems. Procedia CIRP 25:313–319
41. Baykasoğlu A, Karaslan FS (2017) Solving comprehensive dynamic job shop scheduling problem by using a grasp-based approach. Int J Prod Res 55(11):3308–3325
42. Carlucci D, Renna P, Materi S, Schiuma G (2020) Intelligent decision-making model based on minority game for resource allocation in cloud manufacturing. Management Decision
43. Wang J, Yang J, Zhang Y, Ren S, Liu Y (2020) Infinitely repeated game based real-time scheduling for low-carbon flexible job shop considering multi-time periods. J Clean Prod 247:119093
44. Schwung D, Reimann JN, Schwung A, Ding SX (2020) Smart manufacturing systems: a game theory based approach. In: Intelligent systems: theory, research and innovation in applications, Springer, pp 51–69
45. LaValle SM (2006) Planning algorithms. Cambridge University Press
46. Balas E (1969) Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. Oper Res 17(6):941–957

47. Meeran S, Morshed M (2012) A hybrid genetic tabu search algorithm for solving job shop scheduling problems: a case study. J Intell Manuf 23(4):1063–1078
48. Huang X, Yang L (2019) A hybrid genetic algorithm for multi-objective flexible job shop scheduling problem considering transportation time. Int J Intel Comput Cybern 12(2):154–174
49. Van Laarhoven PJ, Aarts EH, Lenstra JK (1992) Job shop scheduling by simulated annealing. Oper Res 40(1):113–125
50. Zäpfel G, Braune R, Bögl M (2010) Metaheuristic search concepts: a tutorial with applications to production and logistics. Springer Science & Business Media
51. Li RK, Shyu YT, Adiga S (1993) A heuristic rescheduling algorithm for computer-based production scheduling systems. Int J Prod Res 31(8):1815–1826
52. Castrogiovanni P, Fadda E, Perboli G, Rizzo A (2020) Smartphone data classification technique for detecting the usage of public or private transportation modes. IEEE Access 8:58377–58391. https://doi.org/10.1109/ACCESS.2020.2982218
53. Fadda E, Mana D, Perboli G, Vallesio V (2018) Sustainable mobility and user preferences by crowdsourcing data: The open agora project. In: 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE) https://doi.org/10.1109/COASE.2018.8560512
54. Breiman L (1999) Random forests. UC Berkeley TR567
55. Horning N 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
56. Santur Y, Karaköse M, Akin E (2016) Random forest based diagnosis approach for rail fault inspection in railways. In: National Conference on Electrical, Electronics and Biomedical Engineering, ELECO, IEEE, pp 745–750
57. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
58. Joachims T (1998) Text categorization with support vector machines: learning with many relevant features. In: European Conference on Machine Learning, Springer, pp 137–142
59. Auria L, Moro RA (2008) Support vector machines (SVM) as a technique for solvency analysis. DIW Berlin Discussion Paper N. 811
60. Pal SK, Mitra S (1992) Multilayer perceptron, fuzzy sets, and classification. IEEE Transactions on Neural Networks 3(5):683–697
61. Singhal S, Wu L Training multilayer perceptrons with the extended Kalman algorithm. In: Advances in neural information processing systems, pp 133–140
62. Gardner MW, Dorling S (1998) Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmos Environ 32(14-15):2627–2636
63. Shrestha A, Mahmood A (2019) Review of deep learning algorithms and architectures. IEEE Access 7:53040–53065. https://doi.org/10.1109/ACCESS.2019.2912200
64. Aloysius N, Geetha M (2017) A review on deep convolutional neural networks. In: International Conference on Communication and Signal Processing (ICCSP), pp 0588–0592. https://doi.org/10.1109/ICCSP.2017.8286426
65. Cuzzocrea A, Gaber MM, Fadda E, Grasso GM (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
66. Arlot S, Celisse A et al (2010) A survey of cross-validation procedures for model selection. Statistics Surveys 4:40–79
67. Larson SC (1931) The shrinkage of the coefficient of multiple correlation. J Educ Psychol 22(1):45
68. Wilhelmstötter F (2019) Jenetics library user's manual v.5.1.0. https://jenetics.io/
69. Harder R (2019) OpenTS tutorial. https://www.coin-or.org/Ots/docs/manual.html
70. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830
71. Bradley AP (1997) The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recogn 30(7):1145–1159
72. Wald N, Bestwick J (2014) Is the area under an ROC curve a valid measure of the performance of a screening or diagnostic test? J Med Screen 21(1):51–56
73. Bard JF, Purnomo HW (2004) Real-time scheduling for nurses in response to demand fluctuations and personnel shortages. In: Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling, Citeseer, pp 67–87
74. Quinton F, Hamaz I, Houssin L (2019) A mixed integer linear programming modelling for the flexible cyclic jobshop problem. Annals of Operations Research. https://doi.org/10.1007/s10479-019-03387-9
75. Obara M, Kashiyama T, Sekimoto Y (2018) Deep reinforcement learning approach for train rescheduling utilizing graph theory. In: IEEE International Conference on Big Data, IEEE, pp 4525–4533