Journal of Control, Automation and Electrical Systems (2021) 32:591–604
https://doi.org/10.1007/s40313-021-00700-5

Scheduling Problems for a Class of Hybrid FMS Using T-TPN and Beam Search

G. Cherif[1] · E. Leclercq[1] · D. Lefebvre[1]

Received: 13 May 2020 / Revised: 2 December 2020 / Accepted: 5 February 2021 / Published online: 3 March 2021
© Brazilian Society for Automatics--SBA 2021

Abstract
This paper is about scheduling problems for a class of flexible manufacturing systems (FMS) that have some operations with total precedence constraints and other operations with full routing flexibility (namely *hybrid FMS*). The objective is to find a control sequence from an initial state to a reference one in minimal time. For that, a systematic multi-level formalism is introduced to model the hybrid FMS based on the hierarchical structuration of the operations. Transition-timed Petri nets (T-TPN) that behave under earliest firing policy are used for that purpose. Then a new cost function is introduced to estimate the residual time to the reference. This estimation is proved to be a lower bound of the true duration. A modified Beam Search algorithm is proposed that uses the cost function to selectively explore the Petri net (PN) state space. Computational experiments illustrate the efficiency of the approach in comparison with other existing methods.

Keywords Complex discrete event systems · T-TPN · Scheduling · Beam search

1 Introduction

Flexible manufacturing systems (FMS) consist of a finite set of operations and resources and can process different types of parts based on a prescribed sequence of operations. In an FMS, there is some amount of flexibility that makes it able to respond to varying conditions. The purpose of this work is to solve scheduling problems for flexible and smart FMS used by the industry of the future. Nowadays production with large series of identical products is no longer competitive. Consumers ask for personalized products. Consequently, the great challenge of Industry 4.0 is to offer personalized products, and despite low manufacturing volumes, to maintain gains. The consumers also like to communicate with the processes and the machines during the production phases. The concept of industry 4.0 or industry of the future corresponds to such new organizations of the means of production called "smart production". Smart production requires, for example, to give flexibility to some operations (and to maintain precedence constraints to other ones) as shown in Fig. 1: in the first job, the controller could start by M_1 then M_2 or by M_2 then M_1. Smart production also requires to use the same machines or servers to achieve different operations: in Fig. 1, M_1 is used for an operation with a duration of 60 s in the first job and for another operation (different from the previous one) with a duration of 100 s in the second job.

In this context, a particular class of FMS that combines operations with total precedence constraints and operations with full routing flexibility (namely *hybrid FMS*) is proposed in this work.

Petri nets (PN) have been widely used for the design, modelling and analysing of FMS with concurrency, synchronization, sequencing and sharing of resources. The optimization of a specific cost function is a main objective of many scheduling problems aiming to allocate a limited number of resources within several operations. However, the scheduling problem of complex FMS is NP-hard and the computational time to obtain an optimal schedule grows exponentially with respect to the problem size. The problem becomes even more challenging in the context of Industry 4.0. Thus, a large literature has been devoted to such optimization problems (Cassandras 1993; Lee and DiCesare 1994; Luo et al. 2015). In this paper, we develop first a systematic structured formalism that is suitable to represent hybrid FMS from the industry of the future. Then, based on the obtained model and on a new cost function that is

✉ G. Cherif
ghassen.cherif@univ-lehavre.fr;edouard.leclercq@univ-lehavre.fr;
dimitri.lefebvre@univ-lehavre.fr

[1] GREAH-UNILEHAVRE, Normandie Université, Caen, France

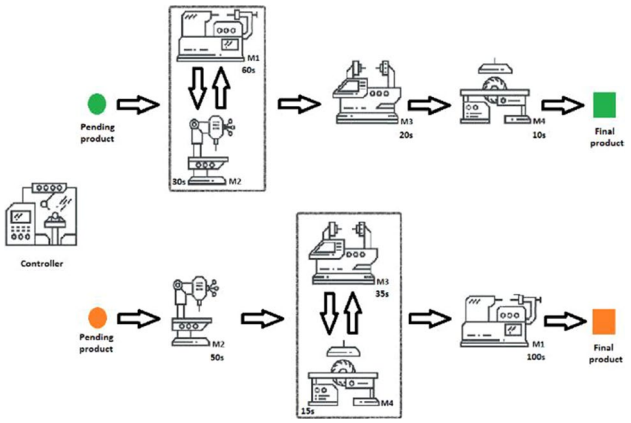


Fig. 1 Example of an FMS organization in Industry 4.0

proved to be a lower bound of the true makespan, a variant of the Beam Search method is detailed to compute sequences of minimal duration by exploring selectively the PN state space. The approach should also avoid deadlocks and dead branches that are a priori unknown for the controller.

The paper is organized as follows. Section 2 is about the state of the art and gives the position of the paper with respect to the usual organizations of FMS and the control issues of scheduling problems. In Sect. 3, we present timed Petri nets (TPN) and Sect. 4 details the method used to model the hybrid jobs with transition-timed Petri nets (T-TPN). In Sect. 5, the approach used for scheduling control of hybrid FMS is described. Finally, computational experiments and results are exposed in Sect. 6.

2 State of the Art

PN have been widely used to model scheduling problems for discrete event systems (DES). Liu and Barkaoui (2016) have listed and discussed the subclasses of PN used for this purpose. In this section, we remind some usual organizations of FMS and the use of PN for scheduling problems.

2.1 Job Shop Modelling

A job shop is a manufacturing process structure in which a set of jobs are processed on multiple resources with total precedence constraints. In a job shop, called also multi-path workshop, each job is composed of a sequence of operations performed according to a well-defined order a priori fixed. Each operation is processed on a specific resource. In order to model the job shop scheduling problem, a subclass of ordinary and conservative PN referred to as S3PR (System of Simple Sequential Processes with Resources) has been defined (Ezpeleta et al. 1995; Xing et al. 1996).

Later, in order to extend the use of resources in job shop, a generalization of S3PR model referred to as S4R (Systems of Sequential Systems with Shared Resources) has been defined by Barkaoui and Ben Abdallah (1996). This extension exceeds the limitation of the use of resources and concerns especially the modelling of concurrently cyclic sequential processes sharing common resources: in S3PR, only one resource is allowed per operation which is not the case for S4R where the use of resources is almost arbitrary and only requires conservativeness.

2.2 Open Shop Modelling

An open shop is a workshop with full routing flexibility where a set of jobs are processed on multiple resources without any precedence constraints. In order to model the open shop scheduling problem, the S3PR nets could be used. The problem is that due to complexity issues, this representation becomes quickly difficult to handle especially for large systems. Therefore, the use of S3PR leads to an exponential explosion of the size of the net with regard to the number of operations because there are $n!$ possible routes for n operations of a single job. An extension of S3PR, referred to as S2OPR (Set of Simple Open Processes with Resources), has been proposed to overcome the problem (Mejía et al. 2017). This extension is a compact representation for the open shop problem, and the net is much smaller compared to its equivalent S3PR model, but it needs to be 1-bounded.

2.3 Scheduling Control

PN are largely used for control issues of DES (Cassandras 1993). They are also a popular graphical and mathematical tool used to handle the scheduling problems with routing flexibility in the presence of shared resources. The key concept is to explore selectively the state space of the PN model with the intention of finding a firing sequence, with the minimum duration. The firing sequence is computed by optimizing a specific cost function.

A first approach concerns global methods where the whole reachability graph is explored (Lefebvre and Daoui 2018; Lefebvre 2018). This approach leads surely to optimal solutions, but the problem is the exponential explosion of the computational time and memory especially for large systems. Consequently, the search space grows exponentially with problem scale and makes the scheduling method applicable to small systems only. To overcome the limitation of global methods, a large literature has been devoted in order to search solutions with weak complexity.

The A* search algorithm is a very well-known algorithm used to accelerate the search of solutions. This technique expands only the most promising branches of a search tree. Lee and DiCesare (1994) proposed an algorithm which

combines the Petri net reachability graph and the A* to schedule FMS. However, the result leads to an explosion of the state space of the net (Pearl 1984). To improve the exploration, researchers developed several improvements on it, for example best-first strategy with controlled backtracking (Xiong and Zhou 1998; Lefebvre 2016), dynamic programing (Xiong et al. 2014), hybrid A* search in order to relax the evaluation scope (Lee and Lee 2010), dynamic window search algorithm in which a policy is applied to select the most promising paths (Reyes et al. 2002), A* algorithm with depth-first strategy (Huang et al. 2008), iterative deepening A* with backtracking (Baruwa et al. 2015) and more informed heuristics (Liu et al. 2014; Moro et al. 2000; Huang et al. 2014; Mejía and Niño 2017).

Contrary to the A* algorithm where the previously explored candidates during search are all conserved, an informed graph search referred to as Beam Search (BS) algorithm (Ow and Morton 1988) conserves only the best β previously explored candidates. The complexity in time and memory remains polynomial thanks to this restriction, but the main drawback is that promising candidates could be eventually eliminated due to the selection method. Several variants of the Beam Search algorithm in combination with Petri nets have been proposed (Mejía and Odrey 2005; Mejía and Montoya 2009; Luo et al. 2015). However, for complex FMS, these improvements remain either insufficient or inapplicable to reach the optimal scheduling.

3 Timed Petri Nets

A PN structure is defined as $<P, T, W_{PR}, W_{PO}>$, where $P=\{p_1,\ldots, p_m\}$ is a set of m places and $T=\{t_1,\ldots, t_q\}$ is a set of q transitions. $W_{PO}\in(\mathbf{N})^{m\times q}$ and $W_{PR}\in(\mathbf{N})^{m\times q}$ are the post- and pre-incidence matrices ($\mathbf{N}$ is the set of non-negative integer numbers), and $W=W_{PO}-W_{PR}\in(\mathbf{Z})^{m\times q}$ ($\mathbf{Z}$ is the set of positive and negative integer numbers) is the incidence matrix. For any node $x\in T\bigcup P$, $x\bullet$ stands for the postset of x and $\bullet x$ stands for the preset of x. $<G, M_0>$ is a PN system with initial marking M_0 and $M\in(\mathbf{N})^m$ represents the PN marking vector. For each marking M, $P(M)\subseteq P$ is defined as the subset of non-empty places at marking M (i.e. the support of M). A transition t_j is enabled at M if there are enough tokens in the preset of t_j: $\forall\, p_i\in P, M(p_i)\geq W_{PR}(p_i,t_j)$ where $M(p_i)$ stands for the number of tokens in place p_i and $W_{PR}(p_i,t_j)$ is the element of matrix W_{PR} at row i and column j. The firing of a transition t at marking M removes $W_{PR}(p, t)$ tokens from each place p in $\bullet t$ and adds $W_{PO}(p, t)$ tokens to each place p in $t\bullet$. This is written as $M\,[t>M'$ (Ezpeleta et al. 1995; Mejía et al. 2017) where M' is the resulting marking when the enabled transition t fires at marking M.

There are several classes of TPN, in particular T-TPN, that associate a firing duration to each transition in the net (Ramchandani 1974), place-timed Petri net (P-TPN) that associate a sojourn duration to each place of the net (Sifakis 1979) and time PN that associate a time interval with each transitions (Merlin 1974). In this paper, T-TPN are used to represent the time required to perform the operations.

A T-TPN is a 6-tuple $N=<P, T, W_{PR}, W_{PO}, D, M_0>$ where $<P, T, W_{PR}, W_{PO}, M_0>$ is a marked PN, and D: $T\rightarrow\mathbf{R}^+$ is a firing time function that assigns a positive real delay $D(t)$ to each transition t of the net. A transition t may fire at earliest after the minimal delay $D(t)$ from the date it has been enabled. The *residual firing time* is defined for each transition t_j at each marking M. If a transition t_j becomes enabled at time $\tau_{\text{enabled}}(t_j)$, it will fire at earliest at time $\tau_{\text{enabled}}(t_j)+D(t_j)$. In this paper, the residual firing time $RFT(t_j)\in[0, D(t_j)]$ of transition t_j is defined as the minimum value between the non-negative difference between the current time τ and the enabled time $\tau_{\text{enabled}}(t_j)$ and the minimal delay $D(t_j)$: $RFT(t_j)=\min(\tau—\tau_{\text{enabled}}(t_j), D(t_j))$.

A timed firing sequence σ of length $|\sigma|=K$ and of duration τ_K is defined as $\sigma=t(\tau_1)t(\tau_2)\ldots t(\tau_K)$ where $\tau_1,\ldots, \tau_K$ represent the times of the firings that satisfy $0\leq\tau_1\leq\tau_2\leq\ldots\leq\tau_K$. The timed firing sequence σ fired at M leads to the timed trajectory $(\sigma,\ M)=M(0)\ [t(\tau_1)>M(1)\ldots.\ M(K-1)\ [t(\tau_K)>M(K)$ with $M(0)=M$. The notation $M\ [\sigma>M(K)$ denotes that there exists such a valid timed firing sequence σ from marking M to marking $M(K)$.

In a PN structure, a *Path* is an orderly succession of K nodes $x_1\ x_2\ \ldots\ x_K$ with $x_k\in T\bigcup P$ such as $x_{k+1}\in x_k\bullet$ for $k=1,\ldots, K-1$. The duration of *Path* is defined as:

$$\chi(Path)=\sum\nolimits_{t\in Path}D(t) \tag{1}$$

A *Path* is said of minimal duration if there exists no other path from x_1 to x_K with a smaller path duration and we refer to the path of minimal duration from x_1 to x_N as to $Path^*(x_1, x_N)$. We refer to the duration of $Path^*(x_1, x_N)$ as to $\chi^*(x_1, x_N)$.

4 Complex FMS Modelling

A FMS is a system composed of several jobs and resources. Each job is composed by a set of operations which perform on the resources. Usual FMS are job shops and open shops but more general organizations could be considered as hybrid FMS studied in this work. Let $\boldsymbol{J}$ and $\boldsymbol{R}$ be, respectively, a set of jobs and a set of resources. Each job $J\in\boldsymbol{J}$ consists of a set of operations $O_J=\{o_i, i\in\{1,\ldots,n\}\}$, where o_i is the operation i of job J. The set of all operations is denoted $O=\cup\{O_J, J\in\boldsymbol{J}\}$. Note that for simplicity the reference of a given operation o_i to the corresponding job J is not explicitly

reported, but since each operation o_i belongs to a single job J there is no confusion.

An *operation* o_i is defined by a duration d_i representing the time required to be performed and a subset of resources $R_i \subseteq \boldsymbol{R}$ required for its execution: $o_i = (d_i, R_i)$. An operation o_i is modelled using one place p_i and one timed transition to_i so-called operation transition with $D(to_i) = d_i$ as shown in Fig. 2. Resource places are connected to the operation transition to_i by means of self-loops: when o_i is executed, the set of resources R_i needed is reserved from the enabling time to the firing time of to_i.

4.1 T-TPN Model of a Sequence of Operations

T-TPN are used to model sequences of operations. In the following, we assume for simplicity that the capacities of the jobs of the FMS are limited to 1. The minimal duration of a single execution of the job J composed by a sequence of operations and without shared resource is given by (2):

$$D(J) = \sum_{to \in J} D(to) \tag{2}$$

An example of a cyclic sequence of two operations with a minimal duration $D(J) = d_1 + d_2$ is shown in Fig. 3. The operations of the job are ordered, the job processes o_1 that requires the resource r_1 and then o_2 that requires r_2.

When the capacity is equal to 1, only one token can pass to perform the job and the others will stand on the start place s_J (s_1 in the example of Fig. 3), waiting for the first token to finish all the operations of the job (o_1 and o_2 in the example of Fig. 3). When the first token passes to perform the job, the capacity becomes 0 (there is no token in the place cp_1 of this example) and no more token in s_J can pass. Then, once the token has passed through all the operations of the job, the capacity returns to 1 and another token from s_J can be used to perform the job.

4.2 T-TPN Model of a Set of Operations with Full Routing Flexibility

T-TPN are also used to model a set of operations performed with full routing flexibility. The open shop could be modelled using S3PR or S2OPR. A comparison of sizes of both the S3PR and the equivalent S2OPR has been done

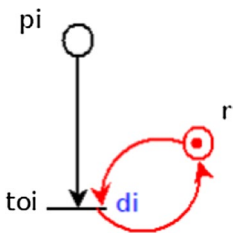


Fig. 2 T-TPN model of an operation with a single resource

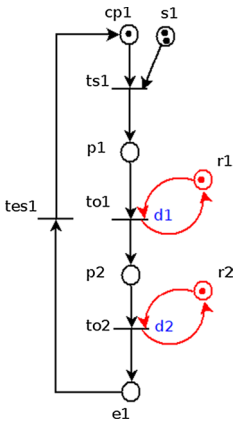


Fig. 3 T-TPN model of a sequence of operations

(Mejía et al. 2017): for large systems where operations are performed with full routing flexibility, the S2OPR net is much smaller than its equivalent S3PR model.

A controller composed by flag and counter places is used with S2OPR (Mejía et al. 2017). For each job with full routing flexibility, a set of flag places $PF = \{f_i, o_i \in O_J\}$ is connected to the postset of the start transition ts to ensure that the system executes each operation only once. In addition, a single counter place c is used to ensure that the system executes all operations. An example of two operations with full routing flexibility is shown in Fig. 4. The S3PR model is shown in Fig. 4a while its equivalent S2OPR model is shown in Fig. 4b. Operations of the job are not ordered: the job can process o_1 that requires the resource r_1 then o_2 that requires r_2 or o_2 then o_1. For S2OPR model, the controller is composed of the flag places f_1 and f_2 and the counter place c_{12}.

A set of operations performed with full routing flexibility without shared resource has a minimal duration $D(J)$ computed with (2). The minimal duration of the example shown in Fig. 4 is $D(J) = d_1 + d_2$.

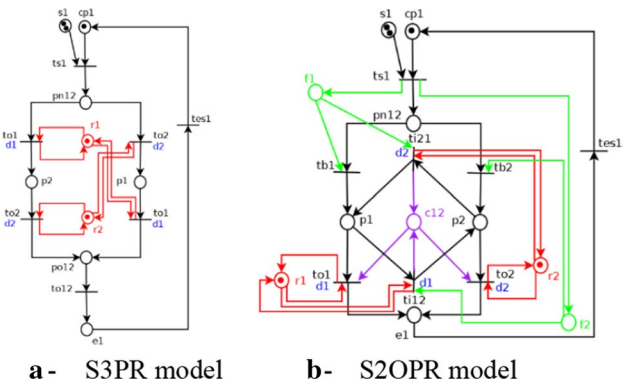


Fig. 4 T-TPN model of a set of operations with full routing flexibility

4.3 T-TPN Model of Hybrid Jobs

A FMS could include more complex organizations (and not only job shops and open shops) with partial routing flexibility. A particular class of complex shops, referred to as *hybrid jobs*, which satisfy some specific organization properties, is considered next. The organization of a hybrid job could be iteratively decomposed in a set of basic jobs. Such jobs could be either sequences of operations performed with total precedence constraints or sets of operations performed with full routing flexibility. The systematic design method introduced in our previous work (Cherif et al. 2018) to encode the iterated organization is improved here to be more efficient for the modelling of complex workshops. The modelling is based on some basic functions.

Operation (d_i, R_i) is the function used to model the operation o_i. The duration d_i of the operation and the set of needed resources R_i are given as inputs. It returns the object "operation" which may be considered as an elementary job.

Sequential (J_1, J_2,…,J_h) is the function used to model a sequence of h jobs (eventually elementary jobs) given as inputs. It returns the resulting job "sequence" which is modelled as a sequence of J_1 to J_h.

Open (J_1, J_2,…,J_h) is the function used to model a set of h jobs (eventually elementary jobs) with full routing flexibility. It returns the resulting job "open" as the set of jobs J_1 to J_h.

In addition, the operations in the hybrid job could share some resources.

Example: An example of hybrid FMS is described below. The hybrid FMS has two jobs J_1 and J_2 with 9 resources including 4 shared resources r_2, r_3, r_7 and r_9. J_1 consists of the sequence of operations o_1, o_2, o_3 and o_4. J_2 is a hybrid job composed of five operations o_5, o_6, o_7, o_8 and o_9. Figure 5 shows the organization of the FMS according to the three basic functions previously described. The T-TPN model of this hybrid FMS is presented in Fig. 6.

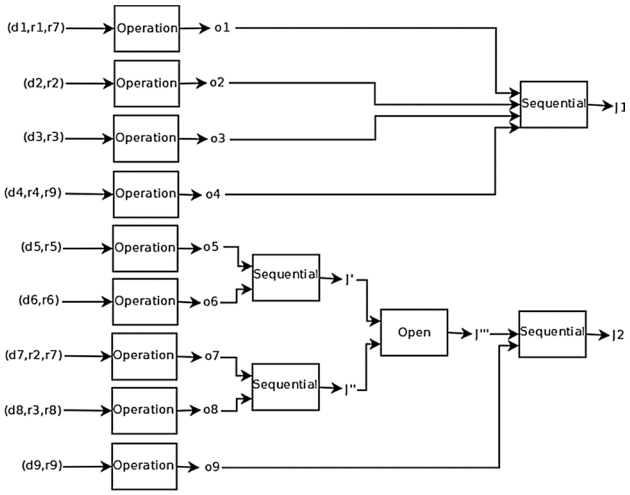


Fig. 5 Hybrid FMS organization

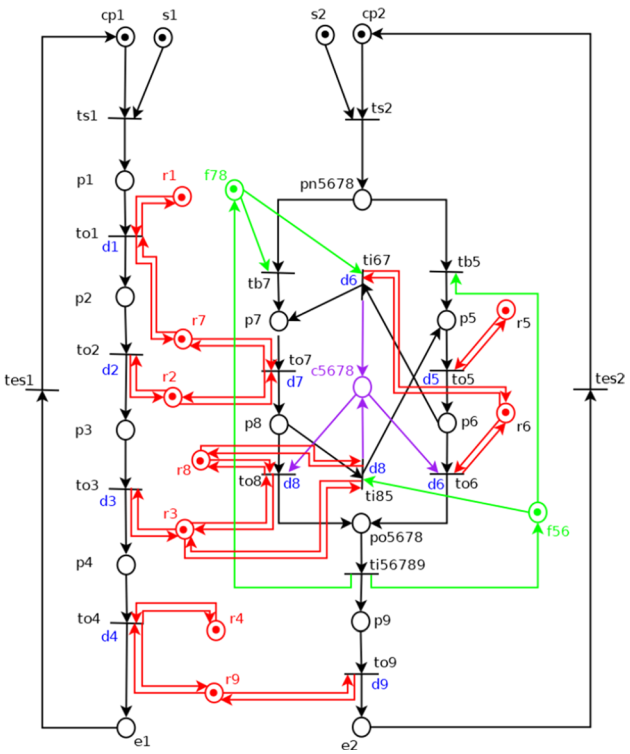


Fig. 6 Model of T-TPN model of the hybrid FMS

5 Beam Search for Scheduling Control

In this paper, an algorithm which is inspired from Hybrid Filtered Beam Search (HFBS) presented in (Mejía and Niño 2017) is proposed. The HFBS combines the main principles of the Beam Search (Sabuncuoglu and Bayiz 1999), the Filtered Beam Search (FBS) (Ow and Morton 1988) and the Beam A* Search (Mejía and Montoya 2009; Mejía and Odrey 2005). The idea of HFBS algorithm is to provide diversification at early steps and intensification at late steps of the search. Indeed, the algorithm aims to select multiple candidates from different branches at the first steps but only fewer candidates at the later steps. The algorithm depends on two parameters: a global filter parameter β_g which presents the maximum number of expanded candidates at one iteration of the algorithm (global beam width) and a local filter parameter β_l which presents the maximum number of expanded successors from one given parent node (local beam width). The search algorithm starts with an initial marking that is the initial population. The search is based on iterations: at the first iteration, the algorithm generates

the successors of the initial population and selects the best β_g candidates, and a new population is created. Then at each iteration, it expands the population obtained at the previous iteration and creates a new population composed of the best β_g candidates. The algorithm terminates in two cases: it may stop when the reference marking is reached or when all candidates have been expanded. One drawback is the selection of the best parameter combination (β_g and β_l) for a given instance which is not obvious even though large values of β_g and β_l do not ensure to result in the optimal solution but certainly to expand unpromising candidates. However, when the selection of the nodes is good, a near optimal solution or even an optimal solution can be obtained. The selection of candidates that will be expanded is based on a heuristic cost function which is defined next. The priority of expansion is given to the candidates that have the lowest values of the cost function.

5.1 Objective Function

In this section, a new cost function is defined to be relevant to complex FMS which consist of hybrid jobs where operations are performed which partial precedence constraints. The objective function calculates a value used to select the candidates to be expanded at each iteration of the search. Consider an initial marking M_0 and a reference marking M_{ref} such that one or more trajectories exist between M_0 and M_{ref}. Imagine that (σ, M_0) is one of these trajectories and that (σ, M_0) is incrementally computed. For each intermediate marking M of (σ, M_0), the firing sequence σ is divided into two parts according to $\sigma = \sigma_1 \sigma_2$. The first part corresponds to the already computed trajectory (σ_1, M_0) from M_0 to M (i.e., $M_0[\sigma_1 > M$), and the second part corresponds to the residual trajectory (σ_2, M) with an unknown sequence σ_2 from M to M_{ref}. The following cost function is reformulated as:

$$f(M_0, \sigma_1, M_{ref}) = g(\sigma_1, M_0) + h(M, M_{ref}) \tag{3}$$

where M is defined as $M_0[\sigma_1 > M$.

For the trajectory (σ_1, M_0) already computed, a systematic algorithm computes the duration $g(\sigma_1, M_0)$ of $(\sigma 1, M_0)$ by transforming the untimed trajectory into a timed trajectory under earliest firing policy (i.e. each transition t in σ_1 fires as soon as its time constraint $D(t)$ is satisfied). The algorithm, detailed in (Lefebvre 2017a, b), updates at each new marking M the remaining durations of the enabled transitions. The estimation of the residual time to the reference is simply the sum of durations of unperformed operations which does not take into account the availability of resources. Indeed, to estimate the residual time with the function $h(M, M_{ref})$, a reduced PN model $N_r = < P_r, T, Wr_{PR}, Wr_{PO}, D >$ is first introduced where $P_r = P \setminus \boldsymbol{R}$ is a set of m_r places where resource places are removed and T is conserved as a set of q transitions. $Wr_{PO} \in (\mathbf{N})^{mr \times q}$ and $Wr_{PR} \in (\mathbf{N})^{mr \times q}$ are the reduced post- and pre-incidence matrices, and $Wr = Wr_{PO} - Wr_{PR} \in (\mathbf{Z})^{mr \times q}$ is the reduced incidence matrix. The marking of the reduced PN is referred to as M_r.

For a sequence J of operations (for example, a job J in job shop problems), several estimation functions $h_J(M, M_{ref})$ of the residual duration of job J have been proposed: in Luo et al. (2015), h_J is based on the search of resource and operational places; in (Lefebvre 2017a), h_J is based on the residual firing count vector. In this work, we propose an estimation function h_J based on the search of the shortest paths from the marked places to a given reference place p_{ref} that is added as the end place to the global model of the FMS (Mejía and Nino 2017; Lefebvre and Mejía 2018). The objective function is improved here and adapted to deal with the modelling of the new organization of the FMS. The estimation h_J is defined depending on the type of the job. For a set of operations with total precedence constraints, the approximation $h_J(M, M_{ref})$ of the cost of the unknown part of the trajectory is given by (4):

$$\begin{aligned} h_J(M, M_{\text{ref}}) = {} & M_r(s_J) D(J) \\ & + \max\{\forall p_i \in P(M_r) \setminus \{s_J\}, \chi^*(p_i, e_J) \\ & - \text{RFT}(to_i) \text{with } to_i \in (p_i)\bullet\} \end{aligned} \tag{4}$$

where $M_r = Reduce(M)$ is the reduced marking obtained from M by removing capacity and resource places, s_J is the unique start place of the job J, e_J is the unique end place of the job J and $RFT(to_i)$ is a correction term that corresponds to the *residual firing time* of the first transition to_i of the path of minimal cost between p_i and e_J. In simple words, $h_J(M, M_{ref})$ corresponds to the sum of the durations of operations that are not yet performed corrected according to the operation that is currently in progress: the first part of Eq. (4) refers to the product of the time needed to perform the job $D(J)$ by the number of tokens in the start place $M_r(s_J)$ (that should wait the end of the current execution of the job), and the second part refers to the maximum of the shortest paths from the marked places p_i to the reference place e_J. One can notice that the reduced model of a set of operations with total precedence constraints is a single path with minimal duration between the unique start place s_J and the unique end place e_J. As long as the capacity of the job is 1, only the start place s_J could be marked with more than one token and there exists at most one place $p_i \in J \setminus \{s_J\}$ such that $M(p_i) = 1$. Consequently, Eq. (4) can be reformulated with (5):

$$\begin{aligned} h_J(M, M_{ref}) = {} & M(s_J) \times D(J) + \chi^*(p_i, e_J) \\ & - RFT(to_i) \quad \text{with } to_i \in (p_i)\bullet \end{aligned} \tag{5}$$

The estimation function $h_J(M, M_{ref})$ of the residual duration for a job J composed of a set of operations with full routing flexibility is also obtained as the corrected sum

of the durations of operations that are not yet performed. Such a duration is no longer computed according to the paths of the PN structure but depends on the sum of the operation durations.

In a set of operations with full flexibility, the status of each operation o_i is defined by an operation place p_i and a flag f_i. Three cases may occur: (1) the flag place f_i is marked and the operation place p_i is unmarked, and then the operation is not yet performed; (2) the operation place p_i is marked and the flag place f_i is unmarked, and then the operation is currently performing; (3) the flag place f_i and the operation place p_i are both unmarked, and then the operation o_i is already performed.

Note that the flag place and the operation place of a given operation could not be marked together. Thus, the approximation $h_J(M, M_{ref})$ of the duration of the unknown part of the trajectory, which is given by the sum of the operations duration that are not yet performed, is equal to the product of the flag place marking $M(f_i)$ by operation duration $D(to_i)$ plus the product of the operation place marking $M(p_i)$ by the operation duration $D(to_i)$. Observe that a correction term is added for the operation that is currently performing. The residual time to M_{ref} is estimated by (6):

$$h_J(M, M_{ref}) = M(s_J) \times D(J) + \sum_{f_i \in PF} M(f_i) \times D(to_i) + \left(M(p_i) \times D(to_i) - \mathrm{RFT}(to_i)\right) \quad \text{with } to_i \in (p_i)\bullet \tag{6}$$

where $PF = \{f_i, o_i \in O_J\}$ is the set of flag places.

To calculate the residual duration for a hybrid job, we use the iterated organization presented in Sect. 4 with a set of basic jobs and we reformulate the whole model as a layered PN with L different levels. The idea, we propose here, is to introduce *dynamic transitions* (that will be represented with small rectangles in the next figures) and *dynamic places* (that will be represented with double circles in the next figures). Dynamic transitions have variable firing times in order to propagate the residual times from one layer to the next one. Dynamic places report the whole marking of a given basic job from one layer to another one. The combined use of dynamic transitions and places embeds the models at levels 1,…, $l-1$ into the dynamic transitions and dynamic places at level l: the marking and residual duration of each basic job at level 1,…, $l-1$ are propagated at level l to change, respectively, the marking of the dynamic places and the firing time of the dynamic transitions. The other places and transitions of the model will be referred to as *static* places and transitions.

Let J be a given job and $J_l = \{J_{1l},..,J_{kl}\}$ be the set of basic jobs at level l and $\{1,..,L\}$ be the set of levels. For any given marking M, the evaluation of $h_J(M,M_{ref})$ is calculated as below:

At level 1, all places and transitions are static ones. The marking of the basic jobs is a simple copy of the PN marking and the firing duration $D(to)$ of any timed transition to is equal to the duration of operation o.

At level $l > 1$, the marking of the static places is a copy of the PN marking the firing duration $D(to)$ of a static transition to is equal to the duration of operation o. On the contrary, the marking of the dynamic places reports the number of tokens in the basic job $J_{k'l'}$ at level $l' < l$, and the firing duration $D(to)$ of a dynamic transition to is equal to the residual duration $h_{J,k'l'}(M,M_{ref})$ of the basic job $J_{k'l'}$ at level $l' < l$. The job $J_{k'l'}$ is defined by the FMS organization.

The estimation $h_J(M, M_{ref})$ of the residual duration for the whole hybrid job J is obtained from the value h_{JL}, obtained for the unique basic job at level L.

Example: Consider the hybrid job J_2 in Fig. 7 with 3 levels and 4 basic jobs $\{J_{11}, J_{21}, J_{12}, J_{13}\}$. The residual durations for the basic jobs J_{11} and J_{21} of the first level are estimated using (5). Then the time parameters d_{11} and d_{21} of the dynamic transitions $\{to_{11}, to_{21}, ti_{1121}, ti_{2111}\}$ of the basic job J_{12} at level 2 are updated. The marking of the dynamic places pi_{11} and pi_{21} is also updated. Then, the residual duration for J_{12} is

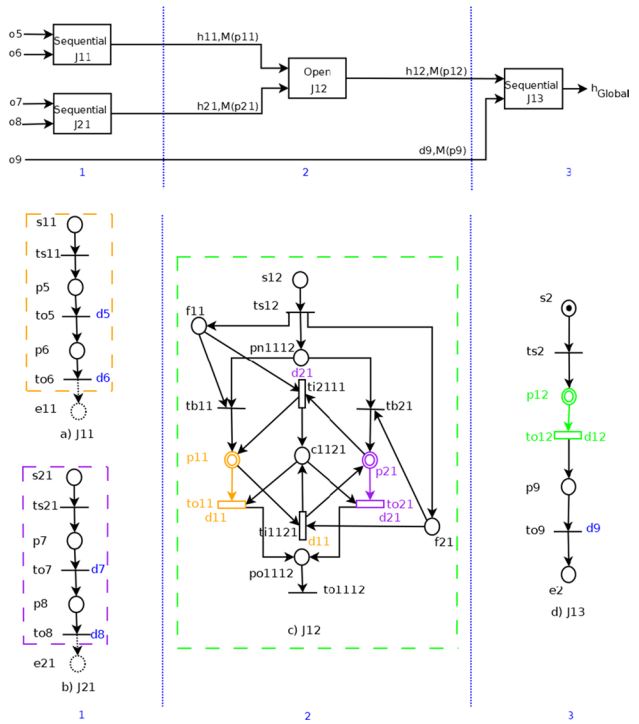


Fig. 7 Propagation of the marking and residual duration for a hybrid job

calculated using (6). The time parameter d_{12} of the dynamic transition to_{12} and the marking of the dynamic place pi_{21} in the basic job J_{13} at level 3 are updated. Thereafter, the estimation of the residual duration for J_{13} is evaluated using (5) and the residual duration of the hybrid job is obtained as $h_J(M, M_{ref}) = h_{13}(M, M_{ref})$.

The residual duration for the FMS is obtained as the maximum value of the residual durations over all hybrid jobs of the FMS:

$$h\left(M, M_{ref}\right) = \max\left\{h_J\left(M, M_{ref}\right) \text{with } J \in \boldsymbol{J}\right\} \quad (7)$$

In order to guarantee that the algorithm can find an optimal solution, $h(M, M_{ref})$ must be admissible (Nilsson 1980) for any reachable marking M, i.e. $h(M, M_{ref}) \leq h^*(M, M_{ref})$, where $h^*(M, M_{ref})$ is the actual minimum time from M to M_{ref} under earliest firing policy.

Proposition: Let us consider a hybrid FMS. The estimation $h(M, M_{ref})$ computed with (7) is a lower bound of the actual duration of the residual trajectory (σ_2, M) from M to M_{ref} (i.e. is admissible).

Proof: According to the iterated organization, previously introduced, each job J of the FMS is described with one or several basic jobs J_{kl} organized in L levels and a set of resources that are shared by the operations. J_{kl} is either a simple operation, or a sequence of jobs or a set of jobs with full routing flexibility.

At level 1, for a sequence of operations J_{k1} with a capacity of 1 and without any resource, the estimation of the residual duration is given by (5). This estimation is the actual duration. When resources are considered, this duration increases if some resources become unavailable. Thus, the true duration is at least equal to (5). The reasoning is similar for a set of operations J_{kl} with full routing flexibility. The actual duration is at least equal to (6) because an additional waiting time should be considered when the resources become unavailable.

At level $l > 1$, the reasoning is quite the same, but sequences of jobs should be considered instead of sequences of operations (resp. sets of jobs with full routing flexibility instead of sets of operations). From the marking M and the computation of the estimation $h_{J,k'l'}(M, M_{ref})$ of the residual duration for the basic jobs $J_{k'l'}$ with $l' < l$, the markings of the dynamic places and the firing durations of the dynamic transitions of basic job J_{kl} are first updated and then the estimation $h_{J,kl}(M, M_{ref})$ of the residual duration for the basic job J_{kl} is computed. This estimation is equal to the actual duration when resources are not considered. Adding shared resources increases the actual duration but does not change the value of $h_{J,kl}(M, M_{ref})$. Propagating the estimations $h_{J,kl}(M, M_{ref})$ up to level L leads to the estimation $h_{JL}(M, M_{ref})$ for job J that is a lower bound of the actual duration required to execute the job J. Finally, (7) is a lower bound of the actual duration required to execute all jobs in the FMS.

5.2 Generation Beam Search Algorithm

In the next, we call generation the new population composed by all direct successors (candidates generated from all the parents of one generation) from a given population of parents. With usual FBS, the comparison is made between the successors of one parent and all other parents. Since the selection of candidates is based on the objective function composed of the cost of the already computed trajectory and the estimation of the residual one which is a lower bound of the real cost, parents have more chance to be selected.

In this section a modified Beam Search algorithm, based on the notion of generation, is proposed in order to give equal chance to all candidates issued from the same generation to be selected. For each parent node, the algorithm starts by exploring their successors, then sorting them according to the total cost f to place the best β_l candidates in a temporary list. Once the expansion of all parent nodes is done, the algorithm selects the best β_g candidates from the temporary list to create a new generation. This routine is iterated according to the successive generations. The main innovation of the proposed algorithm is that the global sorting is made after expansion of all parent nodes which is not the case for the usual HFBS where the global sorting is made after the expansion of only the best parent. We refer to this variant of the FBS algorithm as Generation Filtered Beam Search (GFBS) (Cherif et al. 2019).

The objective is to explore selectively the PN state space according to the successive generations of candidates in order to reach the reference marking M_{ref} starting from an initial marking M_0 with a trajectory of minimal duration. The idea is to provide diversification at early steps and intensification at late steps while maintaining equal chances when selecting the candidates of a given generation. We present below the pseudo-code of the GFBS algorithm. The algorithm aims to return a valid sequence of transition firings *Seq* and the resulting cost $C_{\max}$. The algorithm uses a list, called *OPEN*, to store to-be-expanded candidates of a given generation. The cost function f detailed in Sect. 5.1 is used to sort candidates of one generation for selection purpose. The list OPEN is updated after each iteration (an iteration is the expansion of all candidates of a given generation).

Algorithm: GFBS optimization

Inputs: N, M_0, β_g, β_l, M_{ref} Outputs: Seq, C_{max}

1. Place M_0 into OPEN.
2. For each candidate M remaining into $OPEN$ do
   a. Generate the successors of M.
   b. For each successor M' of M.
      i. If the successor M' is equal to M_{ref}, construct the sequence of transition firings Seq' from M_0 to M_{ref} and return Seq and C_{max} as the duration of Seq.
      ii. Otherwise, calculate the cost function $f(M_0, M', M_{ref})$ and place M' into a temporary list TEMPLIST.
   c. Select up to the β_l best children markings from TEMPLIST and place them into a temporary list GENERATION respecting:
      i. If M' is equal to some markings in GENERATION, verify if $g'(M') < g(M)$. In that case, a new better path was found, then replace M by M'.
      ii. Otherwise, place M' into GENERATION.
   d. Clear TEMPLIST.
3. Clear OPEN.
4. Select up to the β_g best markings from GENERATION and place them into OPEN.
5. Clear GENERATION.
6. If OPEN is empty, exit return $Seq = \varnothing$ and $C_{max} = \infty$, otherwise, go to step 2.

This algorithm avoids implicitly deadlocks and dead branches that are a priori unknown for the controller: an infinite cost is associated with the candidates which could not achieve the objective marking M_{ref} (deadlocks and dead branches). Consequently, these candidates are not selected for the next generation because their cost is higher comparing to other candidates.

6 Computational Experiments and Results

This section is devoted to illustrate the use of GFBS algorithm. The algorithm runs on a 1.8 GHz computer with 16G RAM memory. The first example is the hybrid FMS presented in Fig. 6 and aims to illustrate the proposed method based on the iterated organization. Then, two sets of instances are taken from the literature in order to compare the GFBS to other methods.

6.1 Details of the GFBS Algorithm on a Hybrid FMS

In this section, the GFBS algorithm is performed on the FMS modelled in Fig. 6. The processing times and resources required by the operations are given in Table 1.

Table 1 Operation processing times in TU and resources for the FMS in Fig. 6

Operation	R	Processing time	Operation	R	Processing time
o_1	r_1, r_7	5	o_5	r_5	2
o_2	r_2	4	o_6	r_6	4
o_3	r_3	2	o_7	r_2, r_7	5
o_4	r_4, r_9	5	o_8	r_3, r_8	5
			o_9	r_9	2

The capacity of the jobs and the resources equal to 1: $M_0(r_k) = M_0(cp_J) = 1$, $J = 1,2$, $k = 1,..,9$. The initial marking is such that $M_0(s_J) = 1$. The reference marking is such that $M_{ref}(s_J) = 0$ and $M_{ref}(cp_J) = M_{ref}(r_k) = 1$. With parameters $\beta_g = 3$ and $\beta_l = 2$ the algorithm returns a makespan: $C_{\max} = 21$.

The details of the exploration are given in Fig. 8: the selected candidates at each iteration (the new generation) are shown as the already computed sequence (seq), the duration of the already computed sequence (g) and the estimation to the reference (h). The candidates in each population are generated with firing sequences that have an equal number of transitions, and the selection is made on the best value of $g + h$ cost function. All candidates of a given generation have different markings (if two candidates have the same marking only the one with the minimal cost is saved for the future generations). The idea is to improve the diversification by giving other candidates, with different markings, the chance to be selected. In order to reach the reference and obtain the optimal result, 16 successive iterations were computed as shown in Fig. 8. The obtained makespan is optimal, and this has been validated by applying a global method (Lefebvre and Daoui 2018; Lefebvre 2018). This example illustrates that the proposed method is suitable for scheduling problems with hybrid FMS. Observe that the successive generations are of maximal size 3 and that no more than 2 candidates are generated from each parent.

6.2 Qualitative Comparison with Other Methods

The second example is taken from (Muth and Thompson 1963; Crowston et al. 1963). The scheduling problem is composed of six jobs, and each has to be processed on six resources with total precedence constraints. This well-known benchmark referred to as Ft06 was also tested in numerous papers based on the branch and bound methods (Carlier and Pinson 1989; Perregaard and Clausen 1998; AitZai and Boudhar 2013; Dabah et al. 2016), the genetic algorithms (Asadzadeh and Zamanifar 2010; Koblasa and Kloud 2011; Goncalves and Resende 2014), the taboo search (Watson et al. 2003; Kuo-Ling and Ching-Jong 2008; Peng et al. 2015), the Simulated Annealing Algorithm with Cooperative

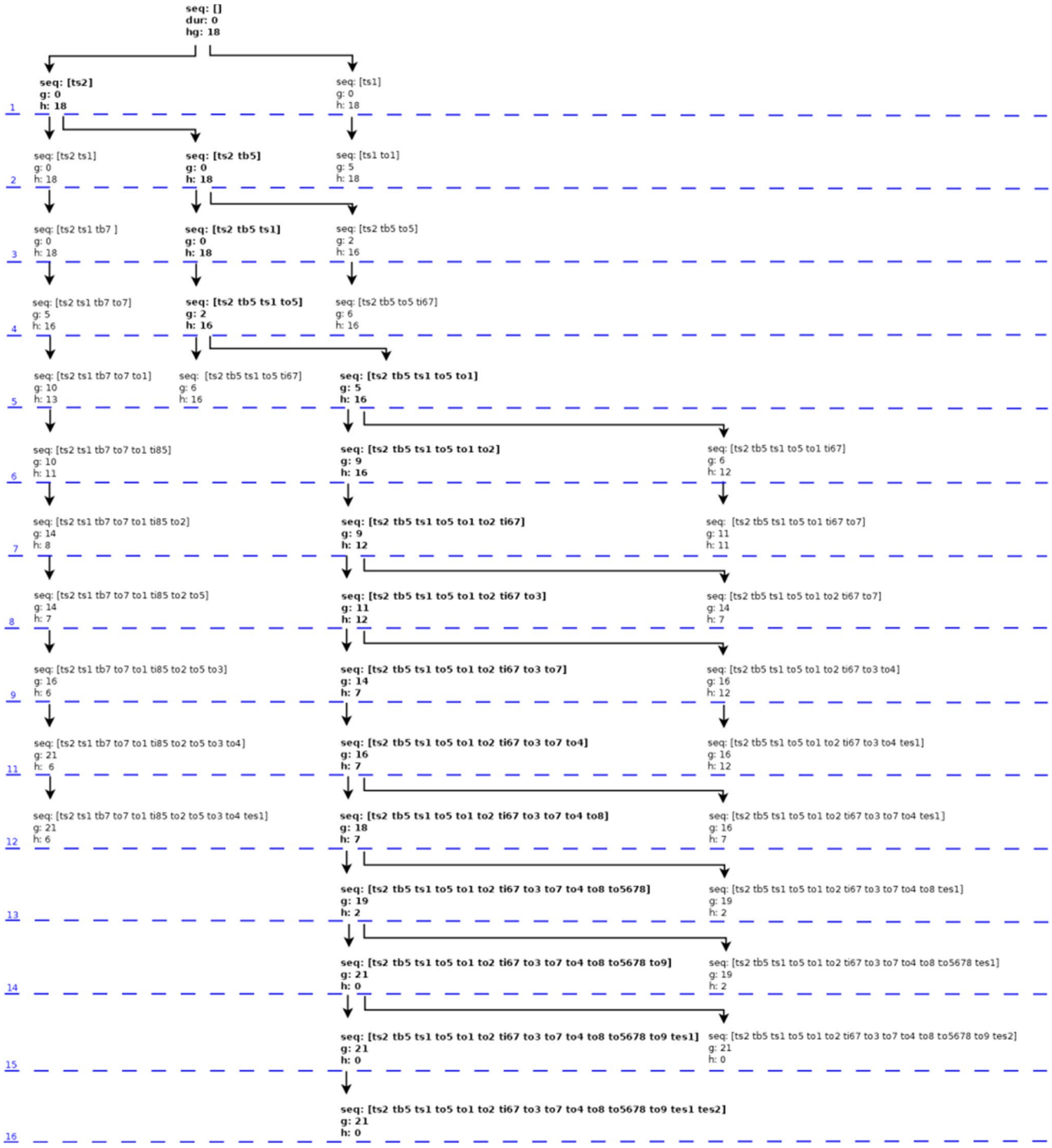


Fig. 8 Details of the exploration for the hybrid FMS of Fig. 6

Threads (SACT) (Cruz-Chávez et al. 2019) and the Particle Swarm Optimization (PSO) (Huang et al. 2019). The details of the problems are given in Table 2. The range of processing times on each resource, shown in Table 3, was from one to ten days. The objective is to perform one part for each job of the FMS.

The optimal solution as given in the literature is $C_{\max} = 55$ days. Using parameters $B_g = 20$ and $B_l = 2$, our approach gives results as good as the ones taken from the literature (i.e., $C_{\max} = 55$ days). In addition, the GFBS does not need more than 1 s, which matches also the literature results. This example also illustrates the large variety of

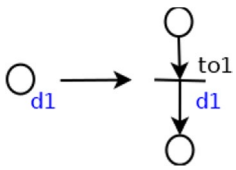
Table 2 Operation processing times in TU and resources for Ft06 benchmark

	J_1		J_2		J_3		J_4		J_5		J_6	
O_J	R_i	d_i	R_i	d_i	R_i	d_i	R_i	d_i	R_i	d_i	R_i	d_i
o_{J1}	3	1	2	8	3	5	2	5	3	9	2	3
o_{J2}	1	3	3	5	4	4	1	5	2	3	4	3
o_{J3}	2	6	5	10	6	8	3	5	5	5	6	9
o_{J4}	4	7	6	10	1	9	4	3	6	4	1	10
o_{J5}	6	3	1	10	2	1	5	8	1	3	5	4
o_{J6}	5	6	4	4	5	7	6	9	4	1	3	1

Table 3 Comparison of methods and performances for Ft06 benchmark

Methods	References	C_{max}	CPU (s)	Characteristics
GFBS		55	≤1	Generations; Populations; T-TPN; Heuristic
Genetic algorithm	Asadzadeh and Zamanifar, (2010), Koblasa and Kloud (2011), Goncalves and Resende (2014)	55	≤1	Generations; Populations; Metaheuristic; Evolutionary algorithm
Branch and bound methods	Carlier and Pinson (1989), Perregaard and Clausen (1998), AitZai and Boudhar (2013), Dabah et al. (2016)	55	≤1	Combinatorial optimization problems; mathematical optimization
Taboo Search method	Watson et al. (2003), Kuo-Ling and Ching-Jong (2008), Peng et al. (2015)	55	≤1	Disjunctive graph of nodes; metaheuristic; evolutionary algorithm
PSO	Huang et al. (2019)	55	≤1	Populations; disjunctive graph of nodes; metaheuristic; evolutionary algorithm
SACT	Cruz-Chávez et al. (2019)	55	≤1	Disjunctive graph of nodes; metaheuristic; evolutionary algorithm

Fig. 9 From P-TPN to T-TPN



existing approaches studied for such kind of scheduling problems.

6.3 Performance Comparison with Other Methods

The third example presented in Han et al. (2015) and used in Mejia and Nino (2017) consists of 20 instances. The FMS is composed of 2 jobs and 4 resources. The example is originally modelled as a P-TPN, and we generate its equivalent T-TPN by replacing each timed place by two places and one timed transition (Sifakis 1979) as shown in Fig. 9.

The P-TPN and its equivalent T-TPN are presented in Fig. 10. Note that the size of the generated T-TPN with 22 places and 17 transitions is bigger than its equivalent P-TPN with 15 places and 10 transitions. Consequently, the size of the reachability graph of the T-TPN model is also bigger than the size of the reachability graph of the P-TPN model. This penalizes our method compared to other methods that are based on P-TPN models.

The processing times of operations are $d_{11}=25$, $d_{12}=23$, $d_{13}=27$, $d_{14}=25$, $d_{21}=25$, $d_{22}=25$ and $d_{23}=25$. Twenty instances with different resource capacities and job sizes are considered.

The instances are denoted by FMS01-FMS20 as shown in Table 4. The GFBS algorithm is performed on these instances with different values of the parameters β_g in the range [20:100] and different values of the parameters β_l in the range [2:10]. The obtained results are given in Table 4.

The results are compared with the Hybrid Particle Swarm Optimization algorithm (HPSO) used in (Han et al. 2015) and the Iterated Hybrid Filtered Beam Search (IHFBS) used in Mejia and Nino (2017). HPSO algorithms used the deadlock controller proposed by (Piroddi et al. 2008). The IHFBS is based on the repeated call of the search algorithm where the value of the objective function of the current call is used as upper bound for the next call.

The comparison with HPSO algorithm can be made from the perspective of the memory and the computational time: in term of memory, the search space of T-TPN model used with GFBS algorithm is much bigger than its equivalents P-TPN used with HPSO.

Contrary to GFBS, the HPSO algorithm uses deadlock controllers which further reduce the search space when it avoids deadlocks and dead branches. In addition, for HPSO the search stops only if a predefined maximum running time

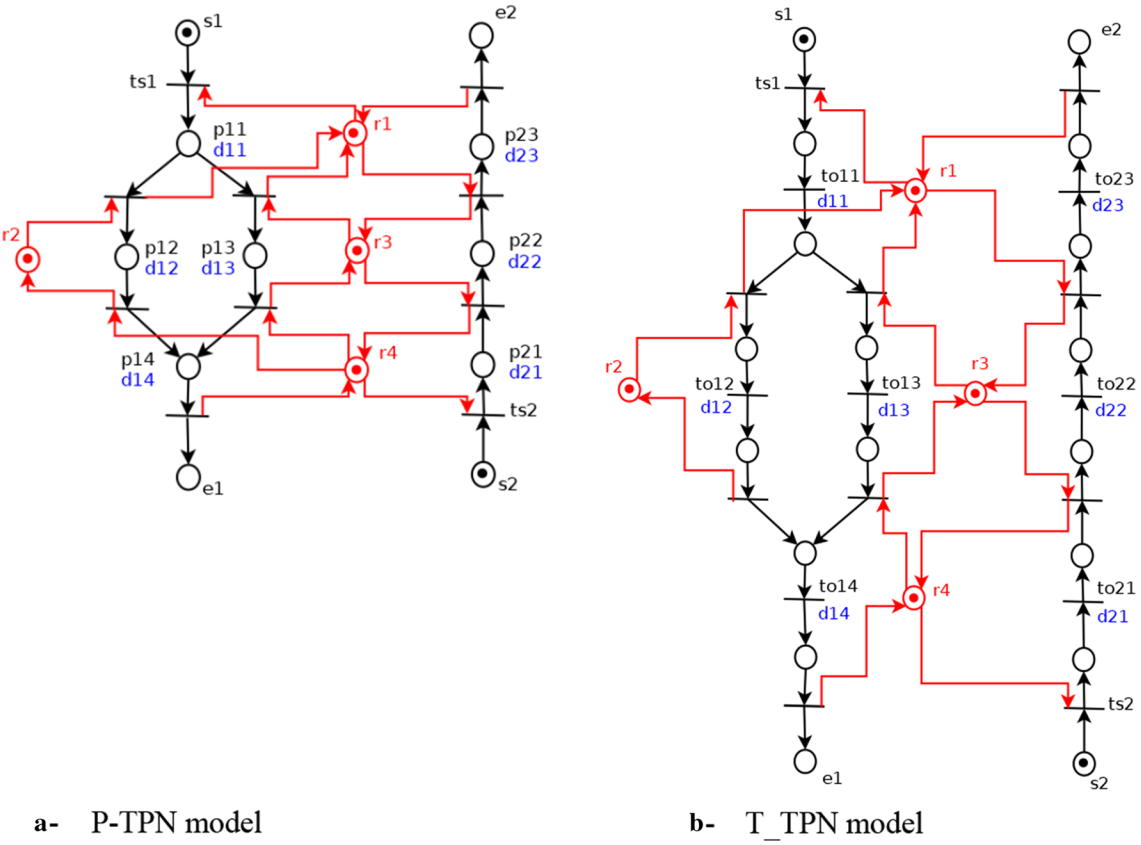


Fig. 10 Model of second set of instances

Table 4 20 instances of the first set

Instance	$M(s_j)$	$M(r_i)$	HPSO	CPU(s)	IHFBS	CPU(s)	GFBS	CPU(s)
FMS1	5	**1**	**293**	500	**293**	40	**293**	**1**
FMS2	10	**1**	**557**	1000	**557**	80	**557**	**3**
FMS3	20	**1**	**1087**	2000	**1087**	160	**1087**	**10**
FMS4	30	**1**	**1617**	3000	**1617**	240	**1617**	**19**
FMS5	50	**1**	**2677**	5000	**2677**	800	**2677**	**31**
FMS6	5	2	**150**	500	**150**	80	**150**	**6**
FMS7	10	2	**273**	1000	**273**	160	**273**	**18**
FMS8	20	2	547	2000	**531**	320	**531**	**26**
FMS9	30	2	833	3000	**795**	480	**795**	**39**
FMS10	50	2	1461	5000	**1325**	800	**1325**	**53**
FMS11	5	3	**106**	500	**106**	120	**106**	**17**
FMS12	10	3	**185**	1000	**185**	240	**185**	**32**
FMS13	20	3	371	2000	**368**	480	**368**	**40**
FMS14	30	3	591	3000	**532**	720	534	**53**
FMS15	50	3	1092	5000	**897**	1200	921	**69**
FMS16	5	**4**	**99**	500	**99**	160	**99**	**22**
FMS17	10	**4**	150	1000	**149**	320	**149**	**39**
FMS18	20	**4**	287	2000	**274**	640	**274**	**51**
FMS19	30	**4**	445	3000	**401**	960	421	**62**
FMS20	50	**4**	877	5000	**664**	1600	687	**80**

is reached. Indeed, even using a processor with higher performances (2.6 GHz computer with 4G RAM memory), the running time of HPSO algorithm is ranged from 500 s on FMS1 to 5000 s on FMS20. For GFBS algorithm, the running time is ranged from 1 s on FMS1 to 80 s on FMS20.

Although its complexity in time and memory, GFBS algorithm found better solutions in 10 out of 20 instances and matches the solutions of the 10 remaining instances.

The comparison with IHFBS algorithm leads to the following comments: in terms of memory, the IHFBS algorithm uses, as well as HPSO, the P-TPN model which is less complicated than its equivalent T-TPN used with GFBS. In addition, contrary to GFBS algorithm which is performed only once, IHFBS is based on repeated call of the search algorithm where the value of the objective function of the current call is used as an upper bound for the next call. Thus, the number of explored candidates is much bigger which influences the computational time needed to perform the algorithm. GFBS algorithm does not need more than 1 s to obtain the result for the smallest instance FMS1 while the time needed for the largest instance FMS20 was 80 s. For IHFBS, even using a processor with higher performances (3.2 GHz computer with 8G RAM memory), the running times ranged from 40 s on FMS1 to 1600 s on FMS20. Although its complexity in time and memory, GFBS algorithm found the same results for 16 of 20 instances. In the remaining 4 instances, the results obtained with IHFBS are slightly better.

Note that we have reported the CPU times as mentioned in the literature but that the processors used were different from our processor: GFBS runs on 1.8 GHz computer, HPSO runs on a 2.6 GHz computer (Han et al. 2015) and IHFBS runs on a 3.2 GHz computer (Mejia and Nino 2017).

Consequently, definitive conclusions about the comparison of the time complexity are difficult to draw. Nevertheless, the values reported in Table 4 illustrate that the time complexity of the approaches developed in (Han et al. 2015) and (Mejia and Nino 2017) exceeds widely the time required by GFBS even if the frequency of the used processor was larger.

7 Conclusion

In this paper, we have studied the modelling of hybrid FMS with partial routing flexibility and solve scheduling problems for such FMS. First, we have developed a T-TPN modelling methodology for FMS where some operations are proceeded with total precedence constraints and others with full routing flexibility. Second, a new scheduling method, which incrementally computes control sequences for hybrid FMS, was presented. The method uses T-TPN, an iterated organization of the operations for hybrid jobs and a new variant of beam search methods that selectively explores the PN state space according to successive generations of candidates. The cost function used by the GFBS method was proved to provide a lower bound of the trajectory duration. A set of instances has been detailed in order to illustrate that the GFBS algorithm is suitable for a large variety of FMS organizations, including job shops, open shops and hybrid FMS.

In our future works, we will introduce, uncertain environments by considering unreliable resources and failures of operations for hybrid FMS. The cost function will be updated in order to take into account the risk of deviation for the computed trajectory. Adding to that, models with more complicated time constraints (for example, maximal time constraints with time Petri nets (Merlin 1974)) will be considered.

References

AitZai, A., & Boudhar, M. (2013). Parallel branch-and-bound and parallel PSO algorithms for job shop scheduling problem with blocking. *International Journal of Operational Research., 16*(1), 14–37.

Asadzadeh, L., & Zamanifar, K. (2010). An agent-based parallel approach for the job shop scheduling problem with genetic algorithms. *Mathematics Computational Model, 52,* 1957–1965.

Barkaoui, K., Ben Abdallah, I. (1996). Analysis of a resource allocation problem in FMS using structure theory of Petri nets. *Proceedings, First International Workshop on Manufacturing and Petri Nets*, pp. 1–15, Osaka, Japan.

Baruwa, O. T., Piera, M. A., & Guasch, A. (2015). Deadlock-free scheduling method for flexible manufacturing systems based on timed colored Petri nets and anytime heuristic search. *IEEE Transaction on System, Man, Cybernetics System, 45*(5), 831–846.

Carlier, J., & Pinson, E. (1989). An algorithm for solving job shop problem. *Management Science, 35*(2), 164–176.

Cassandras, C. (1993). *Discrete event systems: modeling and performances analysis*. Homewood: Aksen Assoc. Inc., Pub.

Cherif, G., Leclercq, E., Lefebvre, D. (2018). Modeling hybrid manufacturing systems using T-TPN with buffers. In: *IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, Vol. 1, pp. 480–487, Turin, Italy

Cherif, G., Leclercq, E., & Lefebvre, D. (2019). Generation filtered beam search algorithm for the scheduling of hybrid FMS using T-TPN. *IEEE 18th European Control Conference (ECC)* (pp. 3225–3230). Italy: Naples.

Cruz-Chávez, M. A., Peralta-Abarca, J. D. C., & Cruz-Rosales, M. H. (2019). Cooperative threads with effective-address in simulated annealing algorithm to job shop scheduling problems. *Applied Sciences, 9,* 3360.

Dabah, A., Bendjoudi, A., El-Baz, D., AitZai, A. (2016). GPU-based two level parallel B&B for the blocking job shop scheduling problem. In: *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, Chicago, IL, USA, pp. 747–755

Ezpeleta, J., Colom, J. M., & Martinez, J. (1995). A Petri Net based deadlock prevention policy for Flexible Manufacturing Systems. *IEEE Transactions on Robotics and Automation, 11,* 173–184.

Goncalves, J. F., & Resende, M. G. C. (2014). An extended akers graphical method with a biased random-key genetic algorithm

for job-shop scheduling. *International Transaction on Operation Research, 21,* 215–246.

Huang, B., Jiang, R., & Zhang, G. (2014). Search strategy for scheduling flexible manufacturing systems simultaneously using admissible heuristic functions and non-admissible heuristic functions. *Computers and Industrial Engineering, 71,* 21–26.

Huang, B., Sun, Y., & Sun, Y. M. (2008). Scheduling of flexible manufacturing systems based on Petri nets and hybrid heuristic search. *International Journal of Production Research, 46,* 4553–4565.

Huang, M., Liu, Q., & Liang, X. (2019). The application of improved hybrid particle swarm optimization algorithm in job shop scheduling problem. In: *Proceedings of the 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*. pp. 49–52, Dalian, China.

Koblasa, F., & Kloud, T. (2011). Solving job shop scheduling with the computer simulation. *The International Journal of Transport and Logistics, 35,* 775–785.

Kuo-Ling, H., & Ching-Jong, L. (2008). Ant colony optimization combined with taboo search for the job shop scheduling problem. *Computer Operative Research, 35,* 1030–1046.

Lee, D. Y., & DiCesare, F. (1994). Scheduling flexible manufacturing systems using Petri nets and heuristic search. *IEEE Transaction on Robotic Atom, 10*(2), 123–132.

Lee, J., & Lee, J. S. (2010). Heuristic search for scheduling flexible manufacturing systems using lower bound reachability matrix. *Computer Industry Engineering, 59*(4), 799–806.

Lefebvre, D. (2016). Deadlock-free scheduling for timed Petri net models combined with MPC and backtracking. In: *Proceedings of the 13th International Workshop on Discrete Event Systems (WODES),* pp. 466–471, Xian, China.

Lefebvre, D. (2017b). Dynamical scheduling and robust control in uncertain environments with petri nets for DESs. *MDPI Processes, 5*(4), 54.

Lefebvre, D. (2017a). Evaluating the robustness of scheduling in uncertain environment with Petri nets. *Valuetools Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*, pp 170–177, Venice, Italy.

Lefebvre,D. (2018). Approximated timed reachability graphs for performance evaluation and control of DES. *IFAC WODES 18*, 29(1), 31–56.

Lefebvre, D., & Daoui, C. (2018). Control design for bounded partially controlled TPNs using timed extended reachability graphs and MDP. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 50*(6), 2273–2283.

Lefebvre, D., & Mejía, G. (2018). Robust scheduling in uncertain environment with Petri nets and beam search. *IFAC-PapersOnLine, 51*(11), 1077–1082.

Liu, G., & Barkaoui, K. (2016). A survey of siphons in Petri nets. *Information Sciences, 363,* 198–220.

Liu, H. X., Xing, K. Y., Zhou, M. C., Han, L. B., & Wang, F. (2014). Transition cover-based design of Petri net controllers for automated manufacturing systems. *IEEE Transaction on System, Man, Cybernetics, System, 44*(2), 196–208.

Luo, J. C., Xing, K., Zhou, M. C., Li, X. L., & Wang, X. N. (2015). Deadlock-free scheduling of automated manufacturing systems using petri nets and hybrid heuristic search. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 45*(3), 530–541.

Mejía, G., Caballero-Villalobos, J., & Montoya, C. (2017). Petri nets and deadlock-free scheduling of open shop manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 48*(6), 1017–1028.

Mejía, G., & Montoya, C. (2009). Scheduling manufacturing systems with blocking: a Petri net approach. *International Journal of Production Research, 47*(22), 6261–6277.

Mejía, G., & Niño, K. (2017). A new hybrid filtered beam search algorithm for deadlock-free scheduling of flexible manufacturing systems using Petri Nets. *Computers and Industrial Engineering, 108,* 165–176.

Mejía, G., & Odrey, N. G. (2005). An approach using Petri nets and improved heuristic search for manufacturing system scheduling. *Journal of Manufacturing Systems, 24*(2), 462–476.

Merlin, P. M. (1974). A study of the recoverability of computing systems, *PhD thesis*, Department of Information and Computer Science, University of California, Irvine, CA.

Moro, A.R., Yu, H., & Kelleher, G. (2000). Advanced scheduling methodologies for flexible manufacturing systems using Petri nets and heuristic search. *IEEE International Robotics* Autom, San Francisco, CA, USA, pp. 2398–2403

Nilsson, N. (1980). *Principles of artificial intelligence*. Palo Alto, CA: Tioga.

Ow, P. S., & Morton, T. E. (1988). Filtered beam search in scheduling. *International Journal of Production Research, 26*(1), 35–62.

Pearl, J. (1984). *Heuristics: intelligent search strategies for computer problem solving*. Boston: Addison-Wesley.

Peng, B., Lu, Z., & Cheng, T. (2015). A tabu search/path relinking algorithm to solve the job shop scheduling problem. *Computational Operation Research, 53,* 154–164.

Perregaard, M., & Clausen, J. (1998). Parallel branch-and-bound methods for the job-shop scheduling problem. *Annals of Operative Research, 83,* 137–160.

Piroddi, L., Cordone, R., & Fumagalli, I. (2008). Selective siphon control for deadlock prevention in Petri nets. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 38*(6), 1337–1348.

Ramchandani, C. (1974). Analysis of asynchronous concurrent systems by timed Petri nets, *PhD thesis,* Massachusetts Institute of Technology, Cambridge, MA, Project MAC Report MAC-TR-120, 1974.

Reyes, A., Yu, H., Kelleher, G., & Lloyd, S. (2002). Integrating Petri nets and hybrid heuristic search for the scheduling of FMS. *Computational Industries, 47*(1), 123–138.

Sabuncuoglu, I., & Bayiz, M. (1999). Job shop scheduling with beam search. *European Journal of Operational Research, 118*(2), 390–412.

Sifakis, J. (1979). Performance evaluation of systems using nets. *Net Theory and Applications*, *Lecture Notes in Computer Science*, Vol. 84, pp. 307–319, Springer, Berlin, Heidelberg.

Watson, J.-P., Beck, J., Howe, A. E., & Whitley, L. (2003). Problem difficulty for tabu search in job-shop scheduling. *Artificial Intelligence, 143,* 189–217.

Xing, K. Y, Xing, K. L, Li, J. M. (1996). Deadlock avoidance controller for a class of manufacturing systems. *Proceedings, IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota, pp. 200–204.

Xiong, H. H., & Zhou, M. (1998). Scheduling of semiconductor test facility via Petri nets and hybrid heuristic search. *IEEE Transaction Semiconducting Manufacturing, 11*(3), 384–393.

Zhang, W., Freiheit, T., & Yang, H. (2014). Dynamic scheduling in flexible assembly system based on timed Petri nets model. *Robotics Computer Integrated Manufacturing, 21,* 550–558.