IEEE Int. Symposium on Industrial Electronics (ISIE'96), Warsaw, Poland, 17-20 June 1996, vol.2, pp.1055-1060.

Optimal Schedules of Manufacturing Cells — Modeling and Analysis Using Timed Petri Nets

W.M. Zuberek

Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada A1C-5S7

Abstract – A method for systematic generation of simple and composite schedules for a large class of manufacturing cells is presented. The generated schedules can easily be transformed into timed Petri net models, and these models evaluated using one of methods developed for analysis of timed Petri nets, for example, invariant analysis. Performance characterization (the cycle time or the throughput) obtained in this way can be used for optimization of the cell's performance.

1. INTRODUCTION

Manufacturing cells (or robotic cells) are groups of machines with a robot that performs cyclic sequences of pickup, move, load, unload and drop operations, transporting the manufactured parts from one machine of the cell to another [1, 2]. This cyclic sequence of robot activities is called a schedule. The throughput of the cell depends on the sequence of robot actions as well as on the sequence in which different parts enter the cell [3]. 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 schedule. Usually the schedules are represented by models which capture the essential characteristics of the schedule, but which remove all details inessential to the evaluation process.

Manufacturing cells can be regarded as discrete event systems, with their behavior represented by 'events' and 'activities'. An activity corresponds to an operation performed by a machine or the robot, and events corresponds to changes of (simultaneous) 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 [4, 5].

Although the application of Petri nets to modeling and analysis of scheduling problems has a long history [6, 7, 8, 9, 10], the Petri nets used in these models did not represent the durations of modeled activities. In order to study performance aspects of modeled systems, these durations 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 [11].

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, but they provide less information than the reachability approach. Quite often, however, all the detailed results of reachability analysis are not really needed, and the performance measures provided by structural methods are quite satisfactory [12]. 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 conflict–free or free–choice elementary nets [13].

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 schedules and selecting the schedule

that minimizes the cell's performance (i.e., minimizes the cycle time). As the number of possible schedules increases rather quickly with the number of machines as well as with the length of a (composite) schedule, an efficient method of generating simple and composite schedules for a large class of manufacturing cells is needed. Such a method is proposed in this paper. The generated schedules can easily be transformed into timed Petri net models, and these models evaluated using one of methods developed for analysis of timed Petri nets, for example, invariant analysis.

2. MODELS OF 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 [2] 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.

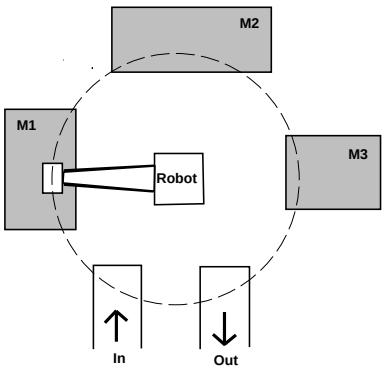


Fig.1. Layout of a three–machine cell.

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 ($M3$), 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:

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

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

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 machines:

- 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 (the initial configurations describe different distributions of parts on machines in time instants when a new part is going to be picked from the input):

$(0,0,0) \to (1,0,0) \to (0,1,0) \to (0,0,1) \to (0,0,0)$ ..... A

$(0,0,1) \to (1,0,1) \to \begin{cases} (0,1,1) \to (0,1,0) \to (0,0,1) & \text{... B} \\ (1,0,0) \to (0,1,0) \to (0,0,1) & \text{... C} \end{cases}$

$(0,1,0) \to (1,1,0) \to (1,0,1) \to \begin{cases} (0,1,1) \to (0,1,0) & \text{... D} \\ (1,0,0) \to (0,1,0) & \text{... E} \end{cases}$

$(0,1,1) \to (1,1,1) \to (1,1,0) \to (1,0,1) \to (0,1,1)$ ..... 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 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 a part is always available in In, and that Out removes manufactured parts sufficiently quickly, so In and Out are not actually shown (they can easily be added to the model, with or without some form of buffering).

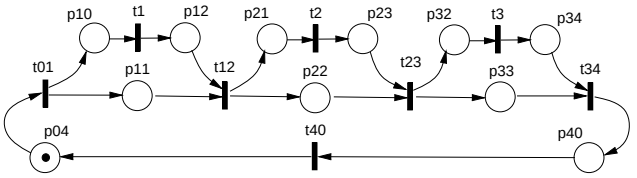


Fig.2. Petri net model of schedule A.

The operations of the robot are represented by a simple cycle (p_{04}, t_{01}, p_{11}, t_{12}, p_{22}, t_{23}, p_{33}, t_{34}, p_{40}, t_{40}, p_{04}), which follows the sequence of robot's operations. The initial marking of place p_{04} represents a new part ready to be picked up from the input conveyor. The 'interpretation' of the transitions is as follows:

	robot operations	*exec time*
t_{01}	pick a part from In, move to M_1, 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	$2y$

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

u denotes the (average) pickup time,
v denotes the (average) unload time,
w denotes the (average) load time,
x denotes the (average) drop time and
y denotes 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).

Fig.3 shows a Petri net model of schedule C. The robot's operations are represented by a more complex cycle (p_{03}, t_{01}, p_{13}, t_{13}, p_{31}, t_{34}, p_{41}, t_{41}, p_{14}, t_{12}, p_{22}, t_{32}, p_{30}, t_{30}, p_{03}), and the 'interpretation' of the transitions is as follows:

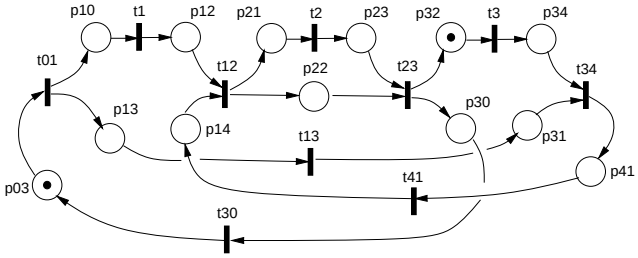


Fig.3. Petri net model of schedule C.

	robot operations	*exec time*
t_{01}	pick a part from In, move to M_1, 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_{30}	move from M_3 to In	$2y$
t_{34}	unload M_3, move to Out and drop	$v + x + y$
t_{41}	move from Out to M_1	$2y$

The initial marking assigns tokens to p_{03} (the next part ready to be picked up from the input conveyor) and p_{32} (another part loaded – in the previous cycle – on machine M_3).

3. MODELS OF COMPOSITE SCHEDULES

Models of composite schedules can be regarded as an interleaved composition of simple schedules, which correspond to consecutive parts entering the cell within one cycle. A systematic generation of composite schedules can be obtained by tracing all possible moves of parts through the cell. Since each composite schedule processes n parts in a single cycle (and is called an n–schedule for simplicity), the states of a cell (i.e., distributions of parts among the machines of a cell) must be denoted in a unique way within a cycle. A typical m–machine cell state is a vector of length m with elements equal to "1" to denote machines processing parts, and elements equal to "0" for idle machines. For composite schedules, a hypothetical "input container" is introduced, which (for an n–schedule) contains initially exactly n (parts); all these parts must be emptied

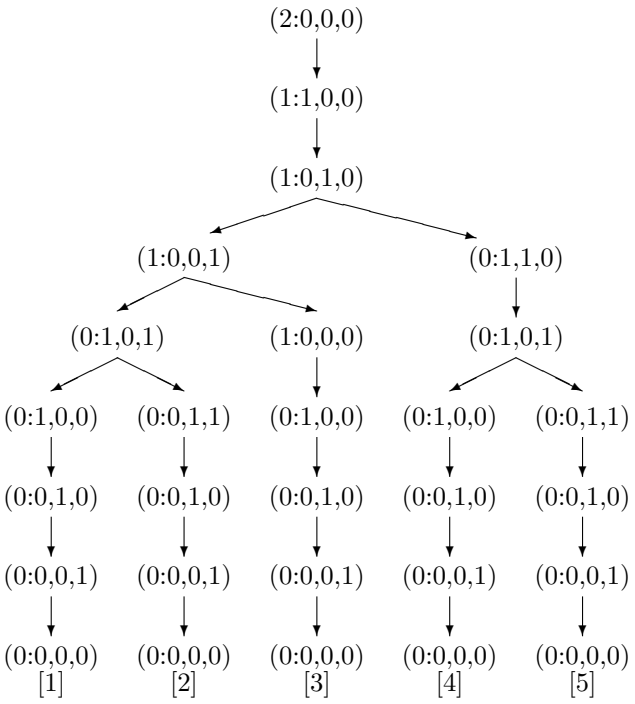
during the schedule. This container is characterized by an additional, initial element of cell descriptions, separated from the remaining (machine) descriptions by a colon (rather than a comma). A typical description of an n–schedule for an m–machine cell is thus:

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

and the 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",
- a configuration $(k_0 : k_1, k_2, ..., 1)$ always derives a configuration $(k_0 : k_1, k_2, ..., 0)$ (this 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 $k_0 > 0$ (this corresponds to moving a new part from the input to 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 from $(k_0 : 0, k_2, ...k_m)$ 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 which are just simple schedules repeated twice. All 34 2–schedules can be systematically derived by iteratively 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 C:

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

schedule [2] 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 [3] is simply a composition of A with itself.

All these schedules can easily be translated into sequences of robot operations (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]: $In \Rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In$

[2]: $In \Rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In$

[3]: $In \Rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In \Rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In$

[4]: $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow In \Rightarrow M_1 \rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In$

[5]: $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow In \Rightarrow M_1 \rightarrow M_2 \Rightarrow M_3 \rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In$

Timed Petri net models can easily be derived from the sequences of robot's action. For example, the net model for schedule [1] is shown in Fig.4. The model contains three transitions representing the machines of the cell (t_1, t_2 and t_3 with average operation times o_1, o_2 and o_3, respectively) and their input and output places. The remaining part of the model represents the actions of the robot (the elements of the schedule A in the composite schedule A+C are denoted by an additional single quote):

	robot operations	*exec time*
t'_{01}	pick a part from In, move to M_1, 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'_{30}	move from M_3 to In	$2y$
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, 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_{34}	unload M_3, move to Out and drop	$v+x+y$
t_{41}	move from Out to In	y

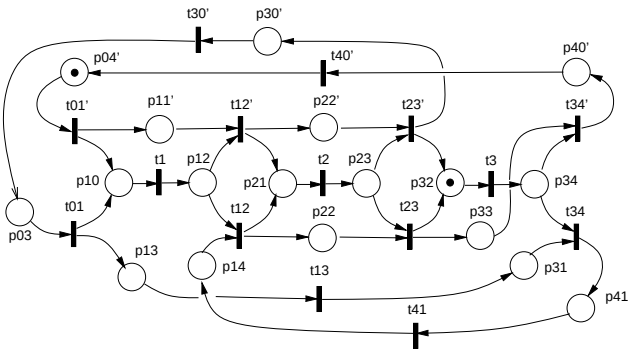
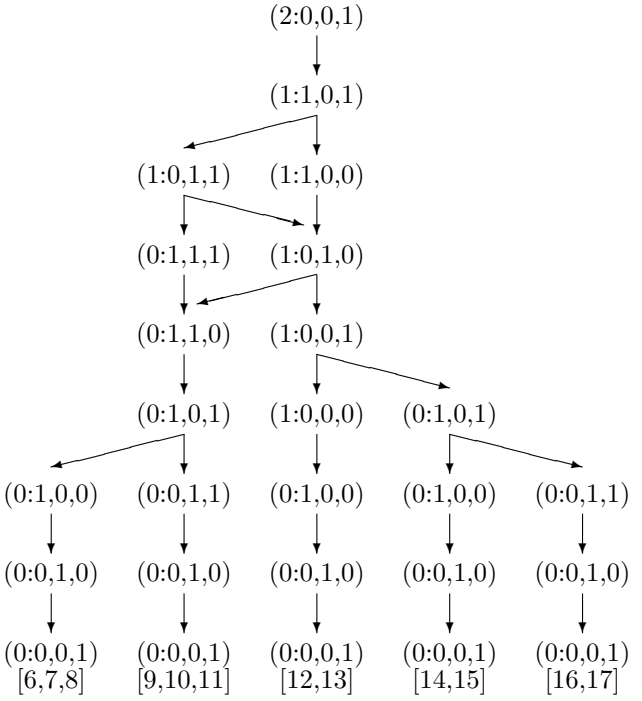


Fig.4. Petri net model of schedule [1]=(A+C).

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]: $In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow In$

There are 12 schedules for the initial configuration (0,1,0) and another 5 schedules for the initial configuration (0,1,1) [14].

4. OPTIMAL SCHEDULES

Performance analysis of simple schedules is extensively discussed in [15]. Net invariants [13, 15] or net transformations [16] can be used to obtain symbolic solutions in terms of operation execution times associated with transitions of the model.

Net invariants can also be used for analysis of composite schedules. The net shown in Fig.4 has six P–invariants which imply invariant subnets with the sets of transitions shown in Tab.1; the entries in Tab.1 are multiplied by the relative frequencies of transition firings, determined by a T–invariant [5, 13] (the net shown in Fig.4 has one T–invariant which assings multiplicity 2 to t_1, t_2 and t_3, and multiplicity 1 to all other transitions).

Since the invariant (2) is a subset of (1), (4) is a subset of (3), and (6) is a subset of (4), the cycle time of the schedule A+C is:

$$T_c^{[1]} \;=\; \max(T_1, T_3, T_4)$$

where T_1, T_3 and T_4 are obtained by adding the firing (or execution) times of transitions in the corresponding invariant subnets:

$$T_1 = 2o_1 + 2o_2 + 2u + 4v + 6w + 7y$$
$$T_3 = 2o_2 + 2o_3 + u + 6v + 5w + 2x + 9y$$
$$T_4 = 2(o_2 + u + 3v + 3w + x + 7y)$$

Tab.1. Sets of transitions of the invariant subnets.

invariants	1	2	3	4	5	6
t_1	2	2	0	0	0	0
t_2	2	0	2	1	0	0
t_3	0	0	2	0	1	0
t'_{01}	1	1	1	1	1	1
t'_{12}	1	1	1	1	1	1
t'_{23}	1	1	1	1	1	1
t'_{30}	1	1	0	1	0	1
t'_{34}	1	1	1	1	1	1
t'_{40}	1	1	1	1	1	1
t_{01}	1	1	0	1	0	1
t_{12}	1	1	1	1	1	1
t_{13}	0	0	0	1	0	1
t_{23}	1	1	1	1	1	1
t_{34}	0	0	1	1	1	1
t_{41}	0	0	1	1	1	1

Other 2–schedules are modeled and analyzed in the same way; some examples are given in [14].

For a 3–machine cell, there are 34 different 2–schedules, so the optimal 2–schedule schedule can easily be selected on the basis of systematic evaluations:

$$T_{opt} \;=\; \min(T_c^{[1]}, T_c^{[2]}, ..., T_c^{[34]})$$

The same approach is used to model and analyze other composite schedules.

5. CONCLUDING REMARKS

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. For simple schedules, such an approach is presented in [14]; a similar approach is expected for composite schedules. Most likely new methods of analyzing such models need to be developed to perform many possible simplifications during the analysis.

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. For example, the composite schedules can correspond to manufacturing parts of different types; the parameters of each component (simple) schedule can ve different to reflect the differences between the different types of parts.

References

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

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

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

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

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

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

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

[8] M. Silva, R. Valette, "Petri nets and flexible manufacturing"; in: "Advances in Petri nets 1989" (Lecture Notes in Computer Science 424), pp.374–417, Springer Verlag 1989.

[9] F. DiCezare, G. Hahalakis, J.M. Orith, M. Silva, F.B. Vernadat, "Practice of Petri nets in manufacturing"; Chapman & Hall 1993.

[10] Proc. IEEE Int. Conf. on Systems, Man and Cybernetics, 1993, 1994, 1995.

[11] W.M. Zuberek, "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.

[12] H.P. Hillion, "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.

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

[14] W.M. Zuberek, "Application of timed Petri nets to modeling and analysis of flexible manufacturing cells"; Technical Report #9503, Department of Computer Science, Memorial University of Newfoundland, St.John's, Canada A1B 3X5, June 1995.

[15] W.M. Zuberek, W. Kubiak, "Modeling simple schedules of manufacturing cells using timed Petri nets"; Proc. 6-th Int. Workshop in Intelligent Systems and Innovative Computations, Tokyo, Japan, pp.38–47, 1994.

[16] W.M. Zuberek, W. Kubiak, "Throughput analysis of manufacturing cells using timed Petri nets"; Proc. IEEE Int. Conf. on Systems, Man and Cybernetics, San Antonio, TX, pp.1328–1333, 1994.