Timed Colored Petri Nets Based Modelling and Scheduling of Aero-engine maintenance

Xinmin Tang， Shisheng Zhong

School of Mechatronics Engineering, Harbin Institute of Technology, Harbin, Heilongjiang 150001, China
txmofhit@hit.edu.cn; zss@hit.edu.cn

Abstract: Aero-engines maintenance procedure can be treated as a typical discrete event dynamic system. A theory that synthesized timed colored Petri net is proposed. The theory adopts a bottom-up approach by merging all sub modules, and each one characterizes the critical resource contention among processes. Nevertheless Petri net seems inapplicable to dynamic scheduling modelling because of its structural rigidity. Shared composition method of Petri net is introduced, which makes its structure adjustable dynamically. Given a set of preemptive dispatching rules, the makespan can be analyzed using linear state equation in the sense of max-plus algebra. To minimize makespan of each aero-engine, genetic algorithm is adopted to arrange the firing sequence of transitions. Dynamic scheduling strategy uses rolling horizon composed by colors is presented. According to the simulation result, the module is verified to be effective and robust. [Nature and Science. 2006;4(2):46-51].

Keywords: timed colored Petri nets; shared composition; static scheduling; dynamic scheduling

1 Introduction

Aero-engine maintenance procedure can be treated as a typical discrete dynamic system (DES). It involves planning, scheduling and controlling of resources, aerial materials, activities, etc. Woxvold proposed a solution to planning and scheduling of aircraft maintenance by combining of MRP (Materials Requirements Planning), PAC (Production Activity Control) and CPM (Critical Path Method) into a unitary structuring technique, which was applied to maintenance of Boeing 737 commercial aircraft[1]. Goldratt used project management technology based on TOC (theory of constraints) to solve maintenance scheduling problem in Israeli air force, which shorten aircraft maintenance duration dramatically by some case studies[2].

In the maintenance work shop, maintenance tasks with different work level may be carried out concurrently, as a result machines resource conflicts are unavoidable. Additionally, maintenance procedures are always interrupted by stochastic events unexpected. Scheduling methods mentioned above do not provide strict mathematical model and are usually applied to static other than dynamic environment. Petri nets as a tool to characterize a concurrent, asynchronous, conflicting, stochastic system is widely used in discrete dynamic system modelling, simulating and scheduling[3]. Timed colored Petri nets is one of high level model which has advantages over basic Petri nets in making the model of a DES much more compact and concise. In this paper, this is gained by introducing "colors" to distinguish among tokens presenting different aero-engines.

The organization of this paper is described as follows. In Sections 2, sub modules based on timed colored Petri nets describe interactions between components of aero-engine and machine resources are built, and then shared composition method of Petri nets is adopted to realize bottom-up modelling . In Section 3, the makespan of each aero-engine is analyzed using max-plus algebra, then Genetic algorithm is adopted to optimize the firing sequence of transitions during static scheduling. In Section 4, to deal with discrete events emerge in workshop, a dynamic scheduling strategy uses rolling horizon is presented, which made the model more adaptive and robust. In Section 5, an example is demonstrated and the performance is analyzed. Finally, conclusions are provided.

2 Bottom-up Modelling of Aero-engine Maintenance Procedure

2.1 Construction of Sub Modules

Suppose that the directional chart G represents the maintenance workflow of an aero-engine component. Each node of the chart is multivocal, firstly it specifies the maintenance process $w_i \in W$, where W represents all processes of the component, secondly it specifies the machine $m_j \in M$ to process w_i, where M represents all machines. It is assumed that one process only uses one machine and the mapping between W and M is $f\!: W \mapsto M$. For $w_1(m_1), w_2(m_2) \in W$, if there exists a directional edge, then the component must be transferred to machine m_2 after accomplished processing in machine m_1.

Definition 1 (*Work Processing Module*). WPM is a state machine $WPM = (P_W, T_W, I_W, O_W, M_{W0})$ and satisfies the following conditions: (1) There exists but only one place $p_{idle} \in P_W$ is called idle place and $M_{W0}(p_{idle}) > 0$.

(2) $\forall p \in \left(P_W - \{p_{idle}\}\right)$ is called operation place and $M_{W0}(p) = 0$.

The rules to construct *WPM* from *G* are described as follows: (1) Build up one by one mapping from nodes W to transitions T_W, $g{:}W \mapsto T_W$. (2) For $w_u,w_v \in W$, if there exists a directional edge, then add place p_v to P_W, let $I(p_v,t_v) = O(p_v,t_u) = 1$ and $M_{W0}(p_v) = 0$. (3) If in-degree $\deg^+(w_i)$ of $w_i \in W$ is zero, then add idle place p_{idle} to P_W, let $I(p_{idle},t_i) = 1$ and $M_{W0}(p_{idle}) = 1$.

Definition 2 (*Resource Sharing Module*). *RSM* is extended from *WPM* and satisfies the following conditions: (1) $\forall p \in P_M$ is called resource place and $|M_{M0}(p)| \geq 1$ represents the quantity of the resource. (2) $\forall t \in T_W$, $\exists p \in P_M$, such that $p \in {}^{\bullet}t \cap t^{\bullet}$, where $I_M(p,t) = O_M(p,t) = 1$, which means resources are not exhaustive and should be released after the process is accomplished.

The rules to extend *RSM* from *WPM* are described as follows: (1) $\forall t \in T_W$, Build up one by one mapping from M to P_M, $h{:}\ M \mapsto P_M$. (2) $\forall p \in P_M$, $\forall t \in T_W$, if $f(g^{-1}(t)) = h^{-1}(p)$, then let $I_M(p,t) = O_M(p,t) = 1$, else let $I_M(p,t) = O_M(p,t) = 0$.

According to definition 2, the resource place looks like "resource dispatching and recycling center". Its initial marking specifies that the resource is available and processes related must compete for the resource[4].

2.2 Extension to Sub Modules

It is obvious that *RSM* reveals the interactions between processes of a single component instance. In order to characterize the resources contention among multiple instances of the component, the original Petri nets should be expanded for more meaningful. An effective resolution is to set up relationship between color set and nodes of Petri nets including places and transitions. Additionally, the deterministic time delay should be related with the transitions of *RSM*.

Definitions 3 (*Timed colored resource sharing module*). *TCRSM* extended by *RSM* is defined as follows: $TCRSM = (P_W \cup P_M, T, C, I, O, M_0, D)$ is timed colored Petri nets, where P_W are maintenance operation state places, P_M are machine resource places; $T = T_W$ are maintenance processes; C is color function defined from places and transitions into color set, $\forall p_i \in P_W$, $C(p_i) = \{a_{i,1},\cdots,a_{i,u_i}\}$, $u_i = |C(p_i)|$, $\forall p_j \in P_M$, $C(p_j) = \{m_j\}$, $m_j \in M$; $\forall t_i \in T_W$, $C(t_i) = C(p_i) \cup \{m_i\}$; $I: C(p) \times C(t) \mapsto N$ is input function that specifies input place of transition t; $O: C(t) \times C(p) \mapsto N$ is output function that specifies output place of transition t; initial marking M_0 is defined from $P_W \cup P_M$ into vector and $M(p)(c)$ represents the token number with a binding color c in the place p; $D: C(T) \mapsto R^+$ is time delay function specify the firing time duration of transitions.

Transition $t_i \in T$ with a binding color $a_{i,h}$ is said to be firing enable at marking a M if $\forall p \in {}^{\bullet}t_i$ $M(p)(a_{i,h}) \geq I(a_{i,h},a_{i,h})$ and $M(p)(m_i) \geq I(m_i,m_i)$, where $m_i \in M$. When enabled transition t_i begins to fire, the new marking is calculated from:

$$\forall p_i \in {}^{\bullet}t_i : M'(p_i)(c) = M(p_i)(c) - I(c,c), c = \{a_{i,h}, m_i\} \quad (1)$$

Once transition t_i begins firing, it will finish after duration $D(t_i)$. The new marking is calculated from:

$$\forall p_j \in t_i^{\bullet} : M'(p_j)(c) = M(p_j)(c) + O(c,c), c = \{a_{i,h}, m_i\} \quad (2)$$

The flowing of colored token from place to place represents the components of aero-engine transferring from one machine to another, which reveals' the busy and idle state of machine resources in the discrete time duration. *TCRSM* based on timed colored Petri nets has properties as follows:

Property 1 (*Conservation*). Note that the *WPM* is a state machine, then $\forall p \in P_W$, such that $|{}^{\bullet}p| \leq 1$ and $|p^{\bullet}| \leq 1$, so $\forall M \in R(M_0)$, such that $\Gamma_{P \to P_W}(M) = \Gamma_{P \to P_W}(M_0)$, where $\Gamma_{X \to Y}(Z)$ represents the projection of Z from X to subset Y. For *RSM*, $\forall p \in P_M$: $I(p,t) = O(p,t)$, in other words, resource tokens are not exhaustive and can be reusable after released, so $\Gamma_{P \to P_M}(M) = \Gamma_{P \to P_M}(M_i)$. Synthetically, there exists vector $\varpi^T = [1,1,\cdots 1]^T$, such that $\varpi^T M = \varpi^T M_0$, where $M(p_i) = \sum n_{i,h} a_{i,h}$, $n_{i,h} = M(p_i)(a_{i,h})$ is the quantity of token with a binding color $a_{i,h}$. So *TCRSM* is conservative.

Property 2 (*Boundedness*). Note that *RSM* is conservative, then there exist $\varpi^T M = \varpi^T M_0$, where $\varpi^T = [1,1,\cdots 1]^T$, that is $\forall p_i \in P_W \cup P_M$, satisfies $\sum_i M(p_i) = \sum_i \sum_h n_{i,h} a_{i,h}$, then $\forall p_i \in P_W \cup P_M$: $M(p_i) \leq \sum_i \sum_h n_{i,h} a_{i,h}$, so *TCRSM* is bounded.

Property 3 (*Deadlock-free*). For *RSM*, $\forall p \in P_M$, $\exists t \in T_W$, such that $I_M(p,t) = O_M(p,t)$, that is ${}^{\bullet}p = p^{\bullet}$, which means *RSM* contains siphons and traps, call them Σ. If for every resource place of Σ, there exists at least one token, that is $\forall p \in \Sigma : M_0(p) > 0$, each siphon contains a marked trap (MST), so *TCRSM* is deadlock-free[4].

2.3 Shared Composition of Sub Modules

To characterize the resources contention among inhomogeneous components have different processing routines, shared composition of sub modules is introduced. Jiang Changjun discussed the state constancy and behavior constancy on shared composition Petri nets[5].

Definition 4 (*Shared composition*). For timed colored Petri nets $TCRSM_i = (P_{Wi} \cup P_{Mi}, T_i, C_i, I_i, O_i, M_i)$, where $P_{W1} \cap P_{W2} = \Phi$, $P_{M1} \cap P_{M2} \neq \Phi$, $i = 1,2$. If $TCRSM = (P,T,C,I,O,M)$ satisfies that: (1) $P = P_1 \cup P_2$,

$P_1 \cap P_2 \neq \Phi$; (2) $T = T_1 \cup T_2$, $T_1 \cap T_2 = \Phi$; (3) $I = I_1 \cup I_2$, $O = O_1 \cap O_2$; (4) if $p \in P_{M1} \cap P_{M2}$, let $C(p) = C_1(p) \cup C_2(p)$, else if $p \in P_{W1} \cup P_{W2}$, let $C(p) = C_i(p)$, $i = 1,2$; (5) if $p \in P_{M1} \cap P_{M2}$, let $M(p) = \sum_{i=1}^{u_i} n_i \cdot m_i$, where $n_i = max\{M_1(p)(m_i), M_{21}(p)(m_i)\}$, else if $p \in P_{W1} \cup P_{W2}$, let $M(p) = M_i(p)$, $i = 1,2$, $TCRSM$ is the shared composition Petri nets of $TCRSM_1$ and $TCRSM_2$, for short, $TCRSM = TCRSM_1 O_p TCRSM_2$.

Property 4 (*State constancy*). Suppose that $TCRSM_i = (P_{Wi} \cup P_{Mi}, T_i, C_i, I_i, O_i, M_i)$ is timed colored Petri nets, $i = 1,2$, and $TCRSM = TCRSM_1 O_p TCRSM_2$. If $\forall M \in R(M_0)$, $\exists M_i \in R(M_{0i})$, such that $\Gamma_{P \to (P_i - (P_1 \cap P_1))}(M) = \Gamma_{P \to (P_i - (P_1 \cap P_1))}(M_i)$, then $TCRSM$ is state constant, where $P_i = P_{Wi} \cup P_{Mi}$, $i = 1,2$.

Proof: Let $\overline{TCRSM_i} = (\overline{P}_i, \overline{T}_i, \overline{F}_i, \overline{M}_{0i})$, $i = 1,2$ be sub nets of $TCRSM_i$ and satisfy the following conditions: (1) $\overline{P}_i = P_{Wi}$; (2) $\overline{T}_i = T_i$; (3) $\overline{F}_i = I_i \cap (P_{Wi} \times T_i) \cup O_i \cap (T_i \times P_{Wi})$; (4) if $p \in P_{Wi}$, then $\overline{M}_{i0}(p) = M_{0i}(p)$. Obviously, $\forall \overline{M}_i \in R(\overline{M}_{i0})$, such that $\Gamma_{P \to P_W}(M) = \Gamma_{P \to \overline{P}_i}(M) \in R(\overline{M}_{0i})$, then $\exists M_i \in R(M_{0i})$, $\Gamma_{P \to P_W}(M) = \Gamma_{P \to P_W}(M_i)$ is satisfied. Since $P_i - (P_1 \cap P_2) = P_{Wi}$, according to definition, $TCRSM$ satisfies state constancy.

State constancy reveals that sub modules keep mark of all places except shared ones unaltered after merged, which means in the merged module, new arrival component does not change the state of processing components. This property provides theoretic support for dynamic scheduling driven by events.

3 GA Based Multiple Aero-engine Static Scheduling

Static scheduling can be described as follows: all components are ready for processing at the beginning and there are no new arrivals, after scheduling the processing orders of all components are assume to be unaltered and machines never break down.

3.1 Static Scheduling Makespan Calculating

There are three ways to dealing with scheduling problem in Petri nets. One is by using reachability graph to determine a firing sequence of transitions to get the optimal schedule. The second way is by heuristic search of reachability graph. The last is using decision rules to solve conflicts[6]. In this paper, genetic algorithm is adopted to determine the firing sequence of transitions Petri nets.

Suppose that the processing order is given, which means the firing sequence of transitions with binding color instances is $\sigma = \{t_i(a_{i,u_i}), t_j(a_{j,u_j}), \cdots, t_k(b_{k,u_k})\}$, then the deterministic state transmission equation can be describe as: $M(n+1) = M(n) + \overline{W} \cdot S(n)$, where $\overline{W}$ is weight function defined from colored arcs, $S(n)$ is the firing vector from n^{th} to $(n+1)^{th}$ step. According to the firing rules of timed Petri nets, if transition t_i with a binding color $a_{i,h}$ is fire enable, the firing procedure is summarized as follows:

1) Before $(n+1)^{th}$ step, suppose that arrival timestamp of the token with color $a_{i,h}$ in input operation place of transition t_i is $f_{t_i}(a_{i,h}, n)$, arrival timestamp of the token with color m_i in input resource place of t_i is $f_{t_i}(m_i, n)$, then the firing timestamp $e_{t_i}(a_{i,h}, n+1)$ of transitions t_i with a binding color $a_{i,h}$ is calculated from:

$$e_{t_i}(a_{i,h}, n+1) = f_{t_i}(a_{i,h}, n) \oplus f_{t_i}(m_i, n) \qquad (3)$$

2) After $(n+1)^{th}$ step, transition t_i accomplishes firing after a duration d_i, the timestamp is $f_{t_i}(a_{i,h}, n+1)$, which is calculated from:

$$\begin{cases} f_{t_i}(a_{i,h}, n+1) = e_{t_i}(a_{i,h}, n+1) \otimes d_i \\ f_{t_i}(m_i, n+1) = f_{t_i}(a_{i,h}, n+1) \end{cases} \qquad (4)$$

In formulas (4) above, max-plus algebra symbols "$\oplus$" and "$\otimes$" are introduced. The two symbols are defined respectively as follows: $a \oplus b = max\{a, b\}$ and $a \otimes b = a + b$. Suppose that initial timestamp is $f_{t_i}(a_{i,h}, 0) = f_{t_i}(m_i, 0) = 0$ and the iterations ends after N steps when there are no enable transitions, then the makespan is calculated from $T = f_{t_1}(c, N) \otimes f_{t_2}(c, N) \cdots \otimes f_{t_u}(c, N)$, where $u = |T|$.

Obviously, given the firing sequence of transitions, makespan can be calculated within linear time complexity. In the next section, genetic algorithm is adopted to determine the firing sequence of transitions Petri nets.

3.2 Genetic Algorithm Based Static Scheduling

Genetic algorithm behaves excellently when dealing with configuration optimization problems. In this paper, Genetic operator encodes the transition firing sequence of Petri nets into chromosome and the scheduling algorithm decodes the chromosome into schedule. In the later genetic algorithm search procedure, population evolves generation by generation to converge to an optimal or near optimal solution.

1) *Encoding* Cheng analyzed the advantages and disadvantages of nine encoding schemas for classical job shop scheduling[7]. In this paper, each gene represents a transition and chromosome represents the transitions firing sequence. Encoding of transitions firing sequence can be carried out by using natural numbers. To be specific, all transitions in the sub module are encoded by the same number which is different from other sub modules, and every transition can be distinguished from each other by order of number in chromosome.

Table 1. Encoding Schema of Chromosme

Process	1-1	2-1	1-2	2-2	2-3	…
Transition	$t_1(h_1)$	$t_1(h_2)$	$t_2(h_1)$	$t_2(h_2)$	$t_3(h_2)$	…
Chromosome	1	2	1	2	2	…

Suppose that two instances of aero-engine module h_1 and h_2 are operated in work shop concurrently, then genes must be either "1" or "2". For example, as shown in table 1. if transitions in sub module of h_2 are described as $t_1(h_2)$, $t_2(h_2)$, …, then the first "2" must represents $t_1(h_2)$, and the later must represents $t_2(h_2)$, the rest may be deduced by analogy.

2) *Selection* Single aero-engine maintenance scheduling aims for minimal makespan, therefore those chromosomes with minor scheduling makespan owned more fitness. Suppose that there are N individuals in the generation G, for each chromosome $L \in G$, its fitness is calculated from:

$$F(L) = (T_{max} - T(L)) / (T_{max} - T_{min}) + \varepsilon \tag{5}$$

where ε is a compensate value to prevent the fitness from being zero, T_{max} and T_{min} are the maximal and minimal makspan values of individuals from G respectively. If the individuals of next generation are chosen by roulette, then those fitter ones will survive at higher possibility.

If multiple aero-engines are operated in work shop concurrently, the fitness is multi-objective function. According to weighted sum approach proposed by Tasahiko, the weighted fitness function $F_{\Sigma}(L)$ is defined as follows:

$$F_{\Sigma}(L) = \sum_{i=1}^{n} w_i \cdot F_i(L) \tag{6}$$

where n is the number of aero-engines, $F_i(L)$ is the fitness function of i^{th} chromosome, w_i is the weighted factor of i^{th} chromosome.

3) *Crossover* Illegal chromosomes maybe results from ordinary crossover operation, to avoid which a new crossover operation strategy is proposed. To be specific, first dividing the components' numbers into two subsets named $set1$ and $set2$, and making sure that the intersection of the two subsets is empty, then scan the two father chromosomes and let son chromosome be:

$$son1 = \Gamma_{parent1 \to set1}(parent1) \cup \Gamma_{parent2 \to set2}(parent2)) \tag{7}$$

where $\Gamma_{parent1 \to set1}(parent1)$ represents the projection of $parent1$ from $parent1$ to subset $set1$, finally exchanging $parent2$ with $parent1$, a new son chromosome is produced after repeating the operation above.

4) *Mutation* Mutation is fairly straightforward. The procedure is to exchange two genes in chosen chromosome randomly.

4 Rolling Horizon Based Dynamic Schduling

Aero-engine maintenance is a dynamic procedure and may be interrupted by stochastic events unexpected, therefore seeking for dynamic scheduling strategies adaptive to the alteration seems to be necessary considerably. Obviously, it is robust and sensitive to unexpected events, which differs from static scheduling.

4.1 Rolling Horizon and Scheduling Sub Module

Inspired by the rolling horizon optimization for the predictive control technology, Fang Jian proposed a period and event-driven rolling horizon scheduling strategy, which means scheduling objects are those components included in rolling horizon[8]. In this paper, corresponding definitions about rolling horizon in timed colored Petri nets are defined as follows:

Definition 5 (*Rolling horizon*). Rolling horizon C_H is defined as the subset of colors of the merged timed colored resource sharing module $TCRSM$, that is $C_H \subseteq C$. The color set number of rolling horizon $|C_H|$ is called capacity.

Definition 6 (*Dynamic scheduling sub module*). For sub nets $CPN_H = (P_H, T_H, C_H, I_H, O_H, M_{H0})$ of $TCRSM$, which satisfies: (1) $P_H = \{p \in P \mid C_H(p) \neq \Phi\} \cup P_M$; (2) $T_H = \{t \in T \mid C_H(t) \neq \Phi\}$; (3) if $p \in P_H$, and $t \in T_H$, let $I_H(p,t) = I(p,t)$; (4) if $p \in P_H$, and $t \in T_H$, let $O_H(p,t) = O(p,t)$, CPN_H is defined as the dynamic scheduling sub module of $TCRSM$ corresponding to Rolling horizon C_H.

There are two ways to construct scheduling sub module: one is to reconstruct the module by "bottom-up modelling" and keep the marking of operation places and resource places unaltered in $TCRSM$, the other is to decomposed the merged $TCRSM$ into two sub ones and one of whose color set is rolling horizon. The scheduling sub module is the reduction of merged system and used to dynamic scheduling.

4.2 Rescheduling Driven by Events and Period

There are two strategies for rescheduling: continuous scheduling and periodical scheduling. The former executes rescheduling once events emerges, which can cope with unexpected events, while the latter executes rescheduling periodically. In this paper, a hybrid strategy is adopted. Some key events including new project arriving, due date changing and machine breaking down are defined as following. If any key event takes place, then executes rescheduling, else executes rescheduling periodically.

1) *New components C_N Arriving* New instances should be added to module, which means new colors should be added to color set of merged $TCRSM$, that is $C'_W(P_W) = C_W(P_W) \cup C_N$, $C'_W(T_W) = C_W(T_W) \cup C_N$, and

keep the sate of processed and processing components unchanged, that is $\forall p \in P_W$, $\forall c \in C_W$: $M'(p)(c) = M(p)(c)$.

2) *Changing of due date* T_{ci} Suppose that $F_i(L)$ is the makespan of i^{th} chromosome, w_i is the weighted factor of i^{th} chromosome, if the variation value of T_{ci} is ΔT_{ci}, then w_i should be adjusted, let $w'_i = w_i \cdot (1 + \Delta T_{ci} / F_i(L))$ and the weighted fitness function $F_\Sigma(L)$ should be adjusted accordingly.

3) *Machine* $m \in M$ *breaking down* If a machine breaks down while machining, then remove the corresponding colored token kept by transition and keep the colored tokens' number $M(p)$ in resource places unchanged, else if the machine is idle and becomes unusable, then let $M'(p) = M(p) - 1$. If the broken down machine is recovered and becomes usable, the let $M'(p) = M(p) + 1$, where $p = h(m)$, h is the mapping function from machines to resource places.

5 Case Studying

Assembly workflow of aero-engine is shown in Figure 1, eight modules of aero-engine at the most left denote branches in the assembly procedure where each branch means one part of engine. Each part of an aero-engine is machined, balanced and assembled. In addition, blocks with the same name denote the same type of machine. The number above a block means the operation time of the operation on this machine. Arrows means sequence of operations.

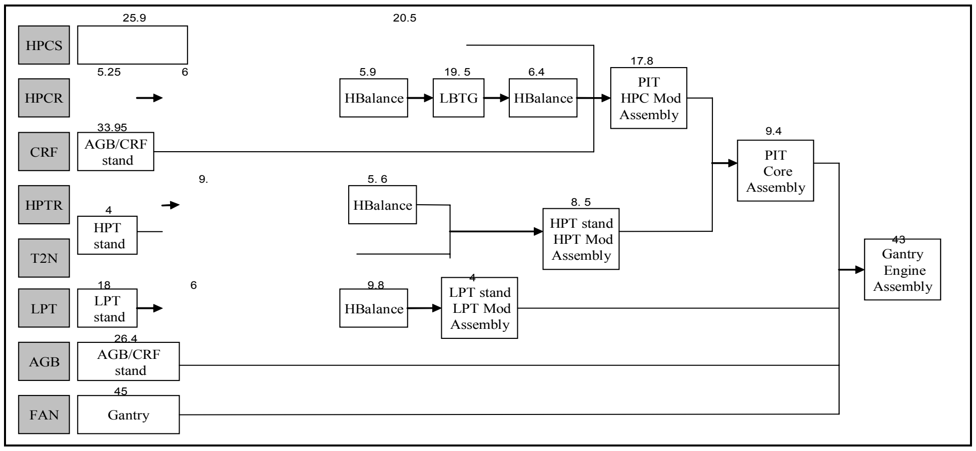

Figure 1. Aero-engine assembly workflow

Aero-engine assembly module represents by Petri nets is shown Figure 2. The mapping procedures from workflow can be described as follows: firstly the Resource Shared Modules of HPC, T2N, HPTR, HPCR, LPT, CRF, AGB, FAN are constructed respectively according to the workflow of each module, then merge all eight Resource Shared Modules by shared composition.

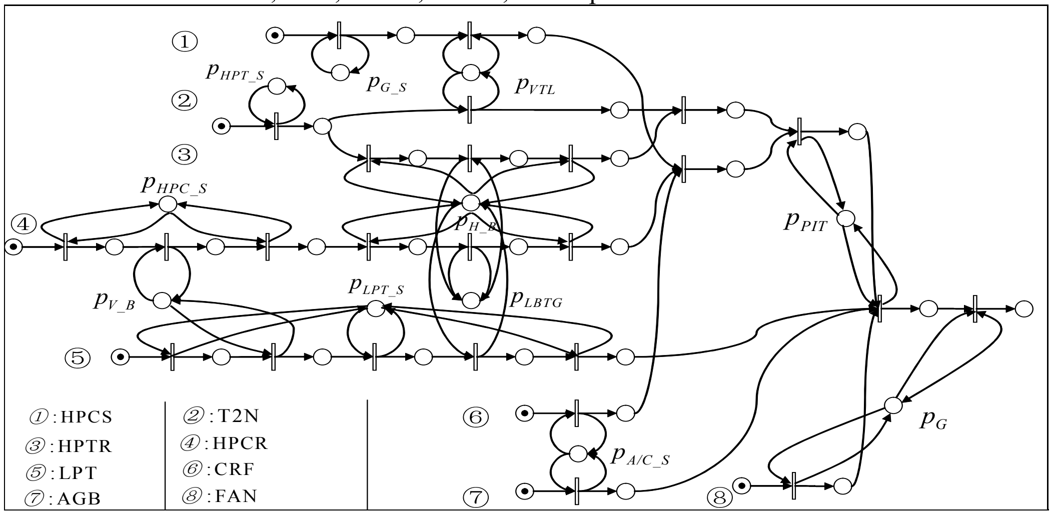

Figure 2. Petri nets module of aero-engine assembly

As shown in Figure 3. we choose 3 aero-engines situation with starting simultaneously. The rationality of Petri nets is verified by three aspects: (1) All processes follow the sequence of workflow strictly. (2) All processes are executed one by one without overlapping. (3) All processes are executed as closely as possible, which reduce the running time and increase the efficiency of utilization of machines.

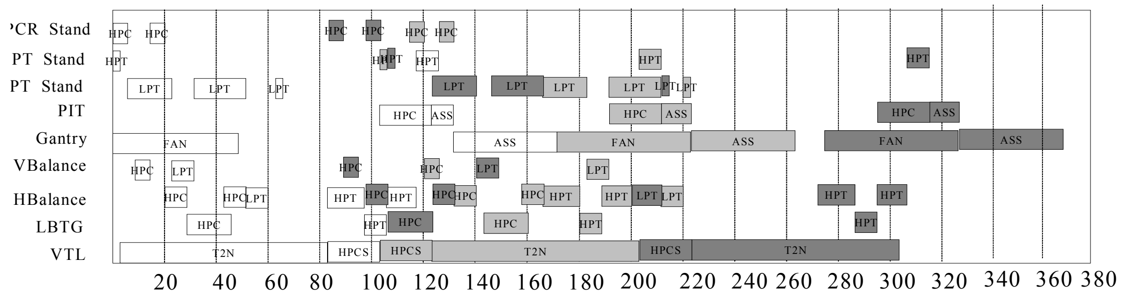

Figure 3. Static scheduling result for 3 aero-engines

From static scheduling result, for 1st aero-engine it takes 174.7 hours to accomplish the assembly procedure, while it takes only 366.4 hours for the 3 aero-engines. This is because the total time duration of aero-engines is only regarding the path which require the longest operation time. We maximize the utility of critical machines to minimize the time duration of processes operated on them. Thus, the makespan is not proportional to the number of aero-engines.

6 Cconclusion

To cope with the problem of concurrency and asynchrony of multiple aero-engines maintenance, shared composition method of Petri net is adopted to realize bottom-up modelling. The constructed model characterizes interactions between components with different processing routines and machine resources. Genetic algorithm is adopted to arrange firing sequence of transitions for static scheduling. To deal with discrete events emerge in workshop, a dynamic scheduling strategy uses rolling horizon composed by colors is presented, which made the model more adaptive and robust.

Acknowledgments

The authors also wish to acknowledge the financial support of the National Natural Science Foundation of China (60373102, 60572174).

Correspondence to:
Xinmin Tang
School of Mechatronics Engineering
Harbin Institute of Technology
Harbin, Heilongjiang 150001, China
Telephone: 01186-451-8641-3847
E-mail: txmofhit@hit.edu.cn

References

[1] P. Samaranayake, Development of engineering structures for scheduling and control of aircraft maintenance, International Journal of Operations & Production Management 2002;22(8): 843-867.
[2] Junyu Chen, The Study of Maintenance Scheduling in Aviation Engines, National Chiao Tung University, Taiwan, 2003: 9-12.
[3] Mu Der Jeng, Frank DiCesare, Synthesis Using Resource Control Nets for Modelling Shared-Resource Systems, IEEE Transactions on Robotics and Automation 1995; 11(3):317-327.
[4] Claude Girault, Rüdiger Valk, Petri Nets for Systems Engineering A Guide to Modelling , Verification, and Application, Springer-Verlag Berlin Heidelberg, 2003:214-215.
[5] Jiang Changjun, Dynamic constancy of Petri nets, Science in China (Series E) 1997; 27(6):567-573.
[6] Zhonghua Huang, Zhiming Wu, Deadlock-Free Scheduling Method for Automated Manufacturing Systems Using Genetic Algorithm and Petri Nets, Proceedings of the 2004 IEEE International Conference on Robotics & Automation 2004;566-571.
[7] Pan Quanke, Multi-Objective Scheduling Optimization of Job-shop in Intelligent Manufacturing System, Nanjing University of Aeronautics and Astronautics, Nanjing, 2003:20-25.
[8] Fang Jian, Xi Yugeng, "The Genetic Algorithms-Based Rolling Horizon Scheduling Strategy," Control Theory and Applictions 1997;14(4):589-594.