Fundamenta Informaticae 67 (2005) 1–12
IOS Press

Extreme Runtimes of Schedules Modelled by Time Petri Nets*

Louchka Popova-Zeugmann†
Logic in Computer Science Group
Department of Computer Science
Humboldt University
popova@informatik.hu-berlin.de

Matthias Werner
Communication and Operating Systems Group
Department for Electrical Engineering and Computer Science
TU Berlin
mwerner@cs.tu-berlin.de

Abstract. In this paper, a method to determine best-case and worst-case times between two arbitrary markings in a bounded TPN is presented. The method uses a discrete subset of the state space of the net and achieves the results, which are integers, in polynomial time. As an application of the method the solution of a scheduling problem is shown.

1. Introduction

1.1. Motivation

Time Petri nets (TPN), first introduced in [9] and also called Interval Petri nets, IPN) provide a suitable method to model, to simulate, and to analyze the behavior of time-dependent systems.

Frequently, objects of consideration in Time Petri nets are the very same properties as for timeless Petri nets, namely reachability, liveliness, boundedness etc. However, in many systems modeled with Time Petri nets the temporal behavior needs to be verified. One of the most prevalent problems in this

*This is a revised version of our paper "Determining Worst-Case Times of Unknown Paths in Time Petri Net" presented at the CS&P 2004.

†Address for correspondence: Logic in Computer Science Group, Department of Computer Science, Humboldt University

area is the meeting of deadlines for a sequence of system processes, i.e., to compare the longest duration of a transition sequence with a given limit. Such considerations are an important part of scheduling studies, especially real-time scheduling. As a base for schedulability analysis, determining the worst-case execution time (WCET) and best-case execution time (BCET) is frequently needed. Computing the best-case time and the worst-case time between two states (i.e. between at least two states) is solvable in exponential time, as shown in [7] and later also in [4].

But, for the case of bounded TPN, and that are the most TPN modeling real systems, we introduce a method using solutions from graph theory which computes the min/max time in polynomial time. This is possible because of the adroit definition of a reachability graph for a TPN by picking out some essential states from the whole state space.

The remainder of this paper is organized in the following way: The second part of this section discusses related work. In Section 2 we introduce the basic notions and definition. The following section sketches our analysis method using a reachability graph defined by picking out only some states from the state space. Section 4 discusses the application of graph theory on the gained reachability graph. In Section 5 an example for a worst-case execution time analysis is given. Finally, the paper concludes with a short summary.

1.2. Related work

Time Petri nets were introduced by Merlin in [9] in order to study recoverability problems in computer systems and the design of communication protocols. Berthomieu and Menasche in [2] res. Berthomieu and Diaz in [1] provide a method to analyze the qualitative behavior of the net. They divide the state space in state classes which are described by a marking and time domain given by inequalities. The reachability graph, that they define consists of these classes as vertices and edges labeled by transitions. This graph has the pretty property, that the graph is finite iff the TPN is bounded. A similar definition for a reachability graph of a TPN delivers [3].

In order to study TPN we use the definition of a reachability graph given in [12]. The property from above, that the graph is finite iff the TPN is bounded, is also true here. In contrast to above, the time appears explicitly in this reachability graph as weights on some edges of the graph and the vertices contain no information. Thus, our reachability graph is a usual directed weighted graph. Therefore, it is possible to use the graph theory in order to accomplish quantitative analysis.

2. Basic Notations and Definitions

In this paper we use the following notations: $\mathbb{N}$ is the set of natural numbers, $\mathbb{N}^+ := \mathbb{N} \setminus \{0\}$. $\mathbb{Q}_0^+$ res. $\mathbb{R}_0^+$ is the set of nonnegative rational numbers res. set of nonnegative real numbers .

Definition 2.1. (Petri net)
The structure $N = (P, T, F, V, m_o)$ is called a Petri net (PN) iff

(a) P, T, F are finite sets with
$P \cap T = \emptyset$, $P \cup T \neq \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ and $dom(F) \cup cod(F) = P \cup T$
(b) $V : F \longrightarrow \mathbb{N}^+$ (weight of the arcs)
(c) $m_o : P \longrightarrow \mathbb{N}$ (initial marking)

A marking of a PN is a function $m : P \longrightarrow \mathbb{N}$, such that $m(p)$ denotes the number of tokens at the place p. The pre-sets and post-sets of a transition t are given by $Ft := \{p \mid p \in P \wedge pFt\}$ and $tF := \{p \mid p \in P \wedge tFp\}$, respectively. Each transition $t \in T$ induces the marking t^- and t^+, defined as follows:

$$t^-(p) = \begin{cases} V(p,t) & \text{,iff} \quad (p,t) \in F \\ 0 & \text{,iff} \quad (p,t) \notin F \end{cases}$$

$$t^+(p) = \begin{cases} V(t,p) & \text{,iff} \quad (t,p) \in F \\ 0 & \text{,iff} \quad (t,p) \notin F \end{cases}$$

Moreover, Δt denotes $t^+ - t^-$. A transition $t \in T$ is enabled (may fire) at a marking m iff $t^- \leq m$ (e.g. $t^-(p) \leq m(p)$ for every place $p \in P$). When an enabled transition t at a marking m fires, this yields a new marking m' given by $m'(p) := m'(p) + \Delta t(p)$ and denoted by $m \xrightarrow{t} m'$.

Definition 2.2. (Time Petri net)
The structure $Z = (P, T, F, V, m_o, I)$ is called a Time Petri net (TPN) iff:

(a) $S(Z) := (P, T, F, V, m_o)$ is a PN.
(b) $I : T \longrightarrow \mathbb{Q}_0^+ \times (\mathbb{Q}_0^+ \cup \{\infty\})$ and $I_1(t) \leq I_2(t)$ for each $t \in T$, where $I(t) = (I_1(t), I_2(t))$.

A TPN is called finite Time Petri net (FTPN) iff $I : T \longrightarrow \mathbb{Q}_0^+ \times \mathbb{Q}_0^+$.

I is the interval function of Z, $I_1(t)$ and $I_2(t)$ the earliest firing time of t ($eft(t)$) and the latest firing time of t ($lft(t)$), respectively. It is not difficult to see (cf. [12]) that considering TPNs with $I : T \longrightarrow \mathbb{N} \times (\mathbb{N} \cup \{\infty\})$ will not result in a loss of generality. Therefore only such time functions I will be considered subsequently. Furthermore, *conflict* is used in the strong sense: two transitions t_1 and t_2 are in conflict iff $Ft_1 \cap Ft_2 \neq \emptyset$. The PN $S(Z)$ referred to as the skeleton of Z.

Within this approach, the definition of a state is of fundamental importance for the ensuing theory. A state is characterized by a marking together with the momentary local time for enabled transitions or the sign $\sharp$ for the disabled transitions.

Definition 2.3. (state)
Let $Z = (P, T, F, V, m_o, I)$ be a TPN and $h : T \longrightarrow \mathbb{R}_0^+ \cup \{\#\}$. $z = (m, h)$ is called a state in Z iff:

(a) m is a reachable marking in $S(Z)$.
(b) $\forall t \ (\ (t \in T \wedge t^- \leq m) \ \longrightarrow \ h(t) \leq lft(t))$.
(c) $\forall t \ (\ (t \in T \wedge t^- \not\leq m) \ \longrightarrow \ h(t) = \#)$.

Interpretation of the notion "state" is as follows: within the net, each transition t has a clock $h(t)$. If t is enabled at a marking m, the clock of t $h(t)$ shows the time elapsed since t became most recently enabled. If t is disabled at m, the clock does not work (indicated by $h(t) = \#$).

Now the dynamic aspects of TPNs – changing from one state into another – can be introduced: The state $z_o := (m_o, h_o)$ with $h_o(t) := \begin{cases} 0 & \text{iff} \quad t \leq m_0 \\ \# & \text{iff} \quad t \not\leq m_0 \end{cases}$ is set as the initial state of the TPN Z. A transition t is ready to fire in the state $z = (m, h)$, denoted by $z \xrightarrow{t}$, iff $t^- \leq m$ and $eft(t) \leq h(t)$. A transition $\hat{t}$, which is ready to fire in the state $z = (m, h)$, may fire yielding a new state $z' = (m', h')$, defined by $m' = m + \Delta\hat{t}$ and

$$h'(t) =: \begin{cases} \# & \text{iff} \quad t^- \not\leq m \\ h(t) & \text{iff} \quad t^- \leq m \wedge t^- \leq m' \wedge Ft \cap F\hat{t} = \emptyset \\ 0 & \text{otherwise} \end{cases} .$$

The state $z = (m, h)$ is changed into the state $z' = (m', h')$ by the time duration $\tau \in \mathbb{Q}_0^+$, denoted by $z \xrightarrow{\tau} z'$, iff $m' = m$ and the time duration τ is possible (formally: $\forall t$ ($(t \in T \wedge h(t) \neq \#) \longrightarrow h(t) + \tau \leq lft(t)$ and $h'(t) := \begin{cases} h(t) + \tau & \text{iff} \quad t \leq m \\ \# & \text{iff} \quad t \not\leq m \end{cases}$). $z = (m, h)$ is called an integer-state iff $h(t)$ is an integer for each enabled transition t in m.

3. Analysis

The state space of an arbitrary Time Petri net is the set of all reachable states of the net, starting from z_0. Of course, this set is in general infinite: On the one hand, because of the markings, reachable in the net – this set is discrete, and it can be infinitely. On the other hand, because of the time of the transitions. Already, the set of all reachable states for a fix marking is infinitely (and dense) in general. Never the less, it is possible to pick up some "essential" states only, so that qualitative and quantitative analysis is possible. In [12] is shown, that the essential states are the integer-states.

The graph $RG_Z(z_o)$ is called the reachability graph of the TPN Z iff its vertices are the reachable integer-states and its edges are defined by the triples (z, t, z') and (z, τ, z'), where $z \xrightarrow{t} z'$ and $z \xrightarrow{\tau} z'$, respectively. This graph is finite iff the set of the reachable markings of the net is finite. And, this set is finite, if the set of reachable markings of the skeleton – the timeless net – is finite. The reverse is not true in general.

Using the parametric description of transition sequences (cf. [11]) minimal and maximal length of time of the sequence can be evaluated. For this reason the construction of the reachability graph is not necessary. The minimal and maximal amount of time is an integer and it can be reached by firing in integer-states. Thus, when the TPN is bounded, a sequence with minimal/maximal amount of time can be found for given source-state and sink-state.

3.1. Parametric description of transition sequences

The state space of a given TPN Z can be formally defined using the following recursive definition:

Definition 3.1. (state space)
Basis: $C_0 := \{z \mid \exists\tau(\tau \in \mathbb{R}_0^+ \wedge z_0 \xrightarrow{\tau} z)\}$

Step: Let C be already defined. Then C' is derived from C by firing $\hat{t}$

(formally $C \xrightarrow{\hat{t}} C'$), iff

$C' := \{z \mid \exists z_1 \exists z_2 \exists\tau(z_1 \in C \wedge \tau \in \mathbb{R}_0^+ \wedge z_1 \xrightarrow{\hat{t}} z_2 \xrightarrow{\tau} z)\}$.

Obviously, the state space of the net is the union of all sets C. In [12] a parametric description for each set C is defined. The Example 3.1 demonstrates this description.

Example 3.1.

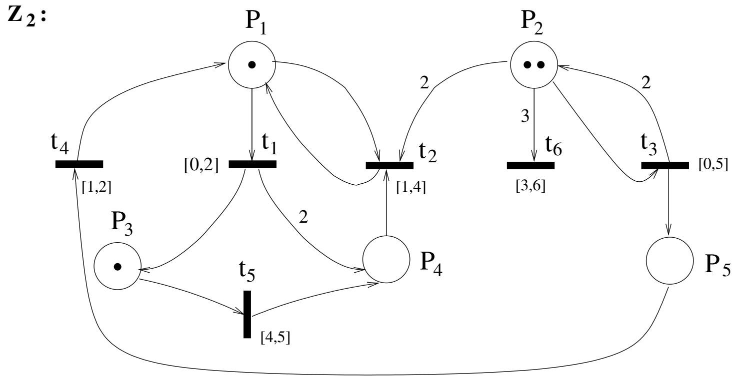

Figure 1. Z_1 - a TPN

Let consider the TPN Z_1. The parametric description of the set C_0 in Z_1 is K_e with

$$K_e = \{\ ((1,2,1,0,0), \begin{pmatrix} x_0 \\ x_0 \\ x_0 \\ \sharp \\ x_0 \\ x_0 \end{pmatrix}) \mid \begin{array}{c} 0 \leq x_0 \\ x_0 \leq 2 \end{array} \}.$$

And let C_1, C_2, C_3 be the sets of all reachable states obtained from C_0 by firing the transition sequence t_1, res. $t_1 t_3$, res. $t_1 t_3 t_4$. The parametric description of C_1, res. C_2, res. C_3 is K_{t_1}, res. $K_{t_1 t_3}$, res. $K_{t_1 t_3 t_4}$ with

$$K_{t_1} = \{\ ((0,2,2,2,0), \begin{pmatrix} \sharp \\ \sharp \\ x_0 + x_1 \\ \sharp \\ x_0 + x_1 \\ \sharp \end{pmatrix}) \left| \begin{array}{ll} 0 \leq x_0, & 0 \leq x_1 \\ x_0 \leq 2 & \\ x_0 + x_1 \leq 5 & \end{array} \right. \},$$

$$K_{t_1t_3} = \{ \ ((0,3,2,2,1), \begin{pmatrix} \sharp \\ \sharp \\ x_2 \\ x_2 \\ x_0+x_1+x_2 \\ x_2 \end{pmatrix}) \left| \begin{array}{l} 0 \le x_0, \quad 0 \le x_1, \quad 0 \le x_2 \\ x_0 \le 2 \\ x_0 + x_1 \le 5 \\ x_0 + x_1 + x_2 \le 5 \\ x_2 \le 2 \end{array} \right. \},$$

$$K_{t_1t_3t_4} = \{ \ ((1,3,2,2,0), \begin{pmatrix} x_3 \\ x_3 \\ x_2+x_3 \\ \sharp \\ x_0+x_1+x_2+x_3 \\ x_2+x_3 \end{pmatrix}) \left| \begin{array}{l} 0 \le x_0, \quad 0 \le x_1 \\ 0 \le x_2, \quad 0 \le x_3 \\ x_0 \le 2 \\ x_0 + x_1 \le 5 \\ x_0 + x_1 + x_2 \le 5 \\ x_2 \le 2 \\ 1 \le x_2 \\ x_3 \le 2 \\ x_2 + x_3 \le 5 \\ x_0 + x_1 + x_2 + x_3 \le 5 \\ x_2 + x_3 \le 6 \end{array} \right. \}$$

Of course, K_{t_1} is derived from K_e, $K_{t_1t_3}$ from K_{t_1} and $K_{t_1t_3t_4}$ from $K_{t_1t_3}$. Let us consider $K_{t_1t_3t_4}$ more detailed. After firing the transition sequence $(t_1t_3t_4)$ transitions t_1, t_2, t_3, t_5 and t_6 are enabled. The clocks of the transitions t_1 and t_2 show the time x_3, the clocks of transitions t_3 and t_6 show the time $x_2 + x_3$ and the clock of transition t_5 shows the time $x_0 + x_1 + x_2 + x_3$ since they become last enabled. The inequalities describe the conditions for the parameters: The time of each enabled transition is bounded above by its lft, e.g. $h(t_3) = x_2 + x_3$ may not exceed $lft(t_3) = 5$. Thus, the condition $x_2 + x_3 \le 5$ has to be fulfilled. Furthermore, all conditions which are true for $K_{t_1t_3}$ have to hold for $K_{t_1t_3t_4}$, too. Finally, the change from $K_{t_1t_3}$ to $K_{t_1t_3t_4}$ is realized by firing t_4. Thus, the parametric time shown by $h(t_4)$ in $K_{t_1t_3t_4}$ have to be not less then $eft(t_4)$, i.e. $1 \le x_2$ should hold in $K_{t_1t_3t_4}$.

3.2. Essential states

From the last section it is clear, that for a given transition sequence σ a parametric description K_σ is defined. Evidently, the sequence σ can fire iff the system of inequalities is solvable. In [12] is shown, that if the system is solvable, then there is an integer solution ("near" the old one), too. That means, we can pick out those states from the state space where clocks (of enabled transitions) show integers only. As already said, the set of all integer states of a TPN is finite iff the net is bounded and finite (all lfts are natural numbers). As a reachability graph for a TPN we use the graph where vertices are the reachable integer-states and its edges are defined by the triples (z, t, z') and (z, τ, z'), where $z \xrightarrow{t} z'$ and $z \xrightarrow{\tau} z'$, respectively. The Example 3.2 illustrates this. This graph is finite iff the set of the reachable markings of the net is finite. And, this set is finite, if the set of reachable markings of the skeleton - the timeless net - is finite. The reverse is not true. Of course, the so defined reachability graph can be reduced: all

Example 3.2.

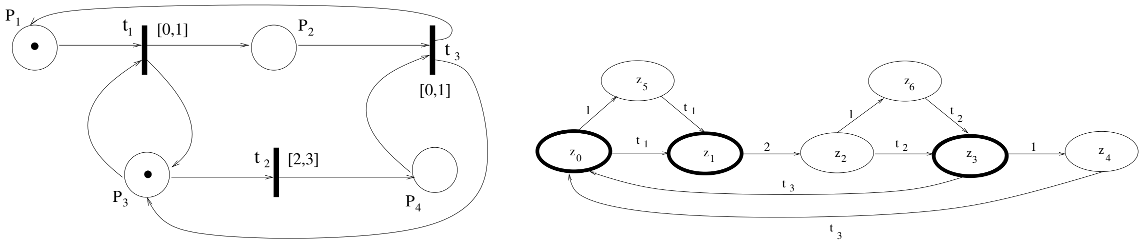

Figure 2. Z_2 - a TPN and their reachability graph RG_{Z_2}

$$\begin{array}{lllll} m_0 = (1,0,1,0) & h_1 = (\sharp,0,\sharp)^T & h_5 = (1,1,\sharp)^T & z_1 = (m_1,h_1) & z_4 = (m_2,h_4) \\ m_1 = (0,1,1,0) & h_2 = (\sharp,2,\sharp)^T & h_6 = (\sharp,\sharp,3)^T & z_2 = (m_1,h_2) & z_5 = (m_0,h_5) \\ m_2 = (0,1,0,1) & h_3 = (\sharp,\sharp,0)^T & z_0 = (m_0,h_0) & z_3 = (m_2,h_3) & z_6 = (m_1,h_6) \\ h_0 = (0,0,\sharp)^T & h_4 = (\sharp,\sharp,1)^T & & & \end{array}$$

vertices, which have input and output edges labeled with time only can be ignored. Each input edge labeled with $\tau_i^{(in)}$ is merged with each output edge labeled with $\tau_j^{(out)}$ and the new edge is labeled with the sum of both labels, i.e. with $\tau_i^{(in)} + \tau_j^{(out)}$. A further reduction can be accomplished as follow: all vertices which have input edges with time only can be ignored, too. Their input edges are merged with their output edges labeled with the both labels (from the input and from the output edges). However the second reduction decreases the number of vertices but the labels of the merged edges are more complex then the former labels.

In case that a TPN is not finite a modified reachability graph can be defined. The modified graph is finite iff the TPN is bounded. For more see [10].

Using the parametric description of transition sequences (cf. [11]) minimal and maximal amount of sequence's time can be evaluated. For this reason, the construction of the reachability graph is not necessary. The minimal and maximal length of time is an integer and it can be realized by firing in integer-states. For more cf. [12].

Thus, if the TPN is bounded, a sequence with minimal/maximal amount of time can be found for given source-state and sink-state without solving a linear programming problem. In this case the minimal and maximal duration of a sequence can be computed on the reachable graph.

4. Determining BCET and WCET

Constructing the reachability graph as described in Sect. 3, methods from graph theory may be applied to determine best-case and worst-case execution times.

Determining the best-case execution leads directly to the well-known problem of the shortest path. Since our reachability graph has nonnegative times only, all common shortest path algorithms are applicable, e.g., Dijkstra's algorithm or Bellman-Ford algorithm. For an overview and discussion of these algorithms see, e.g., [5].

The more important case is the WCET. Determining the worst-case execution is similar to the critical path problem, sometimes called longest path problem.

For our purpose, the problem has to be formulated as followed:

Definition 4.1. (Modified longest path problem)
For a given directed weighted graph $RG_Z = (\mathcal{Z}, E)$, find the length l of a longest path from a source vertex z_s to a goal vertex z_d such that z_d is contained at most once as a last vertex.

Actually we mean, that the length l is infinitely, if a cycle is reachable starting on z_s before passing z_d and otherwise l is the sum of the weights of the longest path.

To solve our modified critical path problem, we use the following algorithm $\mathcal{A}_1$:

1. Remove from the graph RG_Z all edges $(v_d, v_i)\ \forall i$, i.e., all edges that are directed *from* v_d
2. Assign each edge $(z_i, z_j) \in G$ with the weight w_i a new weight $w_i^- = -w_i$. Edges, labeled by transition names obtain the weight 0.
3. Run the Bellman-Ford algorithm.

If Bellman-Ford algorithm returns *false* then l is infinite. Otherwise, $l = -d(v_d)$.

The complexity of this algorithm is dominated by the complexity of the Bellman-Ford algorithm, i.e., it is $\mathcal{O}(|Z| \cdot |E|)$.

The correctness of $\mathcal{A}_1$ is easy to be seen: After the removal of all output edges from z_d no path is possible, which contains z_d at another position than as final vertex. And obviously, the shortest path in the negative weighted graph corresponds to the longest path in the initial graph.

Computing a shortest path and a longest path is implemented in the latest version of INA, a Time Petri net analysis tool [14]. In the next section, we demonstrate the use of INA to determine worst-case execution time for a certain scheduling problem, modelled as a TPN.

5. Application in scheduling

In this section we discuss the application of our approach in task scheduling.

Time Petri nets allow for modeling of multiple processors and other resources, communication and other dependencies and a wide range of scheduling policies.

Whereas optimal solutions exist for simple systems with well-defined conditions (e.g., [8]), there are rarely optimal methods for more complex scenarios. Furthermore, it is a well-known fact that the behavior of multiprocessor systems is sometimes counterintuitive. This fact is expressed in Graham's anomaly theorem:

Theorem 5.1. (Graham's anomalies, [6])
In multiprocessor systems, changing the priorities, increasing the number of processors, reducing execution times, or weakening the precedence constraints can increase the scheduling length.

We selected an example for the practical demonstration of our method, that shows the impact of a Graham anomaly. Please consider Figure 3.
It shows the dependencies between some task in a certain service. The service should run in a two processor environment. Both processors are assumed as equally fast. For every task, the longest and the shortest executions times are known, see Table 1.

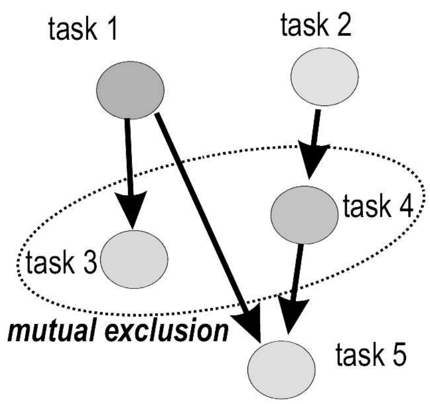

Figure 3. An example for Graham's anomalies (from [13])

Table 1. Minimal and maximal task execution times

task	t_{min_i}	t_{max_i}
task 1	8	10
task 2	5	7
task 3	10	12
task 4	9	13
task 5	11	12

We assume an instantaneous scheduling policy. I.e., each task is scheduled, if all preconditions are met and if all needed resources are available. No task has an affectation for a certain processor. Figure 4 shows the Time Petri net model of the example. Each task's execution is characterized by two events: the start of task execution an its end. These events are modeled by the firing of two transitions.

We assume, that the service will be restarted once it is finished. Thus, our net is connected. Our algorithm, executed with help of the INA tool, identified 25 relevant states. The worst case execution time of the service is 32.

Graham's anomaly becomes visible, if one relaxes the the timing restriction of task 1. We did a second evaluation of our example with the identical parameters, except for the minimal execution time of task 1, which was set to 6 instead of 8.

With the relaxed restriction, INA now calculated a WCET of 44. The reason is obvious, if one looks at Figure 6: Case (b), that consumes because of the mutual exclusion and the instantaneous scheduling policy much more time than case (a), was excluded by the more restrict time settings.

Such kind of anomalies are rather simple to detect in uncomplex situations as our example is. However, since the appearance of anomalies is hardly foreseeable, more sophisticated applications need urgently systematic analysis to detect the critical cases. Our method is a further contribution to such analysis.

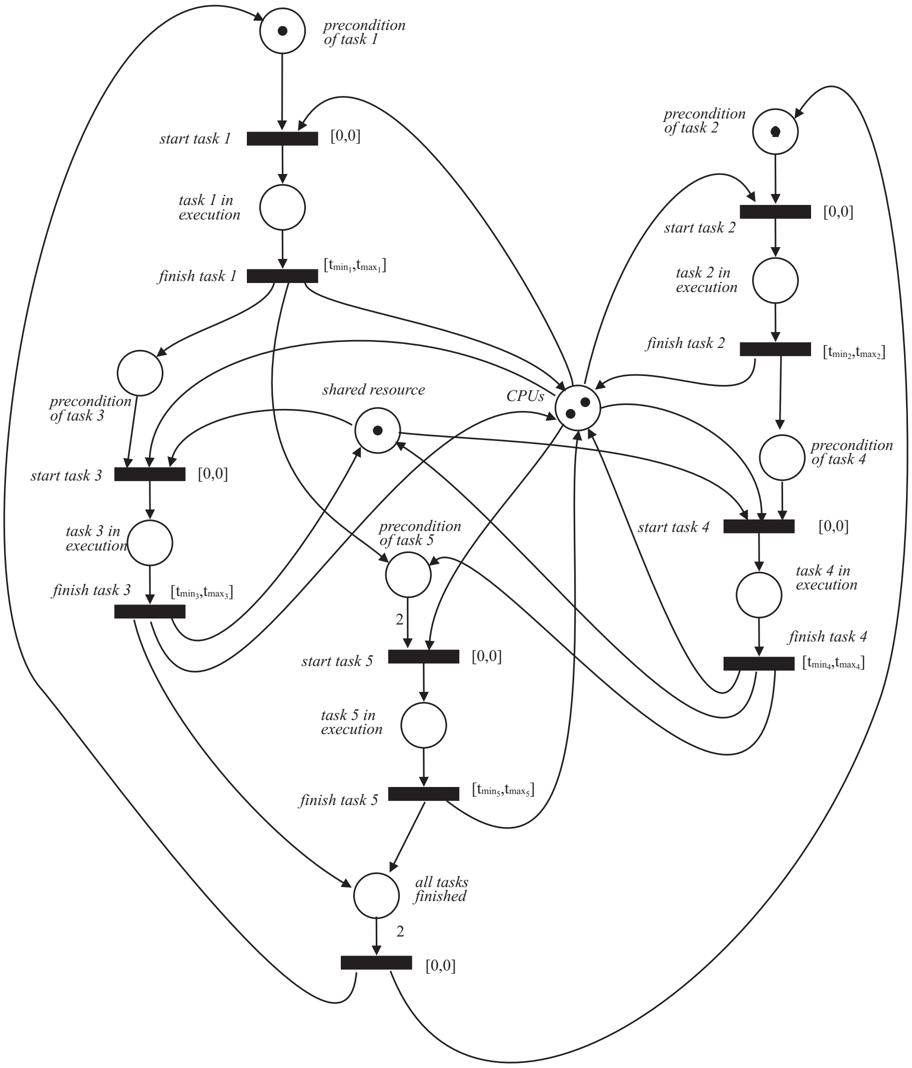

Figure 4. Interval Petri net model

```
Source node nr.> 1
Counting times
Target node nr.> 11
maximal distance = 32
A maximal path:
1 ==> 2 ==> 3 ==> 22 ==> 23 ==> 24 ==> 7 ==> 8 ==> 9 ==> 10 ==> 11
```

Figure 5. INA log for maximal path of the example

(a) Task 2 finishes before task 1

(b) Task 1 finishes before task 2

Figure 6. Two possible executions

6. Conclusions

Within this paper, we introduced a new approach to determine worst-case and best-case times between two states in a TPN in polynomial time and demonstrated the application of our method for a task scheduling problem, facilitating the INA tool.

References

[1] Berthomieu, B., Diaz, M.: Modeling and Verification of Time Dependent Systems Using Time Petri Nets, *Advances in Petri Nets 1984*, 17, No. 3, 1991.

[2] Berthomieu, B., Menasche, M.: An Enumerative Approach for Analyzing Time Petri Nets, *Proceedings IFIP* (R. E. A. Masom (ed.), Ed.), 17, No. 3, North-Holland, 1983.

[3] Boucheneb, H., Berthelot, G.: Towards a Simplified Building of Time Petri Net Reachability Graphs, *Proceedings of Petri Nets and Performance Models PNPM 93*, IEEE Computer Society Press, Toulouse France, 1993.

[4] Bucci, G., Fedeli, A., Sassoli, L., Vicario, E.: Timed State Space Analysis of Real-Time Preemptive Systems, *IEEE Transactions on Software Engineering*, **30**(2), 2004, 97–111.

[5] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C.: *Introduction to Algorithms*, second edition, MIT Press, 2001.

[6] Graham, R.: Bounds on the performance of scheduling algorithms, in: *Computer and job scheduling theory* (E. Coffman, Ed.), John Wiley and Sons, 1976, 165–227.

[7] Heiner, M., Popova-Zeugmann, L.: Worst-case Analysis of Concurent Systems with Duration Interval Petri Nets, *Entwurf komplexer Automatisierungssysteme* (E. Schnieder, D. Abel, Eds.), TU Braunschweig, IfRA, 1997.

[8] Lui, C. L., Layland, J. W.: Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Enviroment, *JACM*, **20**(1), January 1973, 46–61.

[9] Merlin, P.: *A Study of the Recoverability of Communikation Protocols*, Ph.D. Thesis, University of California, Computer Science Dept., Irvine, 1974.

[10] Popova-Zeugmann, L.: *Zeit-Petri-Netze*, Ph.D. Thesis, Humboldt-Universität zu Berlin, 1989.

[11] Popova-Zeugmann, L.: On Parametrical Sequences in Time Petri Nets, *Proceedings of the CS&P'97 Workshop* (H.-D. Burkhard, L. Czaja, P. Starke, Eds.), Warsaw, 1997.

[12] Popova-Zeugmann, L., Schlatter, D.: Analyzing Path in Time Petri Nets, *Fundamenta Informaticae 37, IOS Press*, 1999, 311–327.

[13] Stankovic, J. A., Spuri, M., Natale, M. D., Buttazzo, G. C.: Implications of Classical Scheduling Results for Real-Time Systems, *IEEE Computer*, **28**(6), June 1995, 16–25.

[14] Starke, P. H.: *INA - Integrated Net Analyzer*, Berlin, 1997, Manual.