Technical Report #9503

APPLICATION OF TIMED PETRI NETS TO MODELING AND ANALYSIS OF FLEXIBLE MANUFACTURING CELLS

by

W.M. Zuberek

Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada A1B 3X5

June 1995

Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada A1B 3X5
tel: (709) 737–8627
fax: (709) 737–2009

The Natural Sciences and Engineering Research Council of Canada
partially supported this research through Research Grant A8222.

APPLICATION OF TIMED PETRI NETS TO MODELING AND ANALYSIS OF FLEXIBLE MANUFACTURING CELLS

A b s t r a c t

Timed Petri nets are proposed as models of simple and composite schedules for a large class of manufacturing (or robotic) cells. For simple schedules, exactly one part enters and one leaves the cell in each cycle. Net models of simple schedules can easily be derived from the sequences of robot actions. For composite schedules, several parts enter and leave the cell in each cycle. It appears that models of composite schedules can be obtained by composition of simple schedule. A systematic method of deriving all composite schedules is proposed, and decomposition of derived composite schedules into simple ones is presented. It is shown that simple as well as composite schedules can easily be transformed into timed Petri net models. Invariant analysis of timed net models of schedules is used to derive the cycle times of net models. The solutions are obtained in analytical (or symbolic) form, so they are applicable to a wide spectrum of specific cases. Performance characterization (the cycle time or the throughput) obtained in this way can be used for the maximization of the cell's performance. Because the number of different schedules grows very quickly with the number of machines as well as the length of the (composite) schedule, colored Petri nets are proposed for a uniform representation and analysis of entire classes of schedules. Simple examples illustrate the proposed approach for a robotic cell with three machines.

A c k n o w l e d g e m e n t

Collaboration with Dr. W. Kubiak of the School of Business Administration, Memorial University of Newfoundland, is gratefully acknowledged.

INTRODUCTION

In flexible manufacturing systems, machines are often grouped into manufacturing cells (or robotic cells), in which a robot performs sequences of pickup, move, load, unload and drop operations, transporting the manufactured parts from one machine of the cell to another [S3BK92, Cl83]. The throughput of the cell depends on the sequence of robot activities as well as on the sequence in which different parts enter the cell [DH90]. Any approach to maximizing the throughput of a robotic cell must be able to deal efficiently with two issues: how to generate alternative schedules for a given cell, and how to evaluate these schedules. Usually the schedules are represented by models which capture the essential characteristics of the schedule, but which remove all details which are inessential to the evaluation process.

The behavior of manufacturing cells is represented by 'events' and 'activities'; an activity corresponds to an operation performed by a machine or the robot, and an event corresponds to any change of the cell's activities. Different sets of activities determine the 'states' of the system. In each state, several activities can occur concurrently, for example, several machines can perform their operations simultaneously and the robot can also transport a part. Petri nets provide a simple and convenient formalism for modeling systems that exhibit parallelism and concurrency [Mu89, Re85]. In fact, one of the very first applications of Petri net models was to analyze production schemata [Ha72].

In order to study performance aspects of Petri net models, the duration of activities must also be taken into account and included into model specifications. Several types of Petri nets 'with time' have been proposed by assigning 'firing times' to the transitions or places of a net. In timed nets, transition firings are 'real–time' events, i.e., tokens are removed from input places at the beginning of the firing period, and they are deposited to the output places at the end of this period (sometimes this is also called a "three–phase" firing mechanism). The firing times may be either deterministic or stochastic, i.e., described by some probability distribution function. In both cases the concepts of state and state transitions have been formally defined and used in derivation of different performance characteristics of the model [Zu91].

Analysis of net models can be based on their behavior (i.e., the space of reachable states) or on the structure of the net; the former is called reachability analysis while the latter structural analysis. Invariant analysis seems to be the most popular example of the structural approach. Structural methods eliminate the derivation of the state space, so they avoid the 'state explosion' problem of reachability analysis, but they cannot provide as much information as the reachability approach does. Quite often, however, the detailed results of reachability analysis are not really needed, and some more synthetic performance measures, that can be provided by structural methods, are quite satisfactory [Hi89]. In particular, the throughput of a timed net model can easily be determined from the structure of a net if the net can be decomposed into a set of elementary nets [ZK93].

The steady-state behavior of manufacturing cells is considered for two types of schedules, the so called simple schedules in which exactly one (new) part enters the cell and one leaves the cell in each cycle, and composite schedules which deal with several (new) parts in each cycle. In both cases, timed Petri net models are presented, and are solved using the invariant analysis. The solutions are obtained in symbolic form which means that the analysis needs to be performed only once, and then specific values of performance characteristics can easily be

obtained by simply evaluating the symbolic solutions for different sets of parameter values. Examples of simple and composite schedules for a 3–machine cell illustrate the proposed approach.

Throughput optimization is obtained by systematic analysis of different cell's schedules and selection the one, that minimizes the schedule's cycle time. However, the number of possible schedules increases rather quickly with the number of machines as well as with the length of a (composite) schedule. In order to avoid evaluations of large numbers of different schedules, a uniform approach is proposed which uses colored Petri net models of robotic cells.

In colored Petri nets [Je87], information can be associated with individual tokens. These token attributes are called 'token colors'. Token colors can be quite complex, for example, they can describe the values of (simple or structured) variables or the contents of message packages. Token colors can be modified by (firing) transitions and also the conditions enabling transitions can be different for different colors. The attributes attached to tokens result in net models that contain much fewer places and transitions than would be required in 'ordinary' (or non–colored) Petri nets.

The basic idea of colored nets is to 'fold' an ordinary Petri net. The original set of places is partitioned into a set of disjoint classes, and each class is replaced by a single colored place with token colors indicating which of the original places the tokens belong to. Similarly, the original set of transitions is partitioned into a set of disjoint classes, and each class is replaced by a single colored transition with the occurrence colors indicating which of the original transitions the occurrences belong to.

In colored Petri net models of manufacturing cells, colors are used to represent different schedules of the same cell, so, analyses of several schedules can be performed simultaneously.

TIMED PETRI NETS AND NET INVARIANTS

This section recalls basic concepts of Petri nets timed Petri nets. A more detailed discussion can be found elsewhere [Mu89, Re85, Zu91].

A place/transition net $\mathcal{N}$ is a triple $\mathcal{N} = (P, T, A)$ where:

P is a finite, nonempty set of places,

T is a finite, nonempty set of transitions,

A is a set of directed arcs, $A \subseteq P \times T \cup T \times P$, such that for each transition there exists at least one place connected with it.

For each place p (and each transition t) the input set, $Inp(p)$ (or $Inp(t)$), is the set of transitions (or places) connected by directed arcs with p (or t).

A marked Petri net $\mathcal{M}$ is a pair $\mathcal{M} = (\mathcal{N}, m_0)$ where:

$\mathcal{N}$ is a Petri net, $\mathcal{N} = (P, T, A)$,

m_0 is an initial marking function, $m_0 : P \rightarrow \{0, 1, ...\}$ which assigns a (nonnegative) number of tokens to each place of the net.

Let any function $m : P \to \{0, 1, ...\}$ be called a marking in a net $\mathcal{N} = (P, T, A)$.

A transition t is enabled by a marking m iff every input place of this transition contains at least one token. Every transition enabled by a marking m can fire. When a transition fires, a token is removed from each of its input places and a token is added to each of its output places. This determines a new marking in a net, a new set of enabled transitions, and so on. The set of all markings that can be derived from the initial marking is called the set reachable markings.

A place p is shared iff it is an input place for more than one transition. A net is free–choice if the input sets of all transitions sharing the same place are identical. A net is (structurally or statically) conflict–free is it does not contain shared places. It is (dynamically) conflict–free if for any marking in the set of reachable markings, and for any shared place, at most one of transitions sharing this place is enabled. Only conflict–free nets are considered in this report.

A net $\mathcal{N}$ is regular if each transition has the same numbers of incoming and outgoing arcs. Regular net are conservative, i.e., the total number of token in the net is preserved by (any) firing.

A net $\mathcal{N}_i = (P_i, T_i, A_i)$ is a P_i-implied subnet of a net $\mathcal{N} = (P, T, A)$, $P_i \subset P$, iff:

(1) $T_i \;=\; \{t \in T \mid \exists (p \in P_i)\ (p, t) \in A \vee (t, p) \in A\}$,

(2) $A_i \;=\; A \cap (P_i \times T \cup T \times P_i)$.

Each place/transition net $\mathcal{N} = (P, T, A)$ can conveniently be represented by a connectivity (or incidence) matrix $\mathbf{C} : P \times T \to \{-1, 0, +1\}$ in which places correspond to rows, transitions to columns, and the entries are defined as:

$$\forall (p \in P)\ \forall (t \in T)\quad \mathbf{C}[p, t] = \begin{cases} -1, & \text{if } (p, t) \in A \wedge (t, p) \notin A, \\ +1, & \text{if } (t, p) \in A \wedge (p, t) \notin A, \\ 0, & \text{otherwise.} \end{cases}$$

If a marking m_j is obtained from another marking m_i by firing a transition t_k then (in vector notation) $m_j = m_i + \mathbf{C}[k]$, where $\mathbf{C}[k]$ denotes the k-th column of $\mathbf{C}$, i.e., the column representing t_k.

Connectivity matrices disregard 'selfloops', that is, pairs of arcs (p, t) and (t, p); any firing of a transition t cannot change the marking of p in such a selfloop, so selfloops are neutral with respect to token count of a net. A pure net is defined as a net without selfloops [Re85].

A P–invariant (place-invariant) of a net $\mathcal{N}$ is any positive (column) vector I which is a solution of the matrix equation

$$\mathbf{C}^T \times I = 0,$$

where $\mathbf{C}^T$ denotes the transpose of matrix $\mathbf{C}$. It follows immediately from this definition that if I_1 and I_2 are P–invariants of $\mathcal{N}$, then also any linear (positive) combination of I_1 and I_2 is a P–invariant of $\mathcal{N}$.

A basic P–invariant of a net is defined as a P–invariant which does not have simpler invariants. All basic P–invariants I are binary vectors [Re85], $I : P \to \{0, 1\}$.

It should be observed that in a pure net $\mathcal{N}$, each P–invariant I of a net $\mathcal{N}$ determines a P_I-implied (invariant) subnet of $\mathcal{N}$, where $P_I = \{p \in P \mid I(p) > 0\}$ is sometimes called the support of the invariant I; all nonzero elements of I select rows of $\mathbf{C}$, and each selected row i corresponds to a place p_i with all its input (+1) and all output (–1) arcs associated with it.

Finding basic invariants is a 'classical' problem of linear algebra, and there are known algorithms to solve this problem efficiently [KJ87, MS82].

In timed Petri nets each transition takes a 'real time' to fire, i.e., there is a 'firing time' associated with each transition of a net which determines the duration of transition's firings.

A conflict–free timed Petri net $\mathcal{T}$ is a pair $\mathcal{T} = (\mathcal{M}, f)$ where:

$\mathcal{M}$ is a conflict–free marked Petri net, $\mathcal{M} = (\mathcal{N}, m_0)$, $\mathcal{N} = (P, T, A)$,

f is a firing time function which assigns the nonnegative (average) firing time $f(t)$ to each transition t of the net, $f : T \to \mathbf{R}^{\oplus}$, and $\mathbf{R}^{\oplus}$ denotes the set of nonnegative real numbers.

The behavior of a timed Petri net can be represented by a sequence of 'states' where each 'state' describes the distribution of tokens in places and firing transitions of the net; detailed definitions of states and state transitions are given in [Zu91]. The states and state transitions can be combined into a graph of reachable states; this graph is a semi–Markov process defined by the timed net $\mathcal{T}$. For regular timed conflict–free nets, the reachability graphs are simple cycles which represent the cyclic behavior of such nets. Each such timed Petri net contains a basic invariant subnet with the cycle time equal to the cycle time of the whole net. Moreover, all other basic invariant subnets have cycle times which are not greater than the cycle time of the net, the cycle time of the net is thus equal to the maximum cycle time if its basic invariant subnets.

SIMPLE SCHEDULES

For simple schedules of robotic cells, exactly one part enters and one leaves the cell in each cycle (although the part which leaves the cell may not be the same as the one which enters the cell). It is known [S3BK92] that for a cell with m machines there are $m!$ different simple schedules. For $m = 3$ (Fig.1 shows a sketch of a 3–machine cell), there are six simple schedules, denoted here as A, B, C, D, E and F.

Assuming, for simplicity, that each part follows the same path from the input (In) to machine–1 (M_1), to machine–2 (M_2), to machine–3 (M_3), and finally to the output of the cell (Out), the simple schedules can be described by the following sequences of cell configurations, where each configuration corresponds to a distribution of parts among the machines of the cell (when the robot does not carry a part); more specifically, each configuration is described by an m–tuple of machine descriptions:

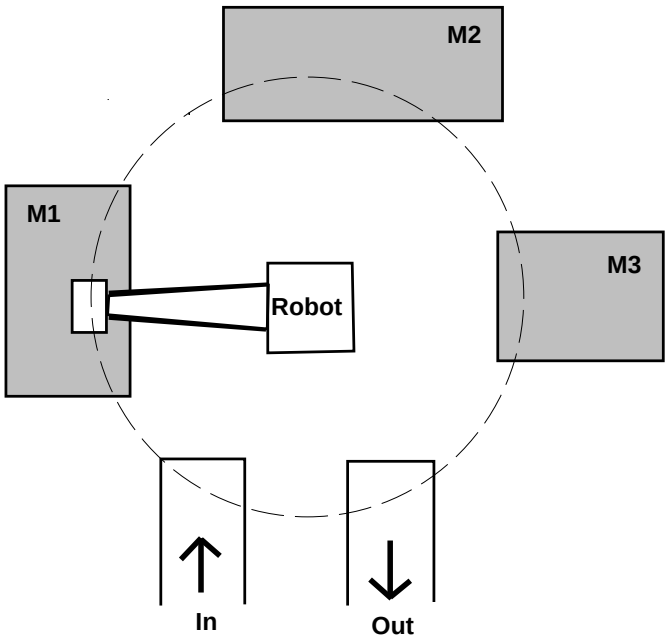


Fig.1. Layout of a three–machine cell.

$$(k_1, k_2, ..., k_m)$$

where each machine description k_i is "1" if the machine M_i is loaded with a part in this configuration, and otherwise is "0" (in the case of multiple machines performing exactly the same operations, the values describing each multi–machine station would assume the values from "0" to "n" where n is the number of identical machines):

A: $(0,0,0) \rightarrow (1,0,0) \rightarrow (0,1,0) \rightarrow (0,0,1) \rightarrow (0,0,0)$
B: $(0,0,1) \rightarrow (1,0,1) \rightarrow (0,1,1) \rightarrow (0,1,0) \rightarrow (0,0,1)$
C: $(0,0,1) \rightarrow (1,0,1) \rightarrow (1,0,0) \rightarrow (0,1,0) \rightarrow (0,0,1)$
D: $(0,1,0) \rightarrow (1,1,0) \rightarrow (1,0,1) \rightarrow (1,0,0) \rightarrow (0,1,0)$
E: $(0,1,0) \rightarrow (1,1,0) \rightarrow (1,0,1) \rightarrow (1,0,0) \rightarrow (0,1,0)$
F: $(0,1,1) \rightarrow (1,1,1) \rightarrow (1,1,0) \rightarrow (1,0,1) \rightarrow (0,1,1)$

Each change of configurations corresponds to a part moving from one machine to another, from the input to the first machine, or from the last machine to the output; all schedules uniformly begin by moving a (new) part from the input to the first machine.

The simple schedules can be generated systematically by applying the following rules, describing all possible "passages" of parts through the cell:

- a configuration $(k_1, ..., k_i, k_{i+1}, ..., k_m)$ derives a configuration $(k_1, ..., k_i - 1, k_{i+1} + 1, ...k_m)$ if and only if the value of k_i is "1" and the value of k_{i+1} is "0", $i = 1, ..., m-1$;

- a configuration $(k_1, k_2, ..., 1)$ always derives a configuration $(k_1, k_2, ..., 0)$ (this derivation corresponds to moving a part from the last machine M_m to the output of the cell),
- it is assumed that each schedule begins by moving a (new) part from the input to the machine M_1, so the first derivation is always from $(0, k_2, ...k_m)$ to $(1, k_2, ..., k_m)$,
- for a cell with m machines, the length of all simple schedules is equal to $m + 1$ (it corresponds to a passage of a part, although not necessarily the same, from the input, through all machines of the cell, to the output).

The six simple schedules of a 3–machine cell correspond to all possible derivations of configurations described by the above rules, applied to four different initial configurations of the cell:

$$(0,0,0) \rightarrow (1,0,0) \rightarrow (0,1,0) \rightarrow (0,0,1) \rightarrow (0,0,0) \text{ – schedule A}$$
$$(0,0,1) \rightarrow (1,0,1) \rightarrow \begin{cases} (0,1,1) \rightarrow (0,1,0) \rightarrow (0,0,1) \text{ – schedule B} \\ (1,0,0) \rightarrow (0,1,0) \rightarrow (0,0,1) \text{ – schedule C} \end{cases}$$
$$(0,1,0) \rightarrow (1,1,0) \rightarrow (1,0,1) \rightarrow \begin{cases} (0,1,1) \rightarrow (0,1,0) \text{ – schedule D} \\ (1,0,0) \rightarrow (0,1,0) \text{ – schedule E} \end{cases}$$
$$(0,1,1) \rightarrow (1,1,1) \rightarrow (1,1,0) \rightarrow (1,0,1) \rightarrow (0,1,1) \text{ – schedule F}$$

Since parts are transported from one machine (or input) to another (or output) by the robot, the sequences of robot's actions can easily be derived from the sequences of configurations by "implementing" the moves of parts corresponding to changes of consecutive configurations. For example, schedule A begins be transporting a part from the input to M_1 and loading it; when the first operation is finished, the robot unloads M_1, moves the part to M_2 and loads it there, and so on. The sequences of robot actions are as follows (the robot moves from X to Y are denoted by $X \Rightarrow Y$ if the robot carries a part and by $X \rightarrow Y$ otherwise):

A: $In \Rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In$
B: $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow In$
C: $In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In$
D: $In \Rightarrow M_1 \rightarrow M_2 \Rightarrow M_3 \rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow In$
E: $In \Rightarrow M_1 \rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \rightarrow In$
F: $In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow M_1 \Rightarrow M_2 \rightarrow In$

Timed Petri net models of simple schedules can easily be derived from the sequences of robot operations. In timed models, net transitions represent (machine and robot) operations while net places represent 'conditions' (in the most general sense). A Petri net model of schedule A is shown in Fig.2. The three machines of Fig.1 (or rather machine operations) are represented by t_1, t_2 and t_3, each of these transition with its input and output place (for 'part loaded' and 'machine operation finished' conditions). The 'firing times' associated with these transitions, $f(t_1) = o_1$, $f(t_2) = o_2$ and $f(t_3) = o_3$, represent the (average) times of performing the operations on machines M_1, M_2 and M_3, respectively. It is assumed that there is always an available part in In and that Out removes manufactured parts sufficiently

quickly, so In and Out are not actually shown although they can easily be added to the model.

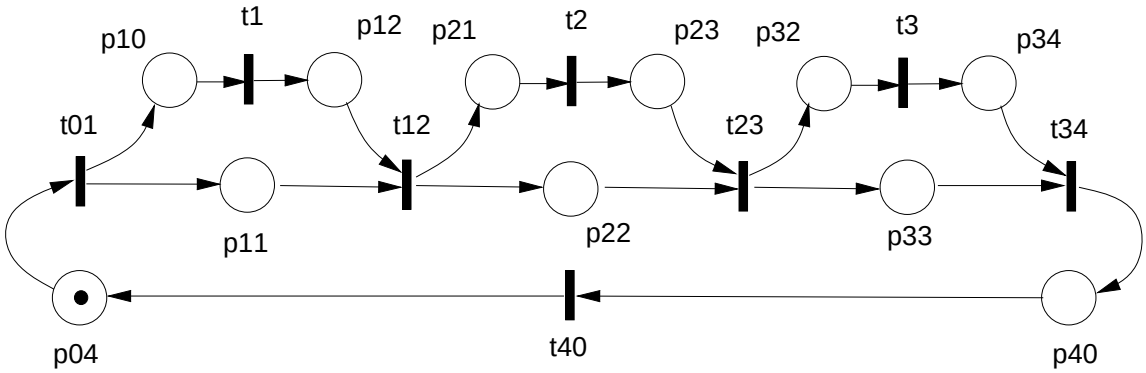


Fig.2. Petri net model of schedule A.

The operations of the robot are represented by the cycle t_{01}, p_{11}, t_{12}, p_{22}, t_{23}, p_{33}, t_{34}, p_{40}, t_{40}, p_{04} and t_{01}, that models the sequence of consecutive steps of the schedule. The 'interpretation' of the transitions is as follows:

	robot operations	*execution time*
t_{01}	pick a part from In, move to M_1 and load	$u + w + y$
t_{12}	unload M_1, move to M_2 and load	$v + w + y$
t_{23}	unload M_2, move to M_3 and load	$v + w + y$
t_{34}	unload M_3, move to Out and drop	$v + x + y$
t_{40}	move from Out to In	y

where the 'execution times' (or firing times of transitions) are given assuming that:

u denotes the (average) pickup time,
v – the (average) unload time,
w – the (average) load time,
x – the (average) drop time and
y – the average 'travel' time between two adjacent machines (assuming, for simplicity, that this time is the same for all adjacent machines, and also the same for M_3 to Out, Out to In and In to M_1 moves).

A Petri net model of schedule E is shown in Fig.3, in which t_1, t_2 and t_3 represent the machine operations, as in Fig.2, and the remaining transitions correspond to robot actions:

	robot operations	*time*
t_{01}	pick a part from In, move to M_1 and load	$u + w + y$
t_{12}	unload M_1, move to M_2 and load	$v + w + y$
t_{20}	move from M_2 to In	$2y$
t_{21}	move from M_1 to M_2	y
t_{23}	unload M_2, move to M_3 and load	$v + w + y$
t_{34}	unload M_3, move to Out and drop	$v + x + y$
t_{41}	move from Out to M_1	$2y$

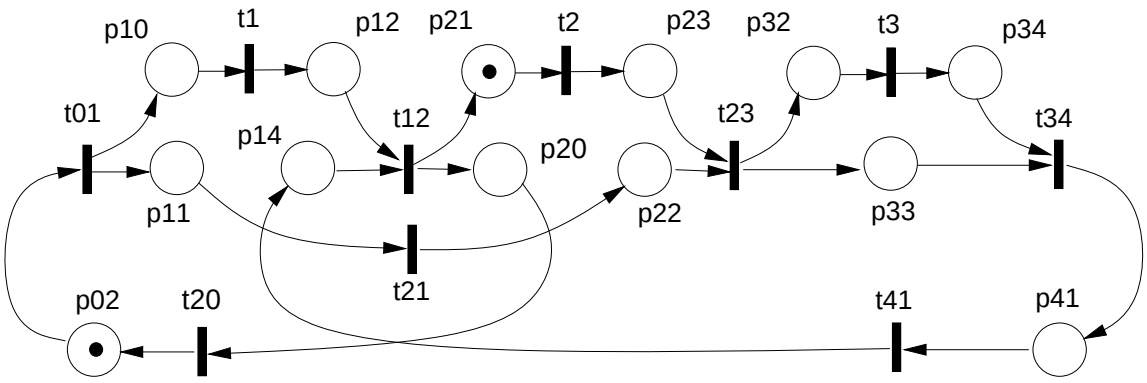


Fig.3. Petri net model of schedule E.

Evaluation of net models using net invariants is described in [ZK93]. It appears that the models of simple schedules have only a few invariants subnets, and this invariant subnets determine the performance of the model. Symbolic formulas for performance characteristics can easily be derived using the invariant analysis. For example, the net shown in Fig.3 (schedule E) has 5 place invariants:

P–invariant	p_{10}	p_{12}	p_{21}	p_{23}	p_{32}	p_{34}	p_{41}	p_{14}	p_{20}	p_{02}	p_{11}	p_{22}	p_{33}
1	1	1	0	0	0	0	0	0	1	1	0	0	0
2	0	0	1	1	1	1	1	1	0	0	0	0	0
3	0	0	1	1	0	0	1	1	0	0	0	0	1
4	0	0	0	0	1	1	1	1	1	1	1	1	0
5	0	0	0	0	0	0	1	1	1	1	1	1	1

which imply the following subnets (see Fig.3):

P–invariant	t_1	t_2	t_3	t_{01}	t_{12}	t_{20}	t_{21}	t_{23}	t_{34}	t_{41}
1	1	0	0	1	1	1	0	0	0	0
2	0	1	1	0	1	0	0	1	1	1
3	0	1	0	0	1	0	0	1	1	1
4	0	0	1	1	1	1	1	1	1	1
5	0	0	0	1	1	1	1	1	1	1

It should be observed that the set of transitions of invariant (3) is a subset of that of (2), and that the set of transitions of invariant (5) is a subset of that of (4). Consequently, the cycle time for this schedule is determined by the maximum cycle time of subnets (1), (2) and (4):

$$\tau_0 \;=\; \max(\tau_1, \tau_2, \tau_4)$$

where:

$$\begin{aligned} \tau_1 &= o_1 + u + v + 2w + 4y, \\ \tau_2 &= o_2 + o_3 + 3v + 2w + x + 5y, \\ \tau_4 &= o_3 + u + 3v + 3w + x + 9y \end{aligned}$$

A similar analysis can be repeated for other simple schedules.

COMPOSITE SCHEDULES

For composite schedules, several parts enter and leave the cell in each cycle. Models of composite schedules can be regarded as an interleaved composition of simple schedules, corresponding to processing of consecutive parts which enter the cell within one cycle and move from one machine to another in consecutive steps of the schedule. A systematic generation of composite schedules can be obtained by a simple extension of the procedure used for generation of simple schedules. Since each composite schedule, which processes n parts withing a single cycle (so it is called an n–schedule for simplicity), may have similar part distributions at different stages of the schedule, a notational extension is needs to uniquely identify all the steps of the schedule. A simple solution is to introduce, in the description of configurations, a hypothetical "input container", which for each n–schedule initially contains exactly n parts, and which must be emptied during the schedule. This container can be represented by an additional element of configurations, for example, the initial element, separated from the remaining machine descriptions by a colon (instead of a comma). So a typical configuration for description of composite schedules of an m–machine cell is:

$$(k_0 : k_1, k_2, ..., k_m)$$

The (revised) rules describing changes of configurations are:

- a configuration $(k_0 : k_1, ..., k_i, k_{i+1}, ..., k_m)$ derives a configuration $(k_0 : k_1, ..., k_i - 1, k_{i+1} + 1, ...k_m)$ if and only if the value of k_i is "1" and the value of k_{i+1} is "0", $k = 1, ..., m - 1$;

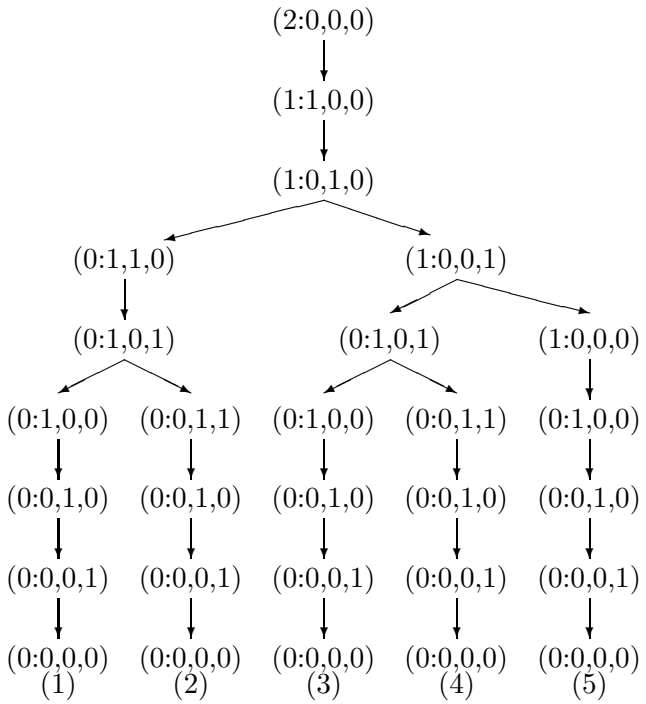
- a configuration $(k_0 : k_1, k_2, ..., 1)$ always derives a configuration $(k_0 : k_1, k_2, ..., 0)$ (this derivation corresponds to moving a part from the last machine M_m to the output of a cell),

- a configuration $(k_0 : 0, k_2, ..., k_m)$ derives a configuration $(k_0 - 1 : 1, k_2, ..., k_m)$ if and only if the value of k_0 is greater than 0 (this derivation corresponds to moving a part from the input to the first machine M_1),

- it is assumed that each schedule begins by moving a (new) part from the input to the machine M_1, so the first derivation is always $(k_0 : 0, k_2, ...k_m)$ to $(k_0 - 1 : 1, k_2, ..., k_m)$,

- for a cell with m machines, the length of all n–schedules is equal to $n * (m + 1)$.

For a 3–machine cell, there are 34 different 2–schedules, including 6 schedules which are just simple schedules repeated twice. All 34 2–schedules can be systematically derived by repeatedly applying the rules to the four initial configurations of the cell. For the initial configuration (0,0,0), there are five 2–schedules:



Each of these schedules can be decomposed into a pair of interleaved simple schedules, for example, schedule (1) is composed of simple schedules A and E:

schedule A		*schedule E*
(0,0,0)		
(1,0,0)		
(0,1,0)	→	(0,1,0)
		(1,1,0)
		(1,0,1)
		(1,0,0)
(0,1,0)	←	(0,1,0)
(0,0,1)		
(0,0,0)		

schedule (4) is a composition of A and B:

schedule A		*schedule B*
(0,0,0)		
(1,0,0)		
(0,1,0)		
(0,0,1)	→	(0,0,1)
		(1,0,1)
		(0,1,1)
		(0,1,0)
(0,0,1)	←	(0,0,1)
(0,0,0)		

and schedule (5) is simply a composition of A with itself.

All these schedules can easily be translated into sequences of robot actions (as before, the robot moves from X to Y are denoted by $X \Rightarrow Y$ if the robot carries a part and by $X \rightarrow Y$ otherwise):

(1): $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow In \Rightarrow M_1 \rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In$

(2): $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow In \Rightarrow M_1 \rightarrow M_2 \Rightarrow M_3 \rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In$

(3): $In \Rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In$

(4): $In \Rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In$

(5): $In \Rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In \Rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In$

Timed Petri net models can easily be derived from the sequences of robot's action. For example, the net model for schedule (1), i.e., A+E, is shown in Fig.4.

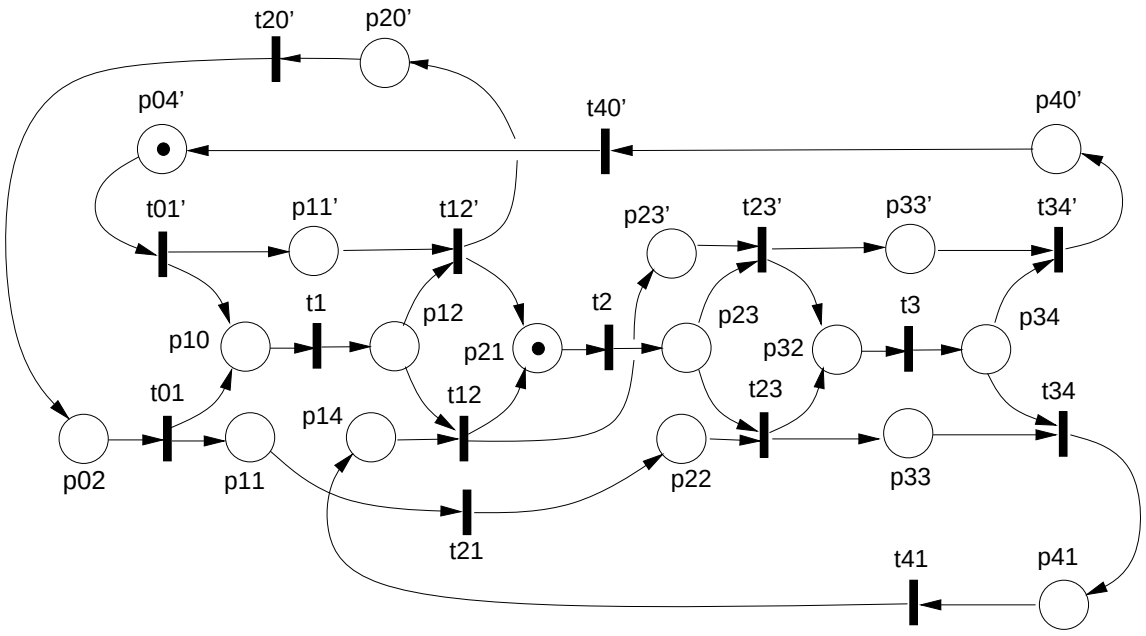


Fig.4. Petri net model of schedule (1)=(A+E).

The transitions correspond to the following actions for the A and E parts of the schedule:

	robot operations	*execution time*
t'_{01}	pick a part from In, move to M_1 and load	$u+w+y$
t'_{12}	unload M_1, move to M_2 and load	$v+w+y$
t'_{20}	move from M_2 to In	$2y$
t'_{23}	unload M_2, move to M_3 and load	$v+w+y$
t'_{34}	unload M_3, move to Out and drop	$v+x+y$
t'_{40}	move from Out to In	y
t_{01}	pick a part from In, move to M_1 and load	$u+w+y$
t_{12}	unload M_1, move to M_2 and load	$v+w+y$
t_{21}	move from M_1 to M_2	y
t_{23}	unload M_2, move to M_3 and load	$v+w+y$
t_{34}	unload M_3, move to Out and drop	$v+x+y$
t_{41}	move from Out to M_1	$2y$

For composite schedules, the cycle times of schedules can be determined in a similar way as for the simple schedules. The net shown in Fig.4 has five P–invariants which imply the following subsets of transitions (all entries equal to 2 correspond to P_i–implied subnets in which the corresponding transitions are implied twice; the implied subnets are free–choice nets [ZK93]):

P–invariant	t_1	t_2	t_3	t'_{01}	t'_{12}	t'_{20}	t'_{23}	t'_{34}	t'_{40}	t_{01}	t_{12}	t_{21}	t_{23}	t_{34}	t_{40}
1	2	0	0	1	1	1	1	1	1	1	1	0	0	0	0
2	0	2	2	1	1	0	1	1	1	0	1	0	1	1	1
3	0	2	0	1	1	0	1	1	1	0	1	0	1	1	1
4	0	0	2	1	1	1	1	1	1	1	1	1	1	1	1
5	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

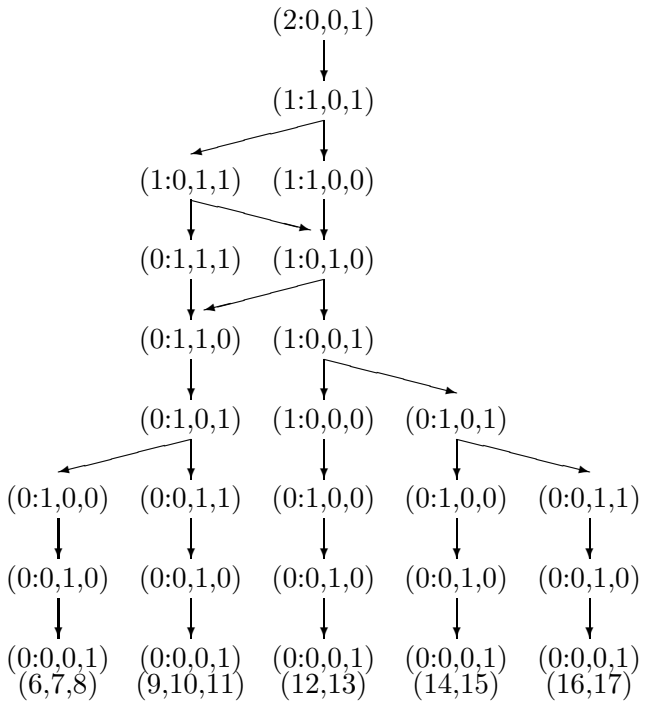
Since, again, the set of transitions of the P–invariant (3) is a subset of that of (2), and the set of transitions of (5) is a subset of that of (4), the cycle time for this schedule is:

$$\tau_0 \;=\; \max(\tau_1, \tau_2, \tau_4)$$

where:

$$\begin{aligned} \tau_1 &= 2o_1 + 2u + 4v + 5w + 9y, \\ \tau_2 &= 2o_2 + 2o_3 + u + 6v + 5w + 2x + 10y, \\ \tau_4 &= 2(o_3 + u + 3v + 3w + x + 7y) \end{aligned}$$

For the initial cell configuration (0,0,1), there are 12 different 2–schedules



and their corresponding robot's sequences of actions are:

(6): $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(7): $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow In \Rightarrow M_1 \rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(8): $In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \rightarrow In \Rightarrow M_1 \rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(9): $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(10): $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow In \Rightarrow M_1 \rightarrow M_2 \Rightarrow M_3 \rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(11): $In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \rightarrow In \Rightarrow M_1 \rightarrow M_2 \Rightarrow M_3 \rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(12): $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In \Rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(13): $In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In \Rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(14): $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(15): $In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(16): $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(17): $In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow In$

Timed Petri net models can be derived for all these schedules in the same way as before. For example, schedule (9) is a composition of simple schedules B and F:

schedule B		*schedule F*
(0,0,1)		
(1,0,1)		
(0,1,1)		
(0,1,1)	→	(0,1,1)
		(1,1,1)
		(1,1,0)
		(1,0,1)
(0,1,1)	←	(0,1,1)
(0,1,0)		
(0,0,1)		

and its Petri net model is shown in Fig.5. The description of transitions is as follows:

	robot operations	*execution time*
t'_{01}	pick a part from In, move to M_1 and load	$u+w+y$
t'_{12}	unload M_1, move to M_2 and load	$v+w+y$
t'_{20}	move from M_2 to In	$2y$
t'_{23}	unload M_2, move to M_3 and load	$v+w+y$
t'_{30}	move from M_3 to In	$2y$
t'_{34}	unload M_3, move to Out and drop	$v+x+y$
t'_{42}	move from Out to M_2	$2y$

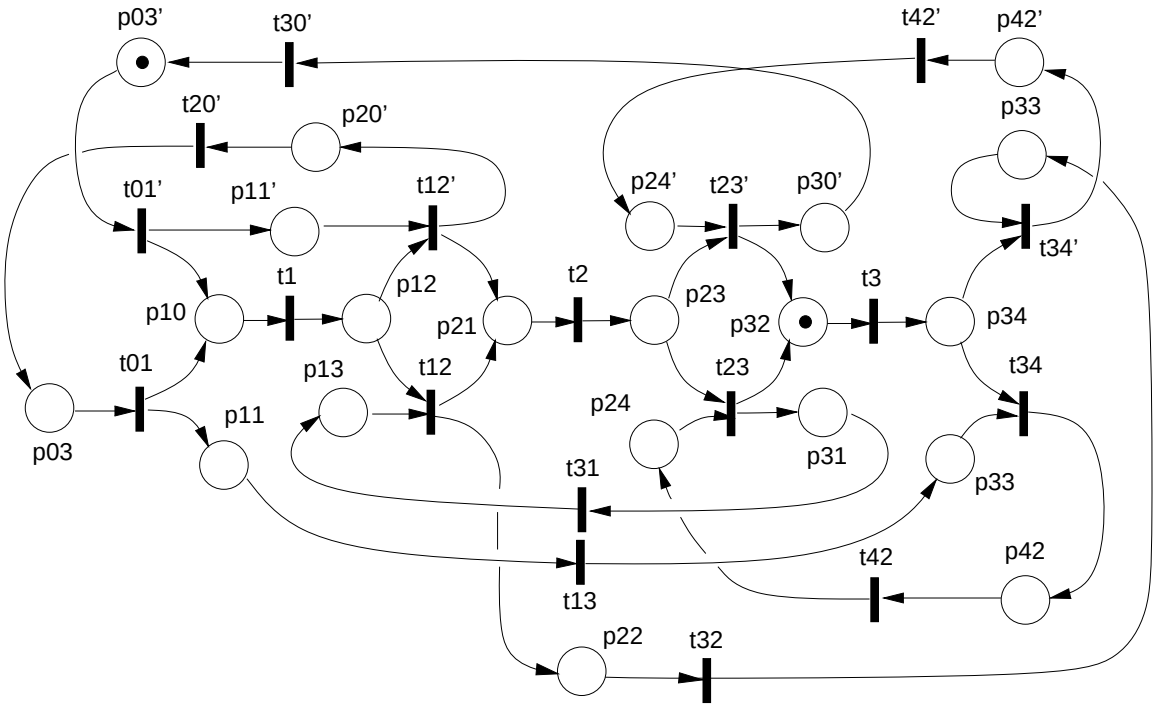


Fig.5. Petri net model of schedule (9)=(B+F).

	robot operations	*execution time*
t_{01}	pick a part from In, move to M_1 and load	$u+w+y$
t_{12}	unload M_1, move to M_2 and load	$v+w+y$
t_{13}	move from M_1 to M_3	$2y$
t_{23}	unload M_2, move to M_3 and load	$v+w+y$
t_{31}	move from M_3 to M_1	y
t_{32}	move from M_2 to M_3	y
t_{34}	unload M_3, move to Out and drop	$v+x+y$
t_{42}	move from Out to M_2	$2y$

The net shown in Fig.5 (composite schedule B+F) has 4 P–invariants which imply the following subnets:

P–invariant	t_1	t_2	t_3	t'_{01}	t'_{12}	t'_{20}	t'_{23}	t'_{30}	t'_{34}	t_{42}	t_{01}	t'_{12}	t'_{13}	t'_{23}	t_{31}	t_{32}	t_{34}	t_{42}
1	2	0	0	1	1	1	1	1	1	1	1	1	0	0	0	1	0	0
2	0	2	0	1	1	0	1	1	0	0	0	1	0	1	1	0	0	0
3	0	0	2	0	0	0	1	0	1	1	0	0	1	1	0	0	1	1
4	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

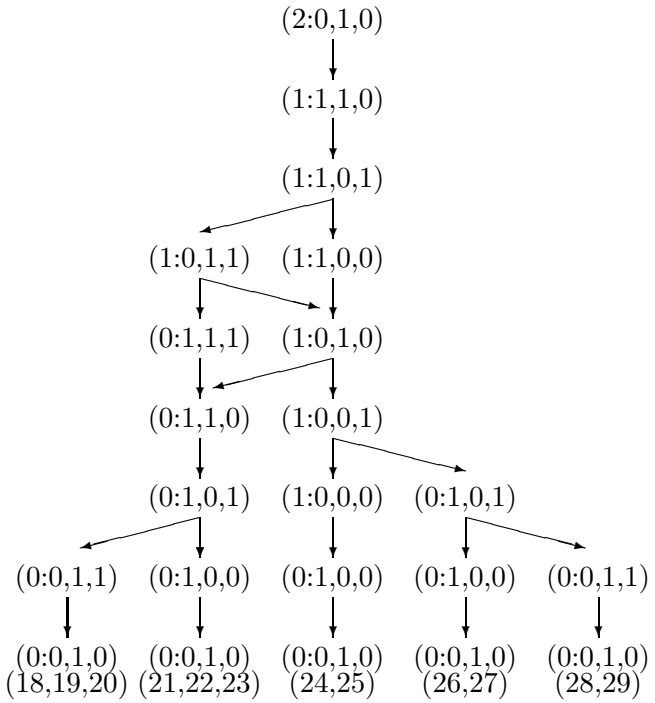
The cycle time is:

$$\tau_0 \;=\; \max(\tau_1, \tau_2, \tau_3, \tau_4)$$

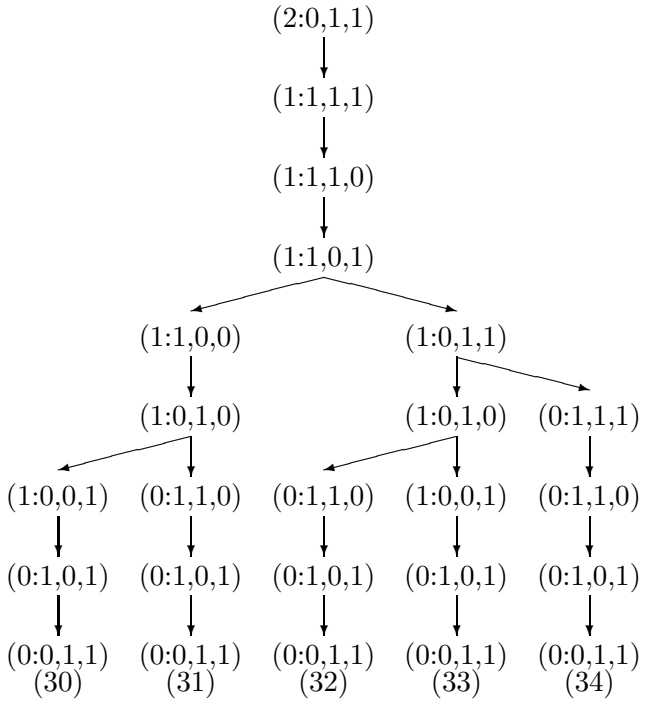
where:

$$\begin{aligned}
\tau_1 &= 2o_1 + 2u + 4v + 5w + x + 13y, \\
\tau_2 &= 2o_2 + u + 4v + 5w + 9y, \\
\tau_3 &= 2o_3 + 4v + 2w + 2x + 6y, \\
\tau_4 &= 2u + 6v + 6w + 2x + 21y)
\end{aligned}$$

There are 12 schedules for the initial configuration (0,1,0):



and another 5 schedules for the configuration (0,1,1):



All these schedules can be evaluated using net invariants, and the schedule with the minimal cycle time is the optimal 2–schedule.

COLORED PETRI NETS

Colored Petri nets and timed colored Petri nets can be considered as an extension of nets without colors by introducing token attributes called colors. Enabled transitions can fire for different combinations of token colors (called occurrences of transitions), and firing transitions can change the colors associated with tokens.

A convenient concept of 'multisets' is frequently used in the definitions which follow. A multiset (or a bag) X over a (nonempty) set A is any function $A \to \mathbf{N}$, $\mathbf{N} = \{0, 1, 2, ...\}$. Intuitively, a multiset is a 'set' which can contain multiple occurrences of the same elements; if X is a multiset over A, then for each $a \in A$, $X(a)$ denotes the number of occurrences of a in X.

A colored Petri net is a 5–tuple $\mathcal{N} = (P, T, A, C, w)$ where:

P is a finite (nonempty) set of places,

T is a finite (nonempty) set of transitions,

A is a (nonempty) set of directed arcs which connect places with transitions and transitions with places, $A \subseteq P \times T \cup T \times P$, such that there are no isolated places or transition; moreover, for each $t \in T$, $Inp(t)$ denotes the set of places which are connected by arcs directed to t, and $Out(t)$ the set of places which are connected by arcs directed from t; $Inp(p)$ and $Out(p)$ are defined similarly,

C is a finite (nonempty) set of colors,

w is the arc function which associates, with each arc of the net, a function from the set of (occurrence) colors into a multiset of token colors, i.e., for each arc $a \in A$, $a = (p,t)$ or $a = (t,p)$, $w(a) : C \rightarrow C \rightarrow \mathbf{N}$.

This definition is a slightly modified version of a colored Petri net matrix [Je87]; the modification is made in order to simplify the definition, and to emphasize the relationship between colored nets and 'ordinary' nets (i.e., non–colored nets [Re85]). It should be observed that ordinary nets correspond to such colored nets in which: (i) the set of colors C contains just one color, and (ii) the arc function assigns weights equal to 1 to all arcs of the net.

A marked colored Petri net is a pair $\mathcal{M} = (\mathcal{N}, m_0)$ where:

$\mathcal{N}$ is a colored Petri net, $\mathcal{N} = (P, T, A, C, w)$,

m_0 is the initial marking function which assigns multisets of token colors (or colored tokens) to places of a net, $m_0 : P \rightarrow C \rightarrow \mathbf{N}$.

Let any function m that maps P into multisets of token colors, $m : P \rightarrow C \rightarrow \mathbf{N}$, be called a marking of the net $\mathcal{N}$.

An occurrence o (or occurrence color) of a transition $t \in T$, $o \in C$, is enabled at the marking m if and only if

$$\forall (p \in Inp(t))\ w(p,t)(o) \leq m(p),$$

where $w(p,t)(o)$ denotes the application of the arc function w of the arc (p,t) to the occurrence color o, and $\leq$ denotes element–wise comparison of multisets.

An occurrence o of the transition t enabled at the marking m_i can fire; firing the occurrence o of t transforms m_i into another marking m_j which is directly (o,t)–reachable (i.e., reachable in 'one step') from m_i

$$\forall (p \in P)\ m_j(p) = m_i(p) - \sum_{t \in Out(p)} w(p,t)(o) + \sum_{t \in Inp(p)} w(t,p)(o)$$

where $\sum$ is used for element–wise addition of multisets. During o of t's firing, tokens are removed from t's input places in numbers corresponding to the (input) arc functions applied to o, and tokens are added to t's output places in numbers corresponding to the (output) arc functions applied to o.

A colored net is *conflict–free* iff no two enabled occurrences share the same place color, i.e., iff

$$\forall (t_i, t_j \in T)\ \forall (o_\ell, o_k \in C)\ p \in Inp(t_i) \cap Inp(t_j)\ \Rightarrow \\ \forall (c \in C)\ w(p,t_i)(o_\ell)(c) * w(p,t_j)(o_k)(c) = 0.$$

Only conflict–free colored nets are considered here.

In timed colored nets, a 'firing time' is associated with each occurrence color of each transition. This firing time may be deterministic or it can be a random variable with some

distribution function, for example, negative exponential distribution. Only deterministic firing times are considered in this report.

A conflict–free timed colored net is a pair, $\mathcal{T} = (\mathcal{M}, f)$ where

$\mathcal{M}$ is a conflict–free marked colored net, $\mathcal{M} = (\mathcal{N}, m_0, \mathcal{N} = (P, T, A, C, w)$,

f is a firing–time function which assigns a nonnegative firing time to each occurrence of each transition of the net, $f : T \to C \to \mathbf{R}^+$, where $\mathbf{R}^+$ denotes the set of nonnegative real numbers.

The behavior of a timed colored net can be described by a sequence of states and state transitions [Zu90]. Any state description of a timed net must take into account the marking of a net (i.e., the distribution of token colors in places) as well as the distribution of occurrence colors in firing transitions.

The arc functions are mappings $C \to C \to \mathbf{N}$, or $(C \times C) \to \mathbf{N}$; the second form can be represented quite conveniently by a rectangular array indexed by occurrence colors (columns) and token colors (rows). The incidence (or connectivity) matrix of a colored net is a $k \times \ell$ matrix A, where k is the number of places and ℓ the number of transitions, and:

$$A[k, \ell] \;=\; w(t_k, p_\ell) - w(p_k, t_\ell)$$

i.e., each element of A is a $C \times C \to \mathbf{N}$ mapping, and the difference is componentwise (for each occurrence color and each token color), so if the arc functions $w(p, t)$ and $w(t, p)$ are represented as $(C \times C)$ matrices of elements of $\mathbf{N}$, the difference is in the sense of the corresponding elements of these matrices.

A place invariant of a colored net is a k–element (column) vector of multisets $C \to \mathbf{N}$ such that

$$A^T * I \;=\; 0$$

where A^T is the transpose of A, and the operation "$*$" is a componentwise application of elements of A to multisets of I, and "0" is the ℓ–element vector of zero multisets $C \to \{0\}$:

$$\forall(1 \le j \le \ell) \sum_{1 \le i \le k} A[i, j](I[i]) \;=\; 0$$

and the sum is performed componentwise (on multisets):

$$\forall(1 \le j \le \ell)\ \forall(c_t \in C)\ \forall(c_p \in C) \sum_{1 \le i \le k} A[i, j][c_p, c_t] * I[i](c_p) \;=\; 0$$

A colored net $\mathcal{M} = ((P, T, A, C, w), m_0)$ is *decoupled* iff the occurrences of transitions do not 'mix' the colors, i.e., iff there is a partition $\mathcal{P}(C)$ of the set of colors C (and an implied equivalence relation H_{eq} on C), such that all occurrences of transitions have their (nonzero) input and output arc mappings in the same equivalence classes of $\mathcal{P}$:

$$\exists(H_{eq} \subset C \times C)\ \forall(t \in T)\ \forall(o \in C)\ \forall(p_i \in Inp(t))\forall(p_j \in Out(t))\ \forall(c_\ell, c_k \in C)$$
$$w(p_i,t)(o)(c_\ell) > 0\ \wedge\ w(t,p_j)(o)(c_k) > 0\ \Rightarrow\ (c_\ell, c_k) \in H_{eq}.$$

It should be observed that if a net is decoupled, it can be analyzed independently for each equivalent class of colors because different classes of colors never interfere one with another.

A colored net model of all simple schedules for a 3–machine cell is shown in Fig.6.

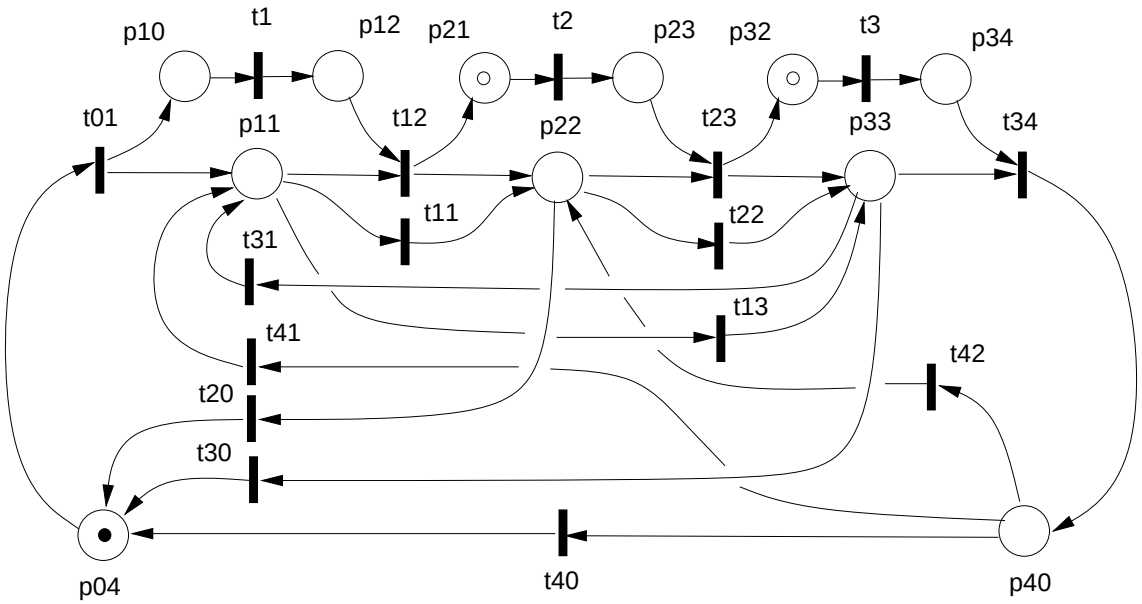


Fig.6. Colored net model of a three–machine cell.

It can be observed that the three machines are represented, as before, by transitions t_1, t_2 and t_3, while the robot actions (for all six simple schedules) are represented by the remaining part of the net, some transitions of which are used by different schedules (i.e., different colors).

There are six basic colors representing the schedules (denoted as the schedules, A, B, ..., F), and five auxiliary colors which are used for elimination of potential conflicts when the models of different schedules are combined together. These auxiliary colors are needed for schedules B, C, D, E and F, and are denoted by b, c, d, e and f; they are used as token colors only, so there are eleven token colors and only six occurrence colors (formally, the arc functions are partial functions which are undefined for the auxiliary occurrence colors).

The transitions correspond to the following actions (the column *schedules* indicates the occurrence colors of the corresponding transitions):

	robot's operation	*schedules*	*execution time*
t_1	M_1 operation	A,B,C,D,E,F	o_1
t_2	M_2 operation	A,B,C,D,E,F	o_2
t_3	M_3 operation	A,B,C,D,E,F	o_3
t_{01}	pick from In, move to M_1 and load	A,B,C,D,E,F	$u + w + y$
t_{11}	move from M_1 to M_2	D,E	y
t_{12}	unload M_1, move to M_2 and load	A,B,C,D,E,F	$v + w + y$
t_{13}	move from M_1 to M_3	C,F	$2y$
t_{20}	move from M_2 to In	E,F	$2y$
t_{22}	move from M_2 to M_3	B,D	y
t_{23}	unload M_2, move to M_3 and load	A,B,C,D,E,F	$v + w + y$
t_{30}	move from M_3 to In	B,C	$2y$
t_{31}	move from M_3 to M_1	D,F	$2y$
t_{34}	unload M_3, move to Out and drop	A,B,C,D,E,F	$v + x + y$
t_{40}	move from Out to In	A,D	y
t_{41}	move from Out to M_1	C,E	$2y$
t_{42}	move from Out to M_2	B,F	$2y$

The *execution times* are the same for all transition occurrences.

The arc functions w are mappings $C \to C \to \mathbf{N}$; for most of the arcs, these functions are (partial) identity functions for the basic colors A, B, ..., F, i.e., for an arc a, an occurrence color $o \in C$ and a token color $c \in C$:

$$w(a)(o)(c) \;=\; \begin{cases} 1, \text{ if } o, c \in \{\text{A,B,C,D,E,F}\} \wedge o = c, \\ 0, \text{ if } o, c \in \{\text{A,B,C,D,E,F}\} \wedge o \neq c, \\ \text{undefined, otherwise.} \end{cases}$$

The definitions of all 'non–standard' arc functions are shown in the following table, in which the occurrence colors correspond to columns and each entry is a function $g : C \to \mathbf{N}$, shown using a simplified notation "$X : i$" that denotes:

$$g(c) \;=\; \begin{cases} i, \text{ if } c = X, \\ 0, \text{ otherwise;} \end{cases}$$

arc	A	B	C	D	E	F
(t_{01}, p_{11})	A:1	B:1	c:1	d:1	e:1	f:1
(t_{12}, p_{22})	A:1	b:1	C:1	d:1	e:1	f:1
(t_{23}, p_{33})	A:1	b:1	c:1	d:1	E:1	f:1
(t_{34}, p_{40})	A:1	b:1	c:1	D:1	e:1	f:1
(p_{11}, t_{11})	—	—	—	d:1	e:1	—
(p_{11}, t_{13})	—	—	c:1	—	—	f:1
(p_{22}, t_{20})	—	—	—	—	e:1	f:1
(p_{22}, t_{22})	—	b:1	—	d:1	—	—
(p_{33}, t_{30})	—	b:1	c:1	—	—	—
(p_{33}, t_{31})	—	—	—	d:1	—	f:1
(p_{40}, t_{41})	—	—	c:1	—	e:1	—
(p_{40}, t_{42})	—	b:1	—	—	—	f:1

It can be observed that, in addition to the typical representation of machines (a transition with one input and one output place for each machine), there is a systematic structure of the net model shown in Fig.6:

- there is a single place corresponding to each of the machines (p_{11}, p_{22} and p_{33} in Fig.4), the *Input* and the *Output* (p_{40} and p_{04} in Fig.4); in general, for an m–machine cell, there are m+2 such places;
- places representing *Input* and all machines (i.e., p_{04} and p_{11}, p_{22}, p_{33}) have three input transitions each (m input transitions in general) representing the possible moves from the 'other' machines and *Output* (and from *Input* in the case of M_1); the 'other' machines do not include the 'next' machine (*Output* is 'the next machine' for M_3); the move from *Input* always carries a part; the moves from *Output* never carry a part, the moves from 'other' machine can carry a part only if the 'other' machine is actually the previous machine; so, for p_{04} the input arcs are from M_2, M_3 and *Output*, for p_{11} the input arcs are from M_3, *Output* and *Input*, for p_{22} the input arcs are from *Output*, M_1 (carrying a part) and M_1 (without a part), and for p_{33} the input arcs are form M_1, M_2 (carrying a part) and M_2 (without a part);
- places representing *Output* and all machines have three output transitions each (m output transitions in general) representing moves to 'other' machines and *Input* (and *Output* in the case of M_3); the 'other' machines do not include the 'previous' machine (*Input* is 'the previous machine' for M_1); so for p_{11} the output arcs are to M_2 (carrying a part), M_2 (without a part) and M_3, for p_{22} the output arcs are to M_3 (carrying a part), M_3 (without a part) and *Input*, for p_{33} the output arcs are to *Output*, *Input* and M_1, and for p_{40} the output arcs are to *Input*, M_1 and M_2;
- the place representing *Input* (p_{04}) has only one output transition (t_{01}, which represents the operations 'pick a part, move and load M_1');
- the place representing *Output* (p_{40}) has only one input transition (t_{34}, which represents the operations 'unload M_3, move and drop),
- the total number of 'scheduling' transitions, resulting from the above rules, is equal to (counting either the input or output arcs) $1 + m * (m + 1)$, so for a three–machine cell there are 13 transitions modeling the possible robot schedules (see Fig.4); a net model of a four–machine cell needs 21 transitions and 6 places to represent all possible robot's schedules, and a model of a five–machine cell, needs 31 such transitions and 7 places.

The colored net shown in Fig.6 is decoupled and the partition of the set of colors is as follows:

$$\mathcal{P}(C) = \{\{A\}, \{B, b\}, \{C, c\}, \{D, d\}, \{E, e\}, \{F, f\}\}$$

Consequently, the invariants are grouped in sections corresponding to different colors (i.e., different schedules). There are 8 invariants for color/schedule A, 5 invariants for

color/schedule B, 6 invariants for color/schedule C, etc.; total number of place–invariants for this model is 33, as shown in Table 1. The corresponding sets of transitions are shown in Table 2 (also grouped in sections corresponding to different schedules). The minimum cycle time of each schedule is determined by the invariant subnet with the maximum total cycle time. Since all invariant subnets are simple cyclic nets, each subnet cycle time is equal to the sum of firing times assigned to all transitions of the (invariant) subnet. For the sets of transitions shown in Table 2, the cycle times of the six schedules are as follows (since the sets of transitions of some invariant subnets are subsets of those of other invariant subnets, not all invariants are used in the formulas); e.g., 2, 3, 4, 5, 6, 7 and 8 are all subsets of 1, 11 is a subset of 9, 13 a subset of 10, etc.):

schedule	*cycle time*
A	$\tau_A = \tau_1$
B	$\tau_B = \max(\tau_9, \tau_{10}, \tau_{12})$
C	$\tau_C = \max(\tau_{14}, \tau_{16}, \tau_{17}, \tau_{19})$
D	$\tau_C = \max(\tau_{20}, \tau_{21}, \tau_{22}, \tau_{23}, \tau_{24})$
E	$\tau_E = \max(\tau_{25}, \tau_{26}, \tau_{28})$
F	$\tau_E = \max(\tau_{30}, \tau_{31}, \tau_{32}, \tau_{33})$

where

$$
\begin{aligned}
\tau_1 &= o_1 + o_2 + o_3 + u + 3v + 3w + x + 5y \\
\tau_9 &= o_1 + o_2 + u + 2v + 3w + 5y \\
\tau_{10} &= o_1 + u + 3v + 3w + x + 9y \\
\tau_{12} &= o_3 + v + w + 5y \\
\tau_{14} &= o_1 + o_2 + u + 2v + 2w + 5y \\
\tau_{16} &= o_2 + o_3 + 3v + 2w + x + 5y \\
\tau_{17} &= o_2 + u + 3v + 3w + x + 7y \\
\tau_{19} &= u + 3v + 3w + x + 10y \\
\tau_{20} &= o_1 + o_2 + o_3 + u + 3v + 3w + x + 5y \\
\tau_{21} &= o_1 + u + 2v + 2w + x + 5y \\
\tau_{22} &= o_2 + 2v + 2w + 4y \\
\tau_{23} &= o_3 + u + 2v + 2w + x + 5y \\
\tau_{24} &= u + 2v + 2w + x + 8y \\
\tau_{25} &= o_1 + u + v + 2w + 4y \\
\tau_{26} &= o_2 + o_3 + 3v + 2w + x + 5y \\
\tau_{28} &= o_3 + u + 3v + 3w + x + 9y \\
\tau_{30} &= o_1 + u + v + 2w + 4y \\
\tau_{31} &= o_2 + 2v + 2w + 4y \\
\tau_{32} &= o_3 + 2v + w + x + 4y \\
\tau_{33} &= u + 3v + 3w + x + 12y
\end{aligned}
$$

Because the optimal schedule is the schedule with the minimum cycle time, so:

$$\tau_{opt} \;=\; \min(\tau_A, \tau_B, \tau_C, \tau_D, \tau_E, \tau_F).$$

Tab.1. P–invariants of the net in Fig.6.

P–invariant	p_{10}	p_{12}	p_{21}	p_{23}	p_{32}	p_{34}	p_{04}	p_{11}	p_{22}	p_{33}	p_{40}
1	A : 1	A : 1	A : 1	A : 1	A : 1	A : 1	A : 1	0	0	0	A : 1
2	A : 1	A : 1	A : 1	A : 1	0	0	A : 1	0	0	A : 1	A : 1
3	A : 1	A : 1	0	0	A : 1	A : 1	A : 1	0	A : 1	0	A : 1
4	A : 1	A : 1	0	0	0	0	A : 1	0	A : 1	A : 1	A : 1
5	0	0	A : 1	A : 1	A : 1	A : 1	A : 1	A : 1	0	0	A : 1
6	0	0	A : 1	A : 1	0	0	A : 1	A : 1	0	A : 1	A : 1
7	0	0	0	0	A : 1	A : 1	A : 1	A : 1	A : 1	0	A : 1
8	0	0	0	0	0	0	A : 1	A : 1	A : 1	A : 1	A : 1
9	B : 1	B : 1	B : 1	B : 1	0	0	B : 1	0	0	b : 1	0
10	B : 1	B : 1	0	0	0	0	B : 1	0	B,b : 1	B,b : 1	b : 1
11	0	0	B : 1	B : 1	0	0	B : 1	B : 1	0	b : 1	0
12	0	0	0	0	B : 1	B : 1	0	0	B : 1	0	b : 1
13	0	0	0	0	0	0	B : 1	B : 1	B,b : 1	B,b : 1	B : 1
14	C : 1	C : 1	C : 1	C : 1	0	0	C : 1	0	0	c : 1	0
15	C : 1	C : 1	0	0	0	0	C : 1	0	C : 1	c : 1	0
16	0	0	C : 1	C : 1	C : 1	C : 1	0	C : 1	0	0	c : 1
17	0	0	C : 1	C : 1	0	0	C : 1	C,c : 1	0	C,c : 1	c : 1
18	0	0	0	0	C : 1	C : 1	0	C : 1	C : 1	0	c : 1
19	0	0	0	0	0	0	C : 1	C,c : 1	C : 1	C,c : 1	c : 1
20	D : 1	D : 1	D : 1	D : 1	D : 1	D : 1	D : 1	0	0	0	D : 1
21	D : 1	D : 1	0	0	0	0	D : 1	0	d : 1	D : 1	D : 1
22	0	0	D : 1	D : 1	0	0	0	D : 1	0	d : 1	0
23	0	0	0	0	D : 1	D : 1	D : 1	d : 1	D : 1	0	D : 1
24	0	0	0	0	0	0	D : 1	D,d : 1	D,d : 1	D,d : 1	D : 1
25	E : 1	E : 1	0	0	0	0	E : 1	0	E : 1	0	0
26	0	0	E : 1	E : 1	E : 1	E : 1	0	E : 1	0	0	e : 1
27	0	0	E : 1	E : 1	0	0	0	E : 1	0	E : 1	e : 1
28	0	0	0	0	E : 1	E : 1	E : 1	E,e : 1	E,e : 1	0	e : 1
29	0	0	0	0	0	0	E : 1	E,e : 1	E,e : 1	E : 1	e : 1
30	F : 1	F : 1	0	0	0	0	F : 1	0	f : 1	0	0
31	0	0	F : 1	F : 1	0	0	0	F : 1	0	f : 1	0
32	0	0	0	0	F : 1	F : 1	0	0	F : 1	0	f : 1
33	0	0	0	0	0	0	F : 1	F,f : 1	F,f : 1	F,f : 1	f : 1

Tab.2. Sets of transitions implied by P–invariants of the net in Fig.6.

P–invariant	t_1	t_2	t_3	t_{01}	t_{11}	t_{12}	t_{13}	t_{20}	t_{22}	t_{23}	t_{30}	t_{31}	t_{34}	t_{40}	t_{41}	t_{42}
1	1	1	1	1	0	1	0	0	0	1	0	0	1	1	0	0
2	1	1	0	1	0	1	0	0	0	1	0	0	1	1	0	0
3	1	0	1	1	0	1	0	0	0	1	0	0	1	1	0	0
4	1	0	0	1	0	1	0	0	0	1	0	0	1	1	0	0
5	0	1	1	1	0	1	0	0	0	1	0	0	1	1	0	0
6	0	1	0	1	0	1	0	0	0	1	0	0	1	1	0	0
7	0	0	1	1	0	1	0	0	0	1	0	0	1	1	0	0
8	0	0	0	1	0	1	0	0	0	1	0	0	1	1	0	0
9	1	1	0	1	0	1	0	0	0	1	1	0	0	0	0	0
10	1	0	0	1	0	1	0	0	1	1	1	0	1	0	0	1
11	0	1	0	1	0	1	0	0	0	1	1	0	0	0	0	0
12	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	1
13	0	0	0	1	0	1	0	0	1	1	1	0	1	0	0	1
14	1	1	0	1	0	1	0	0	0	1	1	0	0	0	0	0
15	1	0	0	1	0	1	0	0	0	1	1	0	0	0	0	0
16	0	1	1	0	0	1	0	0	0	1	0	0	1	0	1	0
17	0	1	0	1	1	1	0	0	0	1	0	0	1	1	0	0
18	0	0	1	0	0	1	0	0	0	1	0	0	1	0	1	0
19	0	0	0	1	0	1	1	0	0	1	1	0	1	0	1	0
20	1	1	1	1	0	1	0	0	0	1	0	0	1	1	0	0
21	1	0	0	1	0	1	0	0	1	0	0	0	1	1	0	0
22	0	1	0	0	0	1	0	0	0	1	0	1	0	0	0	0
23	0	0	1	1	1	0	0	0	0	1	0	0	1	1	0	0
24	0	0	0	1	1	0	0	0	1	1	0	1	1	1	0	0
25	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0
26	0	1	1	0	0	1	0	0	0	1	0	0	1	0	1	0
27	0	1	0	0	0	1	0	0	0	1	0	0	1	0	1	0
28	0	0	1	1	1	1	0	0	0	1	1	0	1	0	1	0
29	0	0	0	1	1	1	0	0	0	1	1	0	1	0	1	0
30	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0
31	0	1	0	0	0	1	0	0	0	1	0	1	0	0	0	0
32	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	1
33	0	0	0	1	0	1	1	1	0	1	0	1	1	0	0	1

Since all simple schedules are modeled by the same (colored) net (as shown in Fig.6), the same modeling net will also represent the composite schedules of a manufacturing cell. For example, to model a composite schedule A+B obtained by combining schedules A and B, two additional basic colors Ab, aB and an auxiliary color ab should be introduced with the following occurrences of transitions (the occurrences of Ab correspond to part A of the schedule AB, while occurrences aB to part B of this schedule):

arc	...	Ab	aB
(t_{12}, p_{22})	...	aB:1	ab:1
(t_{23}, p_{33})	...	Ab:1	ab:1
(t_{34}, p_{40})	...	Ab:1	ab:1
(p_{22}, t_{22})	...	—	ab:1
(p_{33}, t_{30})	...	—	ab:1
(p_{40}, t_{42})	...	ab:1	—

so that the complete robot's path is (p_{04}, t_{01}, p_{11}, t_{12}, p_{22}, t_{22}, p_{33}, t_{34}, p_{40}, t_{42}, p_{22}, t_{23}, p_{33}, t_{30}, p_{04}, t_{01}, p_{11}, t_{12}, p_{22}, t_{23}, p_{33}, t_{34}, p_{40}, t_{40}, p_{04}). Systematic generation of such composite schedules and their analysis need to be investigated in greater detail.

CONCLUDING REMARKS

A systematic approach to modeling and analysis of simple and composite schedules for a large class of manufacturing cells is proposed and is illustrated by example schedules for a 3–machine cell. The derived net models are composed of a relatively small number of conflict–free (for simple schedules) and free–choice (for composite schedules) subnets, which can easily be determined by net invariants.

Invariant analysis of net models provides performance characteristics (the throughput or the average cycle time) in symbolic form which means that specific values of performances can easily be obtained by evaluating the symbolic results for specific values of parameters (i.e., symbols), and then the best schedule (i.e., the one with the smallest cycle time) can be selected to maximize the cell's performance.

Several simplifying assumptions were made during the derivation of Petri net models, e.g., the all parts are identical, that the robot travel times between adjacent machines are the same, etc. It should be noted that all these assumptions were made to simplify the discussion and they can easily be removed by simple modifications of the presented approach. In particular, composite schedules can be used to describe scheduling problems when several different parts enter and leave the cell in one cycle. A decomposition of such a schedule into a number of simple components identifies operations performed om parts of different types. Different parameters can easily be associated with parts of different types because the corresponding operations are represented by independent transitions (see Fig.4 and 5),

The number of schedules (both simple and composite) increases very quickly with the number of machines, and the number of composite schedules also increases rather quickly with the the length of the schedule; for a 3–machine cell, there are 6 simple schedules, 34 different 2–schedules and 198 different 3–schedules. Instead of analyzing all these schedules

one after another, a more general approach can be developed, using colored Petri nets for modeling the whole sets of schedules, with different colors representing different schedules.

The procedure of developing colored net models and then analyzing them can be automated. Since the number of schedules grows very quickly with the number of machines and the length of the schedules, new methods of analysis may be needed in which some reductions are performed at early stages of analysis in order to eliminate all those cases (i.e., schedules) which cannot affect the final results.

References

[Cl83] Claybourne, B.H.: "Scheduling robots in flexible manufacturing cells"; CME Automation, vol.30, no.5, pp.36–40, 1983.

[DH90] Dixon, C., Hill, S.D.: "Work–cell cycle–time analysis in a flexible manufacturing system"; Proc. Pacific Conf. on Manufacturing, Sydney–Melbourne, Australia, vol.1, pp.182–189, 1990.

[Ha72] Hack, M.: "Analysis of production schemata by Petri nets"; Project MAC Technical Report TR–94, 1972.

[Hi89] Hillion, H.P.: "Timed Petri nets and application to multi-stage production system"; in: Advances in Petri Nets 1989 (Lecture Notes in Computer Science 424); pp. 281–305, Springer Verlag 1989.

[Je87] K. Jensen, "Coloured Petri nets"; in: "Advanced Course on Petri Nets 1986" (Lecture Notes in Computer Science 254), G. Rozenberg (ed.), pp.248–299, Springer Verlag 1987.

[KJ87] Krueckeberg, F., Jaxy, M.: "Mathematical methods for calculating invariants in Petri nets"; in: "Advances in Petri Nets 1987" (Lecture Notes in Computer Science 266), G. Rozenberg (ed.), pp.104-131, Springer Verlag 1987.

[MS82] Martinez, J., Silva, M.: "Simple and fast algorithm to obtain all invariants of a generalized Petri net"; in: "Applications and Theory of Petri Nets" (Informatik Fachberichte 52); pp.301–310, Springer Verlag 1982.

[Mu89] Murata, T.: "Petri nets: properties, analysis and applications"; Proceedings of IEEE, vol.77, no.4, pp.541–580, 1989.

[Re85] Reisig, W.: "Petri nets - an introduction" (EATCS Monographs on Theoretical Computer Science 4); Springer Verlag 1985.

[S3BK92] Sethi, S.P., Sriskandarajah, C., Sorger, G., Blazewicz, J., Kubiak, W.: "Sequencing of parts and robot moves in a robotic cell"; Int. Journal of Flexible Manufacturing Systems, vol.4, pp.331–358, 1992.

[Su85] Suri, R.: "An overview of evaluative models for flexible manufacturing systems"; Annals of Operations Research, vol.3, no.1, pp.3–21, 1985.

[Zu90] Zuberek, W.M., "Performance evaluation using timed colored Petri nets", Proc. 33-rd Midwest Symp. on Circuit and Systems (Special Session on Petri Net Models), Calgary, Alberta, pp.779-782, 1990.

[Zu91] Zuberek, W.M.: "Timed Petri nets – definitions, properties and applications"; Microelectronics and Reliability (Special Issue on Petri Nets and Related Graph Models), vol.31, no.4, pp.627–644, 1991.

[ZK93] Zuberek, W.M., Kubiak, W., "Timed Petri net models of flexible manufacturing cells"; Proc. 36-th Midwest Symp. on Circuits and Systems, Detroit MI, August 16–18, 1993.