applied sciences

MDPI

Article

FMS Scheduling under Availability Constraint with Supervisor Based on Timed Petri Nets

Mohamed Ali Kammoun [1,*], Wajih Ezzeddine [1,2], Nidhal Rezg [1,2] and Zied Achour [1,2]

[1] Industrial Engineering, Production and Maintenance Laboratory (LGIPM), University of Lorraine, UFR MIM, île du Saulcy–BP 50128, 57045 Metz, France; wajih.ezzeddine@univ-lorraine.fr (W.E.); nidhal.rezg@univ-lorraine.fr (N.R.); zied.achour@univ-lorraine.fr (Z.A.)

[2] ICN Business School, 54003 Metz-Nancy, France

* Correspondence: mohamed-ali.kammoun@univ-lorraine.fr; Tel.: +33-778-527-746

Academic Editors: Zhiwu Li and MengChu Zhou

Received: 7 March 2017; Accepted: 11 April 2017; Published: 15 April 2017

Abstract: This paper proposes an optimal solution to large-scale Flexible Manufacturing System (FMS) scheduling problems under availability constraints based on Timed Petri Nets (TPNs). First a decomposition method of TPNs is proposed, then a mathematical model is derived based on their properties. The mathematical model is built to determine the optimal firing sequence of TPN transitions to minimize the total manufacturing time. The resulting firing sequence of TPN transitions is used to generate the manufacturing system supervisor operated by TPN and digital controllers. Several numerical examples and comparative studies are provided in this paper in order to prove the new approach's efficiency.

Keywords: scheduling problem; timed Petri net; discrete event system; supervisor synthesis; mathematical model; genetic algorithm

1. Introduction

Flexible Manufacturing Systems (FMS) are widely used since they are swiftly adjustable to different production constraints, which improves their ability to cover different job types without major influence on the production plan [1,2]. Usually, FMS are subjected to failures caused by their resources' degradation process, which has a direct impact on the production planning. To satisfy demand, an efficient production plan under the availability constraint has to be established leading to the identification of the optimal processing time or makespan [3,4]. Therefore, despite the risk of increasing deadlocks, both availability and limited resource constraints are considered when FMS scheduling is optimized.

Several methods dealing with FMS scheduling have been proposed [5–8] where the problem NP-hardness was the major challenge encountered [9]. In FMS scheduling, the most important task is to determine the sequence of production. This is related to the production supervision system for which all scheduling constraints must be considered as stated previously [10,11]. In the literature, several proposed controllers are based on Petri Net (PN) tools [12–15]. Ghaffari et al. [10] used theory of regions to synthetize the PN controller, which is a set of control places added to the initial PN model. Al-Ahmari et al. [16] applied Generalized Stochastic Petri Nets (GSPN) modules to derive a model for the workstation controller.

For PN analysis, several methods can be founded such as enumerative methods, which are adapted to FMS scheduling [8]. In fact, to solve the scheduling problem using enumerative methods, the reachability graph is entirely or partly explored to retrieve the optimal path [14]. Therefore, the optimal path connecting the initial and final markings corresponds to the optimal firing sequence of PN transitions. Several heuristic methods are developed in order to partly explore the reachability

graph and find an optimal or suboptimal production plan. In this perspective, Lee and diCesare [17] present a Timed Petri Net (TPN) modeling for scheduling problems and propose a heuristic algorithm A* [18–20] as the resolution approach. This approach is based on generating and explorating of a limited part of the reachability graph. These models were recently improved by Li et al. [21]. The authors proposed to use an algorithm for the single and multiple lot size scheduling problem and prove the solving heuristic procedure proposed by Xiong and Zhou [22].

The main inconvenience of enumerative methods is the significant number of reachable markings. Consequently, recent studies introduced new methods such as heuristic search algorithms and artificial intelligence methods in PN modeling for scheduling problems [23,24]. For instance, Chung et al. [25] introduced a genetic algorithm [26,27] to solve the sequencing problems relative to Petri net modeling. However, the latter method is limited to FMS with up to five machines. Saitou et al. [28] came up with a colored Petri net to model the resource allocation and the operation schedule of a special FMS class. The authors combined a genetic algorithm with a specific dispatching rule (the Shortest Imminent Operation time (SIO)) to get simultaneously the suboptimal resource allocation and the event-driven schedule of the colored Petri net. They were able to provide results with three different scenarios with up to ten machines.

The current study is motivated by the complexity of the enumerative methods' increasing number leading to state explosion issues. Therefore, the main contribution of this paper is to derive a mathematical model based on TPN properties to efficiently solve the FMS scheduling. To achieve this, a decomposition methodology of TPN is proposed, where the initial model is decomposed into a set of Timed Marked Graphs (TMGs) considered as a subclass of TPN. Several decomposition methods for Petri net variants have been developed [29]. Nishi et al. [30,31] proposed a general PN decomposition method and coordination algorithm in order to optimize the conflict-free routing for an Automated Guided Vehicle (AGV). Ciardo et al. [32] introduced the concepts of near independence and near decomposition where the authors developed an approach for the solution of stochastic Petri net models. The developed approach is applied in the case of a flexible manufacturing system.

The second step of this work, a genetic algorithm, is introduced to solve the mathematical model in the case of larger-scale problems. The last part deals with the implementation of the above proposed solution in order to synthetize a supervisor, coupled with a decomposed TPN model.

The paper is organized as follows: Section 2 presents an overview of the timed Petri net and genetic algorithms. Section 3 illustrates the notations and terminology of the different variables used. Section 4 presents the basic assumptions where our FMS modeling is based. Section 5 derives a method to decompose the initial TPN into a set of Timed Marked Graphs (TMGs). Section 6 details the proposed approaches to solve the scheduling framework under availability constraints. The supervisor synthesis method is illustrated in Section 7. Some illustrative examples and comparative studies are shown in Section 8. Finally, Section 9 is dedicated to the perspectives of this work.

2. Background

2.1. Timed Petri Net

Formally, a TPN [33] is a five-tuple $(P, T, O, W, h(T))$ where:

- $P = \left\{p_0, p_2, \ldots, p_{|P|}\right\}$: is a non-empty finite set of places;
- $T = \left\{t_0, t_2, \ldots, t_{|T|}\right\}$: is a non-empty finite set of transitions;
- $O \subseteq (P \times T) \cup (T \times P)$: are the direct arcs from places to transitions $(P \times T)$ or from transitions to places $(T \times P)$;
- $W : (P \times T) \cup (T \times P) \to \mathbb{N}$: is the affectation function of weights to arcs, where $\mathbb{N}$ is the set of non-negative integers;
- $h(t) \in h(T)$: is the static time linked to t, $t \in T$.

- ${}^{\circ}p$ (p°) : designs the set of input transitions (respectively output) of place p;
- M : is a marking and the initial marking is denoted by M_0;
- $M(p)$: is a place marking;
- $M[t>$: indicates that transition t is fired at marking M.

It is important to note that transition t is enabled by marking M only if $M(p) \geq W(p,t), \forall\, t \in p^{\circ}$. An enabled transition satisfying its time condition can be fired. Furthermore, $P \cup T \neq \varnothing$ and $P \cap T = \varnothing$ where $\varnothing$ designs the empty set. In the literature, several classes could be derived from TPN from where the Timed Marked Graph (TMG) is our endpoint of interest in this paper.

Definition 1. *The TMG is a particular type of TPN where every place possesses one input transition and one output transition, i.e.,* $|{}^{\circ}p| = |p^{\circ}| = 1$. *Furthermore, all weights associated to arcs are equal to one, i.e.,* $W : (P \times T) \cup (T \times P) \rightarrow \{0,1\}$.

Figure 1 illustrates a TMG model. The place p_0 is initially marked, and the enabled transition t_0 can be fired after four time slices ($t.s$). The transition t_1 is fired if p_1 is marked with a delay of 8 $t.s$.

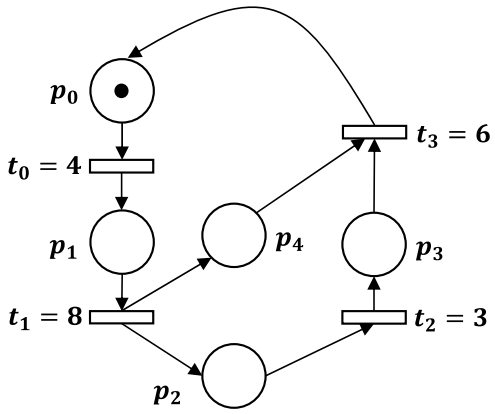

Figure 1. Timed Marked Graph (TMG) example.

2.2. Genetic Algorithm

Genetic Algorithms (GA) are recombination-based metaheuristics and belong to the class of evolutionary algorithms. The search procedures introduced by these algorithms are based on the mechanics of natural genetics. They have been widely applied to a variety of different problems and domains. For a detailed description of the use of genetic algorithms in engineering optimization issues, the reader could refer to Goldberg et al. [34].

In this article, we introduce the genetic algorithm method to solve the mathematical model described in the previous section. The choice of the GA is motivated by:

- Mathematical optimization techniques, such as the branch and bound method and dynamic programming, are limited by the problem dimensionality. These methods are not efficient in the case of large-scale and NP-hard problems, such as our study.
- The heuristic procedure is convenient for a production scheduling problem of a small scale. However, in the case of processing problems for a large scale, heuristics dealing with an NP-hard scheduling problem with a makespan objective have several drawbacks, such as the lack of comprehensiveness [35], and the accuracy of the solution needs to be improved [36].
- The genetic algorithm is convenient for production scheduling problems in view of its solving performance characteristics, such as near optimization and high resolution speed. GA efficiency was proven and discussed in solving scheduling problems having similar characteristics as our present model [37,38].

3. Terminology

In this section are presented the notations used in TPN modeling for FMS scheduling and supervisor synthesis.

- i, i^* : indexes for jobs ($i, i^* = 1 \dots n$);
- j, j^* : indexes for machines ($j = 0 \dots m+1$);
- $t_{i,j}$(if fired) : transition denoting that the job i leaves the machine j;
- $p_{i,j}$(if marked) : place indicating that the job i is already being performed by machine j;
- ${}^{\circ}p_{i,j}$ ($p_{i,j}{}^{\circ}$) : designs the set of input transitions (respectively output) of a place $p_{i,j}$;
- $h\left(t_{i,j}\right) \in h(T)$: is the local-clock linked to $t_{i,j}$; in the proposed modeling method, $h\left(t_{i,j}\right)$ defines the processing duration of i through j;
- $H\left(t_{i,j}\right)$: is the global-clock of transition firing $t_{i,j}$; physically, it defines the end time of job i through machine j;
- $A\left(t_{i,j}, t_{i^*,j-1}\right)$: is a decision variable defining whether the arc from $t_{i,j}$ to $t_{i^*,j-1}$ exists;
- $Z_{i,j+1}$: is a decision variable defining whether job i is performed after the maintenance task on machine $j+1$;
- $\varphi_{t_{i,j}}\left(Z_{i,j+1}\right)$: is the extra time linked to $t_{i,j}$ depending on the $Z(i, j+1)$ value;
- $\pi_{(t_{i,j}, t_{i^*,j-1})}$: unmarked timed elementary path between two transitions;
- $\theta_{(i,i^*)}$: duration associated with $\pi_{(t_{i,j}, t_{i^*,j-1})}$;
- $[B_j, E_j]$: preventive maintenance period where B_j (respectively E_j) is the beginning time (respectively ending time).

In our proposed model for FMS using TPN tools (Section 5), each TMG^i designs the free-dynamics of job i through the manufacturing system. Relations between two different timed marked graphs TMG^i and TMG$^{i^*}$ ($i, i^* = 1 \dots n$ and $i \neq i^*$) are modeled by timed direct arcs $\left(t_{i,j}, t_{i^*,j-1}\right)$ from transition $t_{i,j}$ to transition $t_{i^*,j-1}$. Physically, an established arc $\left(t_{i,j}, t_{i^*,j-1}\right)$ is interpreted to mean that the job i had to be treated on machine j before that the job i^* finishes its treatment on machine $j-1$. $A\left(t_{i,j}, t_{i^*,j-1}\right)$ is analytically expressed as follows:

$$A\left(t_{i,j}, t_{i^*,j-1}\right) = \begin{cases} 1 \text{ if the timed direct arc } \left(t_{i,j}, t_{i^*,j-1}\right) \text{ has taken place} \\ 0 \text{ else} \end{cases} \quad (1)$$

Remark 1. *It is obvious that a timed direct arc between two transitions contains* $(2 \times s)$, $s \in \mathbb{N}$, *places and* $(2 \times s - 1)$ *transitions. In the following, we consider* $s = 1$.

Another important parameter associated with TMG^i is the extra time $\varphi_{t_{i,j}}\left(Z_{i,j+1}\right) \in \mathbb{N}$ added to local-clock $h\left(t_{i,j}\right)$. $Z_{i,j+1}$ corresponds to the job i position, if it is just after the maintenance action on machine $j+1$. The analytical expression of $Z_{i,j+1}$ is:

$$Z_{i,j+1} = \begin{cases} 1 \text{ if the transition } t_{i,\mathrm{j}} \text{ should not be fired} \\ 0 \text{ else} \end{cases} \quad (2)$$

Indeed, $\varphi_{t_{i,j}}\left(Z_{i,j+1}\right)$ is the job i time delay on machine j until the availability of machine $j+1$. Figure 2 illustrates all of the explained parameters.

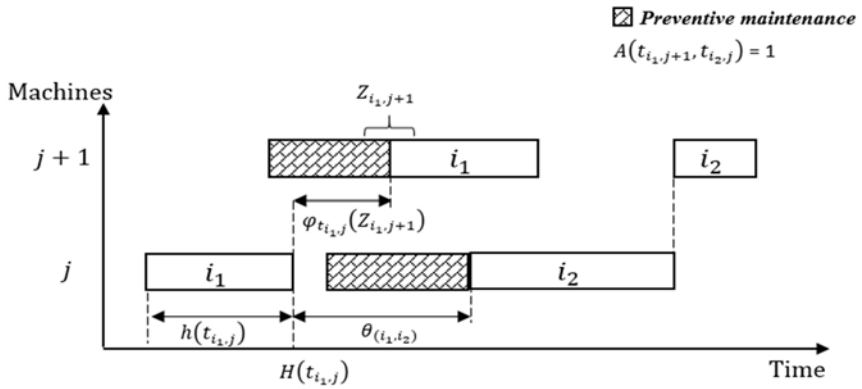

Figure 2. Illustration of Timed Petri Net (TPN) parameters in the production process.

The main contribution of this paper is to carry out an efficient manufacturing supervisor based on an optimal sequence of transitions determined by a mathematical model. In this perspective, the derived supervisor is assumed to be commissioned by two types of controllers defined as follows.

TPN controller: A TPN controller is an unmarked timed elementary path $\pi_{(t_{i,j},t_{i^*,j-1})}$: $\left(p_{1,(i,i^*)}, t_{(i,i^*)}\left[\theta_{(i,i^*)}\right], p_{2,(i,i^*)}\right)$ between two transitions $t_{i,j}$ and $t_{i^*,j-1}$ (i.e., see Remark 1).

The principal role of the TPN controller is to organize the job processing via the time lag $\theta_{(i,i^*)}$. Its computing depends on the $A(t_{i,j}, t_{i^*,j-1})$ value, where ${}^\circ p_{1,(i,i^*)} = \{t_{i,j}\}$ and $p^\circ_{2,(i,i^*)} = \{t_{i^*,j-1}\}$.

Digital controller: A digital controller model is based on the $\varphi_{t_{i,j}}(Z_{i,j+1})$ definition and depends on the decision variable $Z_{i,j+1}$. The role of this controller is to compute the extra time of a fully-operated job in machine j when machine $j+1$ is unavailable.

4. Flexible Manufacturing System Description

This paper covers a special class of FMS, where the flexibility designs a machine that can perform different types of jobs. The time that a machine needs to change the operating mode is negligible. The considered FMS is defined as a set of multiple job types where the processing order of all jobs through the machine set is the same.

The following assumptions are considered:

- n jobs of different type are initially placed in the initial buffer;
- m machines are available; the initial and the final buffer will be modeled respectively by $j = 0$ and $j = m + 1$ as two additional virtual machines;
- Jobs are executed in the same order as defined in the first machine;
- Each machine is subject to a preventive maintenance action expressed by an unavailability period; this action is assumed to be preprogrammed;
- Each machine performs only one job at a time;
- Processing time depends on the job type;
- Job preemption is forbidden.

As mentioned, our objective is to determine the optimal sequence of jobs where the total completion time is subject to optimization. In the second part, we focus on the implementation of this optimal solution in order to synthetize the manufacturing system supervisor coupled to the initial timed Petri net.

In studies involving TPN models, it is often the case that there are computing risks involved, in the case of an important number of machines, so it is necessary to decompose the entire TPN. The main objective is to guarantee an easier solving process.

5. Modeling FMS with the Decomposed TPN to the TMG Set

5.1. Decomposition Methodology

The optimal legal firing sequence problem is impossible to solve when the number of jobs and machines is increased, i.e., scalability issue. Thus, the entire TPN $= (P, T, O, W, h(T))$ explicitly models the physical functioning of FMS decomposed into several timed marked graphs, for the following reasoning. First, the set of transitions T is considered as a disjoint subset T_i, which contains the transitions $t_{i,j}$.

$$T = T_1 \cup T_2 \cup \ldots \cup T_n = \left(\bigcup_{i=1}^{n} T_i \right) \tag{3}$$

Place set P presents disjoint subsets P_i, which contain the places $p_{i,j}$ and the set of resource-places $P_R = \{p_r, 1 \leq r \leq m\}$:

$$P = P_1 \cup P_2 \cup \ldots \cup P_n \cup P_R = \left(\bigcup_{i=1}^{n} P_i \right) \bigcup P_R \tag{4}$$

$$P_i = \{p | \, {}^{\circ}p \subseteq T_i \; and \; p^{\circ} \subseteq T_i\} \tag{5}$$

Each resource-place $p_j \in P_R$ satisfies Equations (4) and (5):

$${}^{\circ}p_j = \left\{ t_{i,j} \middle| 0 \leq j \leq m \right\} \tag{6}$$

$$p_j^{\circ} = \left\{ t_{i,j-1} \middle| 1 \leq j \leq m \right\} \tag{7}$$

The second stage of the modeling methodology aims to remove the resource places from the entire TPN. Indeed, the resulting TPN is divided into n timed marked graphs $\text{TMG}^i = (P_i, T_i, O_i, W_i, h(T_i))$, $1 \leq i \leq n$, which model the free-dynamic of job i. Hence, the place set P_i, direct arcs O_i and affectation weights W_i satisfy the following equations:

$$\bigcup_{i=1}^{n} P_i = P \backslash P_R, \; \forall \; p \in P_i \tag{8}$$

$$O_i(p,t) = O(p,t), \; \forall \; p \in P_i, \; \forall \; t \in T_i \tag{9}$$

$$O_i(t,p) = O(t,p) \; \forall \; p \in P_i, \; \forall \; t \in T_i \tag{10}$$

$$W_i(p,t) = 0, \; \forall \; p \in P_R, \; \forall \; t \in T_i \tag{11}$$

$$W_i(t,p) = 0, \; \forall \; p \in P_R, \; \forall \; t \in T_i \tag{12}$$

Therefore, $\cup_{i=1}^{n} \text{TMG}^i$ represents the free-dynamic of jobs without considering the constraints explained in the above section. This is the reason why new parameters as timed direct arcs and extra times should be considered in the decomposed model (Section 3); parameters are computed using our methodology developed in the following section. Based on the third assumption in Section 4, the TPN model parameterization could be concluded through the $\text{A}(t_{i,1}, t_{i^*,0})$ value determination.

Therefore, Algorithm 1 summarizes the decomposition methodology.

Algorithm 1. The methodology of TPN decomposition.

Step 1. Model the considered system as a set of timed marked graphs taking into account the resource places.
Step 2. Elimination of resource places. The resulting net is $\cup_{i=1}^{n} \text{TMG}^i$ in which each TMG^i models separately the free-dynamics of job i.
Step 3. Add the timed direct arcs between transitions $\left(t_{i,1}, t_{j,0}\right)$, which ensure the right sequence of jobs through operators.
Step 4. Introduce the eventual extra time $\varphi_{t_{i,j}}\left(Z_{i,j+1}\right) \in \mathbb{N}$ that presents the number of times slice delays of job i in machine j, when the next machine $j+1$ is unavailable.

5.2. Illustrative Example

In order to illustrate the proposed TPN decomposition methodology, we consider an FMS composed of two machines $j = 1, 2$ operating two different jobs $i = 1, 2$. The processing time of jobs ($h(t_{i,j})$) and the preventive maintenance periods ($[B_j, E_j]$) are respectively illustrated by Tables 1 and 2.

Table 1. The processing time.

	Machine 1	**Machine 2**
Part 1	3 *t.s*	5 *t.s*
Part 2	4 *t.s*	2 *t.s*

Table 2. The preventive maintenance periods.

	B_j	E_j
Machine 1	3	6
Machine 2	4	8

According to Figure 3, the initial timed Petri net is composed by two timed marked graphs, TMG^1 and TMG^2, designing respectively the free-dynamics of jobs i_1 and i_2. The initial buffer j_0 and the final buffer j_3 are considered as virtual machines. i_1 and i_2 are initially in j_0 corresponding with places $p_{1,0}$ and $p_{2,0}$, which are marked. Besides, the resource places p_{r1} and p_{r2} are designed to satisfy the production capacity constraints of j_1 and j_2. This first modeling step (Algorithm 1), satisfying Equations (12) and (13), represents the partial FMS scheduling with respect to the preventive maintenance and processing sequence.

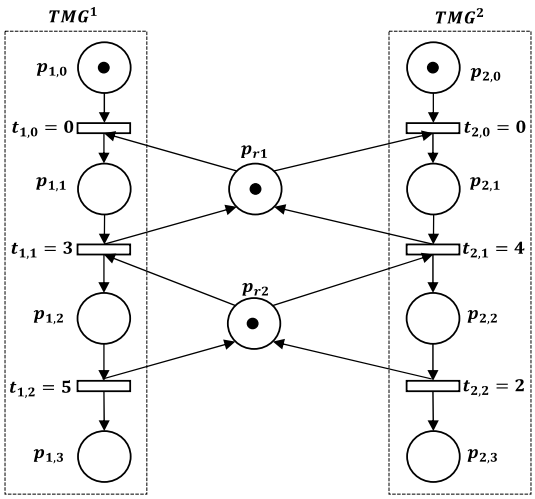

Figure 3. The initial TPN model.

The decomposed TPN is determined as shown in Figure 4. It includes the direct arcs that schedule the possible processing sequences (Step 3). For instance, the timed direct arc $(t_{2,1}, t_{1,0})$ ensures the processing of i_2 before i_1 through the value of $\theta_{(2,1)}$. Besides, to avoid that the jobs conflict, a waiting time (Step 4) is added to corresponding transitions. For example, $\varphi(Z(t_{1,c_1}))$ represents the time delay of i_1 at machine j_1 whether j_2 is unavailable for maintenance.

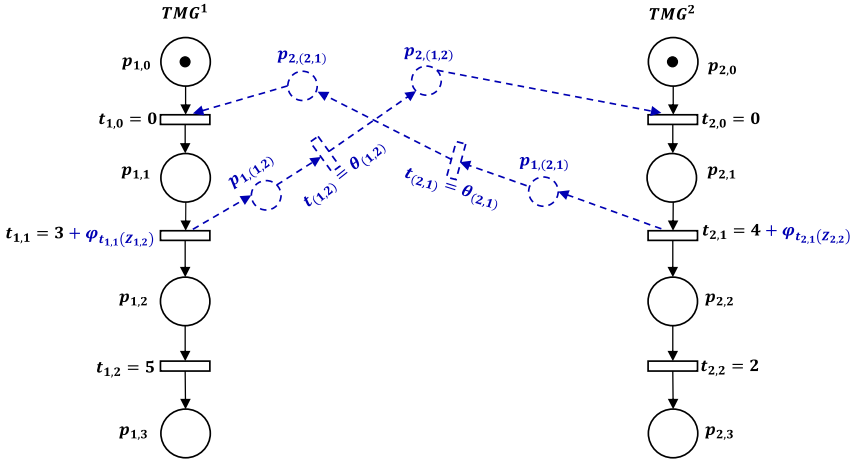

Figure 4. The decomposed TPN model.

The final scheduling through the decomposed TPN is determined by computing the supervisor parameters, which are the subject of the next sections.

6. Mathematical Model

A mathematical model based on decomposed TPN properties is proposed. The model uses the global and local clocks of transition firing (respectively $H(t_{i,j})$ and $h(t_{i,j})$) and the variables' decision $A(t_{i,j}, t_{i^*,j-1})$ and $Z_{i,j+1}$.

6.1. Model Description

A set of machines and jobs is considered. The processing duration of a job i on a machine j is given by $h(t_{i,j})$. The formulation is developed to be in good agreement with the defined parameters of TPN modeling. In addition, a preventive maintenance scheduling constraint will be integrated to get a jointly optimal production and maintenance plan.

The objective function is defined as follows:

$$min\left(\max_{i=1..,n} H(t_{i,m})\right) \tag{13}$$

subject to:

$$H(t_{i,j+1}) - H(t_{i,j}) \geq h(t_{i,j+1}),\ \forall\, i = 1,\ldots n,\ \forall\, j = 0,\ldots m \tag{14}$$

$$H(t_{i^*,j}) - H(t_{i,j}) + \alpha \cdot (1 - A(t_{i,j}, t_{i^*,j-1})) \geq h(t_{i^*,j}),\ \forall\, i, i^* = 1,\ldots n,\ i \neq i^*\ \forall\, j = 1,\ldots m \tag{15}$$

$$H(t_{i,j}) - H(t_{i^*,j}) + \alpha \cdot A(t_{i,j}, t_{i^*,j-1}) \geq h(t_{i,j}),\ \forall\, i, i^* = 1,\ldots n,\ i \neq i^*\ \forall\, j = 1,\ldots m \tag{16}$$

$$H(t_{i,j}) - Z_{i,j} \cdot E_j \geq h(t_{i,j}),\ \forall\, i = 1,\ldots,n,\ \forall\, j = 1,2,\ldots,m \tag{17}$$

$$H(t_{i,j}) - \alpha \cdot Z_{i,j} \leq B_j,\ \forall\, i = 1,\ldots n,\ \forall\, j = 1,2,\ldots,m \tag{18}$$

$$A(t_{i,j}, t_{i^*,j-1}) = A(t_{i,j+1}, t_{i^*,j}),\ \forall\, i, i^* = 1,\ldots,n,\ \forall\, j = 1,\ldots,m-1 \tag{19}$$

$$A(t_{i,j}, t_{i^*,j-1}) = 0 \text{ or } 1\ \forall\, i, i^* = 1,\ldots,n,\ \forall\, j = 1,2,\ldots,m \tag{20}$$

$$Z_{i,j} = 0 \text{ or } 1,\ \forall\, i, i^* = 1,\ldots,n,\ \forall\, j = 1,2,\ldots,m \tag{21}$$

The objective function (13) aims to minimize the firing global clock of last transitions corresponding to the last machine m. The constraint (14) states that a job is processed, on each machine at once, without preemption. The constraints, (15) and (16) ensure that only one job can be performed on the machine at a time. In addition, the firing sequence of transitions (production sequence) has to respect the preventive maintenance period $[B_j, E_j]$ modeled with (17) and (18). Constraint (19) ensures

that the order of jobs remains the same on all machines. Constraints (20) and (21) define the restrictions for the decision variables.

The linear programming (13)–(21) is a Mixed-Integer Linear Programming (MILP) because the decision variables are constrained to be binary.

Remark 2. *In the current study, the end point of interest is the determination of* $A(t_{i,1}, t_{i^*,0})$, $(i, i^* = 1, ..., n)$; *our proposed TPN model assumes that the order of jobs on the machines is the same as the order defined in the first one (Section 4 assumptions).*

As mentioned, our contribution is to provide the optimal firing sequence that will be used to synthetize the scheduling supervisor. Because of large-scale scheduling problems' NP-hardness [39,40], we propose an efficient metaheuristic approach based on the genetic algorithm model.

6.2. Genetic Algorithm

6.2.1. Encoding

The length of the chromosome is equal to $(m + 1) * n$. The first n genes present the jobs' sequence, and the last $(n * m)$ ones are binary corresponding to the job position in relation to the maintenance period on each machine (just before or after the maintenance task).

Figure 5 shows an example of the solution encoding where a set of two jobs and three machines is considered. The job number '2' will be initiated as the first job, followed by the job number '1' (two first genes). For the job number '1' (three next genes), the gene values indicate that it will be operated just before the maintenance period, in the first and the second machine, and after the maintenance period in the third one.

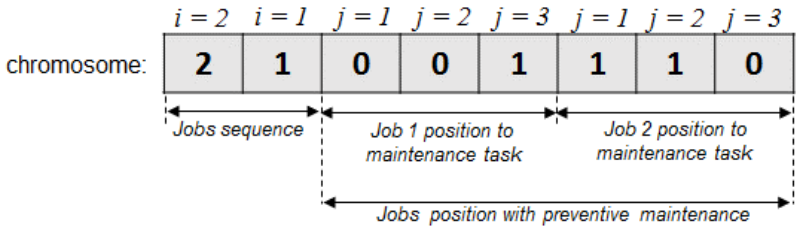

Figure 5. Example of the encoding solution.

6.2.2. Initial Population

An initial population is generated randomly where each chromosome presents a solution for the scheduling problem.

6.2.3. Crossover

There are many ways to apply the crossover operator, which aims to generate new individuals. In this perspective, a uniform crossover with a random binary mask is used. The uniform crossover uses the chromosome to switch between parent's genes. It is possible that the crossover operator could be applied at any point (Figure 6).

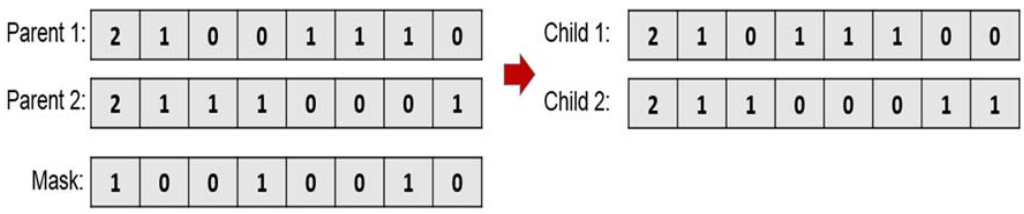

Figure 6. Example of new solution generation via uniform crossover.

6.2.4. Mutation

This operator consists of altering gene values. Figure 7 illustrates an example of the mutation mechanism application. In this example, genes subjected to mutation are Bits 1, 2, 4 and 7.

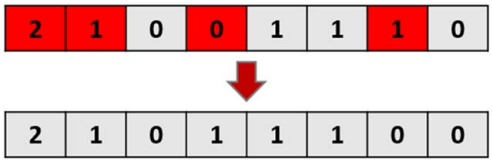

Figure 7. Example of new solution generation via the mutation operator.

Remark 3. *Throughout the model resolution, the random gene values are selected as follows:*

- *For the n first genes (designing jobs), the set of random bit values is* $\{1, \ldots, n\}$*;*
- *For the* $(n * m)$ *last genes (job position to maintenance), the set of random bit values is* $\{1, 0\}$*.*

6.2.5. Constraints

When a solution is generated, the constraints cited in Section 3 have to be respected to ensure that an individual is feasible. In addition, a test should be carried out for every new individual after we decide to reject or save it for fitness computation.

6.2.6. Fitness

The fitness function is illustrated by (13).

6.2.7. Stopping Criteria

The progress of reproduction by crossover and mutation operators and fitness continues until a satisfactory result is obtained or a preset maximum number of iterations is reached. After these iterations, the genetic algorithm prints the best chromosome that corresponds to the minimum makespan H_{max}.

Algorithm 2 recaps the GA steps. Initialization parameters are: population size (P), crossover probability value (P_{cross}), probability mutation value (P_{MUT}) and the preset maximum number of iterations ($Niter$).

Algorithm 2. Determination of the jobs' sequence.

Initializing the program and setting the data values: P, P_{cross}, P_{MUT}, $Niter$, $i = 0$, $k = 0$
Generate randomly a solution
Repeat
$i = 0$
  While $k < P$ do
  Select randomly 2 parents of the population (2 solutions);
  Apply the crossover operator;
  Apply the mutation operator.
  $k = k + 1$
  End while
  Test if a generated solution is feasible;
  Evaluate the obtained and valid new sequences;
  Arrange in ascending order these solutions based on the objective function;
  Delete the worst solutions and record the best ones;
  Record the best solution.
  $i = i + 1$
  Until $i = Niter$
Print the optimal solution corresponding to minimum makespan

7. Synthesis Method of Scheduling Supervisor

The main contribution, in this part, is to develop an efficient supervisor synthesis method (Algorithm 3), which tampers with the decomposed TPN, based on the transitions firing sequence and parameters provided by the mathematical model. The proposed supervisor is managed by two types of controllers: digital controllers and TPN controllers (i.e., see Section 3).

- Step 1. Solving the firing sequence:

This step is essential as it determines the firing global-clock of transitions $H(t_{i,j})$, the decision variable values $A(t_{i,1}, t_{i^*,0})$ and $Z_{i,j}$.

- Step 2. Identifying the chronological order of jobs:

Once $A(t_{i,1}, t_{i^*,0})$ values are determined, one can identify the jobs' order.

- Step 3. Deleting the inexistent direct arcs:

Direct arcs $t_{i,1}, t_{i^*,0}$ corresponding to $A(t_{i,1}, t_{i^*,0}) = 0$ are removed from the decomposed TPN model. Remaining direct arcs ($A(t_{i,1}, t_{i^*,0}) = 1$) design the selected processing sequence.

- Step 4. Determining the digital controllers:

The digital controller reacts when the treatment of job i is accomplished through machine j while $j+1$ is unavailable. Therefore, the controller functioning is displayed through $Z_{i,j+1} = 1$ in which the extra time $\varphi_{t_{i,j}}(Z_{i,j+1})$ of this controller is computed.

- Step 5. Determining the TPN controllers:

The TPN controller ship is to ensure a correct order progression of jobs including among each other a time lag $\theta_{(i,i')}$. The TPN controllers operate in coordination with digital controllers, i.e., the determination of time lag $\theta_{(i,i')}$ takes into account the extra time $\varphi_{t_{i,j}}(Z_{i,j+1})$ of the digital controller. Its value is computed as: $\theta_{(i,i')} = H(t_{i',0}) - (H(t_{i,1}) + \varphi(Z(t_{i,1})))$.

- Step 6. Implementing the supervisor:

The controllers calculated in Steps 4 and 5 take places in the decomposed timed Petri net.

Algorithm 3. Supervisor synthesis based on the generated firing sequence.

---

1. Solve the firing sequence using the mathematical model (13)–(21) and genetic algorithm.

2. Identify the chronological order of jobs.

3. Delete the inexistent direct arcs

For each $A(t_{i,1}, t_{i^*,0}) = 0 \; \forall \; i, \; i^* = 1, 2, \ldots, n, \; i \neq i^*$

Remove the direct arc $(t_{i,1}, t_{i^*,0})$ from the decomposed TPN

4. Determine the digital controllers

For each $Z_{i,j+1} = 1 \; \forall \; j = 1, \ldots, m-2$

Calculate $\varphi_{t_{i,j}}\left(Z_{i,j+1}\right) = H\left(t_{i,j+1}\right) - \left(H\left(t_{i,j}\right) + h\left(t_{i,j+1}\right)\right)$

For each job $s \succ i$

Calculate $\varphi_{t_{s,j}}\left(Z_{i,j+1}\right) = H\left(t_{s,j+1}\right) - \left(H\left(t_{s,j}\right) + h\left(t_{s,j+1}\right)\right)$

5. Determine the TPN controllers

For each $A(t_{i,1}, t_{i^*,0}) = 1 \; \forall \; i, \; i\prime = 1, 2, \ldots, n, \; i \neq i^*$

Create the TPN controller: $\pi_{(t_{i,1}, t_{i^*,0})} : \left(p_{1,(i,i^*)}, t_{(i,i^*)}\left[\theta_{(i,i^*)}\right], p_{2,(i,i^*)}\right)$.

Calculate $\theta_{(i,i^*)} = H(t_{i^*,0}) - (H(t_{i,1}) + \varphi(Z(t_{i,1})))$.

6. Implement the supervisor to decomposed TPN

To solve the mathematical model (13)–(21), we used the Xpress optimizer. Xpress is a mathematical programming solver dealing with linear and quadratic problems in continuous and integer variables. For solving Mixed Integer Programs (MIPs), the optimizer provides a powerful branch and bound framework. The Xpress optimizer essentials, components and interfaces are found in many guides and reference manuals, from which we cite [41]. In the literature, many proposals deal with Xpress solver. Guéret [40] illustrates some applications of the optimization solver programs. Algorithms for solving large linear optimization problems are reviewed. Laundy et al. [42] highlights the Xpress solver's efficiency to optimize mixed linear problems.

To illustrate the supervisor synthesis, we reconsider the decomposed timed Petri net shown in Figure 3 and the preventive maintenance presented in Table 2. The obtained results are shown in Table 3.

Table 3. The optimal solution and decision variables.

$H(t_{1,0}) = 0$	$H(t_{1,1}) = 3$	$H(t_{1,2}) = 13$	$H(t_{2,0}) = 7$	$H(t_{2,1}) = 11$
$H(t_{2,2}) = 15$	$A(t_{1,1}, t_{2,0}) = 1$	$A(t_{2,1}, t_{1,0}) = 0$	$Z_{1,1} = 0$	$Z_{1,2} = 1$
$Z_{2,1} = 1$	$Z_{2,2} = 0$			

Moreover, as explained in Step 2 of Algorithm 3, the decision variable $A(t_{1,1}, t_{2,0}) = 1$ expresses that the job i_1 is treated before the job i_2, ($i_1 \prec i_2$). Hence, the timed direct arc $(t_{2,1}, t_{1,0})$ is removed from the decomposed TPN, i.e., $A(t_{2,1}, t_{1,0}) = 0$ (Figure 8).

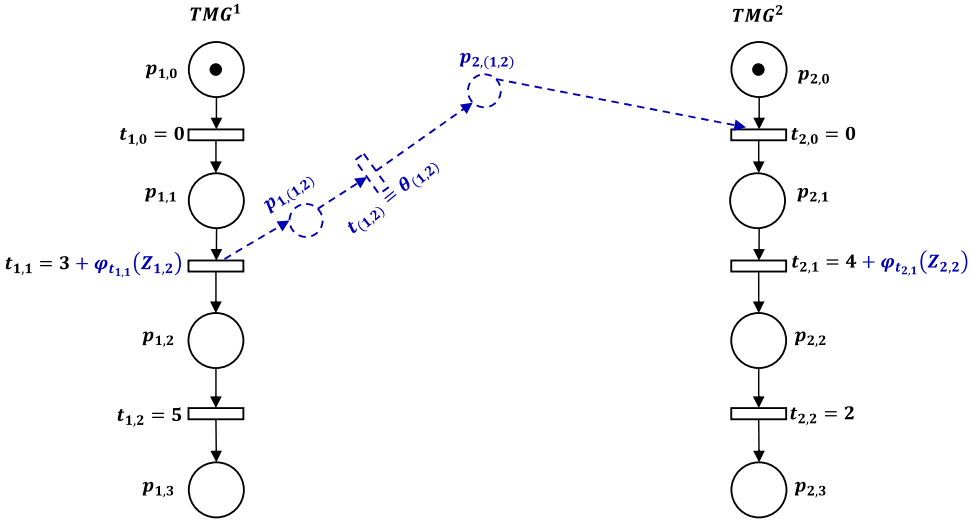

Figure 8. The remaining arc for supervisor synthesis.

According to Step 3, the determination of digital controllers is related to $Z_{1,2} = 1$, as follows:

$$\Longrightarrow \begin{cases} \varphi_{t_{1,1}}(Z_{1,2}) = H(t_{1,2}) - (H(t_{1,1}) + h(t_{1,2})) = 5\ t.s \\ As(i_2 \succ i_1) \Longrightarrow \varphi_{t_{2,1}}(Z_{1,2}) = H(t_{2,2}) - (H(t_{2,1}) + h(t_{2,2})) = 2\ t.s. \end{cases}$$

Indeed, the two digital controllers $\varphi_{t_{1,1}}(Z_{1,2})$ and $\varphi_{t_{2,1}}(Z_{1,2})$ require a waiting time respectively for i_1 and i_2 at j_1. Jobs i_1 and i_2 are respectively delayed by 5 $t.s$ and 3 $t.s$.

Afterward, the TPN controller $\pi_{(t_{1,1},t_{2,0})} : \left(p_{1,(1,2)}, t_{(1,2)}\left[\theta_{(1,2)}\right], p_{2,(1,2)}\right)$ permits a good sequence between i_1 and i_2. Its time lag $\theta_{(1,2)}$ is determined as follows:

➢ $A(t_{1,1}, t_{2,0}) = 1$

$$\Longrightarrow \theta_{(1,2)} = H(t_{2,0}) - (H(t_{1,1}) + \varphi(Z(t_{1,1}))) = 1\ t.s$$

Finally, the decomposed TPN coupled with the synthetized supervisor, shown in Figure 9, guarantees the optimal manufacturing scheduling; i.e., the optimal sequence firing of transitions is $\sigma = t_{1,0}t_{1,1}t_{2,0}t_{2,1}t_{1,2}t_{2,2}$.

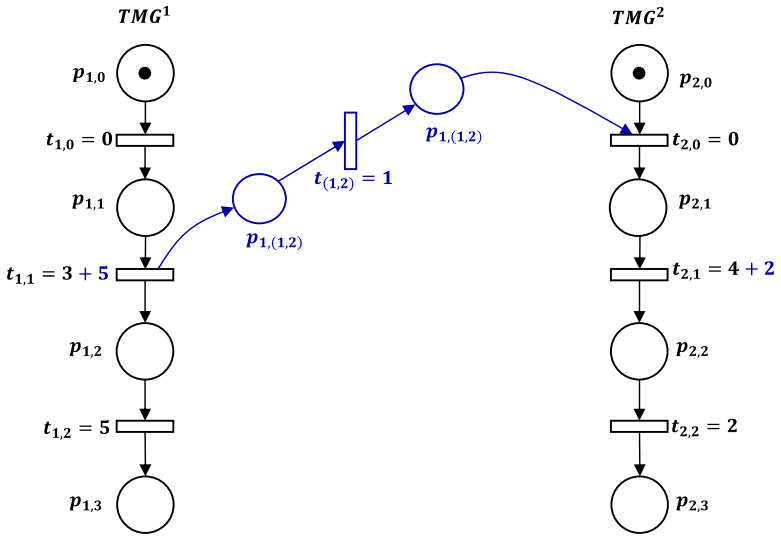

Figure 9. The controlled timed Petri net.

Figure 10 illustrates the optimal manufacturing scheduling obtained from the controlled TPN model.

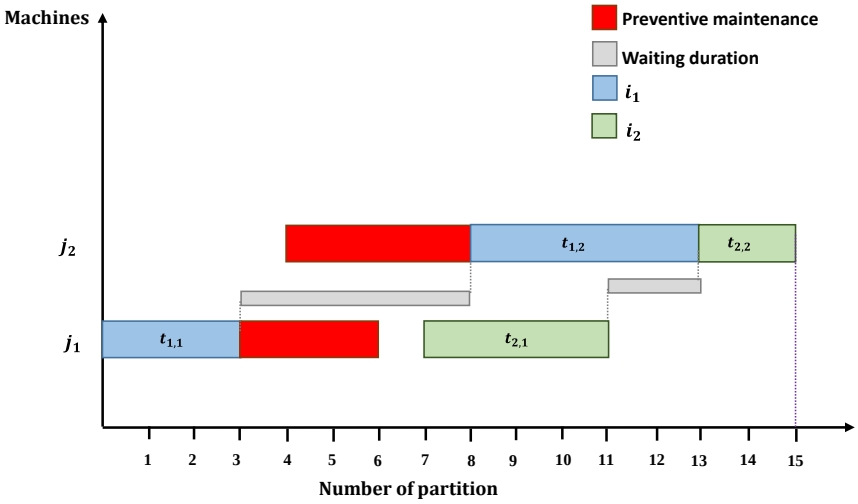

Figure 10. Optimal manufacturing scheduling.

8. Numerical Examples

A fully-automated flexible manufacturing system based at the Engineers' National School of Metz represents the application object of our proposed approaches.

8.1. System Description

The manufacturing system consists of three machines, an initial buffer and a final buffer, which are arranged in series around a central conveyor (Figure 11). Thereby, every machine is designed to realize an operation on a glass part. Each machine function is described as follows:

- j_0 : initial buffer;
- j_{Eng} : engraving of glass part by the laser;
- j_{Ass} : assembly of glass part engraved at a stand;
- j_{Pac} : packaging of the glass part engraved and assembled in a case;
- j_4 : final buffer.

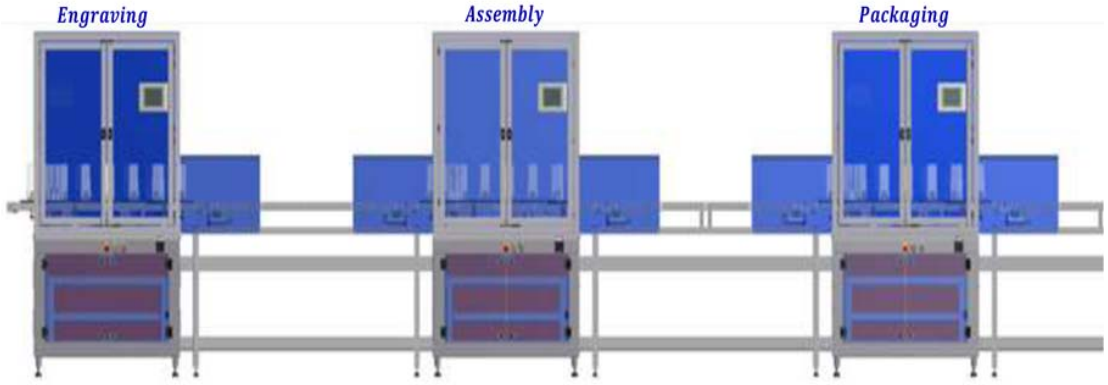

Figure 11. FMS real case environment.

We consider that the system produces three jobs (glass parts), which are differentiated by their color: blue (i_b), red (i_r) and green (i_g). The processing time of every job is presented in Table 4.

Table 4. Operating time.

	j_{Eng}	j_{Ass}	j_{Pac}
Red glass part	3	2	3
Blue glass part	2	3	3
Green glass part	4	4	2

The maintenance action periods for three machines are illustrated by Table 5.

Table 5. The preventive maintenance periods.

	B_j	E_j
j_{Eng}	2	7
j_{Ass}	0	6
j_{Pac}	10	13

8.2. Illustrative Example for the Genetic Algorithm

In view of the problem NP-hardness, in the case of more than two machines apart from the initial and final buffer since they do not increase the complexity of the problem (their operating time is zero), we have proposed Algorithm 2 to solve the (13)–(21) model.

We consider the database given by Tables 4 and 5. The genetic algorithm parameters are used as follows:

- $P = 100$ individuals;
- $P_{cross'} = 0.95$;
- $P_{MUT} = 0.05$;
- $Niter = 200$

Table 6 illustrates the sub-optimal value for decision variables and firing sequence.

Table 6. The optimal solution of three jobs and three machines.

The Optimal Firing Sequence
$\sigma = t_{b,0}t_{b,Eng}t_{b,Ass}t_{g,Eng}t_{b,Pac}t_{g,Ass}t_{r,Eng}t_{g,Pac}t_{r,Ass}t_{r,Pac}$
The resulting decision variables values
$H(t_{b,0}) = 0$; $H\left(t_{b,Eng}\right) = 7$; $H(t_{b,Ass}) = 13$; $H(t_{g,0}) = 9$; $H(t_{g,Eng}) = 13$; $H(t_{b,Pac}) = 16$; $H\left(t_{g,Ass}\right) = 18$; $H(t_{r,0}) = 14$; $H(t_{r,Eng}) = 18$; $H(t_{g,Pac}) = 20$; $H(t_{r,Ass}) = 21$ $H_{max} = H(t_{r,Pac}) = 24$; $A\left(t_{b,Eng}, t_{g,0}\right) = 1$; $A(t_{g,Eng}, t_{r,0}) = 1$; $Z_{b,Ass} = 1$; $Z_{g,Eng} = 1$; $Z_{b,Pac} = 1$.

In the same way as the previous described example, the computational results (Table 3) are used to determine supervisor parameters as shown in Figure 12.

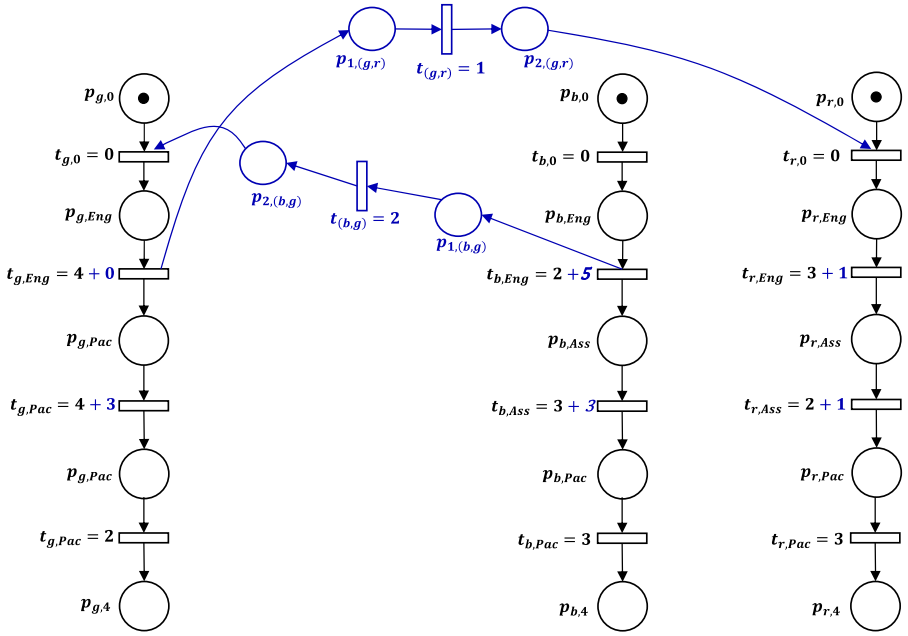

Figure 12. The resulting controlled TPN using GA.

Table 7 illustrates decision variables resulting from the Algorithm 2 resolution for a set of different jobs and machines. In the fourth and fifth column, we have considered new jobs: i_{b^1} and i_{b^2}, having the same operating time as blue glass i_b, i_{g^1} and i_{g^2} as green glass i_g.

The effectiveness of the genetic algorithm is tested by evaluating the makespan computed with that determined by the mathematical model (through the Xpress optimizer). In order to make the comparison, we have computed the makespan for two machines as a function of the job number (Figure 13).

On the basis of the results shown in Table 7 and Figure 13, the GA yields a good average makespan compared to the optimal solutions given by the mathematic model. The actual computing time for each problem was nearly the same for the two resolutions of the frameworks.

Table 7. Genetic algorithm results.

	2 (b,g)	**3** (b,g,r)	**4** $\left(b^1,b^2,g,r\right)$	**5** $\left(b^1,b^2,g^1,g^2,r\right)$
2 $\left(j_{Eng},j_{Ass}\right)$	$H\left(t_{g,Ass}\right)=18$ $A\left(t_{b,Eng},t_{g,0}\right)=1$ $Z_{b,Ass}=1$ $Z_{g,Eng}=1$	$H(t_{r,Ass})=19$ $A\left(t_{b,Eng},t_{r,0}\right)=1$ $A\left(t_{r,Eng},t_{b,0}\right)=1$ $Z_{b,Ass}=1$ $Z_{r,Eng}=1$	$H(t_{r,Ass})=25$ $A\left(t_{b^2,Eng},t_{r,0}\right)=1$ $A\left(t_{r,Eng},t_{g,0}\right)=1$ $A\left(t_{g,Eng},t_{b^1,0}\right)=1$ $Z\left(t_{b^2,Ass}\right)=1$ $Z_{r,Eng}=1$	$H\left(t_{g^1,Ass}\right)=33$ $A\left(t_{b^1,Eng},t_{g^2,0}\right)=1$ $A\left(t_{g^2,Eng},t_{b^2,0}\right)=1$ $A\left(t_{b^2,Eng},t_{r,0}\right)=1$ $A\left(t_{r,Eng},t_{g^1,0}\right)=1$ $Z_{b^1,Ass}=1, Z_{g^2,Eng}=1$
3 $\left(j_{Eng},j_{Ass},j_{Pac}\right)$	$H\left(t_{g,Pac}\right)=22$ $A\left(t_{b,Eng},t_{g,0}\right)=1$ $Z_{b,Ass}=1$ $Z_{g,Eng}=1$	$H(t_{r,Pac})=24$ $A\left(t_{b,Eng},t_{g,0}\right)=1$ $A\left(t_{g,Eng},t_{r,0}\right)=1$ $Z_{b,Ass}=1$ $Z_{g,Eng}=1$ $Z_{b,Pac}=1$	$H\left(t_{b^2,Pac}\right)=30$ $A\left(t_{b^1,Eng},t_{r,0}\right)=1$ $A\left(t_{r,Eng},t_{g,0}\right)=1$ $A\left(t_{g,Eng},t_{b^2,0}\right)=1$ $Z_{b^1,Ass}=1$ $Z_{r,Eng}=1$ $Z_{b^1,Pac}=1$	$H(t_{r,Ass})=48$ $A\left(t_{b^1,Eng},t_{g^1,0}\right)=1$ $A\left(t_{g^1,Eng},t_{b^2,0}\right)=1$ $A\left(t_{b^2,Eng},t_{r,0}\right)=1$ $A\left(t_{r,Eng},t_{g^2,0}\right)=1$ $Z_{b^1,Ass}=1, Z_{b^1,Pac}=1$ $Z\left(t_{g,Eng}\right)=1$

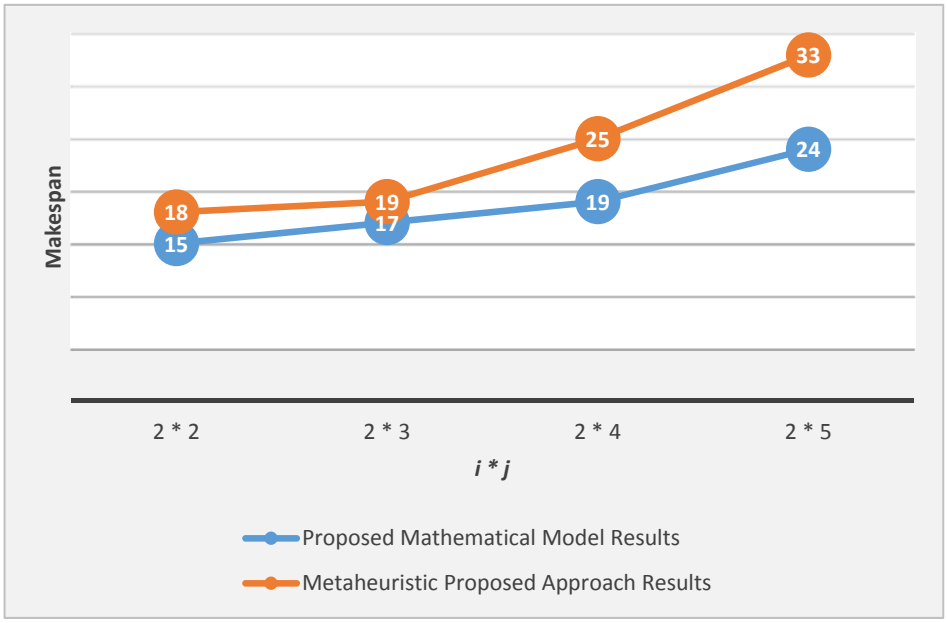

Figure 13. Makespan evolution as a function of models.

8.3. Comparative Study

In this section, a comparison between the above approaches' results and those given by the works of [21] is established. We have chosen this study to compare, since the authors deal with the same problem, which is the FMS scheduling problems, where the production process could be modeled by Petri net properties, but in different resolution framework conditions. In order to compare various parameter specifications under the same computational load, model performances are determined through the makespan value.

Li et al. [21] did not consider the availability constraint in their proposal. Results shown in Table 8 illustrate the comparison of their model adding the constraints of availability and our mathematical model. On the basis of makespan (*Ms*) values, it can be concluded that the results given by our approach are in good agreement with those given by Li et al.'s [21] approach. In addition, the efficiency

of our approach could also be highlighted by the fact that [21] assume that the order of jobs could differ in machines (more relaxed).

Table 8. Comparison of the two models applied with the availability constraints.

Case	Size $i \times j$	Lot Size	*Ms* [21]	*Ms*
1	3×3	1	43	36
2	5×6	1	64	57
Processing time for Case 1		5, 3, 3; 5, 4, 5; 4, 1, 3		
Processing time for Case 2		1, 2, 4, 5, 1, 5; 5, 2, 2, 3, 4, 1; 5, 4, 5, 2, 4, 2; 2, 3, 5, 3, 1, 3; 3, 3, 4, 3, 3, 2		
Maintenance periods for Case 1		[2, 7]; [0, 6]; [10, 13]		
Maintenance periods for Case 2		[2, 7]; [0, 6]; [10, 13]; [20, 23]; [30, 35]; [41, 43]		

9. Conclusions

Reachability marking numbers increase exponentially, which leads to FMS state saturation. Consequently, previously-proposed models in the literature perform poorly. Therefore, in this study, we proposed a mathematical model, based on the decomposed TPN property, to solve the scheduling problem and minimize the processing time. In addition, a genetic algorithm is introduced, to provide efficiency solutions in the case of large-scale scheduling problems. Promising results indicate that the new model is efficient compared with similar models.

As perspectives, further studies should be considered to extend the proposed approaches for a parallel machine purpose with several practical constraints, such as limited buffer capacity.

Acknowledgments: This work is sponsored by the University of Lorraine and is a part of supervisor implementation project for a flexible manufacturing system in National Engineering School of Metz (ENIM). This work is a result of a coordination between two research teams in the industrial engineering, production and maintenance laboratory.

Author Contributions: Mohamed Ali Kammoun and Wajih Ezzeddine developed the approaches and conceived the experiments. Zied Achour contributed to the literature review. Nidhal Rezg provided assistance in models development and supervised results.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dubey, R.; Ali, S.S. Identification of flexible manufacturing system dimensions and their interrelationship using total interpretive structural modelling and fuzzy MICMAC analysis. *Intell. J. Flex. Syst. Manuf.* **2014**, *15*, 131–143. [CrossRef]
2. Scholze, S.; Barata, J.; Stokic, D. Holistic Context-Sensitivity for Run-time Optimization of Flexible Manufacturing Systems. *Sensors* **2017**, *17*, 455. [CrossRef] [PubMed]
3. Kolisch, R. *Project Scheduling under Resource Constraints: Efficient Heuristics for Several Problem Classes*; Springer: New York, NY, USA, 2013.
4. Zhang, L.; Li, X.; Gao, L.; Zhang, G. Dynamic rescheduling in FMS that is simultaneously considering energy consumption and schedule efficiency. *Int. J. Adv. Manuf. Technol.* **2016**, *87*, 1387–1399. [CrossRef]
5. Baruwa, O.T.; Piera, M.A. Identifying FMS repetitive patterns for efficient search-based scheduling algorithm: A colored Petri net approach. *J. Manuf. Syst.* **2015**, *35*, 120–135. [CrossRef]
6. Huang, B.; Jiang, R.; Zhang, G. Search strategy for scheduling flexible manufacturing systems simultaneously using admissible heuristic functions and nonadmissible heuristic functions. *Comput. Ind. Eng.* **2014**, *71*, 21–26. [CrossRef]
7. Maccarthy, B.L.; Liu, J. Addressing the gap in scheduling research: A review of optimization and heuristic methods in production scheduling. *Int. J. Prod. Res.* **1993**, *31*, 59–79. [CrossRef]
8. Xie, C.; Allen, T.T. Simulation and experimental design methods for job shop scheduling with material handling: A survey. *Int. J. Adv. Manuf. Technol.* **2015**, *80*, 233–243. [CrossRef]
9. Jeng, M.; Lin, C.; Huang, Y. Petri net dynamics-based scheduling of flexible manufacturing systems with assembly. *J. Intell. Manuf.* **1999**, *10*, 541–555. [CrossRef]

10. Ghaffari, A.; Rezg, N.; Xie, X. Feedback control logic for forbidden-state problems of marked graphs: Application to a real manufacturing system. *Trans. Autom. Control* **2003**, *48*, 18–29. [CrossRef]
11. Lin, F.; Vaz, A.F.; Wonham, W.M. Supervisor specification and synthesis for discrete event systems. *Int. J. Control* **1988**, *48*, 321–332. [CrossRef]
12. Kammoun, M.A.; Rezg, N.; Achour, Z. New approach for air traffic management based on control theory. *Int. J. Prod. Res.* **2014**, *52*, 1711–1727. [CrossRef]
13. Kammoun, M.A.; Rezg, N.; Achour, Z.; Rezig, S. State Space Search for Safe Time Petri Nets Based on Binary Decision Diagrams Tools: Application to Air Traffic Flow Management Problem. *Stud. Inf. Control* **2016**, *25*, 39–50.
14. Murata, T. Petri nets: Properties, analysis and applications. *Proc. IEEE* **1989**, *77*, 541–580. [CrossRef]
15. Peterson, J.L. *Petri Nets Theory and the Modelling of Systems*; Prentice-Hall: Englewood Cliffs, NJ, USA, 1981.
16. Al-Jaar, R.Y.; Desrochers, A.A. Performance evaluation of automated manufacturing systems using generalized stochastic Petri nets. *Trans. Robot. Autom.* **1990**, *6*, 621–639. [CrossRef]
17. Lee, D.Y.; DiCesare, F. Scheduling flexible manufacturing systems using Petri nets and heuristic search. *IEEE Trans. Robot. Autom.* **1994**, *10*, 123–132. [CrossRef]
18. Der Jeng, M.; Chen, S.C. A heuristic search approach using approximate solutions to Petri net state equations for scheduling flexible manufacturing systems. *Intell. J. Flex. Manuf. Syst.* **1998**, *10*, 139–162. [CrossRef]
19. Pearl, J. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, 1rd ed.; Addison-Wesley Pub: Reading, MA, USA, 1984.
20. Yu, H.; Reyes, A.; Cang, S.; Lloyd, S. Combined Petri net modelling and AI based heuristic hybrid search for flexible manufacturing systems—Part 1. Petri net modelling and heuristic search. *Comput. Ind. Eng.* **2003**, *44*, 527–543. [CrossRef]
21. Li, C.; Wu, W.; Feng, Y.; Rong, G. Scheduling FMS problems with heuristic search function and transition-timed Petri nets. *J. Intell. Manuf.* **2015**, *26*, 933–944. [CrossRef]
22. Xiong, H.H.; Zhou, M.C. Scheduling of semiconductor test facility via Petri nets and hybrid heuristic search. *Semicond. Manuf.* **1998**, *11*, 384–393. [CrossRef]
23. Ingimundardottir, H.; Runarsson, T.P. Supervised learning linear priority dispatch rules for job-shop scheduling. In Proceedings of the International Conference on Learning and Intelligent Optimization, Rome, Italy, 17–21 January 2010; pp. 263–277.
24. Özgüven, C.; Özbakır, L.; Yavuz, Y. Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Appl. Math. Model.* **2010**, *34*, 1539–1548. [CrossRef]
25. Chung, Y.Y.; Fu, L.C.; Lin, M.W. Petri Net based modeling and GA based scheduling for a flexible manufacturing system. *Proc. Decis. Control* **1998**, *4*, 4346–4347.
26. Dasgupta, D.; Michalewicz, Z. *Evolutionary Algorithms in Engineering Applications*, 1st ed.; Springer: New York, NY, USA, 2013.
27. Davis, L. *Handbook of Genetic Algorithms*; Van No Strand Reinhold: New York, NY, USA, 1991.
28. Saitou, K.; Malpathak, S.; Qvam, H. Robust design of flexible manufacturing systems using, colored Petri Net and genetic algorithm. *J. Intell. Manuf.* **2002**, *13*, 339–351. [CrossRef]
29. Barbierato, E.; Dei Rossi, G.L.; Gribaudo, M.; Iacono, M.; Marin, A. Exploiting product forms solution techniques in multiformalism modeling. *Electron. Not. Theor. Comput. Sci.* **2013**, *296*, 61–77. [CrossRef]
30. Nishi, T.; Maeno, R. Petri net decomposition approach to optimization of route planning problems for AGV systems. *Trans. Autom. Sci. Eng.* **2010**, *7*, 523–537. [CrossRef]
31. Nishi, T.; Tanaka, Y. Petri net decomposition approach for dispatching and conflict-free routing of bidirectional automated guided vehicle systems. *Trans. Syst. Man Cybern. A Syst. Hum.* **2012**, *42*, 1230–1243. [CrossRef]
32. Ciardo, G.; Trivedi, K.S. A decomposition approach for stochastic Petri net models. In Proceedings of the Fourth International Workshop on Petri Nets and Performance Models (PNPM91), Melbourne, Australia, 2–5 December 1991; pp. 74–83.
33. Al-Ahmari, A. Optimal robotic cell scheduling with controllers using mathematically based timed Petri nets. *Inf. Sci.* **2016**, *329*, 638–648. [CrossRef]
34. Goldberg, D.E.; Samtani, M.P. Engineering optimization via genetic algorithm. In *Electronic Computation, Proceedings of the Ninth Conference on Electronic Computation, Birmingham, AL, USA, 23–26 February 1986*; American Society of Civil Engineers: New Yor, NY, USA, 1986; pp. 471–482.

35. Sarin, S.; Lefoka, M. Scheduling heuristic for the n-job m-machine flow shop. *Omega* **1993**, *21*, 229–234. [CrossRef]
36. Garey, M.R.; Johnson, D.S.; Stockmeyer, L. Some simplified NP-complete graph problems. *Theor. Comput. Sci.* **1976**, *1*, 237–267. [CrossRef]
37. Murata, T.; Ishibuchi, H.; Tanaka, H. Genetic algorithms for flowshop scheduling problems. *Comput. Ind. Eng.* **1996**, *30*, 1061–1071. [CrossRef]
38. Reeves, C.R. A genetic algorithm for flowshop sequencing. *Comput. Oper. Res.* **1995**, *22*, 5–13. [CrossRef]
39. Framinan, J.M.; Gupta, J.N.; Leisten, R. A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *J. Oper. Res. Soc.* **2004**, *55*, 1243–1255. [CrossRef]
40. Guéret, C.; Prins, C.; Sevaux, M. *Applications of Optimization with Xpress-MP*, 1rd ed.; Dash Optimization Ltd.: Leamington Spa, UK, 1999.
41. Daniel, B. *Xpress-Optimizer Reference Manual*, 1rd ed.; Fair Isaac Corporation: Leamington Spa, UK, 2009.
42. Laundy, R.; Perregaard, M.; Tavares, G.; Tipi, H.; Vazacopoulos, A. Solving hard mixed-integer programming problems with Xpress-MP: A MIPLIB 2003 case study. *J. Comput.* **2009**, *21*, 304–313. [CrossRef]