Chapter 6
Coupling Robust Optimization and Model-Checking Techniques for Robust Scheduling in the Context of *Industry 4.0*

Pascale Marangé, David Lemoine, Alexis Aubry, Sara Himmiche, Sylvie Norre, Christelle Bloch, and Jean-François Pétin

Abstract This chapter presents a generic methodology when considering robustness in production systems of *Industry 4.0*. It is the first milestone for coupling Operations Research models for robust optimization and Discrete Event Systems models and tools for property checking. The idea is to iteratively call Operations Research and Discrete Event Systems Models for converging towards a solution with the required robustness level defined by the decision-maker.

6.1 Introduction

In recent years, a new type of industry is emerging that aims to be more adaptable, agile, and flexible. This industry called "Industry 4.0" promises to adapt to the personalized needs of customers, thanks to the integration and generalization of new Information and Communication Technologies (IoT, Big Data, RFID, Digital Twin, etc.) into the production system such that new features can emerge:

- dynamical adaptation to the high market volatility and the need for tailor-made product solutions.

P. Marangé (✉) · A. Aubry · S. Himmiche · J.-F. Pétin
Université de Lorraine, CRAN, CNRS 7039, Nancy, France
e-mail: Pascale.Marangé@univ-lorraine.fr

D. Lemoine
IMT Atlantique, LS2N UMR CNRS 6004, Nantes, France
e-mail: david.lemoine@imt-atlantique.fr

S. Norre
Université Clermont Auvergne, LIMOS UMR CNRS 6158, Aubière, France
e-mail: sylvie.norre@uca.fr

C. Bloch
Université Bourgogne Franche-Comt., FEMTO-ST Institute, CNRS 6174, Montbéliard, France
e-mail: christelle.bloch@univ-fcomte.fr

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_6

- Communication with other systems and their environment.
- Distributed intelligence: each component is able to sense and to decide.

However some enablers are needed to support the realization of this new paradigm (Panetto et al. 2019). In particular, mass customization in shorter and shorter delay leads to a difficulty in knowing the quantity and type of demand, the flow of products and their fluctuations and thus increases perturbations into the production model due to the great diversity of the manufactured products and their shortened life cycle. Perturbations can be categorized as follows: (1) Uncertainties: as the difference between predicted and actual information (uncertainties about the volume of demand, the duration of operations, etc.); (2) Hazards are defined by the occurrence of uncontrollable Event in production or in the environment (machine failure, urgent order, etc.).

To incorporate perturbations into the problem, different types of models exist in the literature. (Ierapetritou and Jia 2007) have listed the three common models for integrating perturbations into production models: delimited form or scenario description, probability description or stochastic models, and fuzzy modeling.

The main disadvantage of stochastic models was the need to have knowledge about historical data for identifying the right probability distribution and its parameters. However, *Industry 4.0* and integration of big data technologies promise to have access to data coming from the shop-floor such that this historical data and their analysis should help to build the right stochastic model of perturbations.

Such models can be integrated into classical Operations Research Models for Robust Optimization (Bertsimas and Sim 2004). The Operations Research models and associated solution tools are particularly efficient for tending towards the optimal solution despite the complexity of the problem. But the counterpart of this efficiency is often a dedicated static model whose price of adaptation when considering a new characteristic can be very high. By essence, production systems in Industry 4.0 will be highly dynamical and reconfigurable. Discrete Event Systems (DES) models and tools are particularly efficient to capture and model the dynamics of a system through the modeling of states and Event (Cassandras and Lafortune 2009).

The objective of this chapter is to present a generic methodology to assess the impact of perturbations into production systems in order to define a solution with a good balance between performance and robustness. This methodology is the first milestone for combining the advantages of robust optimization and Discrete Event Systems models and tools. The idea beside is to iteratively call robust optimization and Discrete Event Systems models for reaching the robustness level required by the decision-maker. This methodology is shown to be relevantly applied in the context of robust production scheduling when considering uncertainties on operation execution durations.

This chapter is built as follows: the first section presents the generic hybrid approach between Operations Research models for robust optimization and Discrete Event Systems models and tools for property verification. The second section presents the instantiation of this methodology to a scheduling problem under perturbations in a workshop with parallel machines. The third section illustrates and discusses the results on a use case. Finally the last section concludes the chapter by recalling the obtained results and by opening the discussion considering general considerations about perturbations and *Industry 4.0*.

6.2 A Hybrid Approach for Optimization Under Perturbations

In this section, we begin by introducing Linear Programming and robustness, then Discrete Event Systems concepts are also presented. We finish by describing the proposed methodology to deal with the robustness level wanted by the decision-maker for a solution, thanks to a combination of a robust linear programming approach and Discrete Event Systems Models.

6.2.1 Linear Programming and Robustness

Linear programming is one of the most powerful tools in Operations Research. It allows to model a wide variety of practical problems (particularly in logistics) and is often able to solve them to optimality. Among these logistic problems, we can quote scheduling, production planning, vehicle routing, time tabling, etc.

According to Papadimitriou and Steiglitz (1998), a Mixed Integer Linear Programming (MILP) can be expressed as:

$$\text{Maximize} \quad \sum_{j=1}^{n} c_j x_j \tag{6.1}$$

s.t.

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad \forall i = 1, \cdots, m \tag{6.2}$$

$$x_j \in \mathbb{N} \quad \forall j = 1, \cdots, p \tag{6.3}$$

$$x_j \in \mathbb{R}_+, \quad \forall j = p+1, \cdots, n \tag{6.4}$$

where

- $(c_j)_{j=1\ldots,n}$, $(b_i)_{i=1\ldots,m}$, $(a_{ij})_{(i,j)=\{1\ldots,m\}\times\{1\ldots,n\}}$ are real variables which represent the problem's parameters (for instance, costs, distances, capacities, etc.).
- $X = (x_j)_{j=1\ldots,n}$ are the decision variables. They represent the solution we seek to determine.

- Function (6.1) is a linear form which represents the criterion we seek to optimize (in this case, maximize. For instance, it can be some logistics costs, customer's satisfaction, etc.).
- Equation (6.2) is a set of affine constraints that any solution of the modeled problem must satisfy (it describes the problem specificities).
- Equations (6.3) and (6.4) are integrity and positivity constraints.

For more information about linear programming, readers can also refer to Chvátal (1983), Wolsey (1998), and Nemhauser and Wolsey (1999).

Usually, when using such modeling, all parameters are assumed to be well known and deterministic. Nevertheless, this situation is very rarely encountered in real life. Therefore, solutions determined by this method may be unrealistic in practice. To avoid this, one possibility is to introduce uncertainties on parameters in order to better model reality and to try to find solutions able to absorb these perturbations without unreasonably degrading their quality. This kind of approach is usually referred to as robust (Billaut et al. 2013).

In linear programming, several robust approaches have been designed depending on the type of parameters on which the uncertainties fall on. Here, we focus on issues where uncertainties are related to (a_{ij}) parameters. More precisely, we assume that each parameter a_{ij} takes its values in a bounded interval $\left[\bar{a}_{ij} - \hat{a}_{ij}, \bar{a}_{ij} + \hat{a}_{ij}\right]$. That is to say that there is a random real variable ζ_{ij} which takes its values in $[-1, 1]$ such that

$$a_{ij} = \bar{a}_{ij} + \zeta_{ij}\hat{a}_{ij}$$

Thus, according to these assumptions, a MILP that takes into account these uncertainties can be formalized as follows:

$$\text{Maximize} \sum_{j=1}^{n} c_j x_j \tag{6.5}$$

s.t.

$$\sum_{j=1}^{n} \bar{a}_{ij} x_j + \sum_{j=1}^{n} \zeta_{ij}\hat{a}_{ij} x_j \leq b_i \quad \forall i = 1, \cdots, m \tag{6.6}$$

$$x_j \in \mathbb{N} \quad \forall j = 1, \cdots, p \tag{6.7}$$

$$x_j \in \mathbb{R}_+, \quad \forall j = p+1, \cdots, n \tag{6.8}$$

where $\sum_{j=1}^{n} \zeta_{ij}\hat{a}_{ij} x_j$ models the uncertainty in constraint (6.6).

The main idea of robust approaches presented in this chapter is to try to reasonably protect oneself from this uncertainty by taking into account the risk, thanks to a set of deterministic functions $\left(\beta_i^{\Omega_i}(x)\right)_{i=1,\cdots,m}$, where $(\Omega_i)_{i=1,\cdots,m}$ are parameters tuned in order to meet the degrees $\left(\Gamma_i^{ref}\right)_{i=1,\cdots,m}$ of protection the decision-maker wants to implement, depending on the criticality of the constraint.

In other words, $(\Omega_i)_{i=1,\cdots,m}$ have to be set up to be sure that the probability that the uncertainty does not exceed $\beta_i^{\Omega_i}(x)$ is greater or equal to Γ_i^{ref}, for $i = 1, \cdots, m$:

$$\mathbb{P}\left[\sum_{j=1}^{n} \zeta_{ij}\hat{a}_{ij}x_j \leq \beta_i^{\Omega_i}(x)\right] \geq \Gamma_i^{ref}, \quad \forall i = 1, \cdots, m \tag{6.9}$$

Thus, if one solution $(\Omega_i)_{i=1,\cdots,m}$ can be set up such that Eq. (6.9) is satisfied, solving the following optimization problem will ensure to have a solution $X = \left(x_j\right)_{j=1,\ldots,n}$ which can resist to uncertainty with degrees wanted by the decision-maker:

$$\text{Maximize} \sum_{j=1}^{n} c_j x_j \tag{6.10}$$

s.t.

$$\sum_{j=1}^{n} \bar{a}_{ij}x_j + \beta_i^{\Omega_i}(x) \leq b_i \quad \forall i = 1, \cdots, m \tag{6.11}$$

$$x_j \in \mathbb{N} \quad \forall j = 1, \cdots, p \tag{6.12}$$

$$x_j \in \mathbb{R}_+, \quad \forall j = p+1, \cdots, n \tag{6.13}$$

In Bertsimas and Sim (2004), the authors propose to use the following set of functions:

$$\beta_i^{\Omega_i}(x) = \max_{\sum_{j=1}^{n}|\zeta_{ij}| \leq \Omega_i} \left(\sum_{j=1}^{n} \zeta_{ij}\hat{a}_{ij}x_j\right) \tag{6.14}$$

and they prove that this non-linear formulation can be linearized and is equivalent to a MILP. Thus traditional Linear Programming technics can be used for solving the initial problem. This kind of MILP is called a Robust Linear Programming Model.

Nevertheless, tuning $\Omega = (\Omega_i)_{i=1,\cdots,m}$ for satisfying (6.9) can be very difficult. In Bertsimas and Sim (2004), the authors show that if for all i, each ζ_{ij} is independent and symmetrically distributed in $[-1, 1]$, Ω_i can be analytically determined. But, such a hypothesis is not often verified in real industrial problems.

6.2.2 Discrete Event Systems Models for Evaluating Solution Robustness

Industrial systems can be modeled by Discrete Event Systems (DES) that allow a representation of the behavior of a system by considering the state and Event that allow it to evolve. The event is seen as an instantaneous occurrence of an action or phenomenon in the system environment. Changes due to the event can be deterministic when the behavior is known with certainty or stochastic when the occurrence of an event can lead to different states. These modeling tools can be Petri Nets, State Automata, Statecharts, Bayesian Networks (Cassandras and Lafortune 2009).

To model the behavior of industrial systems and perturbations, we should be able to represent many dynamic characteristics such as the communication between elements of the workshop (jobs, resources), the time, and the probabilistic behavior of perturbations. Many stochastic Discrete Event Systems languages allow the modeling of these characteristics. For instance, Stochastic Petri Nets (Chiola et al. 1993), Stochastic Automata (Alur and Dill 1994), Stochastic Automata Networks (Plateau and Atif 1991).

The language chosen here is the Stochastic Timed Automata (STA). In fact, it is an extension of the well-known Timed Automata (Alur and Dill 1994) which is enriched with shared variables, synchronizing Event and probabilistic characteristics (Larsen et al. 1997).

Definition 6.1 Formally a Stochastic Timed Automaton is presented as the following n-tuple:
$A = (L, V, E, C, Inv, Pr, T, L_m, l_0, v_0)$ where

- L is a finite set of locations.
- V is a finite set of variables.
- E is a finite set of synchronizing Event
- C is a finite set of clocks.
- Inv is a set of invariants (conditions in location).
- Pr is a set of probabilities: (i) discrete for the set of transitions (from a location, probabilistic transitions allow to attend different locations l_i with a given probability p_i, with $\sum p_i = 1$). (ii) Continuous for the variables (the crossing condition of a transition is defined randomly by a probability distribution).
- T is a finite set of transitions $(l, e, g, m, l') \in L \times E \times G \times M \times L$, where l and l' are, respectively, the starting and arriving locations. On a transition, three optional elements are defined: (i) a guard (condition on variables) g from the set of guards G, (ii) an update (on variables) m from the set of updates M, and (iii) a synchronizing event e from the set E.
- $L_m \subseteq L$ is the set of marked locations.
- $l_0 \in L$ is the initial location of the automaton.
- v_0 is the initialization vector of variables.

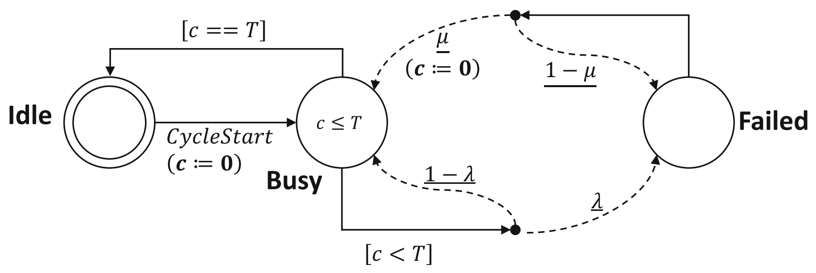


Fig. 6.1 A Stochastic Timed Automaton representing a machine

The elements of a STA can be graphically represented as follows (see example in Fig. 6.1). Locations are represented by vertices and transitions by arcs. An initial active location is represented by a double vertex. The invariants are represented inside the associated vertex (location). Guards are represented between brackets "[ ]". Synchronizing Event is represented in italics. The update of variables and clocks are represented between parenthesis "( )". Discrete probabilities are modeled by dotted arcs and associated probabilistic values are underlined. For continuous probabilities, they are directly linked with the definition of variables.

The automaton $MACHINE$ in Fig. 6.1 represents the behavior of a machine that can be subjected to failures. For this purpose, the machine can be represented by three states: Idle, Busy, Failed. Moreover, the failure rate is represented by λ and the repair rate by μ.

Initially, the MACHINE is in the location **Idle** waiting for the $CycleStart$ event. After the occurrence of this event, the local clock c is initialized ($c := 0$) and the MACHINE becomes **Busy**, i.e. is used for executing a cycle (that lasts normally T time units). Before T times units, a failure may occur with a probability λ (the MACHINE reaches the location **Failed**) or the machine continues its cycle with the probability $1 - \lambda$. In the location **Failed**, the MACHINE is repaired with the probability μ and, in this case, the cycle restarts to zero ($c := 0$). In this example, a failure has a big impact because the cycle restarts from zero after being repaired.

Discrete Event systems models can be used either to control, i.e. to inhibit certain state transitions to avoid unwanted behaviors, or to evaluate performance, i.e. to check properties such as the reachability of a state or the execution of an events sequence. This "Model-Checking" ability can be used for evaluating the impact of a perturbation on a system.

Properties that are traditionally desirable for an industrial system concern the reliability, maintainability, and safety. And DES models and tools are usually used for evaluating these properties. For instance, in Morel et al. (2009), Reliability is defined as *the ability of a device or system to perform a required function under stated conditions for a specified period of time*. This property is often measured by the probability $R(t)$ that a system will operate without failure before time t (depending on the failure rate λ), i.e. the probability that the Time To Failure TTF is greater than the time t:

$$R(t) = \mathbb{P}(TTF > t) \tag{6.15}$$

Now, we will show how DES models and tools can be used for evaluating reliability. In the example in Fig. 6.1, reliability can be the probability that the **Failed** state will be never reached before a cycle time T (meaning $TTF > T$, such that the failure does not happen during the cycle, if not the cycle has to restart from zero). For stochastic DES models, such a property can be expressed in PCTL (Probabilistic Computation Tree Logic). This language is a probabilistic extension of CTL (Computation Tree Logic) (Baier and Kwiatkowska 1998). This type of logic allows to express properties like "*What is the probability that the model is in the state $Failed$, in the precise interval [0,T]?*" This question can be transcribed in PCTL as in the expression (6.16).

$$P = ?[F \leq T \text{ ``}MACHINE.Failed''] \tag{6.16}$$

where P =? means that we want to assess the probability that the property that is inside the brackets [ ] is reached. This property can be translated as follows:

- $F \leq T$ means "There exists in the future in a time that is less or equal to T."
- "MACHINE.Failed" means "a state where the stochastic timed automata MACHINE is in the marked location Failed."

Finally the obtained result assesses $1 - R(T)$. So Model-Checking can be used for evaluating the reliability of a system. And we could do the same for the maintainability and safety.

The implementation of property verification is done by model-checking. The input of the model-checker is a system model and a property. At the output, the model-checker indicates whether the property is checked and, if not, a counter-example is returned (i.e., an example that shows that the property is not checked). In the case of stochastic system modeling, model-checking can be done numerically or statistically:

- Numerical model-checking uses accurate valuation methods to determine the probability value of a property. This type of model control ensures the accuracy of the given solution, but it is not suitable for large problems.
- Statistical model-checking generates different execution paths and verifies, after each execution, the satisfaction of a property. Statistical model-checking is similar to the Monte Carlo simulation. Monte Carlo simulation is a method for estimating a numerical quantity using random numbers. At each simulation step, the expectation of the variable is calculated and the simulation stops when the statistical parameters are satisfied. This avoids the combinatorial explosion and is therefore adapted to check real systems (Ballarini et al. 2011).

Actually, reliability can be seen as a robustness property. The notion of robustness has different definitions in literature that converge to the same idea: a robust system should maintain or guarantee some performances despite perturbations and variations generated by the system or its environment (Billaut et al. 2013).

When considering perturbations modeled as stochastic variables, the concept of “service level” can be used for assessing the robustness (Dauzères-Pérès et al. 2010). A service level $\mathcal{SL}$ is defined as *the probability that a criterion is smaller (resp. larger) or equal to a given value*. Thus, the assessed robustness level $\mathcal{SL}$ can be translated as the probability $\mathbb{P}$ that a system state z is lower than a value z_{max} (or larger than a value z_{min}) as in the following equation:

$$\mathcal{SL} = \mathbb{P}(z \leq z_{max}) \tag{6.17}$$

We can see that reliability falls within this definition. DES models and tools are thus good candidates for evaluating the robustness level of a system.

6.2.3 Proposed Methodology for Combining the Two Approaches

As said before, fine-tuning the $\Omega = (\Omega_i)_{i=1,\cdots,m}$ for satisfying (6.9) can be very difficult in general cases. Therefore, we propose a methodology for iteratively and numerically tuning Ω, thanks to Discrete Event Systems Models and associated Model-Checking tools. Figure 6.2 sketches the proposed approach.

Here, we suppose that:

- the problematic we want to solve and which relies on the system-of-interest can be formalized by a Mixed Integer Linear Programming model,
- the system-of-interest and its dynamic we want to study can be modeled, thanks to a Discrete Event Systems Model (DES),
- the decision-maker is able to define the different robustness indicators $\left(\Gamma_i^{ref}\right)_{i=1,\ldots,m}$ for each constraint i that must be satisfied by the solution X.

To determine the parameters $(\Omega_i)_{i=1,\cdots,m}$ that lead to the robustness level wanted by decision-makers, we proposed a methodology based on the following three iterative modules:

Module 1: ***The Operations Research Module (OR Module)***. According to the current value of $(\Omega_i)_{i=1,\cdots,m}$, the Robust Mixed Integer Programming Model using Bertsimas and Sim’s framework is designed (Bertsimas and Sim 2004). Then, this is input into a Solver to get the optimal solution taking into account the robustness parameters. The obtained solution X is then sent to the second Module.

Module 2: ***The Discrete Event Systems Module (DES Module)***. Considering the system-of-interest and the solution proposed by the Operations Research module, a Stochastic Timed Automata model is first designed. Then, the different robustness levels Γ_i (as instantiations of the service level $\mathcal{SL}$) to be assessed are defined as properties to be checked on the resulting

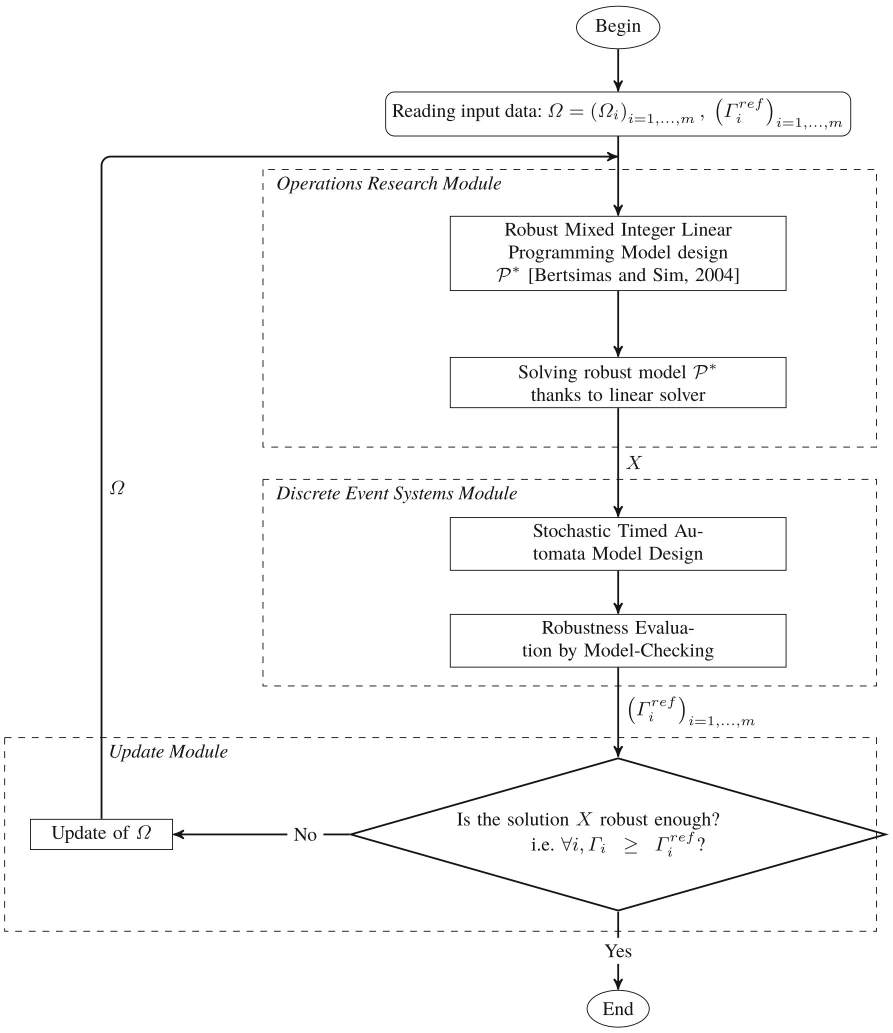

Begin
Reading input data: $\Omega = (\Omega_i)_{i=1,\dots,m}$, $\left(\Gamma_i^{ref}\right)_{i=1,\dots,m}$
Operations Research Module
Robust Mixed Integer Linear Programming Model design $\mathcal{P}^*$ [Bertsimas and Sim, 2004]
Solving robust model $\mathcal{P}^*$ thanks to linear solver
X
Ω
Discrete Event Systems Module
Stochastic Timed Automata Model Design
Robustness Evaluation by Model-Checking
$\left(\Gamma_i^{ref}\right)_{i=1,\dots,m}$
Update Module
Update of Ω
No
Is the solution X robust enough? i.e. $\forall i, \Gamma_i \geq \Gamma_i^{ref}$?
Yes
End

Fig. 6.2 The proposed methodology

model by a model-checker. The resulting $(\Gamma_i)_{i=1,\dots,m}$ are sent to the third module.

Module 3: ***Update Module***. Depending on the robustness levels $(\Gamma_i)_{i=1,\dots,m}$ assessed by the Discrete Event Systems Module, if the robustness levels required by the decision-maker are reached, then the process is stopped and the solution is given. Otherwise, $(\Omega_i)_{i=1,\cdots,m}$ are updated and sent back to the Operations Research Module for a new iteration.

6.3 Application to the Problem of Scheduling Under Perturbations

6.3.1 Scheduling Under Perturbations

The issue of production scheduling is an important decision-making problem in industrial processes. Actually, to guarantee the production performances, the decision-maker has to find an adapted schedule to its production system and the associated constraints. A production scheduling problem consists usually in (1) allocating the workshop resources to operations needed to make the jobs, (2) sequencing the operations on resources (defining the execution order of operations on resources), and (3) eventually defining the starting and ending dates of each operation. The schedule obtained should satisfy the workshop constraints (precedence constraints, non-preemption of operations, etc.). Indeed, each type of workshop has its own constraints in order to satisfy the production objective (like minimizing the total completion time of operations, number of late jobs, production cost, etc.).

Monostori et al. (2016) consider robust scheduling as one of the six main challenges in Research and Development for Cyber-Physical Production Systems. Others like Zhong et al. (2017) prefer to talk about a need of intelligent scheduling able to generate, from captured data, a reliable schedule in real time.

6.3.2 Instantiation of the Approach to Scheduling Under Perturbations

Here, we present an illustration of our methodology applied to a scheduling problem in a production area composed of two non-identical parallel machines: this means that the machines can perform the same operations but with different processing times. Then scheduling problem in this production cell involves both machine allocation and sequencing, rather than simply sequencing (Mokotoff 2001). Figure 6.3 shows the considered production cell.

In our case, we seek to minimize the completion time of the last scheduled job: this criterion is usually called the Makespan and is denoted as C_{max}. The main assumptions of our problem are the following:

- All jobs are available at time 0,
- The two machines are always available (no breakdown, . . .),
- Processing times for the jobs are independents,
- A machine cannot process more than one product at any time.

This problem which is referred to as $R2||C_{max}$ has been shown to be NP-hard in the weak sense (Lenstra et al. 1977). Here, we also assume that processing times are not deterministic.

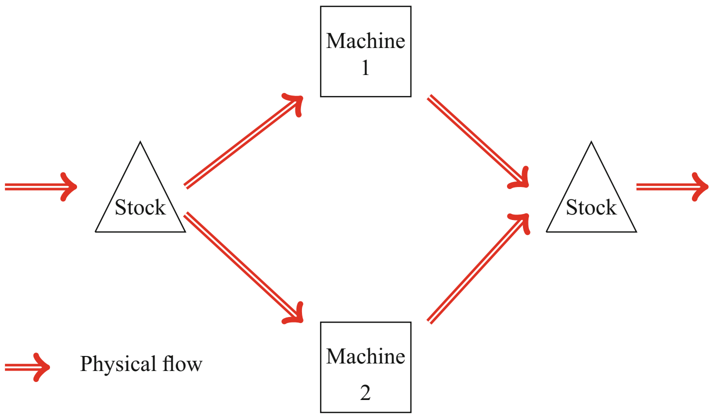


Fig. 6.3 The production area

6.3.2.1 Instantiation of the Methodology

The methodology presented in Fig. 6.2 can be instantiated as in Fig. 6.4. The MILP formulation of the problem $R2||C_{max}$ under uncertainties is presented in Sect. 6.3.2.2. The DES Module is presented in Sect. 6.3.2.3. The Update Module is presented in Sect. 6.3.2.4.

6.3.2.2 Operations Research Module

First, we give the MILP formulation of this scheduling problem when all the processing times are deterministics.

The parameters of the model are given in Table 6.1.

The decision variables are summarized in Table 6.2.

The $R2||C_{max}$ problem can be formulated as follows:

$$\text{Minimize } C_{max} \tag{6.18}$$

s.t.

$$\sum_{k=1}^{2} x_{jk} = 1 \quad \forall j \in \{1, \dots, N\} \tag{6.19}$$

$$C_{max} - \sum_{j=1}^{N} t_{jk} x_{jk} \geq 0 \quad \forall k \in \{1, 2\} \tag{6.20}$$

$$C_{max} \geq 0 \tag{6.21}$$

$$x_{jk} \in \{0, 1\} \quad \forall (j, k) \in \{1, \dots, N\} \times \{1, 2\} \tag{6.22}$$

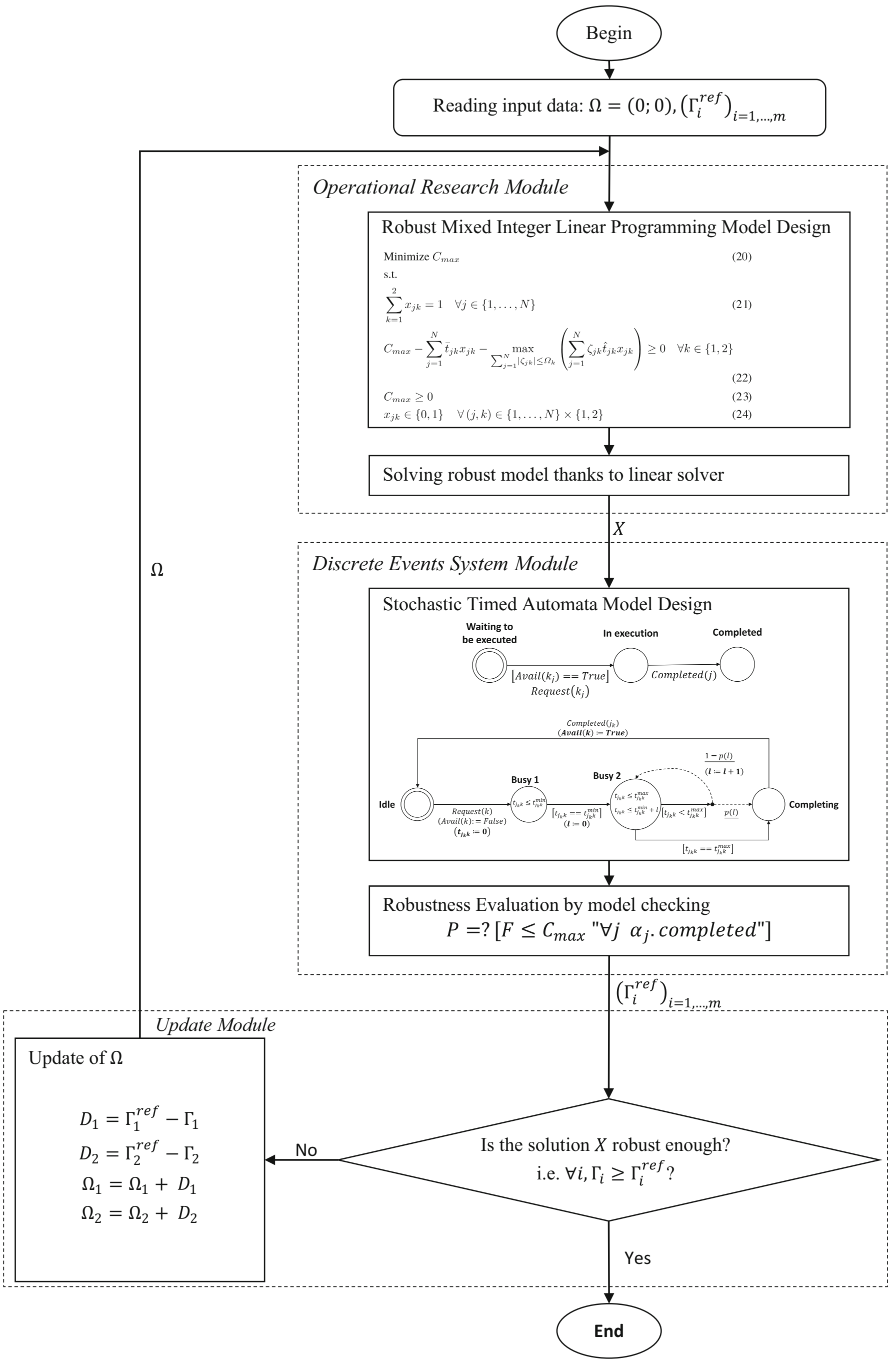

Begin
Reading input data: $\Omega = (0;0), (\Gamma_i^{ref})_{i=1,\dots,m}$
Operational Research Module
Robust Mixed Integer Linear Programming Model Design
Minimize C_{max} (20)
s.t.
$\sum_{k=1}^{2} x_{jk} = 1 \quad \forall j \in \{1,\dots,N\}$ (21)
$C_{max} - \sum_{j=1}^{N} \bar{t}_{jk}x_{jk} - \max_{\sum_{j=1}^{N}|\zeta_{jk}|\le\Omega_k} \left(\sum_{j=1}^{N} \zeta_{jk}\hat{t}_{jk}x_{jk}\right) \ge 0 \quad \forall k \in \{1,2\}$ (22)
$C_{max} \ge 0$ (23)
$x_{jk} \in \{0,1\} \quad \forall (j,k) \in \{1,\dots,N\} \times \{1,2\}$ (24)
Solving robust model thanks to linear solver
X
Ω
Discrete Events System Module
Stochastic Timed Automata Model Design
Waiting to be executed
In execution
Completed
$[Avail(k_j) == True]$ $Request(k_j)$
$Completed(j)$
$Completed(j_k)$ $(\boldsymbol{Avail(k) := True})$
Idle
Busy 1
Busy 2
Completing
$t_{j_k k} \le t_{j_k k}^{min}$
$t_{j_k k} \le t_{j_k k}^{max}$
$t_{j_k k} \le t_{j_k k}^{min} + l$
$Request(k)$ $(Avail(k) := False)$ $(\boldsymbol{t_{j_k k} := 0})$
$[t_{j_k k} == t_{j_k k}^{min}]$ $(\boldsymbol{l := 0})$
$[t_{j_k k} < t_{j_k k}^{max}]$
$\underline{1 - p(l)}$ $(\boldsymbol{l := l + 1})$
$\underline{p(l)}$
$[t_{j_k k} == t_{j_k k}^{max}]$
Robustness Evaluation by model checking
$P =? [F \le C_{max}$ "$\forall j \ \alpha_j.completed$"$]$
$(\Gamma_i^{ref})_{i=1,\dots,m}$
Update Module
Update of Ω
$D_1 = \Gamma_1^{ref} - \Gamma_1$
$D_2 = \Gamma_2^{ref} - \Gamma_2$
$\Omega_1 = \Omega_1 + D_1$
$\Omega_2 = \Omega_2 + D_2$
No
Is the solution X robust enough? i.e. $\forall i, \Gamma_i \ge \Gamma_i^{ref}$?
Yes
End

Fig. 6.4 Instantiated methodology to $R2||C_{max}$

Table 6.1 Parameters of the model

N: Number of jobs we have to schedule
t_{jk}: Processing time for job j on the machine k, $(j,k) \in \{1,\ldots,N\} \times \{1,2\}$

Table 6.2 Decision variables of the model

$X = \left(x_{jk}\right)_{jk}$:	$\begin{cases} x_{jk} = 1 & \text{if machine } k \text{ is allocated to job } j \\ x_{jk} = 0 & \text{otherwise} \end{cases}$ $(j,k) \in \{1,\ldots,N\} \times \{1,2\}$
C_{max}	: is the makespan value

Equation (6.18) is the objective function we seek to minimize. Equation (6.19) ensures that every job is executed by a single machine. Constraint (6.20) requires that total completion time C_{max} is higher than the completion time on each machine. Equations (6.21) and (6.22) are positivity and integrity constraints.

Now, we suppose that there are some uncertainties related to the jobs' processing times. As presented in the robustness section, there is a random variable ζ_{jk} which takes its values in $[-1, 1]$ such that

$$t_{jk} = \bar{t}_{jk} + \zeta_{jk}\hat{t}_{jk}$$

According to the Bertsimas and Sim (2004) approach, we can formulate the robust model as follows:

$$\text{Minimize } C_{max} \tag{6.23}$$

s.t.

$$\sum_{k=1}^{2} x_{jk} = 1 \quad \forall j \in \{1,\ldots,N\} \tag{6.24}$$

$$C_{max} - \sum_{j=1}^{N} \bar{t}_{jk}x_{jk} - \max_{\sum_{j=1}^{N}|\zeta_{jk}| \le \Omega_k} \left(\sum_{j=1}^{N} \zeta_{jk}\hat{t}_{jk}x_{jk} \right) \ge 0 \quad \forall k \in \{1,2\} \tag{6.25}$$

$$C_{max} \ge 0 \tag{6.26}$$

$$x_{jk} \in \{0,1\} \quad \forall (j,k) \in \{1,\ldots,N\} \times \{1,2\} \tag{6.27}$$

In this context, Ω_k represents the maximal deviation (using the $\mathcal{L}^1$-Norm) that is taking into account in the model for each machine k. If $\Omega_k = 0$, which means that no uncertainties are taken into account. In fact, the constraints (6.25) become equivalent to the constraints (6.20) and the robust formulation becomes equivalent to the deterministic formulation. On the contrary, if we want to consider all the uncertainties, Ω_k must be chosen as equal to N. If it is the case, the most conservative solution will be obtained. In fact, the constraints (6.25) become equivalent to the following:

$$C_{max} - \sum_{j=1}^{N} \bar{t}_{jk} x_{jk} - \sum_{j=1}^{N} \hat{t}_{jk} x_{jk} \geq 0, \ \forall k \in \{1, 2\} \tag{6.28}$$

Thus, this corresponds to the worst-case formulation: i.e. the most conservative, considering that the worst case (all the ζ_{jk} are equal to 1) is more important than the other cases.

Here, the idea is to fix Ω_k (as a "maximum amount of deviation" on the operation durations) but with guaranteeing that the desirable robustness levels Γ_k^{ref} are reached.

6.3.2.3 Discrete Event Systems Module

This section presents the DES formulation such that the allocation $X = \left[x_{jk}\right]_{jk}$ resulting from the solving of the robust MILP formulation given in the previous section can be evaluated regarding its robustness level and the result is sent to the Update Module for updating accordingly $(\Omega_k)_{k=1,2}$ (and a new iteration is launched) or not. The DES module contains two steps:

Step 1: Stochastic Timed Automata Model design: using STA for modeling the behavior of jobs and machines when executing the allocation X.

Step 2: Robustness evaluation by Model-Checking: evaluating the robustness levels Γ_k associated with each machine k.

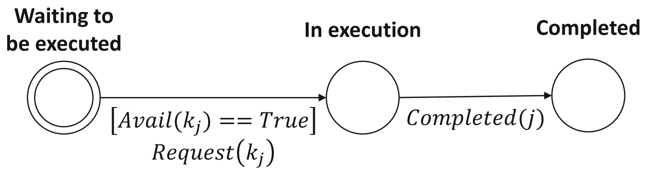
Stochastic Timed Automata Model Design

First we propose to model the behavior of the jobs and the machines when they are not subjected to uncertainties. We define a job pattern that will be instantiated for each job j and a machine pattern that will be instantiated for each machine k.

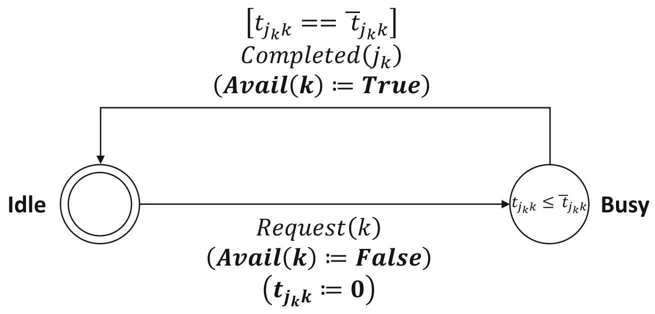
In the following, we denote as k_j the machine that is allocated to the job j (defined by X coming from the Operations Research Module). Formally $k_j = \sum_k x_{jk}.k$.

The job pattern (named α_j) is presented in Fig. 6.5a. First, in the **Waiting to be executed** location, the job j waits the availability of its allocated machine k_j (through the guard $\left[Avail\left(k_j\right) == True\right]$). When the guard is satisfied, the job pattern sends a request to the machine pattern (by the synchronizing event $Request\left(k_j\right)$) and reaches the **In execution** location waiting for its completion (the reception of the synchronizing event $Completed\,(j)$). After its completion, the job reaches the **Completed** location.

The machine pattern is represented in Fig. 6.5b. In this model, the job that is going to be executed is denoted as j_k. First, in the **Idle** location, the machine waits a request from a job (the synchronizing event $Request\,(k)$) and then reaches the **Busy** location after updating its availability status ($Avail(k) := False$) and initializing the local clock $t_{j_k k}$ to 0. In the **Busy** location, the machine executes the job until the



(a)


$[t_{j_k k} == \bar{t}_{j_k k}]$
$Completed(j_k)$
$(\boldsymbol{Avail(k) := True})$
Idle
Busy
$t_{j_k k} \leq \bar{t}_{j_k k}$
$Request(k)$
$(\boldsymbol{Avail(k) := False})$
$(\boldsymbol{t_{j_k k} := 0})$

(b)

Fig. 6.5 STA models of job and machine without considering perturbations. (**a**) Job STA: α_j. (**b**) Deterministic machine STA

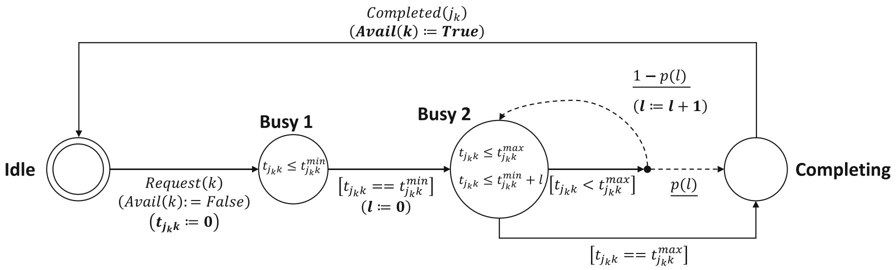

$Completed(j_k)$
$(\boldsymbol{Avail(k) := True})$
$1 - p(l)$
$(\boldsymbol{l := l + 1})$
Busy 1
Busy 2
Idle
$t_{j_k k} \leq t_{j_k k}^{min}$
$t_{j_k k} \leq t_{j_k k}^{max}$
$t_{j_k k} \leq t_{j_k k}^{min} + l$
Completing
$Request(k)$
$(Avail(k) := False)$
$(\boldsymbol{t_{j_k k} := 0})$
$[t_{j_k k} == t_{j_k k}^{min}]$
$(\boldsymbol{l := 0})$
$[t_{j_k k} < t_{j_k k}^{max}]$
$p(l)$
$[t_{j_k k} == t_{j_k k}^{max}]$

Fig. 6.6 Perturbed machine STA

local clock reaches the deterministic duration $\bar{t}_{j_k k}$. When the duration is reached, the job can be informed of its completion (by the synchronizing event $Completed\ (j_k)$) and the machine updates its availability status ($Avail(k) := True$). The machines then go back to the **Idle** location.

Now, we integrate the uncertainties on the job duration into the machine pattern. The resulting updated machine pattern is represented in Fig. 6.6. In the following, we denote the upper value of the job duration as $t_{j_k k}^{max} = \bar{t}_{j_k k} + \hat{t}_{j_k k}$ and the lower value of the job duration as $t_{j_k k}^{min} = \bar{t}_{j_k k} - \hat{t}_{j_k k}$.

In the **Busy 1** location, the machine waits to reach the minimal duration $t_{j_k k}^{min}$ (through the guard $\left[t_{j_k k} == t_{j_k k}^{min}\right]$). Moreover, the iteration counter l is initiated to

0. The idea is to let the duration increase according to a discrete probability $p(l)$ that is evolving depending on the iterations number l. In the **Busy 2** location, if the maximal duration is reached ($\left[t_{j_k k} == t_{jk}^{max}\right]$), then the machine reaches the **completing** location. If it is not the case ($\left[t_{j_k k} < t_{jk}^{max}\right]$), there are two possible probabilistic choices: (1) with the probability $1 - p(l)$, the duration can increase and the iteration counter is updated ($l := l + 1$) or (2) with the probability $p(l)$, the current duration is the final duration.

Finally, the probability that $t_{j_k k} = t_{j_k k}^{min} + l$ is the probability to loop into the **Busy 2** location $l - 1$ times and to get out from the loop in the l^{th} iteration.

Actually, $p(l)$ is a probabilistic parameter that can be calculated from the probability distribution followed by $t_{j_k k}$.

Modeling the execution of the job as previously presented allows to not be restricted to any kind of probability distribution (symmetric or not, discrete or not, etc.). We could even imagine to cut the interval of the job duration in several sub-intervals in which the probability distributions could be different. That makes this approach a good complement to the robust linear programming of the Operations Research module.

Robustness Evaluation by Model-Checking

In the second step, model-checking tools are used to assess the robustness level of X.

In a scheduling problem, we can instantiate the service level presented in Eq. (6.17) as follows: z is the total completion time despite the considered uncertainties and z_{max} is the referential completion time C_{max} associated with X given by the Operations Research module. So we define the robustness level as the probability that the executed makespan is smaller or equal than the referential completion time C_{max}. Formally, this metric is given by Eq. (6.29):

$$\mathcal{SL} = \mathbb{P}\left(C_{max}\left(X, U\right) \leq C_{max}\right) \tag{6.29}$$

where $C_{max}\left(X, U\right)$ is the executed makespan of an allocation X subjected to uncertainties U.

So to assess the value of $\mathcal{SL}$ using DES models and associated Model-Checking, the property to check is: "*What is the probability that all the paths lead to a global state where all the job models α_j are in the marked location **Completed** in a time that is less or equal to* C_{max}?"
Using PCTL, this property can be expressed as follows:

$$P =? \left[F \leq C_{max} \text{``}\forall j\ \alpha_j.Completed''\right] \tag{6.30}$$

where $P = ?$ means that we want to assess the probability that the property that is inside the brackets [ ] is reached. This property can be translated as follows:

- "$F \leq C_{max}$" means "There exists in the future in a time that is less or equal to C_{max}."
- "$\forall j \ \alpha_j.Completed$" means "a state where, for all j, all the stochastic timed automata α_j are in the marked location **Completed**."

That means that the formula $\left[F \leq C_{max} \text{ ``}\forall j \ \alpha_j.Completed\text{''}\right]$ is a PCTL expression for: $C_{max}(X, U) \leq C_{max}$.

Here, two robustness levels associated, respectively, with each machine can be defined. They consist to consider only the uncertainties are only taken into account on machine 1 or machine 2. So, we can evaluate which machine is more sensitive than the other. These two robustness levels are defined as follows:

$$\begin{aligned} \Gamma_1 &= \mathbb{P}\left(C_{max}\left(X, \left(\hat{t}_{j1}\right)_{j1}\right) \leq C_{max}\right) \\ \Gamma_2 &= \mathbb{P}\left(C_{max}\left(X, \left(\hat{t}_{j2}\right)_{j2}\right) \leq C_{max}\right) \end{aligned} \tag{6.31}$$

where $\left(\hat{t}_{j1}\right)_{j1}$ (resp. $\left(\hat{t}_{j2}\right)_{j2}$) are the uncertainties on the operation durations when considering that there are no uncertainties on the machine 2 (resp. 1). Finally, these robustness levels assess whether the inequation (6.9) is satisfied or not. This result is used in the Update Module for updating or not Ω.

Moreover, we are able to evaluate a general robustness level considering the global uncertainties $\left(\hat{t}_{jk}\right)_{jk}$ as follows:

$$\Gamma = \mathbb{P}\left(C_{max}\left(X, \left(\hat{t}_{jk}\right)_{jk}\right) \leq C_{max}\right) \tag{6.32}$$

As the two machines are independent, we have $\Gamma = \Gamma_1 \times \Gamma_2$.

6.3.2.4 Update Module

Update of Ω

Here we assumed that the decision-maker is able to fix a robustness level Γ^{ref} he would like to be achieved by the system. This robustness level assessed the minimal acceptable probability that the executed makespan is smaller than the reference makespan C_{max}. Moreover, we considered that:

- the machine are independent: $\Gamma^{ref} = \Gamma_1^{ref} \times \Gamma_2^{ref}$
- the contributions of each machine to the global robustness level are equivalent (no machine is more critical than the other).

Thus, Γ_k^{ref} (defined in the inequation (6.9)) can be fixed as follows:

$$\forall k \in \{1, 2\}, \Gamma_k^{ref} = \sqrt{\Gamma^{ref}}$$

Following the assessment of Γ, Γ_1, and Γ_2 by the Discrete Event Systems Module, the following algebraic distances to the required minimal robustness level Γ^{ref}, Γ_1^{ref}, and Γ_2^{ref} can thus be calculated:

$$D = \Gamma^{ref} - \Gamma \tag{6.33}$$

$$D1 = \Gamma_1^{ref} - \Gamma_1 \tag{6.34}$$

$$D2 = \Gamma_2^{ref} - \Gamma_2 \tag{6.35}$$

If $D1 > 0$ or $D_2 > 0$, which means that the required robustness levels are not reached and the parameters Ω_1 and Ω_2 have to be updated. We propose to do it as follows:

$$\Omega_1 = \Omega_1 + D_1 \tag{6.36}$$

$$\Omega_2 = \Omega_2 + D_2 \tag{6.37}$$

We can note that these update formulas are arbitrarily defined. However, they express the fact that the further away from the objective (the bigger D_k), the more the parameters Ω_k must be amplified.

6.4 Application

In the application, 10 jobs are considered, with execution times having the uncertainties defined in Table 6.3. Moreover, Γ^{ref} is fixed to 0.90: meaning that the probability that the executed makespan will be effectively less than or equal to the optimal value is at least equal to 0.90.

Table 6.3 Characteristics of jobs

	t_{1k}^{min}	t_{1k}^{max}	t_{2k}^{min}	t_{2k}^{max}	t_{3k}^{min}	t_{3k}^{max}	t_{4k}^{min}	t_{4k}^{max}	t_{5k}^{min}	t_{5k}^{max}
machine k = 1	1	1	1	3	2	8	1	5	3	13
machine k = 2	1	3	1	1	1	3	2	6	3	9
	t_{6k}^{min}	t_{6k}^{max}	t_{7k}^{min}	t_{7k}^{max}	t_{8k}^{min}	t_{8k}^{max}	t_{9k}^{min}	t_{9k}^{max}	t_{10k}^{min}	p_{10k}^{max}
machine k = 1	1	3	2	6	1	3	3	5	1	1
machine k = 2	4	6	1	5	1	7	1	3	1	1

Table 6.4 Iterations for the application

Iteration k	Input Ω	OR module		DES module		Update module
		Solution	C_{max}	$\{\Gamma, \Gamma_1, \Gamma_2\}$	$\{D, D_1, D_2\}$	Output Ω
0	[0, 0]	X_1	12	{0.65, 0.85, 0.77}	{0.25, 0.10, 0.18}	[0.10, 0.18]
1	[0.10, 0.18]	X_1	12.54	{0.65, 0.85, 0.77}	{0.25, 0.10, 0.18}	[0.21, 0.35]
2	[0.21, 0.35]	X_1	13.05	{0.80, 0.93, 0.88}	{0.10, 0.02, 0.07}	[0.23, 0.42]
3	[0.23, 0.42]	X_1	13.26	{0.80, 0.93, 0.88}	{0.10, 0.02, 0.07}	[0.25, 0.49]
4	[0.25, 0.49]	X_1	13.47	{0.80, 0.93, 0.88}	{0.10, 0.02, 0.07}	[0.27, 0.56]
5	[0.27, 0.56]	X_1	13.54	{0.80, 0.93, 0.88}	{0.10, 0.02, 0.07}	[0.29, 0.63]
6	[0.29, 0.63]	X_2	13.58	{0.80, 0.85, 0.95}	{0.10, 0.10, 0.00}	[0.39, 0.63]
7	[0.39, 0.63]	X_2	13.78	{0.80, 0.85, 0.95}	{0.10, 0.10, 0.00}	[0.49, 0.63]
8	[0.49, 0.63]	X_1	13.89	{0.80, 0.92, 0.87}	{0.10, 0.03, 0.07}	[0.51, 0.70]
9	[0.51, 0.70]	X_2	14.02	{0.90, 0.93, 0.99}	{0.00, 0.02, −0.04}	[0.53, 0.66]
10	[0.53, 0.66]	X_1	13.98	{0.92, 0.96, 0.95}	{−0.02, −0.01, 0.00}	∅

Table 6.4 gives the different iterations of the combined approach. We started with $\Omega = [0, 0]$ (meaning that no uncertainty is considered). Two solutions are explored during different iterations. The solution X_1 allocates the first machine to jobs 1, 4, 6, 7, 8 and the second machine to jobs 2, 3, 5, 9, 10. The solution X_2 allocates the first machine to jobs 1, 4, 6, 7, 8, 10 and the second machine to jobs 2, 3, 5, 9.

This application shows that combining the two approaches allows to converge to a solution with a good robustness level without degrading too much the makespan. Initially (without perturbations), the makespan was of 12 and the associated robustness level was of 0.65. If the decision-maker accepts to degrade this makespan of around 20% (increasing the makespan to 14), then the robustness level reaches 0.90. Moreover, this approach is a good means for tuning the Ω parameters even if the probability distribution associated with the uncertainties is not symmetrical. We can note here that the makespan for a robustness level of 1 is $C_{max} = 18$ (namely the most conservative solution). Thus, the couple ($C_{max} = 13.98$, $\Gamma = 0.9$) is a good compromise between optimality and robustness.

6.5 Conclusions

Among the major issues related to *Industry 4.0*, risk management and, consequently, robust decision support play a significant role in the concerns of decision-makers.

In order to provide an efficient answer to this problem, we have proposed a generic method combining robust mathematical programming and Discrete Event Systems models. This allows to reach the level of robustness desired by the decision-maker by finely assessing the degree of robustness of the solutions provided by the optimization module, regardless of the probability distributions that follow the uncertainties on the model input data. We have illustrated the latter on the case of a scheduling problem with parallel machines.

However, as far as our methodology is generic, it will have to be adapted to the context of use. In particular, the mechanism for updating robustness coefficients Ω can be designed more efficiently to increase the rate of convergence of the methodology towards a solution with the required robustness level. In addition, instead of considering an equidistribution of the levels of robustness to be obtained over all the constraints of the model, a more specific distribution of these can be considered, taking into account, for example, the configuration of the production system, the criticality of certain machines (for instance, requiring greater robustness for bottleneck machines, etc.).

With the development of *Industry 4.0* and, more particularly, the increasing use of digital twins, these hybridization between decision support and performance evaluation models are likely to develop. The advent of Big Data and its consequences in terms of model calibration (and in particular through a more realistic estimation of probability laws modeling data uncertainties) combined with ever-increasing computing power will make it possible to implement this type of methodology in decision support tools in an industrial context.

Acknowledgments This chapter is the result of a collaboration between the two working groups Bermudes[1] and SED[2] from the French National Centre for Scientific Research (CNRS).

The Bermudes group has been created in 1996. It is labeled by the GDR RO[3] and the GDR MACS [4], which are two research groups depending on the French CNRS. Focused at the beginning on classical scheduling issues such as classical scheduling workshop problems (Job Shop, Flow Shop, Generalized Job Shop, Hybrid Flow Shop, etc.), scheduling problems in manufacturing systems (Flexible Manufacturing Systems, Hoist Scheduling Problems), and Resource Constrained Project Scheduling Problems, its research topics have evolved following Industry 4.0 context and include now integrated scheduling problems taking into account both several related activities (maintenance, transport, etc.) and different constraints (environmental, human, energetic, etc.). Determination of robust or reactive scheduling has become an important issue.

The Discrete Event Systems Working Group (DES) has been created in 2014. It is a French working group from the GDR MACS. Its objectives are to promote exchanges between the various specialists, whether they come from the world of automation, computer science, or mathematics, and thus to provide a better knowledge of the problems related to DES and the solutions that can be provided. The topics covered include (1) the study of the syntax and semantics of DES formalisms; (2) the application of these formalisms for modeling based on specifications, system performance analysis, simulation, property verification, system control, supervision, observation, detection, diagnosis, decision support, architecture selection, reconfiguration, etc.

[1]http://www.gt-bermudes.fr/.

[2]https://sites.google.com/site/gtsedmacs/.

[3]http://gdrro.lip6.fr/.

[4]https://gdr-macs.cnrs.fr/.

References

Alur, R., & Dill, D. L. (1994). A theory of timed automata. *Theoretical Computer Science, 126*(2), 183–235.

Baier, C., & Kwiatkowska, M. (1998). Model checking for a probabilistic branching time logic with fairness. *Distributed Computing, 11*(3), 125–155.

Ballarini, P., Djafri, H., Duflot, M., Haddad, S., & Pekergin, N. (2011). Petri nets compositional modeling and verification of Flexible Manufacturing Systems. In *IEEE International Conference on Automation Science and Engineering*, pp. 588–593. New York: IEEE.

Bertsimas, D., & Sim, M. (2004). The price of robustness. *Operations Research, 52*(1), 35–53.

Billaut, J.-C., Moukrim, A., & Sanlaville, E. (2013). *Flexibility and robustness in scheduling*. London: Wiley.

Cassandras, C. G., & Lafortune, S. (2009). *Introduction to discrete event systems*. New York: Springer Science & Business Media.

Chiola, G., Marsan, M. A., Balbo, G., & Conte, G. (1993). Generalized stochastic petri nets: A definition at the net level and its implications. *IEEE Transactions on Software Engineering, 19*(2), 89–107.

Chvátal, V. (1983). *Linear programming*. *A series of books in the mathematical sciences*. New York: W. H. Freeman.

Dauzères-Pérès, S., Castagliola, P., & Lahlou, C. (2010). Service level in scheduling. In J.-C. Billaut, A. Moukrim, & E. Sanlaville (Eds.), *Flexibility and robustness in scheduling* (chapter 5, pp. 99–121). Hoboken: Wiley-ISTE.

Ierapetritou, M. G., & Jia, Z. (2007). Short-term scheduling of chemical process including uncertainty. *Control Engineering Practice, 15*(10), 1207–1221. Special Issue - International Symposium on Advanced Control of Chemical Processes (ADCHEM).

Larsen, K. G., Pettersson, P., & Yi, W. (1997). Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer, 1*(1–2), 134–152.

Lenstra, J., Kan, A. R., & Brucker, P. (1977). Complexity of machine scheduling problems. In P. Hammer, E. Johnson, B. Korte, & G. Nemhauser (Eds.), *Studies in integer programming*. *Annals of discrete mathematics* (Vol. 1, pp. 343–362). Amsterdam: Elsevier.

Mokotoff, E. (2001). Parallel machine scheduling problems: A survey. *Asia-Pacific Journal of Operational Research, 18*(2), pp. 193–242.

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

Morel, G., Pétin, J.-F., & Johnson, T. L. (2009). *Reliability, maintainability, and safety* (pp. 735–747). Berlin/Heidelberg: Springer.

Nemhauser, G. L., & Wolsey, L. A. (1999). *Integer and combinatorial optimization*. *Wiley-Interscience series in discrete mathematics and optimization*. New York, NY: Wiley.

Panetto, H., Iung, B., Ivanov, D., Weichhart, G., & Wang, X. (2019). Challenges for the cyber-physical manufacturing enterprises of the future. *Annual Reviews in Control, 47*, 200–213.

Papadimitriou, C. H., & Steiglitz, K. (1998). *Combinatorial optimization: Algorithms and complexity*. Mineola, NY: Dover Publications.

Plateau, B., & Atif, K. (1991). Stochastic automata network of modeling parallel systems. *IEEE Transactions on Software Engineering* **17**(10), 1093–1108.

Wolsey, L. A. (1998). *Integer programming*. *Wiley-Interscience series in discrete mathematics and optimization*. New York: Wiley.

Zhong, R. Y., Xu, X., Klotz, E., & Newman, S. T. (2017). Intelligent manufacturing in the context of industry 4.0: A review. *Engineering, 3*(5), 616–630.