Chapter 13
The Tolerance Scheduling Problem in a Single Machine Case

Daniel Alejandro Rossit, Fernando Tohmé, and Gonzalo Mejía Delgadillo

Abstract This chapter introduces the *Tolerance Scheduling* problem, which involves the decision-making issues in rescheduling processes. The solutions to this problem can be incorporated in the design of Decision Support Systems (DSS) in Industry 4.0 environments. We present here the mathematical foundations for the solutions of the Tolerance Scheduling problem as well as the technical requirements of their embodiment in Industry 4.0's DSS. We illustrate these ideas in a case study with a single machine, in which we analyze the performance of the model at different tolerances.

13.1 Introduction

The last years have witnessed an explosive increase in the penetration of digital technologies like the Internet of Things (IoT), facilitating significant advances in production systems by augmenting their flexibility and the autonomy of their control systems (Lee et al. 2015). This, in turn, promotes new business models on the basis of the intensive and massive personalization of production processes (Monostori et al. 2016; Bortolini et al. 2017). The autonomy of control systems allows the autonomous

D. A. Rossit (✉)
Departamento de Ingeniería, Universidad Nacional del Sur, Bahía Blanca, Argentina

INMABB UNS CONICET, Instituto de Matemática de Bahía Blanca, Bahía Blanca, Argentina
e-mail: daniel.rossit@uns.edu.ar

F. Tohmé
INMABB UNS CONICET, Instituto de Matemática de Bahía Blanca, Bahía Blanca, Argentina

Departamento de Economía, Universidad Nacional del Sur, Bahía Blanca, Argentina
e-mail: ftohme@criba.edu.ar

G. M. Delgadillo
Universidad de la Sabana, Faculty of Engineering, Campus Universitario del Puente del Común, Chía, Cundinamarca, Colombia
e-mail: gonzalo.mejia@unisabana.edu.co

B. Sokolov et al. (eds.), *Scheduling in Industry 4.0 and Cloud Manufacturing*,
International Series in Operations Research & Management Science 289,
https://doi.org/10.1007/978-3-030-43177-8_13

solution of problems that previously required human intervention, enriching the capacities of decision support systems (Ivanov et al. 2016; Rossit and Tohmé 2018).

Cyber-Physical Systems (CPS) are the main drivers of this accelerated automatization process (Monostori 2014). A CPS is a single system that integrates the physical part of the device (which carries out the actual production) with a digital component (Lee 2008). These components communicate through IoT technologies, increasing the autonomy of the system, thanks to the incorporation of large information-processing capacities into the production process (Lee et al. 2015). Many of the classical operations in manufacturing planning will become delegated to CPS to either fully automatized them or to provide a powerful support to decision makers (Almada-Lobo 2016).

These considerations apply naturally to scheduling processes, which will also be affected by these technologies (Monostori 2014; Ivanov et al. 2016; Rossit et al. 2019b). In this sense, numerous advances have been already made in this field (Ivanov et al. 2018; Zhang et al. 2019; Rossit et al. 2019d). In this chapter we present some tools that will profit from their implementation in Industry 4.0 technologies. They are intended to solve scheduling problems, in particular *Rescheduling* ones. The problem of rescheduling may arise once the production process is getting executed according a predefined schedule and an unexpected (not anticipated by the plan) event happens, which affects the performance of the production (Vieira et al. 2003). To solve this problem, corrective actions have to be exerted, according to the magnitude of the impact of the event. The possibilities are either to initiate a rescheduling process or not (Pinedo 2012). Our proposal involves analyzing it in the light of the Tolerance Scheduling Problem (Rossit et al. 2019b). Basically, the tolerance scheduling problem captures explicitly how a human modeler faces a rescheduling situation. Its formal structure allows incorporating its solution into a Decision Support System (DSS) applied to production process (Rossit et al. 2019b). In the following text, we will analyze in detail the fundamentals of this problem and of its incorporation in DSS, evaluating the approach in the context of a study case.

13.2 Scheduling and Rescheduling

Scheduling is the process of assigning resources to jobs for specified time periods in order to optimize a single or multiobjective function. The resources and jobs may vary from organization to organization. The former may include the machines in a workshop, squads of workers dedicated to maintain or build things, IT people, and airfields. In turn, jobs may include the operations in a production process, stages in a construction project, runs of computer programs, takeoff and landings in airports.

The objectives may also differ between organizations. A usual one is the minimization of the finishing time of the last job. With such flexible and all-embracing view, it can be said that scheduling problems are pervasive in all fabrication and production systems as well as in information-processing activities.

Among the main features of scheduling processes, we can highlight the following ones: (1) they are complex decision-making processes requiring the detailed planning of production orders (assigning jobs or operations to resources and machines). The trend toward deep customization of production processes only increases their complexity, (2) the lifetimes of schedules is short, requiring their repetition over short temporal horizons, (3) a critical feature is the delivery dates of the products, affecting directly the production costs, and (4) schedules obtained from the most structured decision-making processes in an organization, requiring well-defined data, constraints, and objects (Framinan et al. 2014).

Among the decision processes in scheduling we can distinguish two levels, associated with different time horizons: one involves the classical problem of designing a production schedule *before* production starts. The other level involves the case in which a schedule has already been executed, but new issues arise in real time (e.g., a machine breaks down or the delivery of input materials is delayed). The latter is known as the Rescheduling problem (Vieira et al. 2003; Peng et al. 2018). This problem has been studied widely in the literature since it is a feature of the real praxis of production processes (Vieira et al. 2003). In real-world production contexts, rescheduling is almost mandatory for the minimization of the impact of perturbations on the plan. These perturbations induce delays in processing times, problems in the quality of products and the lack of necessary material. In order to face these events, there exist different possibilities. One approach is to try to foresee these events and endow the schedule with the ability to "absorb" them. Another approach involves facing each of the perturbations once they occur, handling them in a singular and reactive way (Ouelhadj and Petrovic 2009).

A distinctive predictive approach involves using robust scheduling processes, which compute the optimal schedule taking into account the probabilities of the possible perturbations. The resulting schedules are able to absorb or at least minimize the impact of future events. The solutions are usually obtained by using either stochastic optimization (Birge et al. 1990; Arnaout 2014) or robust optimization (Al-Hinai and ElMekkawy 2011; Rahmani and Heydari 2014). Despite its obvious benefits, this approach requires detailed statistical information that is not always fully available in industrial contexts.

The reactive approach addresses issues only when they appear. A usual strategy involves using dispatch rules (Ouelhadj and Petrovic 2009). These generate schedules by comparing and ranking the jobs, for example, according to which one requires less time or has closer due dates. Solutions are accordingly always found in a fast way (Pinedo 2012). A shortcoming is that these dispatch rules are very sensitive to the kind of problem at hand. For different objective functions the performance of the same dispatch rule may vary considerably (Vieira et al. 2003). Even so, dispatch rules are common in industrial contexts in which it is necessary to rearrange quickly and consistently the production schedule (Framinan et al. 2014).

Most rescheduling strategies are event-driven. That is, disruptive events trigger the application of rescheduling mechanisms (Dong and Jang 2012). This event-driven logic is embodied by the Decision Support System that helps managing schedules in a shop floor (Rossit and Tohmé 2018). The human scheduler, for instance, when

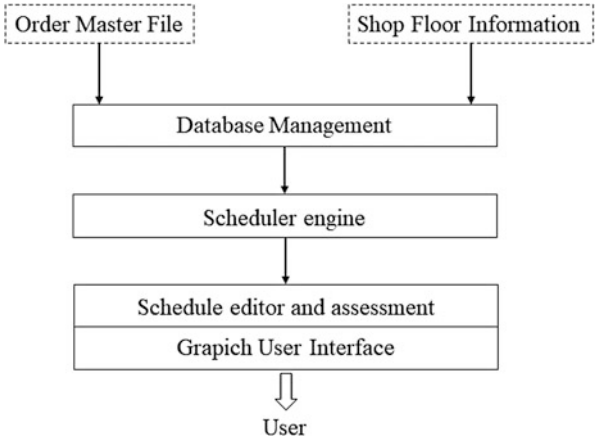

Fig. 13.1 Scheduling system (Pinedo 2012)

noticing a difference between the due date and the actual date of delivery of the production, uses the DSS to simulate potential scenarios and the impact on them of the delay. Upon this assessment the scheduler decides whether or not to initiate the rescheduling process, either following a predefined strategy or, if possible, choosing the best possible strategy (Katragjini et al. 2013).

In the decision-making process of the scheduler, we can identify two initial stages. In the first one she analyzes the disruptive event. In the second stage she checks whether the event requires triggering a rescheduling process (Rossit et al. 2019a). If not, the alarm is dismissed. Otherwise, a next stage of analysis evaluates the impact of the event on the current schedule. Then, again, the scheduler assesses the costs and benefits of rescheduling and according to that decides whether to trigger or not the rescheduling process (Rossit et al. 2019b). Figure 13.1 depicts the rescheduling process according to the system described in the study by Pinedo (2012). It shows that the system feeds in data from the production objectives (Order Master file) and from the shop floor.

13.3 The Impact of Industry 4.0

The concept of Industry 4.0 tries to capture all the relevant features of the transformations that are changing the classically accepted manufacturing practices (Lee et al. 2015). These changes are mostly due to the advent of cyber-physical systems (CPSs) and Internet of Things (IoT), which allow redesigning manufacturing processes oriented towards highly personalized products (Ivanov et al. 2016; Rossit et al. 2019a). CPSs integrate the physical world with virtual environments in which the information drawn from the former can be analyzed using Big Data tools (Lee 2008). This makes possible the real-time analysis of the physical system. This, in turn, endows the production system with the possibility of adapting flexibly itself to new

circumstances. IoT, in turn, provides connections among distributed CPSs allowing the transmission of data collected by a CPS to other CPSs or to a DSS (Dolgui et al. 2019; Ivanov et al. 2019). All these possibilities give rise to the concept of Cyber-Physical Production Systems (CPPSs) (Monostori 2014). An Industry 4.0 environment running under a CPPS will be managed by CPSs, given that they can incorporate AI programs able to solve highly complex problems (Monostori et al. 2016).

This implies that CPSs constitute a key technology for Industry 4.0. It becomes highly relevant, thus, to understand how they work. Lee et al. (2015) present the basic architecture of a CPS based on layers establishing a parallel between the physical and virtual spaces working in consonance inside the CPS. Table 13.1 shows a representation of this. In the first level we find the Connection Level that connects the virtual to the physical world in real time, in such a way that all the information processed by the CPS is obtained at this level. The data are collected either through wireless sensors or by plugged-in ones. The CPS processes the data at the Conversion Level. In this layer, the CPS transforms the data in information giving them a meaning in terms of the environment in which it operates. A multidimensional analysis fits the information to a model of how the CPS works. This, in turn, yields the inputs for the Cyber Level of the architecture. At this layer the CPS evaluates whether the process runs as planned or not. At this level it becomes also possible to compare the internal states of the CPS working together on a single production process. The next layer up is the Cognition Level, in which the models of the previous stage are used to simulate potential cases and choose the best courses of action for the system. Finally, the results are conveyed to the Configuration Level for the autonomous implementation of the actions decided on at the Cognition Level.

Table 13.1 Cyber-physical 5C's architecture

Level	Attribute
I. Connection level	• Plug-in • Tether-free communication • Sensor network
II. Conversion level	• Data-to-information • Multidimensional data correlation • Smart analytics
III. Cyber level	• Virtual modeling • Clustering information • Controllability
IV. Cognition level	• Integrated simulation and synthesis • Collaborative diagnostics and decision making • Early awareness
V. Configuration level	• Self-configuration • Self-optimization

13.3.1 Cyber-Physical Production Systems

The possibility of integrating the functionalities of industrial information systems with physical production assets in the CPS allows the production system to be conceived globally as a Cyber-Physical Production System (CPPS) (Monostori 2014; Monostori et al. 2016; Rossit and Tohmé 2018). The CPPS incorporates the different business functions of an industrial organization into a single CPS-based system (Monostori 2014). Among these built-in business functions are some functions with direct interference in production such as Manufacturing Execution Systems (MES), which are responsible for programming, executing, and programming production processes. To show how a CPPS can program production processes (almost autonomously), we will use the well-known ANSI/ISA-95 control standard as a reference architecture.

The ANSI/ISA 95 standard allows to link control systems and production facilities through an automated interface. It can provide a common environment for the communication of all participants in a production process and offers a representation to model and use the information. It organizes the different levels of decision making hierarchically. This standard is based on the "Purdue Enterprise Reference Architecture" (PERA), which establishes five different hierarchies, as shown in Fig. 13.2. Level 0 is associated with the physical manufacturing process. Level 1 involves the smart devices that measure and manipulate the physical process. Typical instruments at this level are sensors, analyzers, effectors, and related instruments. Level 2 represents the control and supervision of the underlying activities, a typical case of ISA-95. Level 2 systems are Supervisory Control and Data Acquisition (SCADA) or Programmable Logic Controllers (PLC). Level 3

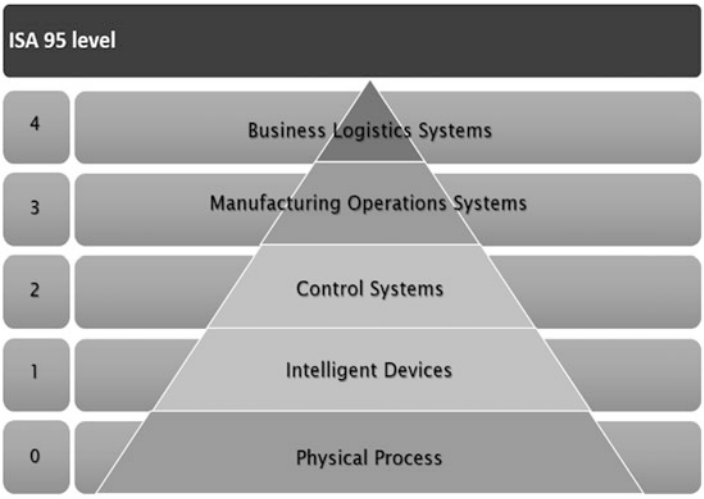

Fig. 13.2 Control structure ANSI/ISA 95 (Rossit and Tohmé 2018)

involves the management of operations and the workflow with respect to production. A clear case of systems belonging to level 3 are manufacturing execution/operations management systems (MES/MOMS). Therefore, this level is of special importance for our work, since this is where the scheduling process is developed. Finally, level 4 is associated with the commercial activities of the entire company. This architecture represents the different activities and functions of a production system hierarchically. In addition, determine the way in which the different levels communicate. In traditional production environments, in particular, each level interacts only with its adjacent levels (Rossit and Tohmé 2018).

13.3.2 Decision Making in CPPS

CPPSs, due to their functional characteristics, will directly impact the decision-making processes of industrial planning and control. To present our perspective on this, we show in Fig. 13.3 the levels of ISA 95 that would be managed by a CPPS in an integrated way. This integration is derived from the capabilities of CPPSs, which can implement physical processes (level 0), measure and manage the instruments that read physical processes (level 1), and implement control actions on their operations (level 2). In addition, given the computational capability of CPPSs, they can also plan, evaluate, and manage the whole production process flow (level 3).

Being able to integrate these functionalities within the CPPS will allow increasing the capacity to respond to unexpected events that occur in production, endowing it with high flexibility. In addition to other advantages, one that should be highlighted is the improvement of the transmission of information, since the same system that

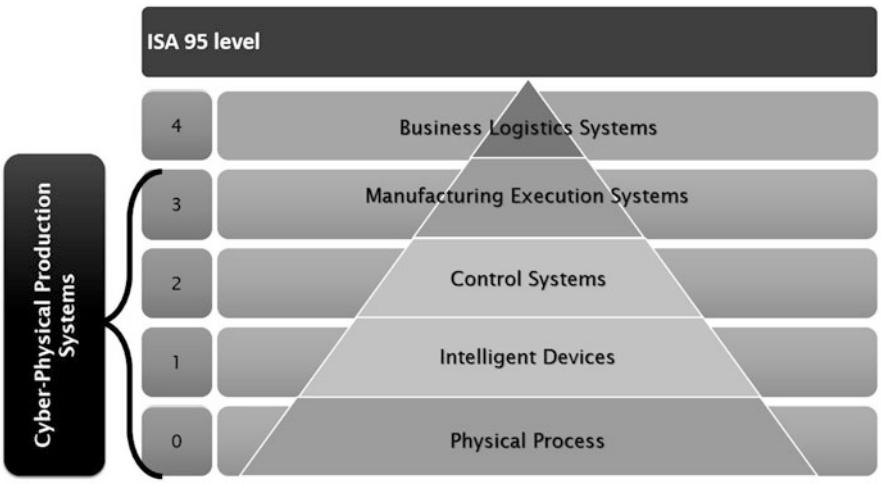

Fig. 13.3 Control structure of a CPPS

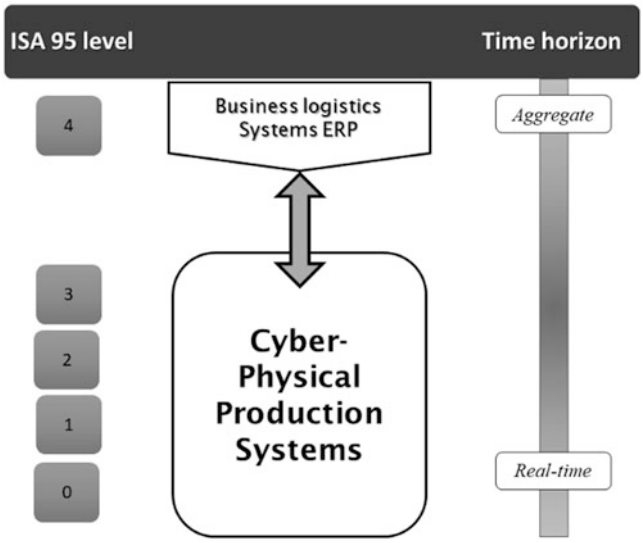

Fig. 13.4 Distribution of ISA 95 levels between ERP and CPPS. The representation of time is drawn from the model of the Manufacturing Enterprise Solutions Association (MESA) International

collects the data is the one that processes and analyzes them. This reduces the impacts of the restrictions imposed by adjacencies inherited from the PERA architecture.

In addition, the integration of these functionalities will have a direct impact on decision making related to production planning. This will lead to the ISA 95 architecture levels being managed by two large systems: the ERP (Enterprise Resource Planning) and the CPPS, as shown in Fig. 13.4. It can be seen there that the levels of the ISA 95 architecture are managed by the ERP for level 4, where the most strategic and aggregate-level decisions are made, while the rest of the levels will be administered by the CPPS. In this perspective, a CPPS should be considered as a set of autonomous (or almost autonomous) elements that collaborate with each other to achieve the objectives set by the ERP. An important consequence of this perspective is that the MES systems, whose tasks are the management of the production flow and its schedule, are embedded within the CPPS. This will generate better information, useful both for making decisions at this level (level 3, MES) and to minimize response times, improving flexibility.

The CPPS will manage a good part of the decisions made by traditional ERP systems (such as inventory control, database management, and information management on suppliers). However, we leave both systems separate to indicate at what point the system becomes autonomous and to what extent human interventions may be necessary, particularly in the area of production planning. Most of these interventions will be associated with the definition of objectives or decisions of aggregate nature. Then, the ERP (properly prepared to work with CPPSs) will translate these decisions to the CPPSs that handle the production flow and the

schedule. Consequently, the CPPSs will not be completely autonomous, since they run on an open loop with the ERP, at least on production planning decisions (Rossit et al. 2019d). In this sense, Almada-Lobo (2016) states that although Industry 4.0 integrates production execution systems, this integration will not include the highest levels of management that define more general aspects of manufacturing.

Within the levels that will be integrated by the CPPSs, it is important to point out that the ISA-95 level 3 will be integrated, even horizontally, allowing access to all the systems at that level: inventory management systems, quality control, historical records, and maintenance planning. This allows CPPSs to work on both scheduling and material inventory problems, which can affect the performance of the production process (Kuo and Kusiak 2019). The systems associated with maintenance work better when monitored in real time, since overrequirements may result in temporary maintenance stops, affecting their availability (Lee et al. 2017; Rossit et al. 2019c). While many of these aspects of maintenance or the lack of materials can have an indirect effect on the schedule, access to that information improves the results of computing the optimal schedule. The proposed design seeks to incorporate these functions into CPPSs, in order to improve the quality of scheduling processes.

In addition to integrating other functions, CPPSs will have access to the data and systems used as inputs for traditional scheduling, as detailed in Fig. 13.1, although with a larger advantage over traditional systems. Essentially, CPPSs will improve the management of the physical processes, integrating execution, control, and analysis in the same system. Moreover, if the decision-making process is delegated to the CPPSs, they will collect the data, make decisions, and execute them. This will result in a complete vertical integration, improving the quality of the results and the responsiveness of the system. The DSS proposed in this work aims to benefit from this capacity of CPPSs and is intended as a step toward implementing the aforementioned vertical integration in Smart Manufacturing environments.

This integration allows to manage the different industries increasingly in a virtual way. This lends the production system resilience in the face of extreme disruptions such as the one generated by the COVID-19 pandemic of 2020 (Ivanov and Das 2020). The possibility of managing the system virtually/remotely allows to reduce the number of people in the shop floor, as well as to run a large number of critical operations online (Ivanov and Dolgui 2020). This virtualization alone, does not by itself guarantee the protection of all personnel, but certainly provides a stronger shield against a pandemic scenario.

13.4 Tolerance Scheduling Problem

In this section we present a summary of how Smart Scheduling Scheme works. Basically, we illustrate the concepts of classical and inverse scheduling. Then, we show how inverse scheduling allows gaining new insights on the scheduling problem. From these insights, we derive the notion of tolerance. Finally, we present the flow of decisions to be made to address the tolerance scheduling problem.

13.4.1 Inverse and Reverse Scheduling

The problem of inverse scheduling is based on the concepts developed for inverse optimization in the study by Ahuja and Orlin (2001). The essential idea of inverse optimization is to start from a given solution (i.e., defined values of the decision variables), to adjust the value of the parameters as to make optimal the given solution. That is, in inverse optimization, the roles are reversed, the variables become fixed, and the parameters become "adjustable." These parameters can be the coefficients of the objective function or the coefficients of the inequalities of the constraints of the classic optimization problem. The problems of inverse scheduling tend to be similar to the latter, modifying internal parameters of the restrictions, being typical cases where the adjustable parameters are due dates (Brucker and Shakhlevich 2009) or processing times (Koulamas 2005).

Unlike the classic scheduling problems in which the parameters are known in advance, in inverse scheduling problems they are unknown and must be determined to make a given schedule optimal (Brucker and Shakhlevich 2011). Generally, the values that these parameters can take are restricted to certain intervals. For example, Brucker and Shakhlevich (2009) seek to analyze schedules in a single machine case with the goal of minimizing the maximum tardiness. The tardiness of a job j (T_j) is defined as the excess of the actual processing time with respect to its due date. It is calculated by

$$T_j(\pi, d) = \max\left\{C_j(\pi) - d_j, 0\right\} \tag{13.1}$$

Here π is a given schedule, d_j is the due date of job j, and C_j the completion time of job j under schedule π. Then, maximum tardiness is obtained according to

$$T_{max}(\pi, d) = \max_{j \in \mathbb{N}} \left\{T_j(\pi, d)\right\} \tag{13.2}$$

The classic scheduling problem involves finding π^* such that $T_{max}(\pi, d)$ is minimal, that is:

$$T_{max}\left(\pi^*, d\right) \leq T_{max}(\pi, d), \quad \text{for any schedule } \pi. \tag{13.3}$$

Assume now the case of the single machine problem with maximum tardiness as objective. The only parameters we need are p_j and d_j, due to the processing times and due dates of work j, respectively. Suppose that only due dates can be modified, so for each job j we will have an interval $d_j \in \left[\underline{d}_j; \overline{d}_j\right]$. Therefore, the adjusted delivery date, $\hat{d}_j$, must be such that $\hat{d}_j \in \left[\underline{d}_j; \overline{d}_j\right]$ for each job j, producing a vector $\hat{d} = \left(\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_n\right)$. Inverse scheduling amounts to minimize the Euclidean difference $\left\|\hat{d} - d\right\|$ (or any other norm) of d, making π optimal. Thus, the inverse scheduling problem is here

$$\min \left\| \hat{d} - d \right\| \tag{13.4}$$

$$s.t. \quad T_{max}\left(\pi, \hat{d}\right) \leq T_{max}\left(\sigma, \hat{d}\right), \tag{13.5}$$

$$\text{For any schedule } \sigma, \ \underline{d}_j \leq \hat{d}_j \leq \overline{d}_j, j \in N. \tag{13.6}$$

Other norms to measure the deviation $\hat{d} - d$ may yield different results (Ahuja and Orlin 2001).

Another problem arises when the goal is to achieve a fixed value of the objective function. Starting from an initial solution, the initial values of the parameter are adjusted so as to reach a given value of the objective function. This problem is known as the Reverse Scheduling problem, which in the case of minimizing maximum tardiness on a single machine is as follows. We seek to find $\hat{d}_j \in \left[\underline{d}_j; \overline{d}_j\right]$, such that the value T^* of the objective function T_{max} remains the same or gets improved.

There are other schemes that involve modifying the parameters, such that from an initial solution, adjust them so as to make the initial solution get a higher objective function value. This problem is called, as said, Reverse Scheduling Problem (Brucker and Shakhlevich 2009). For the previous case (a single machine with maximum tardiness), the Reverse Scheduling problem would look as follows:

$$\min \left\| \hat{d} - d \right\| \tag{13.7}$$

$$s.t. \quad T_{max}\left(\sigma, \hat{d}\right) \leq T^*, \tag{13.8}$$

$$\text{For any schedule } \sigma, \ \underline{d}_j \leq \hat{d}_j \leq \overline{d}_j, j \in N. \tag{13.9}$$

Here T^* is the given objective function value that the Schedule σ should improve by modifying the parameters of the problem. Other parameters are subject to choice so as to solve inverse and reverse scheduling problems, a sample of which can be found in Heuberger (2004).

13.4.2 The Tolerance Scheduling Problem

Rescheduling processes are triggered when the production process deviates from the established plan. These deviations derive from unexpected, and thus unplanned, events, so they were not considered in the design of the production schedule. However, since these events affect the performance of the schedule, corrective actions must be taken and decided according to rescheduling processes.

Rescheduling processes are event-driven. But the events that can trigger rescheduling processes must be distinguished from events that cannot start such processes. Furthermore, determining whether an event triggers or not, a rescheduling process is not enough since it is necessary to analyze also the magnitude of the event. The sensitivity of rescheduling processes to differences in the nature and magnitude of potential triggering events becomes an additional source of instability, making the production process subject to further uncertainty (Pinedo 2012; Framinan et al. 2014). To buffer the system against this, our proposal is to incorporate tolerances as to absorb disruptive events of low impact.

The Tolerance Scheduling problem can be conceived as a mathematical programming one (Rossit et al. 2019b). The decision-making process starts with the solution of a standard scheduling problem. The parameters have given values and the decision variables are related with the features of the production process. A solution is an optimal (or almost optimal) schedule. Then, we proceed to look for a range of tolerances for which the initial solution remains being acceptable. These tolerance ranges are computed assuming some shocks on the parameters of the initial problem, representing disruptive events that are not at the discretion of the scheduler. The issue here is to determine ranges of tolerance within which the produced goods are still considered appropriate, even if some degree of imperfection in the plan is allowed. Consider, for example, situations in which the actual processing times differ from the processing times specified by the original schedule. It is obvious that this affects the performance of the production process (e.g., worsening the makespan), calling for rescheduling the plan if the advantages of doing this exceed its costs.

To introduce the Tolerance Scheduling problem, consider the case previously presented by Brucker and Shakhlevich (2009). It is a simple single machine problem that can be solved in an inverse way, that is, by adjusting the parameters in such a way that a given schedule π becomes optimal. While in the problem of reverse scheduling, the idea is to make minor adjustments to the parameters with respect to initial reference values, in order to ensure that the π solution becomes good enough, in the Tolerance scheduling problem, a range of parameter values has to be found for which the objective function is acceptable and does not require being rescheduled.

Formally, given an optimal or near-optimal schedule π, $F(\pi) \approx T^*$, and the families of parameters d_j and p_j, we seek the maximal interval of variations for them, according to an *inertia factor*, δ, expressing the weight given to the stability of the system. A high δ indicates that the design favors high stability, meaning that few events can trigger reschedules. Then:

$$\max \left\| \hat{d} - d \right\| \tag{13.10}$$

$$s.t. \quad T_{max}\left(\pi, \hat{d}\right) \leq T_{max}\left(\sigma, \hat{d}\right) \cdot (1+\delta)\,, \tag{13.11}$$

$$T_{max}\left(\pi, \hat{d}\right) \leq T^* \cdot (1+\delta)\,, \tag{13.12}$$

$$\text{For any schedule } \sigma, \ \underline{d}_j \leq \hat{d}_j \leq \overline{d}_j, \delta \geq 0, j \in N. \tag{13.13}$$

That is, the goal is to maximize the distance between the d parameters, while ensuring that schedule π improves over the original objective function up to an inertia factor $\delta \geq 0$. This provides a tool that not only detects possible rescheduling events but also determines whether or not to proceed with the rescheduling process. The choice of δ is not arbitrary: if the idea is to reschedule only at high levels of disruption, δ must be large. On the contrary, a low inertia system should be ready to react, requiring a lower δ.

This procedure is rather easy to automatize, providing another tool to be added to the DSS embedded in the CPPS, making the latter more prone to autonomous behavior. The value of δ should, in that case, be set at the design stage.

13.4.3 The Scheme of Smart Scheduling[1]

Smart Scheduling is introduced as a framework for scheduling in production planning by using the tools of Smart Manufacturing and Industry 4.0 environments. As shown in Fig. 13.4, this is handled mainly by a CPPS. The goal of Smart Scheduling is to automatize the solution to the scheduling problems in the integrated frame of CPPS.

We have to prove that the functionalities of a MSS (Manufacturing Scheduling System) (shown in Fig. 13.1), which is part of the MSS-scheduler ensemble in an autonomous CPPS, are captured in our approach. First, let us analyze the MSS in itself, then the scheduler and finally their combination. The functionalities provided by the MSS are easily integrated into a CPPS, since they can be connected to different business functions. CPPSs are by design computer (and physical) systems wired in such a way as to support all those functions (Monostori 2014). This amounts to say that CPPSs can replace and even improve over the MSS, being able to run quality analyses or failure diagnoses, increasing the global efficiency of the system.

This is not the case of the skills of the scheduler, which are not easy to integrate into the basic design of CPPSs (Lee et al. 2015). To extend the functionalities of CPPSs in this respect, we introduce the Smart Scheduling framework, which fundamentally addresses the Tolerance Scheduling Problem. Our proposal provides reliable optimization-based tools to assess the criticality of events, in terms of their nature and magnitude. Endowed with this ability, the CPPS can trigger reschedules only when the goals of the schedule become considerably affected, improving the resilience of the system and decreasing its sensitivity to the noise generated by the environment.

Smart Scheduling follows the logic of dynamic scheduling, as proposed in Fig. 13.5. In a first stage the problem, in either the standard or stochastic version,

[1]The majority of the analysis in this section is based upon the study by Rossit et al. (2019b).

is solved, yielding an initial schedule. Next, the tolerances are set to be used in solving the Tolerance Scheduling problem. In a second stage the production process is started, following the original schedule, until a disruptive event is detected. The event is analyzed to determine if it requires triggering a rescheduling procedure. If not, the production process continues. But if, by its very nature, the event belongs to the class of those that might trigger reschedules, the Tolerance Scheduling procedure is invoked. If the current schedule falls within range of the tolerance, the production process continues. Otherwise, a new schedule is generated to address the disruption caused by the event. Then, once the schedule is obtained applying the dynamic scheduling strategy, new tolerances are set for a future eventual solution of the Tolerance Scheduling

13.5 Examples

To show how Smart Scheduling works on the tolerance scheduling problem, we present a case study. Consider a dynamic scheduling problem in a single machine context with the goal of minimizing Total Weighted Tardiness (TWT). The resulting production process is subject, in the real world, to numerous disruptive events affecting the plan. In our case we will consider machinery breakage or failure, the usual instances studied in the literature (Zandieh and Gholami 2009; Ahmadi et al. 2016; Liu et al. 2017). We will draw data instances from the OR-library.

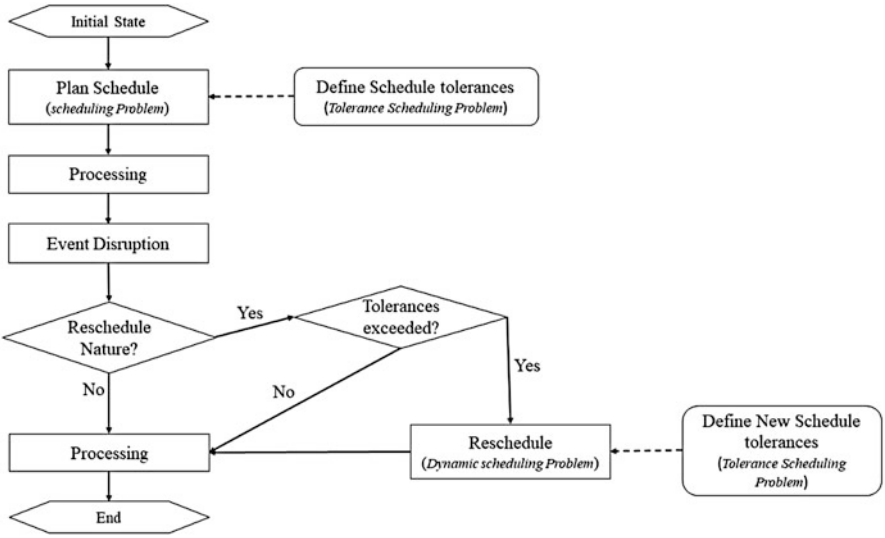

Fig. 13.5 Smart scheduling scheme

We will focus on the following issues: (1) obtain a quasi/optimal schedule, (2) defining tolerances, and (3) analyze events in the light of the schedule and the tolerances. (1) Has been widely studied in the literature and there exist a large variety of methods to solve the problem (Ruiz and Maroto 2005; Ruiz and Vázquez-Rodríguez 2010; Rossit et al. 2018), as well as a way of modeling the problem (Ivanov et al. 2012; Battaïa and Dolgui 2013; Pinedo 2012) and strategies to address it (Blazewicz et al. 2012; Sokolov et al. 2018; Dolgui et al. 2019). For our purposes we choose a heuristical method based on the ATC (Apparent Tardiness Cost) rule, which analyzes jobs according to two classical dispatch rules: WSPT (Weighted Shortest Processing Time) and Minimum Slack First. For details, see Sect. 14.2 in Pinedo (2012).

Then, (2) requires defining the tolerances for the schedule obtained with the ATC rule. In order to do this, we consider the model defined by (13.10)–(13.13) on the initial π. We have to define an inertia factor δ, which in turn yields the tolerance tol_π, defined as in (Rossit et al. 2019e):

$$tol_\pi = TWT \cdot \delta \tag{13.14}$$

The larger the values of δ, the larger will be the magnitude of events that the system is able to "absorb" without triggering a reschedule. On the contrary, lower values of δ make the system more reactive.

The full Smart Scheduling Scheme represented in Fig. 13.5 connects (1) and (2) distinguishing between events that require or not a reschedule. In our case, machine failures are the events that may or may not lead to reschedules. If it involves a small failure (e.g., in terms of the time required to be back in operation), its impact on the objective function can be assessed (via simulations).

We consider 40 cases from the OR-Library and generate different scenarios according to classical examples in the literature (Zandieh and Gholami 2009), with a different number of failures: 2 and 5. The repairing time of failures is represented by means of a uniform distribution over [30, 70], with an expected value of 50, similar to the processing times. Thus, each failure involves an extra operation, in average.

Each scenario is solved under different values of tolerances, according to Eq. (13.14). We considered different values of δ, namely 0%, 15%, 30%, 50%, 75%, and 100%. The results of the experiments are presented in Table 13.2.

Table 13.2 shows the average values of TWT for ten replicas in each case, as well as the corresponding standard deviations. Each row represents either two or five failures in the planning horizon. The first thing to notice is the impact of considering either two or five failures: the average TWT duplicates under no tolerance (0%), increasing from 3447.1 to 6646.8. With two failures and low tolerances, no changes in the average value of the objective function can be detected. For higher values (over 50%), the objective function gets impacted. On the other hand, for five failures, we can see that the tolerances at which the objective function changes are higher than those for two failures (100%). These variations in the objective function can be explained by the lower reactivity in the face of failures, leading to a higher tolerance to their impact and thus to less readjustments of the schedule.

Table 13.2 Experimental results

	Tolerance											
	0%		15%		30%		50%		75%		100%	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Two failures	3447.1	717.3	3447.1	717.3	3447.1	717.3	3651.7	317.2	3651.7	317.2	3793.2	440.1
Five failures	6646.8	1253.7	6646.8	1253.7	6646.8	1253.7	6646.8	1253.7	6646.8	1253.7	6761.9	1035.4

Ref: *SD* standard deviation

13.6 Conclusions

In this chapter we presented the Tolerance Scheduling Problem, discussing its solution and its performance in a case study. We can see that tolerances allow modeling the capacity of the system to absorb deviations with respect to the initial schedule. We find that the larger that tolerance, the larger the deviation of the original plan that ensues.

Future work involves the analysis of this problem under the same objective function (TWT) in more complex production environments, like flow shop or job shop systems.

Acknowledgments This work was partially supported by the CYTED Ciencia y Tecnología para el Desarrollo project [P318RT0165].

References

Ahmadi, E., Zandieh, M., Farrokh, M., & Emami, S. M. (2016). A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms. *Computers & Operations Research, 73*, 56–66.

Ahuja, R. K., & Orlin, J. B. (2001). Inverse optimization. *Operations Research, 49*(5), 771–783.

Al-Hinai, N., & ElMekkawy, T. Y. (2011). Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. *International Journal of Production Economics, 132*(2), 279–291.

Almada-Lobo, F. (2016). The Industry 4.0 revolution and the future of manufacturing execution systems (MES). *Journal of Innovation Management, 3*(4), 16–21.

Arnaout, J. P. (2014). Rescheduling of parallel machines with stochastic processing and setup times. *Journal of Manufacturing Systems, 33*(3), 376–384.

Battaïa, O., & Dolgui, A. (2013). A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics, 142*(2), 259–277.

Birge, J., Frenk, J. B. G., Mittenthal, J., & Kan, A. R. (1990). Single-machine scheduling subject to stochastic breakdowns. *Naval Research Logistics (NRL), 37*(5), 661–677.

Blazewicz, J., Ecker, K. H., Schmidt, G., & Weglarz, J. (2012). *Scheduling in computer and manufacturing systems*. Berlin: Springer Science & Business Media.

Bortolini, M., Ferrari, E., Gamberi, M., Pilati, F., & Faccio, M. (2017). Assembly system design in the Industry 4.0 era: A general framework. *IFAC-PapersOnLine, 50*(1), 5700–5705.

Brucker, P., & Shakhlevich, N. V. (2009). Inverse scheduling with maximum lateness objective. *Journal of Scheduling, 12*(5), 475–488.

Brucker, P., & Shakhlevich, N. V. (2011). Inverse scheduling: Two-machine flow-shop problem. *Journal of Scheduling, 14*(3), 239–256.

Dolgui, A., Ivanov, D., Sethi, S. P., & Sokolov, B. (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.

Dong, Y. H., & Jang, J. (2012). Production rescheduling for machine breakdown at a job shop. *International Journal of Production Research, 50*(10), 2681–2691.

Framinan, J. M., Leisten, R., & García, R. R. (2014). *Manufacturing scheduling systems. An integrated view on models, methods and tools*. New York: Springer.

Heuberger, C. (2004). Inverse combinatorial optimization: A survey on problems, methods, and results. *Journal of Combinatorial Optimization, 8*(3), 329–361.

Ivanov, D., & Das, A. (2020). Coronavirus (COVID-19 / SARS-CoV-2) and supply chain resilience: A research note. *International Journal of Integrated Supply Management*. in press.

Ivanov, D., & Dolgui, A. (2020). Viability of intertwined supply networks: Extending the supply chain resilience angles towards survivability. A position paper motivated by COVID-19 outbreak. *International Journal of Production Research*. in press.

Ivanov, D., Dolgui, A., & Sokolov, B. (2012). Applicability of optimal control theory to adaptive supply chain planning and scheduling. *Annual Reviews in Control, 36*(1), 73–84.

Ivanov, D., Dolgui, A., Sokolov, B., Werner, F., & Ivanova, M. (2016). A dynamic model and an algorithm for short-term supply chain scheduling in the smart factory industry 4.0. *International Journal of Production Research, 54*(2), 386–402.

Ivanov, D., Sethi, S., Dolgui, A., & Sokolov, B. (2018). A survey on control theory applications to operational systems, supply chain management, and Industry 4.0. *Annual Reviews in Control, 46*, 134–147.

Ivanov, D., Dolgui, A., & Sokolov, B. (2019). The impact of digital technology and Industry 4.0 on the ripple effect and supply chain risk analytics. *International Journal of Production Research, 57*(3), 829–846.

Katragjini, K., Vallada, E., & Ruiz, R. (2013). Flow shop rescheduling under different types of disruption. *International Journal of Production Research, 51*(3), 780–797.

Koulamas, C. (2005). Inverse scheduling with controllable job parameters. *International Journal of Services and Operations Management, 1*(1), 35–43.

Kuo, Y. H., & Kusiak, A. (2019). From data to big data in production research: The past and future trends. *International Journal of Production Research, 57*(15–16), 4828–4853.

Lee, E. A. (2008, May). Cyber physical systems: Design challenges. In *2008 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*. IEEE (pp. 363–369).

Lee, J., Bagheri, B., & Kao, H. A. (2015). A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters, 3*, 18–23.

Lee, J., Jin, C., & Bagheri, B. (2017). Cyber physical systems for predictive production systems. *Production Engineering, 11*(2), 155–165.

Liu, F., Wang, S., Hong, Y., & Yue, X. (2017). On the robust and stable flowshop scheduling under stochastic and dynamic disruptions. *IEEE Transactions on Engineering Management, 64*(4), 539–553.

Monostori, L. (2014). Cyber-physical production systems: Roots, expectations and R&D challenges. *Procedia CIRP, 17*, 9–13.

Monostori, L., Kádár, B., Bauernhansl, T., Kondoh, S., Kumara, S., Reinhart, G., Sauer, O., et al. (2016). Cyber-physical systems in manufacturing. *CIRP Annals, 65*(2), 621–641.

Ouelhadj, D., & Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling, 12*(4), 417.

Peng, K., Pan, Q. K., Gao, L., Zhang, B., & Pang, X. (2018). An improved artificial bee colony algorithm for real-world hybrid flowshop rescheduling in steelmaking-refining-continuous casting process. *Computers & Industrial Engineering, 122*, 235–250.

Pinedo, M. (2012). *Scheduling*. New York: Springer.

Rahmani, D., & Heydari, M. (2014). Robust and stable flow shop scheduling with unexpected arrivals of new jobs and uncertain processing times. *Journal of Manufacturing Systems, 33*(1), 84–92.

Rossit, D., & Tohmé, F. (2018). Scheduling research contributions to smart manufacturing. *Manufacturing Letters, 15*, 111–114.

Rossit, D. A., Tohmé, F., & Frutos, M. (2018). The non-permutation flow-shop scheduling problem: A literature review. *Omega, 77*, 143–153.

Rossit, D. A., Tohmé, F., & Frutos, M. (2019a). An Industry 4.0 approach to assembly line resequencing. *The International Journal of Advanced Manufacturing Technology, 105*(9), 3619–3630.

Rossit, D. A., Tohmé, F., & Frutos, M. (2019b). Industry 4.0: Smart scheduling. *International Journal of Production Research, 57*(12), 3802–3813.

Rossit, D. A., Tohmé, F., & Frutos, M. (2019c). A data-driven scheduling approach to smart manufacturing. *Journal of Industrial Information Integration, 15*, 69–79.

Rossit, D. A., Tohmé, F., & Frutos, M. (2019d). Production planning and scheduling in Cyber-Physical Production Systems: A review. *International Journal of Computer Integrated Manufacturing, 32*, 385–395.

Rossit, D. A., Tohmé, F., & Frutos, M. (2019e). Designing a scheduling logic controller for Industry 4.0 environments. *IFAC-PapersOnLine, 52*(13), 2164–2169.

Ruiz, R., & Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research, 165*(2), 479–494.

Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research, 205*(1), 1–18.

Sokolov, B., Dolgui, A., & Ivanov, D. (2018). Optimal control algorithms and their analysis for short-term scheduling in manufacturing systems. *Algorithms, 11*(5), 57.

Vieira, G. E., Herrmann, J. W., & Lin, E. (2003). Rescheduling manufacturing systems: A framework of strategies, policies, and methods. *Journal of Scheduling, 6*(1), 39–62.

Zandieh, M., & Gholami, M. (2009). An immune algorithm for scheduling a hybrid flow shop with sequence-dependent setup times and machines with random breakdowns. *International Journal of Production Research, 47*(24), 6999–7027.

Zhang, J., Ding, G., Zou, Y., Qin, S., & Fu, J. (2019). Review of job shop scheduling research and its new perspectives under Industry 4.0. *Journal of Intelligent Manufacturing, 30*(4), 1809–1830.