アルゴリズム　30－15
（1992.　11.　21）

時間付ペトリネットにおける スケジューリングのための優先順位表構成法

山内雅弘　渡辺敏正

広島大学工学部第二類(電気系)
724 東広島市鏡山一丁目4-1
(電話) 0824-22-7111 内3442(渡辺)
(ファクス) 0824-22-7195
(電子メール) watanabe@huis.hiroshima-u.ac.jp

あらまし　本稿の主題は時間付ペトリネットにおけるスケジューリングに用いるための優先順位表を新しく2つ提案することである．優先順位表は通常は時間付ペトリネット中のボトルネックと呼ばれる部分ネットに着目して構成されるが，その際にはSifakisの下界値(完了時間の下界値)を利用して定まるボトルネックが広く使われている．しかしSifakisの下界値計算はスケジューリングの実行可能性は全く考慮しないため，実際の最小完了時間はこの下界値よりははるかに大きくなる傾向がある．このことは優先順位表に基づくスケジューリングの精度を下げる原因となる．提案する二つの優先順位表はいずれもスケジューリングの実行可能性を考慮に入れて構成されたものであり，実験によりSifakisの下界値による優先順位表よりも優れていることを示す．

和文キーワード　スケジューリング，時間付ペトリネット，優先順位表，ボトルネック，複数発火規則，発火系列

Constructing Priority-Lists for Scheduling of Timed Petri Nets

Masahiro Yamauchi and Toshimasa Watanabe

Department of Circuits and Systems, Faculty of Engineering, Hiroshima University
4-1, Kagamiyama 1-chome, Higashi-Hiroshima, 724 Japan
Phone:+81-824-22-7111 Ext.3442(Watanabe), Fax:+81-824-22-7195
E-mail:watanabe@huis.hiroshima-u.ac.jp

Abstract　The subject of the paper is to propose two new priority-lists for scheduling of timed Petri nets. Priority-lists are usually constructed based on subnets, called *bottlenecks* in timed Petri nets, and bottlenecks by means of the *Sifakis bound*, a lower bound on completion time, have been widely used. Since no feasibility of scheduling is considered in its computation, actual minimum completion time tends to be much greater than this bound, possibly preventing priority-list scheduling algorithm utilizing this bound from producing good approximate solutions. Both of the proposed priority-lists are constructed by taking feasibility into consideration, and our experimental evaluation shows their superiority over those by the Sifakis bound.

英文 key words　Schedulings, timed Petri nets, priority-lists, bottlenecks, infinite-server semantics, firing sequences

1. Introduction

Two new priority-lists for priority-list scheduling of timed Petri nets under infinite server semantics are proposed. It is experimentally evaluated that scheduling algorithms using these priority-lists produce better solutions than FM_DPLA that has been showing best performance among those proposed in [26,27].

Scheduling theory is one of research fields that have been well investigated from both practical and theoretical viewpoints. The results are summarized in [4,16,17] for classical results: see [6,7,8] for bounding on approximate solutions and complexity results, [10,11] for scheduling in parallel processing. Although timed Petri nets are useful models in scheduling theory, related research results are much less than those using task graphs: see [5,9,23,24] for scheduling in marked graphs, [12,13,14,21,25] for minimum cycle time problems, [1,19] for periodic scheduling in timed Petri nets, [26,27] for priority-list scheduling in timed Petri nets, [28,29,30] for minimizing initial markings of ordinary or timed Petri nets.

Timed or ordinary Petri nets have two extreme possibilities in interpreting transition firing: *infinite-server semantics* and *single-server semantics*. The first semantics allows any transition to fire concurrently with itself, while this is not the case with the second one. Various processors and their total numbers are explicitly represented as places (called *processor pools*) and tokens residing within them, providing flexible models for scheduling problems. It is very likely that average completion time in cyclic scheduling can be reduced if cyclic structure of Petri net models is fully utilized. These explain some advantages of timed Petri nets over task graphs that have been used in ordinary scheduling problems.

We consider priority-list scheduling in timed Petri nets, that is, scheduling is done by choosing a transition of top priority from a priority-list. Priority-lists are constructed based on bottlenecks (each being a certain set of transitions defined later), which are counterparts of critical paths commonly used in ordinary scheduling. These lists are fixed as predetermined or can be changed dynamically. [26,27] proposed four algorithms SPLA, DPLA, FM_SPLA and FM_DPLA. SPLA and FM_SPLA (DPLA and FM_DPLA, respectively) are based on fixed (dynamic) priority-lists. It is reported in [26,27] that experimental results for more than 25290 total test data shows superiority of FM_DPLA among them.

The bottlenecks used in constructing their priority-lists are extracted by means of the well-known *Sifakis bounds* [25], which are lower bounds on completion time and which have been widely used in performance evaluation of Petri net models. Since no feasibility of scheduling (that is, firability of sequences of transitions in timed Petri nets) is considered in the computation of the Sifakis bound, actual minimum completion time tends to be much greater than this bound, possibly preventing scheduling algorithms, utilizing priority-lists constructed by means of this bound, from producing good approximate solutions. This is observed from experimental results on FM_DPLA: it is very often that, because of firability checking, transitions of middle priority are selected instead of those of high priority.

The subject of the paper is to propose two new ways of constructing priority-lists. The first one is modification of markings used in the computation of the Sifakis bound. An initial marking M_0 has been used in computing this bound, while our computation replaces it by a marking M_a, called an *active marking*, which consists of only tokens having possibility to be used in subsequent firing of transitions. The bound obtained by this modification is no less than the Sifakis bound, improving a lower bound on completion time. This modification incorporates firability to some extent.

It is experimentally observed that FM_DPLAM has better performance than FM_DPLA, where FM_DPLAM is FM_DPLA using priority-lists constructed from bottlenecks based on this modified bound.

The second one is completely new. For each transition t_f that can fire on a current marking of a Petri net, it finds a depth-first-search tree by starting from t_f and by searching edges in their direction. This tree intends to represent how tokens produced by firing t_f once are used by other transitions. Each of places p and transitions t of the tree has a weight supply(p) or rate(t), which is computed during the search. A weight, supply(p), of a place p intends to represent as a ratio how many tokens, among those that can be brought into p, are produced by firing t_f once. Another weight, rate(t), of a transition t intends to denote as a ratio how many tokens, among those deleted from input places of t, are produced by firing t_f once. The total sum of all rate(t) is denoted as effect(t_f). The value effect (t_f) is expected to show, as the sum of such ratios, to what extent firing t_f once hepls other transitions become firiable. A new priority-list is constructed according to values effect(t_f) of all transitions t_f that can fire on a current marking: transitions t_f with larger values of effect(t_f) get higher priority. This priority on transitions considers their firability as the most significant measure rather than time required by their subsequent firing.

Experimental results show that YW_PLA has better performance than FM_DPLA and is slightly better than FM_DPLAM, where YW_PLA is a scheduling algorithm based on this new priority-lists. The running time of YW_PLS is much less than those of FM_DPLA and FM_DPLAM.

2. Basic definitions

We assume that the reader is familiar with graph algorithms and Petri net theory (see [6,20,22], for example). A *digraph* is denoted by G=(V,A), where V and A are the sets of *vertices* and *directed edges* (often called *arcs*), respectively. We denote a directed edge e from u to v by e=(u,v). Let $*u=\{w|(w,u)\in A\}$, $u*=\{v|(u,v)\in A\}$ for $u\in V$. If $|*u|=0$ ($|u*|=0$) then u is called a *source* (a *sink*) of G. The graph obtained from G by replacing each directed edge with an undirected one is called the *underlying graph* of G. G is *weakly connected* if the underlying graph of G has an undirected path between any pair of vertices.

A *Petri net* is a simple bipartite digraph $PN=(P,T,E,\alpha,\beta)$, where P is the set of *places*, T is that of *transitions*

$$P\cap T=\phi,\ E=E_{in}\cup E_{out},\ E_{in}\subseteq K(T,P)=\{(u,v)|u\in T, v\in P\},$$

$$E_{out}\subseteq K(P,T)=\{(u,v)|u\in P, v\in T\},$$

$\alpha:E_{out}\to Z^+$ (nonnegative integers) and $\beta:E_{in}\to Z^+$ are weight functions. If $\alpha(e)=\beta(e')=1$ for any $e,e'\in E$, or if weight functions are independent of discussion then PN is denoted simply as PN=(P,T,E). We always consider PN to be a simple directed digraph unless otherwise stated. PN is a *marked graph* if $(\forall p\in P)|*p|,|*p|\le 1$. PN is a *state machine* if $(\forall t\in T)$ $|*t|,|t*|\le 1$. Let $C=C^+-C^-=[c_{ij}^+]-[c_{ij}^-]$ denote a $|P|\times|T|$ matrix, called the *place-transition incidence matrix* of PN, which is defined by

$$c_{ij}^+=\begin{cases}\beta(t_j,p_i) & \text{if } (t_j,p_i)\in E_{in},\\ 0 & \text{otherwise},\end{cases}\qquad c_{ij}^-=\begin{cases}\alpha(p_i,t_j) & \text{if } (p_i,t_j)\in E_{out},\\ 0 & \text{otherwise}.\end{cases}$$

A *marking* M of PN is a function $M:P\to Z^+$. We denote $|M|=\Sigma_{p\in P}M(p)$. A marking initially given is called an *initial marking*. A transition t is *firable* on a marking M consecutively k times ($k \geq 1$) if $M(p) \geq k \cdot \alpha(p,t)$ ($\forall p \in {}^*t$). *Firing* such a transition t on M consecutively $k'(\leq k)$ times is to define a marking M' such that, for $\forall p \in P$, $M'(p)=M(p)+k' \cdot \beta(t,p)$ if $p \in t^*-{}^*t$, $M'(p)=M(p)-k' \cdot \alpha(p,t)$ if $p \in {}^*t-t^*$, $M'(p)=M(p)-k' \cdot \alpha(p,t)+k' \cdot \beta(t,p)$ if $p \in t^* \cap {}^*t$ and $M'(p)=M(p)$ otherwise. We denote $M'=M[t>$ if $k'=1$. *Single-server semantics* is to restrict k' as $k'=1$ even if $k \geq 2$; *infinite-server semantics* is to allow k' to take any value with $1 \leq k' \leq k$. In this paper we use the term "Petri nets" under infinite-server semantics unless otherwise stated. Let $\delta=t_{i1} \ldots t_{is}$ be a sequence of transitions, called a *firing sequence*, and $\bar{\delta}(t)$ be the total number of occurrences of t in δ. $\bar{\delta}=[\bar{\delta}(t_1) \ldots \bar{\delta}(t_n)]^{tr}$ (transposition of a matrix or a vector) is the *firing count vector* of δ. For a marking M, δ is *legal* on M if t_{ij} is firable on M_{j-1}, where $M_0=M$ and $M_j=M_{j-1}[t_{ij}>$, $j=1,\ldots,s$. M_s is denoted as $M[\delta>$. For a |T|-dimensional vector $X=[x_1 \ldots x_n]^{tr}$ with $n=|T|$, δ is *legal* on M *with respect to* X if δ is legal on M and $\bar{\delta}=X$. We denote $|X|=\Sigma_{t \in T}X(t)$. $M[\delta>$ is *reachable* from M. For any subset $T' \subseteq T$, the subnet $PN_{T'}=(P',T',E')$ (generated by T') is defined by $P'=\{p \in P| {}^*p \cap T' \neq \emptyset$ or $p^* \cap T' \neq \emptyset\}$ and $E'=\{(p',t), (t,p'')|$ $t \in T'$, $p',p'' \in P'\}$. For a subset $S \subseteq P \cup T$, let PN-S denote the Petri net obtained from PN by deleting all element of S, where deleting $v \in S$ means deletion of v as well as all edges incident upon v. Let **0** (**1**, respectively) denote a vector with every component equal to 0 (1). A |T|-dimensional vector X with every component being a nonnegative integer is called a *T-invariant* of PN if $X \neq \mathbf{0}$ and $C \cdot X=\mathbf{0}$. A |P|-dimensional vector Y with every component being a nonnegative integer is called a *P-invariant* of PN if $Y \neq \mathbf{0}$ and $Y^{tr} \cdot C=\mathbf{0}$. Any linear combination X' of some T-invariants of PN is also a T-invariant if all elements of X' are nonnegative. A T-invariant X is called *elementary* if no linear combination of other T-invariants of PN is equal to it. Similarly elementary P-invariant is defined.

3. Timed Petri nets and scheduling

3.1. Timed Petri nets.

A *timed Petri net* is a Petri net $PN=(P,T,E,\alpha,\beta)$ with a *delay function* $D:T \rightarrow Z^+$. D(t) is called the *delay* of $t \in T$. It is often denoted as $PN=(P,T,E,\alpha,\beta,D)$ in the following. In this paper we assume that any transition t has $D(t)>0$. When we consider a timed Petri net PN, time instant or time interval is always associated with markings and firing of transitions of PN. (A Petri net without time is sometimes called an *ordinary* Petri net in order to distinguish it from a timed one.) An initial marking means a marking at time instant $0 \in Z^+$. A marking M at time instant λ is often denoted as $M^{<\lambda>}$. Firability of transitions is the same as those of ordinary ones. The difference exists in a resulting marking. If a transition t fires on a marking $M^{<\lambda>}$ then at the same time (more precisely, at time instant $\lambda+\varepsilon$ for a very small rational number $\varepsilon>0$) M is changed to another marking M' such that, for $\forall p \in P$, $M'(p)=M(p)-\alpha(p,t)$ if $p \in {}^*t$, and $M'(p)=M(p)$ otherwise. We formally define relation of $M^{<\lambda>}$ and $M^{<\lambda+\omega>}$, $\omega \in Z^+$; $\omega>0$.

Suppose that X' (X", respectively) is a |T|-dimensional vector such that X'(t') (X"(t')) denotes the total number of firing of $t' \in T$ whose firing begins at time instant τ', $\lambda \leq \tau' \leq \lambda+\omega$ (whose firing ends at time instant τ'', $\lambda \leq \tau'' \leq \lambda+\omega$). We define two m-dimensional vectors

$$B(t,\alpha)=[X'(t) \cdot b_1(t,\alpha),\ldots,X'(t) \cdot b_m(t,\alpha)]^{tr},$$

$$B(t,\beta)=[X''(t) \cdot b_1(t,\beta),\ldots,X''(t) \cdot b_m(t,\beta)]^{tr}$$

for $P=\{p_1,\ldots,p_m\}$ ($m=|P|$) such that

$$b_s(t,\alpha)=\begin{cases} -\alpha(p_s,t) & \text{if } p_s \in {}^*t, \\ 0 & \text{otherwise,} \end{cases} \quad b_s(t,\beta)=\begin{cases} \beta(t,p_s) & \text{if } p_s \in t^* \\ 0 & \text{otherwise,} \end{cases}$$

$s=1, \ldots, m$. Then $M^{<\lambda+\omega>}$ is defined by

$$M^{<\lambda+\omega>}=M^{<\lambda>} + \Sigma_{t \in T}(B(t,\alpha)+B(t,\beta)).$$

$M^{<\lambda+\omega>}$ is reachable from $M^{<\lambda>}$. If t fires at time instant λ and no other transition fires until time instant $\lambda+D(t)$ then we denote a marking at time instant $\lambda+\varepsilon$ by

$$M^{<\lambda+\varepsilon>}=M^{<\lambda>}[t,\alpha>=M^{<\lambda>}+B(t,\alpha)$$

and a marking at time instant $\lambda+D(t)$ by

$$M^{<\lambda+D(t)>}=M^{<\lambda>}[t> = M^{<\lambda>} + B(t,\alpha) + B(t,\beta).$$

In this paper we assume that a timed Petri net $PN=(P,T,E,\alpha,\beta,D)$ has a specified set $L=\{h_1,\ldots,h_r\} \subseteq P$, $r \geq 1$, such that $\cup_{1 \leq i \leq r}T(h_i)=T$, where

$$T(h_i)=\{t \in T|h_i \in {}^*t \cap t^*\} \text{ and } \alpha(h_i,t)=\beta(t,h_i)=1$$

for any $t \in T(h_i)$, $i=1,\ldots,r$. Each $h_i \in L$ is called a *processor pool* (or simply a p-pool) of type i. PN'=PN-L is called the *underlying Petri net* of PN.

3.2. Scheduling in timed Petri nets.

We define scheduling in a timed Petri net PN with a set L of processor pools. Suppose that nonnegative integers $q_1,\ldots,q_r$ are given, and let M_0 be any initial marking of PN satisfying that $M_0(h_i)=q_i$, $i=1,\ldots,r$. This means that there are r types $1,\ldots,r$ of processors (represented by processors pools $h_1,\ldots,h_r$) and that total $q_i(=M_0(h_i))$ processors of type i are available initially for $i=1,\ldots,r$. In the following, unless otherwise stated, we assume that any initial marking of PN is as above. All processors of type i has the same capability and are numbered $1,\ldots,q_i$, for each i, $1 \leq i \leq r$. Let $q_{max}=\max\{q_1,\ldots,q_r\}$. For a given |T|-dimensional vector X, let

$$\Psi(T,X)=\{(t,1),\ldots, (t,X(t))|t \in T\},$$

where X(t) denotes the element of X corresponding to $t \in T$.

We assume that timed Petri nets satisfy the following conditions (C-1) - (C-4) unless otherwise stated.

(C-1) *no wait*: any transition has to fire as soon as it becomes firable.

(C-2) *nonpreemptive*: once a transition t starts firing at some time instant $j \in Z^+$ then it keeps firing through $j+D(t)$ and cannot be interrupted during this interval.

(C-3) Only time instant that is an integer is considered, where we consider both $M^{<\lambda>}$ and $M^{<\lambda+\varepsilon>}=M^{<\lambda>}[t,\alpha>$ as markings at time instant λ.

(C-4) Transitions can fire only at some time instant.

Suppose that we are given a timed Petri net $PN=(P,T,E,\alpha,\beta,D)$, an initial marking M_0 and a |T|-dimensional vector X. Let

$Z_r=\{1,...,r\}$, $\Delta=\{1,...,q_{max}\}$, and

$L(t)=\{h_i\in L|h_i\in {}^*t\cap t^*\}$ for each $t\in T$.

For any subset

$$S=\{\{i_1,j_1\},...,\{i_{r'},j_{r'}\}\}\subseteq 2^{Z_r\times\Delta}$$

with $r'\le r$, $1\le i_1\le...\le i_{r'}\le r$, $1\le j_k\le q_{i_k}$ $(k=1,...,r')$, let

$$Z(S)=\{i_1,...,i_{r'}\} \text{ and } \Delta(S)=\{j_1,...,j_{r'}\}.$$

A *scheduling* is a function

$$\sigma:\Psi(T,X)\rightarrow Z^+\times 2^{Z_r\times\Delta}$$

satisfying (1)-(5), where $\sigma((t,x))$ is written as $\sigma(t,x)$ for notational simplicity, and the first or second element of $\sigma(t,x)$ is denoted as $\sigma(t,x)_1\in Z^+$ (time instant) or $\sigma(t,x)_2\in 2^{Z_r\times\Delta}$ (a set of processors of some types), respectively:

(1) if M is a marking at time instant $\sigma(t,x)_1$ then t is firable on M for any $(t,x)\in\Psi(T,X)$;

(2) $\sigma(t,x)_1$ is time instant when firing of t is supposed to begin for each $(t,x)\in\Psi(T,X)$;

(3) if $\sigma(t,x)_2=\{\{i_1,j_1\},...,\{i_{r'},j_{r'}\}\}$ for some $r'\le r$ then (i) and (ii) hold:

(i) $\{h_{i_1},...,h_{i_{r'}}\}=L(t)$;

(ii) the x-th firing of t is associated with processing by the j_k-th processor of type i_k, which is selected from available ones, for each k, $k=1,...,r'$;

(4) if there is any pair $\sigma(t_1,x_1)_2$ and $\sigma(t_2,x_2)_2$ such that $t_1\neq t_2$ and $\{i_k,j_k\}\in\sigma(t_1,x_1)_2\cap\sigma(t_2,x_2)_2$ then

$\sigma(t_1,x_1)_1\ge\sigma(t_2,x_2)_1+D(t_2)$ or

$\sigma(t_1,x_1)_1+D(t_1)\le\sigma(t_2,x_2)_1$;

(5) $|\theta(j,\lambda)|\le q_j$ for any type j, $1\le j\le r$, and any time instant $\lambda\in Z^+$, where

$$\theta(j,\lambda)=\{(t,x)\in\Psi(T,X)|j\in Z(\sigma(t,x)_2),\ \sigma(t,x)_1\le\lambda\le\sigma(t,x)_1+D(t)\}.$$

(Note that the following (6) will be added in handling single-server semantics: (6) if there is any pair $\sigma(t,x_1)_2$ and $\sigma(t,x_2)_2$ with $x_1<x_2$ then $\sigma(t,x_1)_1+D(t)\le\sigma(t,x_2)_1$.) If $r'=1$ then $\{\{i_1,j_1\}\}$ is denoted as $\{i_1,j_1\}$ to avoid extra brackets. Let

$$\tau(\sigma)=\max\{\sigma(t,x)_1+D(t)|(t,x)\in\Psi(T,X)\}.)$$

Such σ is called a *scheduling of completion time* $\tau(\sigma)$ with respect to M_0, X and PN. If we obtain a marking $M^{<\tau(\sigma)>}$ equal to a given initial marking $M_0(=M^{<0>})$ then σ is called a *cyclic scheduling* of *period* $\tau(\sigma)$ with respect to M_0, X and PN.

We define the *Scheduling Problem of Timed Petri Nets* **PLS($r;q_1,...,q_r$)**:

Instance: A timed Petri net $PN=(P,T,E,\alpha,\beta,D)$ with a set $L=\{h_1,...,h_r\}\subseteq P(r\ge1)$ of processor pools, r nonnegative integers $q_1,...,q_r$, a firing vector X and an initial marking M_0.

Question: Find a scheduling σ of minimum completion time $\tau(\sigma)$ with respect to M_0, X and PN.

Example 1. We show an example of **PLS($r;q_1,...,q_r$)** with $r=2$, $q_1=q_2=1$. Suppose that there are two jobs J_1, J_2 with $J_1(J_2$ respectively) consisting of two (three) tasks, denoted as $J_1=\{t_1,t_2\}$, $J_2=\{t_3,t_4,t_5\}$, and that we have two types of processors, one processor for each type, denoted as $Type_1=\{h_1\}$, $Type_2=\{h_2\}$. The following constraints are imposed on the tasks:

(i) t_1 is to be processed in a unit time by the processor h_1 after t_2 is finished.

(ii) t_2 is to be processed in a unit time by the processor h_1 after both t_1 and t_3 are finished.

(iii) t_3 is to be processed in a unit time by the two processors h_1 and h_2 after t_5 is finished.

(iv) t_4 is to be processed in a unit time by the processor h_2 after t_3 is finished.

(v) t_5 is to be processed in a unit time by the processor h_2 after both t_2 and t_5 are finished.

A timed Petri net $PN=(P,T,E,\alpha,\beta,D)$, representing this situation schematically, is constructed as follows (see Fig. 1):

$P=\{p_i|i=1,...,5\}\cup\{h_1,h_2\}$, $L=\{h_1,h_2\}$,

$T=\{t_i|i=1,...,5\}$,

$E=E'\cup E_L$ with

$E'=\{(t_1,p_1),(p_1,t_2),(t_2,p_2),(p_2,t_1),$
$(t_3,p_1),(p_2,t_5),(t_3,p_3),(p_3,t_4),$
$(t_4,p_4),(p_4,t_5),(t_5,p_5),(p_5,t_3)\}$,

$E_L=\{(h_1,t_i),(t_i,h_1)|i=1,2,3\}\cup\{(h_2,t_i),(t_i,h_2)|i=3,4,5\}$,

$$\alpha(u,v)=\begin{cases}2 & \text{if } (u,v)=(p_1,t_2),\\ 1 & \text{otherwise,}\end{cases}\quad \beta(u,v)=\begin{cases}2 & \text{if } (u,v)=(t_2,p_2),\\ 1 & \text{otherwise.}\end{cases}$$

Let $X=[1,1,1,1,1]^{tr}$ be the T-invariant of PN defined in Example 1. Every transition $t\in T$ is assumed to have delay $D(t)=1$, and

$\Psi(T,X)=\{(t_i,1)|i=1,2,3,4,5\}$.

For the initial marking $M=[1,0,0,0,1,1,1]^{tr}$, where the i-th element denotes $M(p_i)$, the following mapping σ is a scheduling with $\tau(\sigma)=3$ (see Fig. 2):

$\sigma(t_1,1)=(2,\{1,1\})$, $\sigma(t_2,1)=(1,\{1,1\})$,

$\sigma(t_3,1)=(0,\{\{1,1\},\{2,1\}\})$, $\sigma(t_4,1)=(1,\{2,1\})$,

$\sigma(t_5,1)=(2,\{2,1\})$.

That is, t_3 is firable on $M^{<0>}(=M)$ and it fires at time instant 0, reaching the marking $M^{<1>}=[2,0,1,0,0,1,1]^{tr}$. Then both t_2 and t_4 are firable on $M^{<1>}$, and firing them makes the marking $M^{<2>}=[0,2,0,1,0,1,1]^{tr}$. Both t_1 and t_5 are fiable on $M^{<2>}$, and their firing results in the marking $M^{<3>}=M^{<0>}$, assuring that X is a T-invariant of PN. In this case $\tau(\sigma)=3$ is the minimum total completion time, and σ is a solution to **PLS(2;1,1)**.

3.3. Average completion time.

Let σ be a cyclic scheduling with respect to M_0, X and PN, where X is a T-invariant of PN. For some integer $k\ge1$, suppose that there is a scheduling σ' with respect to M_0, kX and PN. Clearly σ' is a cyclic scheduling. Then the ratio $\tau(\sigma')/k$ is called *average completion time* (with respect to M_0, X and PN). The value

$\min\{\tau(\sigma')/k\ |k\in Z^+,\ k\ge1\}$

is called the *minimum average completion time* (with respect to M_0, X and PN). If there is $k\ge2$ such that $\tau(\sigma')/k<\tau(\sigma)$ then this implies that fully utilizing cyclic structures in cyclic scheduling of timed Petri nets may reduce average completion time. Task graphs can only produce a cyclic schduling σ' with

with respect to M_0, kX and PN such that $\tau(\sigma')=k\cdot\tau(\sigma)$. Finding such an integer k giving smaller average completion time can be done by using timed Petri nets, and this is one of advantages over task graphs in handling cyclic scheduling. Fig. 3 schematically explains this situation by means of Gantt charts having $\tau(\sigma')=k\cdot\tau(\sigma)=10$ and $\tau(\sigma')=9<k\cdot\tau(\sigma)=10$, where $\tau(\sigma)=5$ and $k=2$.

4. Priority-list scheduling

In this section only timed Petri nets having at least one T-invariant as well as at least one P-invariant are considered unless otherwise stated. Suppose that we are given any instance of **PLS**$(r;q_1,\dots,q_r)$, that is, a time Petri net $PN=(P,T,E,\alpha,\beta,D)$ with r processor pools h_i in which q_i processors are available initially for $i=1,\dots,r$, an initial marking M_0, and a firing vector X that is a T-invariant. In [26,27] four priority-list scheduling algorithms SPLA, FM_SPLA, DPLA and FM_DPLA are proposed. Experimental results for more than 25290 total test data show superiority of FM_DPLA. The difference of the four algorithms is construction of priority-lists, and they are combined as procedure PLS(θ) in [26,27], where $\theta=1$ if SPLA, $\theta=2$ if DPLA, $\theta=3$ if FM_SPLA and $\theta=4$ if FM_DPLA.

Priority-list scheduling algorithms are represented as a general scheme GS(θ) as follows, where priority-lists are fixed as predetermined if $\theta=1$, while they are changed dynamically if $\theta=2$. Note that $\theta=0$ for each of the three algorithms FM_DPLA, FM_DPLAM and YW_PLS to be considered in the following. The priority-list L to be used in GS(θ) consists of transitions that are sorted in nonincreasing order of priority (the first element has the highest priority). The formal description of **PLS**(θ) is as follows.

```
procedure GS(θ);
   begin
1.  τ←0; M<τ>←M0;
    for each t∈T do X'(t)←0;
    for i=1 to r do j_i←1; /* the least index of available
                                          processor of type i */
2.  Construct a priority-list L.
3.  Choose from L a transition t of highest priority, with
       X'(t)<X(t), among those that are firable on M<τ>;
       /* Note that X'(t)<X(t) means X(t)>0 */
       /* t is firable if and only if j_{i_k}≤q_{i_k}, k=1,...,r(t),
                             where L(t)={h_{i_1},...,h_{i_r(t)}} */
4.  if a transition t, with X'(t)<X(t),
            that is firable on M<τ> is found then
      begin
      σ(τ,X'(t))←(τ,{{i_1,j_{i_1}},...,{i_r(t),j_{i_r(t)}}})
                             for L(t)={h_{i_1},...,h_{i_r(t)}};
      X'(t)←X'(t)+1;
      for k=1 to r(t) do j_{i_k}←j_{i_k}+1;
      for each p∈*t do M<τ>(p)←M<τ>(p)-α(p,t);
      if θ is even then goto Step 2
      else goto Step 3
      end
   else /* find the nearest time instant τ'>τ at which
             some transitions end firing */
      if there is t'∈T with X'(t')<X(t') then
        begin
        τ'←min{σ(τ'',X'(t))_1+D(t)| t∈T, τ''<τ,
                  σ(τ'',X'(t))_1<τ<σ(τ'',X'(t))_1+D(t)};
        for each t∈T with σ(τ'',X'(t))_1+D(t)=τ' do
           begin
           for each p∈P do
              if p∈t* then
                   M<τ'>(p)←M<τ''>(p)+β(t,p)
              else M<τ'>(p)←M<τ''>(p);
           for each h_{i_k}∈L(t)={h_{i_1},...,h_{i_r(t)}} do
                j_{i_k}←j_{i_k}-1
           end;
        τ←τ';
        goto Step 3
        end
      else
        halt
end;
```

Let O(PL) denote time complexity for computing L, and let

$|X|=\Sigma_{t\in T}X(t)$, $\chi=\Sigma_{t\in T}X(t)\cdot D(t)$ and $Q=\Sigma_{1\le i\le r}q_i$.

Time complexity of PLS(θ) is as follows:

$O(PL(1)+|P||T||X|+\chi Q)$ if $\theta=1$;

$O((PL(2)+|P||T||X|)|X|+\chi Q)$ if $\theta=2$.

5. Bottlenecks and priority-lists

We define S-bottlenecks, bottlenecks by the Sifakis bounds [25] on completion time of timed Petri nets. Also explained is some ways of constructing priority-lists based on S-bottlenecks. They are used in the four algorithms SPLA, DPLA, FM_SPLA and FM_DPLA [26, 27]. Two new ways of constructing priority-lists are also given.

5.1 Bottlenecks.

Given an initial marking M_0, a P-invariant Y and a |T|-dimensional vector X of PN, let

$\omega(X,Y)=(Y^{tr}\cdot C^{-}\cdot\bar{D}X)/Y^{tr}\cdot M_0$

be called the *Sifakis bound* [25] with respect to X and Y, where $\bar{D}$ is a $|T|\times|T|$ diagonal matrix with elements d_{ij} such that

$$d_{ij}=\begin{cases} D(t_i) & \text{if } i=j,\\ 0 & \text{otherwise,}\end{cases}$$

for $i,j=1,\dots,|T|$, where $T=\{t_1,\dots,t_{|T|}\}$. Let

$\omega(X,PN)=\max\{\omega(X,Y')\mid Y' \text{ is an elementary P-invariant of } PN\}$.

We call $\omega(X,PN)$ the Sifakis bound of PN (with respect to X). Let Y be a P-invariant with $\omega(X,Y)=\omega(X,PN)$,

$P_Y=\{p\in P'\mid Y(p)>0\}$ and $T_Y=\cup_{p\in P_Y}(^{*}p\cup p^{*})$,

where P' is the place set of the underlying Petri net PN' of PN. The set T_Y is called a *S-bottleneck* of PN (or the S-bottleneck with respect to X and Y). The following proposition shows that $\omega(X,PN)$ is a lower bound on the period of any cyclic scheduling of timed Petri nets.

Proposition 1 [25]. Suppose that PN is a timed Petri net having a T-invariant X and at least one P-invariant, and let σ be a cyclic scheduling of period $\tau(\sigma)$ with respect to M, X and PN. Then

$\tau(\sigma) \geq \omega(X,PN)$. ♦

The value $\omega(X,PN)$ can be computed by means of a linear programming as shown in the following proposition.

Proposition 2 [2,3]. The Sifakis bound $\omega(X,PN)$ of Proposition 1 can be computed from an optimum solution Y given by solving the following linear programming problem:

$$\text{maximize} \quad \omega = Y^{tr} \cdot C^{-} \cdot D \cdot X$$

$$\text{subject to} \quad Y^{tr} \cdot C = \mathbf{0},\ Y \geq 0 \text{ and } Y^{tr} \cdot M_0 = \mathbf{1}.\ ♦$$

Example 2. PN of Fig. 1 has two elementary T-invariants

$X_1=[2,1,0,0,0]^{tr}$, $X_2=[0,1,2,2,2]^{tr}$,

where the j-th element of X_i denotes $X_i(t_j)$. It also has four elementary P-invariants

$Y_1=[1,1,0,0,1,0,0]^{tr}$, $Y_2=[0,0,1,1,1,0,0]^{tr}$,

$Y_3=[0,0,0,0,0,1,0]^{tr}$, $Y_4=[0,0,0,0,0,0,1]^{tr}$,

where the j-th element of Y_i denotes $Y_i(p_j)$. Let

$X=[1,1,1,1,1]$,

which is a T-invariant of PN. Then we have

$\omega(X,Y_1)=5/2=2$, $\omega(X,Y_i)=3/1=3$, $i=2,3,4$.

Hence

$\omega(X,PN)=3$,

$P_{Y_1}=\{p_1,p_2,p_5\}$, $\quad T_{Y_1}=\{t_1,t_2,t_3,t_5\}$,

$P_{Y_2}=\{p_3,p_4,p_5\}$, $\quad T_{Y_2}=\{t_3,t_4,t_5\}$,

$P_{Y_3}=\{h_1\}$, $\quad T_{Y_3}=\{t_1,t_2,t_3\}$,

$P_{Y_4}=\{h_2\}$, $\quad T_{Y_4}=\{t_3,t_4,t_5\}$. ♦

5.2. Priority lists.

We explain three measures MR_i, $i=1, 2, 3$, used in determining priority among transitions in [26, 27]. We fix PN, M_0 and X in this section. The first measure is

MR1: if $D(t)>D(t')$ then t has priority over t'.

Let Y be a P-invariant with $\omega(X,PN)=\omega(X,Y)$ and T_Y be the bottleneck with respect to X and Y. The second measure MR2 is defined as follows:

MR2: if $t \in T_Y$ and $t' \notin T_Y$ then t has priority over t'. The third Measure MR3 is a little complicated, and is based on the costs given by the following procedure. We provide a $|T| \times \zeta$ matrix μ in which any element $\mu(i,j)=0$ initially, where ζ is the total number of elementary P-invariants of PN.

procedure AC;

Step1. Find all elementary P-invariants $Y_1,\ldots,Y_\zeta$ of PN by using the Fourier-Motzkin method (see [14]);

Step2. For each Y_j, repeat Steps 3 and 4;

Step3. Compute $\omega_j=\omega(X,Y_j)$;

Step4. For each $t_i \in T_{Y_j}$, $\mu(i,j) \leftarrow \omega_j$;

Step5. For each $t_i \in T$, sort $\mu(i,1),\ldots,\mu(i,\zeta)$ in nonincreasing order, and reindex them as $\mu(i,1) \geq \ldots \geq \mu(i,\zeta)$;

Step6. Let $\mu(i)$ denote the ζ-dimensional vector $[\mu(i,1),\ldots,\mu(i,\zeta)]^{tr}$, $i=1,\ldots,|T|$;

Sort $\mu(1),\ldots,\mu(|T|)$ in lexicographically nonincreasing order, and reindex them as $\mu(1) \geq \ldots \geq \mu(|T|)$;

MR3 is defined by using $\mu(1),\ldots,\mu(|T|)$:

MR3: if $\mu(i)>\mu(j)$ then t_i has priority over t_j.

Example 3. PN of Fig 1 has four elementary P-invariants Y_j, $j=1,\ldots,4$, as given in Example 2. Let $X=[1,1,1,1,1]$, which is a T-invariant of PN. For each t_i, $i=1,\ldots,5$, $\mu(i,j)=\omega(X,Y_j)$ (below left) and the lexicographically sorted result (below right) are given as follows, where the priority is denoted by the right most numbers in parentheses:

	Y_1	Y_2	Y_3	Y_4
t_1	5/2	0	3	0
t_2	5/2	0	3	0
t_3	5/2	3	3	3
t_4	0	3	0	3
t_5	5/2	3	0	3

→

t_1	3	5/2	0	0	(4)
t_2	3	5/2	0	0	(5)
t_3	3	3	3	5/2	(1)
t_4	3	3	0	0	(3)
t_5	3	3	5/2	0	(2)

.

We define the following priority among MRi, i=1,2,3.

PR1: MR2>MR1; PR2: MR3>MR1,

where MR2>MR1, for example, means that if t and t' has the same priority concerning MR2 then the one of higher priority concerning MR1 is chosen; if t and t' has the same priority concerning MR1 then we assign priority such that the one with smaller index has higher priority, among all such transitions. Hence transitions of T are totally ordered. For each PRi, i=1,2, we construct a list $PL(i)=\{t_1,\ldots,t_{|T|}\}$ of transitions of PN, where t_j has higher priority over t_{j+1} and is chosen before t_{j+1} according to PRi, for $j=1,\ldots,|T|-1$. The list PL(i) is called the *priority list of type* i (with respect to X and PRi), i=1,2, and these lists are used in the four approximation algorithms SPLA, FM_SPLA, DPLA and FM_DPLA proposed in [26,27].

5.3. Modification of the S-bottleneck computation.

We explain modification of the S-bottleneck computation. Computation of the Sifakis bound is modified so that it may reflect firability of transitions. After this modified bound is obtained, we find bottlenecks based on it and priority-lists are constructed as in 5.2. FM_DPLA using these modified priority-lists is denoted as FM_DPLAM. Only modification of the Sifakis bound is described in the following.

Let M_0, Y and X be an initial marking, a P-invariant and a firing count vector of PN, respectively. Let M be any marking reachable from M_0. Since

$$Y^{tr} \cdot M_0 = Y^{tr} \cdot (M + C^{+} \cdot X_{fire}),$$

we have

$$\omega = (Y^{tr} \cdot C^{-} \cdot \bar{D} \cdot X_{rest}) \,/\, Y^{tr} \cdot (M + C^{+} \cdot X_{fire}),$$

where X_{fire} is a $|T|$-dimensional vector showing that each $t \in T$ has fired $X_{fire}(t)$ times so far and

$$X_{rest}(t) = X(t) - X_{fire}(t) \text{ for } \forall t \in T.$$

This equivalent formula computing ω implies that there are two kinds of tokens, *active tokens* and *dead ones* : an active one has possibility to be used by subsequent firing of some transitions, and a dead one has no such possibility. It seems that incorporating dead tokens in computing ω may make the value less than ω much less than actual minimum completion time. We define a marking M_a, called an *active marking*, defined as follows :

$$M_s = M + C^{+} \cdot X_{fire},$$

$$M_d = C^{-} \cdot X_{rest},$$

and, for each $p \in P$,

$$M_a(p) = \begin{cases} M_s(p) & \text{if } M_s(p) < M_d(p), \\ M_d(p) & \text{otherwise.} \end{cases}$$

Define a new bound ω' by

$\omega' = (Y^{tr} \cdot C^{-} \cdot \bar{D}.X_{rest}) / Y^{tr} \cdot M_a$.

Since $Y^{tr} \cdot M_0 \geq Y^{tr} \cdot M_a$, we have $\omega' \geq \omega$.

M_a is computed by the following procedure. It runs in $O(|P|+|T|+|E|)$ time, and actual computation time for ω' is almost the same as that for ω.

```
procedure active_marking(M);
  begin
  for each p∈P do
    begin
    Ma(p)←0; Ms(p)←M(p); Md(p)←0;
    for each t∈*p with Xfire(t)>0 do
        Ms(p)← Ms(p)+Xfire(t)·β(t,p);
    for each t∈p* with Xrest(t)>0 do
        Md(p)← Md(p)+Xrest(t)·α(p,t);
    if(Ms(p)<Md(p)) then Ma(p)← Ms(p)
    else Ma(p)← Md(p)
    end
  end;
```

5.4. A new priority-list.

We propose a new method for determing priority on transitions, by taking firability into consideration. As mentioned in Section 1, this method is a depth-first-search during which weights are assigned to places and transitions. These weights intend to represent how firing a transition once on a current marking affects subsequent firing of other transitions.

The outline determining priority on transitions is stated in the following, and the formal description will be given later as procedure **comp_effect**(M).

Suppose that a marking M is reached from an initial marking M_0 by firing each $t \in T$ by $X_{fire}(t)$ times, and let X_{rest} be defined by a given firing count vector X as

$$X_{rest}(t)=X(t)-X_{fire}(t),\ t \in T.$$

Let $t_f \in T$ be any transition that is firable on M and $X_{rest}(t_f)>0$. We compute a value effect(t_f) by the following Steps 1-5.

Step 1. Define a marking M_v by

$$M_v=M-M'+C^{+} \cdot X_{fire},$$

where M' is another marking defined, for each $p \in P$, by

$$M'(p) = \begin{cases} \alpha(p,t_f) & \text{if } p \in {}^*t_f, \\ 0 & \text{otherwise.} \end{cases}$$

M'(p) represents tokens necessary for firing t_f once. Suppose that M is a marking at time instant τ_0. $M'+C^{+} \cdot X_{fire}$ is a marking at some time instant τ, with $\tau>\tau_0$, such that any firing that has begun at time instant $\tau' \leq \tau_0$ is finished at τ, under the assumption that no other transitions begin their firing at any time instant τ'' with $\tau_0 \leq \tau'' \leq \tau$.

Step 2. Execute a depth-first-search starting at t_f and tracing unvisited edges in their direction. Each edge is originally marked "UNVISITED", and will be marked "VISITED" once it is visited by the search: every edge is visited at most once. At each place p visited from $t \in {}^*p$ during the search, we compute a value max(p) by executing

$$\max(p) \leftarrow \max\{\max(p), \beta(t,p)\}.$$

This means that we will obtain

$$\max(p)=\max\{\beta(t,p) \mid t \in {}^*p \text{ and } (t,p) \text{ is marked "VISITED"}\}$$

after the completion of the search. Subsequent search from p to $t' \in p^*$ is stopped if

$$\max(p)+M_v(p)<\alpha(p,t') \text{ or } M_v(p) \geq \alpha(p,t').$$

This is because all tokens produced by firing transitions visited so far are to be used without passing through p (meaning that tokens produced by firing t_f once cannot expand further) or because t' can fire even if no such tokens are brought to p (firing of t' is independent of that of t_f), respectively.

Step 3. After the completion of the search we compute a value supply(p) for each $p \in P$ as follows:

$$\text{supply}(p)=\beta'(p)/\beta_{sum}(p),$$

where

$$\beta_{sum}(p)= \sum_{t \in {}^*p} \beta(t,p),\quad \beta'(p)= \sum_{t \in visit(p)} \beta(t,p),$$

$$visit(p)=\{t \in {}^*p \mid (t,p) \text{ is marked "VISITED"}\}.$$

The value supply(p) intends to represent as a ratio how many tokens, among those that can be brought into p, are produced by firing t_f once.

Step 4. We compute a value rate(t) for each $t \in T$ as follows. Let

$$T'=\{t \in T \mid X_{rest}(t)>0\},$$

and define rate(t) by

$$\text{rate}(t)=\alpha'(t) \cdot d(t)/\alpha_{sum}(t),$$

where

$$\alpha_{sum}(t)= \sum_{p \in {}^*t} \alpha(p,t),\quad \alpha'(t)= \sum_{p \in {}^*t} \alpha(p,t) \cdot \text{supply}(p).$$

The value rate(t) intends to represent as a ratio how many tokens, among those deleted from input places of t, are produced by firing t_f once. Finally define a value effect(t_f) of t_f by

$$\text{effect}(t_f)= \sum_{t \in T} \text{rate}(t).$$

Note that effect(t)=0 if t is not firable on M or if $X_{rest}(t)=0$. We are expecting effect(t_f) to show, as the sum of rate(t), to what extent firing t_f once helps other transitions become firable.

Now we determine priority on transitions. First compute effect(t_f) for every t_f, with $X_{rest}(t_f)>0$, which is firable on M, and then set effect(t)=0 for other transitions t. Define priority on transitions of T according effect(t): those t with larger value of effect(t) get higher priority.

This completes the outline of constructing a new priority-list to be proposed. The formal description of procedure **comp_effect(M)** computing effect(t), $t \in T$, is given in the following.

```
procedure search(t,Mv,state,max);
  begin
  for each p∈t* do
    begin
    if (state(t,p)=UNVISITED) then
      begin
      state(t,p)←VISITED;
      if (max(p)<β(t,p)) then max(p)←β(t,p);
      for each t'∈T do
        begin
        if (state(p,t')=UNVISITED)
           ∧(Mv(p)<α(p,t)≤max(p)+Mv(p))
           ∧(Xrest(t)>0 ) then
           begin
           state(p,t')←VISITED;
           searching(t',Mv,state,max)
           end
        end
      end
```

```
        end
    end;

procedure comp_effect(M);
    begin
1.  for each t∈T do effect(t)←0;
2.  F←{t∈T|X_rest(t) > 0, M(p) ≥ α(p,t) for any p∈*t},
3.  for each t_f∈F do
        begin
4.      for each p∈P do begin max(p)←0; supply(p)←0;
            β_sum(p)←0; β'(p)←0 end;
5.      for each t∈T do
            begin
6.          rate(t)←0; α_sum(t)←0; α'(t)←0;
7.          for each p∈*t do state(p,t)←UNVISITED;
8.          for each p∈t* do state(t,p)←UNVISITED
            end
9.      for each p∈P do
            if (p∈*t_f) then M'(p)←α(p,t) else M'(p)←0;
10.     M_v←M - M' + C+·X_fire;
11.     search(t_f,M_v,state,max);
12.     for each p∈P do
            begin /* computation of supply(p) */
13.         for each t∈*p with X_rest(t)>0 do
                begin
14.             β_sum(p)←β_sum(p)+β(t,p);
15.             if (state(t,p)=VISITED) then
                    β'(p)←β'(p)+β(t,p)
                end;
16.         supply(p)←β'(p)/β_sum(p)
            end;
17.     for each t∈T with X_rest(t)>0 do
            begin /* computation of rate(t) */
18.         for each p∈*t do
                begin
                α_sum(t)←α_sum(t)+α(p,t);
19.             α'(t)←α'(t)+α(p,t)·supply(p)
                end;
20.         rate(t)←α'(t)·d(t)/α_sum(t)
            end;
21.     for each t∈T do effect(t_f)←effect(t_f)+rate(t)
        end
    end;
```

Clearly procedure **comp_effect(M)** runs in $O(|P|+|T|+|E|)$ time, and a new priority-list can be constructed in $O(|T|(|P|+|T|+|E|))$ time, which is $O(|T||E|)$.

6. Experimental evaluation

We experimentally evaluate FM_DPLAM and YW_PLS by comparing with those results by FM_DPLA shown in [27]. All three algorithms FM_DPLA, FM_DPLAM and YW_PLS are implemented on a workstation, DATA GENERAL AV300 (CPU: 88100: 16.7MHz), by means of C language codes. Test nets are generated manually or randomly by the authors. The details are omitted due to shortage of space: see[26,27,30,31].

Delay D(t) of each $t\in T$ is set by one of the following: (a) $D(t)=D(t')$ for $\forall t,t'\in T$ (EQUAL); (b) D(t) is created randomly (RANDOM). X is set to 1. Sizes of Petri nets used as input data in our experimentation are as follows:

$24\le|P|\le191$; $14\le|T|\le94$; $82\le|E|\le516$; $1\le D(t)\le100$.

The total number of test nets we have tried so far is 1800: 50 state machines(sm), 50 marked graphs(mg), and 50 general Petri nets(gn) as underlying Petri nets, two kinds of delays EQUAL and RANDOM ("eq" and "ra" for short, respectively), two combinations of processor pools and the number of processors ($1\le r\le2$; $q_1=3$ if r=1, $q_1=1$ anf $q_2=2$ if r=2), and, three values of k, $k\in\{1,5,10\}$.

Table 1 shows a part of our experimental results for the cases with k=5. The column CT gives average completion time $\tau(\sigma')/k$, where σ' is scheduling with respect to kX. The column CPU denotes CPU time. The column S-b is the Sifakis bound.

Table 2 shows two statistical data for each combination of Petri nets, delays and algorithms, where the data are taken over 200 nets among 300 total test nets and they are fixed. Each integer appearing in upper raw denotes the total number of test nets for each of which feasible scheduling is found by the corresponding algorithm, while each figure shown in the lower raw does the average of the ratio CT(*)/(S-b).

Let CT(YW_PLS) and CPU(YW_PLS) denote completion time and CPU time by YW_PLS, and similarly for others. Concerning average completion time,

CT(YW_PLS)≤CT(FM_DPLAM)≤CT(FM_DPLA)

in general. On the other hand, concerning CPU time, we have

CPU(FM_DPLA)≤CPU(FM_DPLAM),

while CPU(YW_PLS) is better than the others or worse than one or all of them, depending upon test nets. Other statistical data will be given at presentation.

It should be mentioned that there are many test nets, each having an integer k such that $\tau(\sigma')/k<\tau(\sigma)$, where σ (σ', respectively) is a scheduling with respect to X (kX). That is, utilizing cyclic structures in scheduling of timed Petri nets may lead to shorter average completion time. This assures one of advantages of timed Petri nets over task graphs in handling cyclic scheduling. It is also noted that YW_PLS produces an optimum solution to an example (see [27]) whose worst approximation by FM_DPLA or FM_DPLAM cannot be bounded by a constant.

7. Concluding remarks

The followings are left for future research:

(1) incorporating time required by subsequent firing of transitions as the second measure in constructing priority-lists to be used in YW_PLS;

(2) providing more experimental results for $r\ge3$ or $q_i\ge4$;

(3) theoretical evaluation of approximate solutions;

(4) estimating integers k with $\tau(\sigma')/k<\tau(\sigma)$, where σ (σ', respectively) is a scheduling with respect to X (kX).

Acknowledgements

The research of T. Watanabe was partly supported by the Telecommunications Advancement Foundation (TAF), Tokyo, Japan; and is partly supported by The Okawa Institute of Information and Telecommunication, Tokyo, Japan.

References

[1] H.Arai, A.Fujimori and T.Hisamura, Applications of Timed Petri Net to Scheduling Problems of Repetitive Processes and Their Restoring Strategies in Emergency Stops, Trans. Society of Instrument and Control Engineers, 22(1986), 9, pp.955-961. (in Japanese)

[2] J.Campos, G.Chiola and M.Silva, Ergodicity and Throughput Bounds of Petri Nets with Unique Consistent Firing Count Vector, IEEE Trans. Software Engineering, SE-17, 2(1991), pp117-125.

[3] J.Campos, G.Chiola, J.M.Colom and M.Silva, Tight Polynomial

Bounds for Steady-State Performance of Marked Graphs, Proc. 3rd International Workshop on Petri Nets and Performance Models (December 1989), pp.200-209.

[4] E. G. Coffman. Computer and Job-Shop Scheduling Theory, John Wiley & Sons, N. Y. (1976).

[5] F. Commoner A. W. Holt, S. Even and A. Pnueli, Marked Directed Graphs, J. Computer and System Sciences, 5, 5, pp. 511-523(Oct. 1971).

[6] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, Freeman, San Francisco, CA, 1978.

[7] R. L. Graham. Bounds on Multiprocessing Timing Anomalies, SIAM J. Appl. Math., 17, 2(1969), pp.416-429.

[8] R. L. Graham, E. L. Lawler, J. K. Lenstra and A. H. G. Rinnooy Kan, Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey, Annals of Discrete Mathematics 5(1979), pp.287-326.

[9] H. P. Hillion and J. M. Proth, Performance Evaluation of Job-Shop Systems Using Timed Event-Graphs, IEEE Trans. Automatic Control, 34, 1(1989), pp.3-9.

[10] K. Iwano and S.Yeh, An Efficient Algorithm for Optimal Loop Parallelization, Research Report RT 0043, IBM Research, Tokyo Research Laboratory, Chiyoda-Ku, Tokyo, Japan (March, 1990).

[11] H. Kasahara and S. Narita, A Practical Optimal/Approximate Algorithm for Multi-Processor Scheduling Problem, Trans. IEICE, J67-D, 7(1984), pp.792-799. (in Japanese)

[12] J. Magott, Performance Evaluation of Concurrent Systems Using Petri Nets, Information Processing Letters, 18(1984), pp.7-13.

[13] J. Magott, Performance Evaluation of Systems of Cyclic Sequential Processes with Mutual Exclusion Using Petri Nets, Information Processing Letters 21(1985), pp. 229-232.

[14] J. Magott, New NP-Complete Problems in Performance Evaluation of Concurrent Systems Using Petri Nets, IEEE Trans. Software Engineering, SE-13, 5(1987), pp.578-581.

[15] J.Martinez and M.Silva, A Simple and Fast Algorithm to Obtain All Invariants of a Generalized Petri Net, Fachberichte Informatik, Vol.52, (C.Girault and W.Reisig (eds)), Springer Verlag, pp.301-310 (1982).

[16] R. R. Muntz and E. G. Coffman, Jr., Optimal Preemptive Scheduling on Two-Processor Systems, IEEE Trans. on Computers, Vol. C-18, No.11(1969), pp.1014-1020.

[17] I. Nabeshima, Sukejuringu-Riron (Theory of Scheduling), Morikita Shuppan Pub. Co. Ltd., 1974. (in Japanese)

[18] K. Onaga, Scheduling of Extended Marked Graphs, Trans. IEICE, J69-A, 2(1986), pp.241-251. (in Japanese)

[19] K. Onaga, T. Watanabe and M. Silva, On Periodic Schedules for Deterministically Timed Petri Nets Systems, Proc. 4th International Workshop on Petri Nets and Performance Models (PNPM91) (Dec., 1991), pp.210-215.

[20] J. L. Peterson: Petri Net Theory and the Modeling of Systems, Prentice-Hall, Englewood Cliffs, N.J., 1981.

[21] C. V. Ramamoorthy and G. S. Ho, Performance evaluation of asynchronous concurrent systems using Petri nets, IEEE Trans. Software Eng., Vol. 6, No.5(1983), pp.440-449

[22] W. Reisig: Petri Nets/An introduction, Springer-Verlag, Berlin, 1982.

[23] R.Reiter, Scheduling Parallel Computations, J. Assoc. Computing Math., 15, pp.590-599 (1968).

[24] R.Reiter, On Assembly-Line Balancing Problems, Opns. Res., 17, 4, pp.685-700(1969).

[25] J. Sifakis, Modeling and Performance Evaluation of Computer Systems, Proc. Third International Symposium on Measuring, Modeling and Evaluating Computer Systems, (H.Beilner and E.Gelenbe (eds.)), North-Holland, (1977), pp.75-93.

[26] T.Tanida, T.Watanabe, K.Masuoka and K.Onaga, Scheduling in a Timed Petri Net Model of a Repeatedly Executing Set of Tasks — Priority-List Scheduling —, IEICE of Japan, Tech. Rep., COMP92-94, pp.41-48 (March 1992).

[27] T.Tanida, T.Watanabe, M.Yamauchi and K.Onaga, Priority-List Scheduling in Timed Petri Nets, Trans. IEICE of Japan, Vol. E 75, No. 10 pp. 1394-1406.

[28] T. Watanabe, Y. Mizobata and K. Onaga, Minimum initial marking problems of Petri nets, Trans. IEICE of Japan, E72(1989), No. 12, pp.1390-1399.

[29] T. Watanabe, Y. Mizobata and K. Onaga, Time Complexity of Legal Firing Sequence and Related Problems of Petri Nets, Trans. IEICE of Japan, E72(1989), No. 12, pp.1400-1409.

[30] T.Watanabe, T.Tanida, M.Yamauchi and K.Onaga, The Minimum Initial Marking Problem for Scheduling in Timed Petri Nets, Trans, IEICE of Japan, Vol. E 75, No. 10 pp. 1407-1421.

[31] M.Yamauchi, T. Tanida, T. Watanabe and K.Onaga, Scheduling in Timed Petri Net Model of a Repeatedly Executing Set of Tasks — Minimum Initial Marking Problems —, IEICE of Japan, Tech. Rep., COMP91-93, pp.29-40 (March 1992).

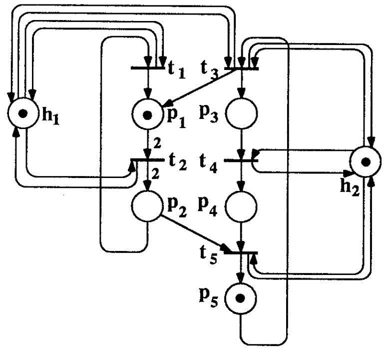

Fig. 1. An example of a (timed) Petri net. This also represents a set of tasks $\{t_1,t_2,t_3,t_4,t_5\}$ of Example 1.

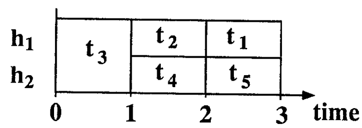

Fig. 2. A Gantt chart for the scheduling σ of Example 1.

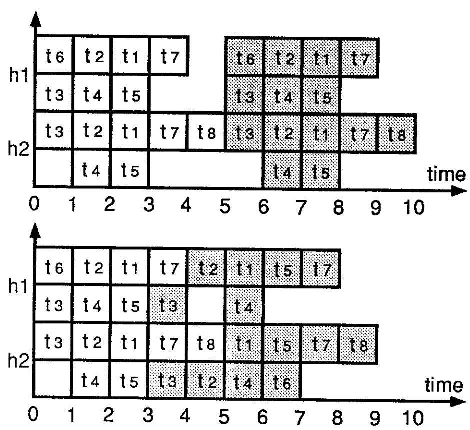

Fig. 3. Gantt charts schematically explaining the situation with $\tau(\sigma')=k\cdot\tau(\sigma)$ and $\tau(\sigma')<k\cdot\tau(\sigma)$. (1) $\tau(\sigma')=k\cdot\tau(\sigma)=10$; (2) $\tau(\sigma')=9<k\cdot\tau(\sigma)=10$, where $\tau(\sigma)=5$ and $k=2$.

DATA				FM_DPLA			FM_DPLAM			YW_PLA			
No.	|P|	|T|	|E|	CT	CPU	ratio	CT	CPU	ratio	CT	CPU	ratio	S-b
sm3-eq	14	16	64	16	55	1.00	16	69	1.00	16	92	1.00	16.00
sm5-eq	13	16	62	15	51	1.00	15	63	1.00	15	67	1.00	15.00
sm7-eq	34	40	158	39	566	1.00	39	772	1.00	39	475	1.00	39.00
sm10-eq	29	40	160	40	545	1.00	40	720	1.00	40	394	1.00	40.00
sm11-eq	28	40	158	39	531	1.00	39	701	1.00	39	455	1.00	39.00
sm12-eq	27	40	160	40	536	1.00	40	701	1.00	40	390	1.00	40.00
sm13-eq	45	54	214	53	1640	1.00	53	2145	1.00	53	918	1.00	53.00
sm16-eq	43	54	216	54	1322	1.00	54	1797	1.00	54	774	1.00	54.00
sm20-eq	58	70	278	28.6	3301	1.24	28.2	4254	1.23	25.4	7094	1.10	23.00
sm21-eq	57	70	280	70	3441	1.00	70	4500	1.00	70	1485	1.00	70.00
sm22-eq	55	70	278	69	3366	1.00	69	4391	1.00	69	1817	1.00	69.00
sm23-eq	55	70	278	37.8	3311	1.10	38	4249	1.10	37.6	6595	1.09	34.50
sm24-eq	53	70	280	28.8	2770	1.23	28.8	3760	1.23	24.6	7736	1.05	23.33
sm25-eq	75	85	340	31.4	6534	1.11	30.4	8281	1.07	30.2	16696	1.07	28.33
sm26-eq	73	85	340	85	6528	1.00	85	8443	1.00	85	2545	1.00	85.00
sm27-eq	72	85	340	45.4	5850	1.07	46	7781	1.08	42.6	14527	1.00	42.50
sm28-eq	70	85	340	32.6	5793	1.15	32.2	7618	1.14	28.4	19457	1.00	28.33
sm29-eq	69	85	340	85	5803	1.00	85	7800	1.00	85	2449	1.00	85.00
sm30-eq	67	85	340	45.4	5733	1.07	46	7542	1.08	42.6	16292	1.00	42.50
sm3-ra	14	16	64	84	56	1.00	84	70	1.00	84	81	1.00	84.00
sm5-ra	13	16	62	87	54	1.00	87	67	1.00	87	91	1.00	87.00
sm7-ra	34	40	158	216	596	1.00	216	799	1.00	216	456	1.00	216.00
sm8-ra	32	40	158	121.6	586	1.09	121.6	779	1.09	115.2	505	1.03	111.50
sm9-ra	31	40	158	118	579	1.01	118	767	1.01	118.8	614	1.02	116.50
sm10-ra	29	40	160	240	579	1.00	240	755	1.00	240	403	1.00	240.00
sm11-ra	28	40	158	234	563	1.00	234	726	1.00	234	449	1.00	234.00
sm12-ra	27	40	160	267	560	1.00	267	723	1.00	267	378	1.00	267.00
sm13-ra	45	54	214	317	1648	1.00	318.2	2138	1.00	317	961	1.00	317.00
sm16-ra	43	54	216	346	1400	1.00	346	1872	1.00	346	816	1.00	346.00
sm19-ra	60	70	280	152.2	3366	1.08	155.2	4673	1.10	148.2	2555	1.05	141.33
sm21-ra	57	70	280	399	3274	1.00	399	4348	1.00	399	1509	1.00	399.00
sm22-ra	55	70	278	394	3408	1.00	394	4424	1.00	394	1601	1.00	394.00
sm24-ra	53	70	280	127	2901	1.08	127	3894	1.08	120.4	3194	1.02	118.00
sm25-ra	75	85	340	182	6400	1.15	178.8	8309	1.13	161.4	6038	1.02	157.67
sm26-ra	73	85	340	448	6396	1.00	448	8350	1.00	448	2494	1.00	448.00
sm27-ra	72	85	340	241	5977	1.08	235.4	8041	1.06	222.8	4714	1.00	222.50
sm28-ra	70	85	340	173	5936	1.16	175	7871	1.17	153.4	7588	1.03	149.33
sm29-ra	69	85	340	419	6016	1.00	419	7865	1.00	419	2515	1.00	419.00
sm30-ra	67	85	340	209.8	5967	1.00	209.8	7823	1.00	210.8	4386	1.01	209.50
mg2-eq	33	30	124	20	305	1.00	20	418	1.00	20	326	1.00	20.00
mg9-eq	79	76	308	41 4	5754	1.09	41.4	7051	1.09	39.8	4805	1.05	38.00
mg10-eq	79	74	304	37.7	5525	1.02	37.2	6748	1.01	37.2	5904	1.01	37.00
mg16-eq	94	91	368	45.6	9855	1.00	45.6	12522	1.00	45.6	10114	1.00	45.50
mg17-eq	94	89	364	46	9650	1.03	46	11627	1.03	45.2	12018	1.02	44.50
mg18-eq	94	87	360	43.6	8952	1.00	43.6	11085	1.00	43.8	12231	1.01	43.50
mg19-eq	94	87	360	44	8917	1.01	43.6	11095	1.00	43.6	12238	1.00	43.50
mg20-eq	94	86	358	44	10112	1.02	43	12304	1.00	43.8	12004	1.02	43.00
mg21-eq	94	84	354	42	14121	1.00	42	16376	1.00	46	9164	1.10	42.00
mg23-eq	94	91	368	47.6	9331	1.05	47.4	12219	1.04	47.6	11974	1.05	45.50
mg24-eq	94	89	364	44.6	9228	1.00	44.6	11750	1.00	44.6	9926	1.00	44.50
mg30-eq	94	91	368	45.6	9400	1.00	47.2	12327	1.04	45.6	10483	1.00	45.50
mg2-ra	33	30	124	105.8	323	1.01	105.8	441	1.01	105.8	286	1.01	105.00
mg9-ra	79	76	308	213.6	4968	1.04	208.6	7108	1.01	208	2715	1.01	206.00
mg10-ra	79	74	304	200.4	4996	1.02	199	6813	1.01	200.2	2665	1.02	196.50
mg16-ra	94	91	368	267.8	9613	1.06	260	12190	1.03	259.3	5108	1.03	252.00
mg17-ra	94	89	364	242.4	8663	1.01	241.6	11697	1.01	241	5753	1.01	239.00
mg18-ra	94	87	360	262.6	9432	1.08	248.6	11739	1.02	250.6	4677	1.03	243.50
mg19-ra	94	87	360	237.4	9362	1.01	241.6	11816	1.03	236.8	5086	1.01	234.50
mg20-ra	94	86	358	241.4	10362	1.05	237.4	12605	1.04	237	5117	1.03	229.00
mg21-ra	94	84	354	234	14531	1.04	234	16737	1.04	234	4157	1.04	225.00
mg23-ra	94	91	368	272.1	9628	1.08	266	12211	1.06	264	4816	1.05	250.50
mg24-ra	94	89	364	268.4	8236	1.11	255	11484	1.05	248.4	4724	1.02	242.50
mg30-ra	94	91	368	263.8	8952	1.08	258.2	12294	1.06	253.8	5057	1.04	244.00
gn7-eq	38	36	150	*	*	*	*	*	*	36	92	1.00	36.00
gn8-eq	37	41	160	*	*	*	*	*	*	24	254	1.00	24.00
gn13-eq	49	54	214	53	488	1.00	53	619	1.00	53	265	1.00	53.00
gn14-eq	49	53	214	27	468	1.02	27	594	1.02	27	749	1.02	26.50
gn18-eq	50	51	210	19	430	1.12	19	541	1.12	20	773	1.18	17.00
gn19-eq	66	70	278	*	*	*	*	*	*	25	2109	1.00	25.00
gn21-eq	65	69	274	68	1050	1.02	*	*	*	*	*	*	66.67
gn22-eq	63	69	276	69	985	1.00	69	1267	*	*	*	*	69.00
gn25-eq	87	79	328	28	1918	1.06	28	2359	1.06	28	2659	1.06	26.33
gn26-eq	87	80	330	80	2283	1.00	80	2755	1.00	80	680	1.00	80.00
gn27-eq	87	80	328	40	2218	1.01	40	2686	1.01	40	2681	1.01	39.50
gn28-eq	86	80	328	27	2243	1.03	27	2716	1.03	27	3294	1.03	26.33
gn7-ra	38	36	150	*	*	*	*	*	*	198	100	*	*
gn8-ra	37	41	160	154	308	1.31	154	365	1.31	156	167	1.32	118.00
gn13-ra	49	54	214	317	468	1.00	317	599	1.00	317	269	1.00	317.00
gn14-ra	49	53	214	157	462	1.01	157	587	1.01	158	394	1.02	155.00
gn16-ra	49	52	210	332	439	1.01	332	562	1.01	330	236	1.00	330.00
gn18-ra	50	51	210	120	445	1 13	120	561	1.13	125	443	1.18	106.33
gn23-ra	63	68	270	183	1012	*	*	*	*	*	*	*	*
gn25-ra	87	79	328	205	1874	1.33	205	2353	1.33	210	1233	1.37	153.67
gn26-ra	87	80	330	447	2291	1.00	447	2831	1.00	447	677	1.00	447.00
gn27-ra	87	80	328	247	2219	1 19	241	2691	1.16	250	1465	1.21	207.00
gn28-ra	86	80	328	164	2313	1.19	*	*	*	187	1772	1.36	137.67

Table 1. A part of our experimental results for the cases with k=5. The three columns FM_DPLA, FM_DPLAM and YW_PLS show the results given by the corresponding algorithms. The column No. denotes the data identification: for example, sm3-eq means data #3 which is a state machine with equal delays on all transitons, and "ra" does RANDOM(delays randomly generated), while "mg" and "gn" denote marked graphs and general Petri nets, respectively. The column CT gives average completion time $\tau(\sigma')/k$ (in the number of time units), where σ' is scheduling with respect to kX. The column CPU denotes CPU time in 1/60 second. The column S-b is the Sifakis bound.

Table 2. Two statistical data for each combination of net structures (sm, mg, gn), generation of delays (eq, ra) and scheduling algorithms (FM_DPLA, FM_DPLAM, YW_PLS), where the data are taken over 200 nets among 300 total test nets and they are fixed. Each integer appearing in upper raw denotes the total number of test nets for each of which feasible scheduling is found by the corresponding algorithm, while each figure shown in the lower raw does the average of the ratio CT(†)/(S-b), where † denotes any one of the three algorithms. * in the table shows that datum is not available.

net	delay	FM_DPLA	FM_DPLAM	YW_PLA
sm	eq	196	196	196
		1.042012	1.041743	1.039423
	ra	196	196	196
		1.042663	1.041709	1.039405
mg	eq	116	116	116
		1.031385	1.029885	1.024109
	ra	116	116	116
		1.041961	1.038714	1.038671
gn	eq	52	53	67
		1.033969	1.033969	1.03098
	ra	40	38	56
		1.130128	1.11886	1.131068