Hanife APAYDIN ÖZKAN / IU-JEEE Vol. 16(2), (2016), 2073-2079

Shortest Path Algorithms For Petri Nets

Hanife APAYDIN ÖZKAN

Department of Electrical and Electronics Engineering, Anadolu University, Eskişehir, Turkey
hapaydin1@anadolu.edu.tr

Abstract: *Petri net is a mathematical and graphical tool for modeling and analysing discrete event systems. This paper focuses on a driving Petri net system from a given initial state to a desired state via minimum number of operations, that is, through the shortest transition sequence which is called as the shortest path problem. Thereby, two algorithms are developed to obtain the shortest path for Petri nets. The first algorithm, namely Forward Algorithm, uses integer programming approach and makes the process start from the initial state towards the desired state. The second algorithm, namely Backward Algorithm, uses most promising state to generate the shortest path and makes the process start from the desired state back to the initial state. Proposed algorithms do not deal with the reachability tree or graph of the net under analysis and use memory only for storing the obtained paths unlike the approaches based on the reachability tree. Moreover, the algorithms can be applied to general Petri nets without any restriction. Simulation results demonstrate that the proposed algorithms reduces the computational time and complexity significantly.*
Keywords: *Petri net, Shortest path, Algorithms.*

1. Introduction

Petri nets are frequently used for modeling and analysing Discrete Event Systems such as communication protocols, manufacturing systems, transport systems and others [1,2,3,4].

In most cases, it is interested in driving a system from a given initial state to a desired state, and obtaining the required operation sequence. For minimizing the cost, it is aimed to reach the desired state by minimum number of operations. This problem is called as *shortest path problem* for Petri nets and aims to compute the shortest transition sequence, i.e., transition sequence with the minimum number of transitions, firing which from the initial state drives the system to the desired state.

Reachability analysis techniques and shortest path problems are strictly connected, since the reachability analysis techniques are utilized to solve shortest path problems. The main reachability analysis technique of Petri nets is the reachability tree (or reachability graph) approach which is obtained by enumerating all reachable states. But, constructing the reachability tree and investigating the paths towards a desired state would be rather difficult task. Many works have been presented on this subject. The author in [7] deals the shortest path problem directly and the length of the path is obtained for some subclasses of Petri nets in that work. But the transition sequence of the corresponding path is not referred. In [5], Petri nets without transition invariants are studied, while only strictly monotone Petri nets are considered in [6]. In [3], reachability condition of some subclasses of Petri nets is exhibited. The authors of [7] propose a method to generate a reduced reachability tree for a class of unbounded generalized Petri nets. The work in [8] proposes a shortest path based on ant colony algorithm, while in [9] a Petri net based shortest path algorithm is presented. Although these works upgrade the reachabiliy analysis techniques and propose new perspectives for the solution of path problems for some subclasses, their complexity and expansions are still open.

In this work, two algorithms are developed to obtain the shortest path leading the system from the given initial state to the desired state. The first algorithm, namely Forward Algorithm, uses integer programming approach and makes the process start from the initial state towards the desired state. The second algorithm, namely Backward Algorithm, which is the improved version of the one presented in [10] by the author, uses the most promising state to generate the shortest path and makes the process start from the desired state back to the initial state.

The main contributions of this paper are as follows: The proposed algorithms do not deal with the reachability tree or reachability graph of the net under analysis and use memory only for storing the obtained path as distinct from the approaches based on reachability tree. Moreover, developed algorithms can be applied to general Petri nets without any restriction. Simulation results demonstrate that the proposed algorithms reduces the computational time and complexity, significantly.

Received on: 28.02.2016
Accepted on: 12.04.2016

2. Petri Nets

This section provides a background on Petri nets related to the discussion of this work and presents an introductory example. The readers are recommended to see [2,3] for more detailed information.

Petri net is a tuple $G = \langle P, T, N, O \rangle$ where P is the set of *places*, T is the set of transitions, $\boldsymbol{N}: P \times T \to Z^+$ is the *input matrix* that specifies the weights of arcs directed from places to transitions, $\boldsymbol{O}: P \times T \to Z^+$ is the *output matrix* that specifies the weights of arcs directed from transitions to places. Here, Z^+ is the set of non-negative integer numbers.

$\boldsymbol{m}: P \to Z^+$ is a *state vector* or *marking vector* (shortly, *state* or *marking*, respectively), $m(p)$ indicates the number of tokens assigned by marking $\boldsymbol{m}$ to place $p \in P$, and the *initial state* of the system is denoted by $\boldsymbol{m}_0$. When an initial state is introduced to the net model G, the model is called as Petri net system and noted by (G, m_0)

A transition $t \in T$ is *enabled* if and only if

$$m(p) \geq N(p,t) \qquad \forall p \in P \tag{1}$$

which is called as *enability condition*. The set of transitions which are enabled at a marking $\boldsymbol{m}$ is denoted by $E(G, \boldsymbol{m})$.

A *transition sequence* **g** is defined as firing sequence of enabled transitions $t_1 t_2 \mathrm{K}\, t_k$ where $t_1, t_2, \mathrm{K}\, t_k \in T$. A marking $\boldsymbol{m}'$ is said to *reachable* from $\boldsymbol{m}$ if there exists a transition sequence $\boldsymbol{g} = t_{i1}, t_{i2}, \mathrm{K} \quad 1 \leq i_k \leq |T|, \mathrm{k} = 1,2,\mathrm{K}$ which can be fired starting from $\boldsymbol{m}$ (i.e., the first transition of the sequence fires at $\boldsymbol{m}$) and yielding $\boldsymbol{m}'$ (i.e., the final transition of the sequence yields $\boldsymbol{m}'$) according to the following so-called *state equation*:

$$\boldsymbol{m}' = \boldsymbol{m} + \boldsymbol{C} \cdot \boldsymbol{\sigma}_g \tag{2}$$

where, $\boldsymbol{C} = \boldsymbol{O} - \boldsymbol{N}$ denotes the incidence matrix, $\boldsymbol{\sigma}_g : T \to Z^+$ denotes the firing count vector whose j^{th} element (represented by $[\boldsymbol{\sigma}_g]_j$) indicates how many times t_j is fired in the transition sequence $\boldsymbol{g}$. Besides, $\boldsymbol{m} \xrightarrow{g} \boldsymbol{m}'$ denotes that the marking $\boldsymbol{m}'$ is reachable from a marking $\boldsymbol{m}$ by firing the transition sequence $\boldsymbol{g}$. The set denoted by $\mathrm{R}(G, \boldsymbol{m})$ is the set of all markings reachable from $\boldsymbol{m}$. The set of all reachable markings from $\boldsymbol{m}_0$ is called *reachability set* of Petri net and represented by $\mathrm{R}(G, \boldsymbol{m}_0)$.

If $\exists \boldsymbol{y} \geq 0$ such that $\boldsymbol{y} \cdot \boldsymbol{C} = 0$ then, $\boldsymbol{y}$ is called as P-semiflow of the net and every marking $\boldsymbol{m}$ reachable from $\boldsymbol{m}_0$ satisfies:

$$\boldsymbol{y} \cdot \boldsymbol{m} = \boldsymbol{y} \cdot \boldsymbol{m}_0 \tag{3}$$

This provides a *token balance law* which is necessary for marking $\boldsymbol{m}$ to be reachable from $\boldsymbol{m}_0$, thus for satisfying $\boldsymbol{m} \in \mathrm{R}(G, \boldsymbol{m}_0)$.

For a given Petri net system, as many *new* markings as the number of the enabled transitions can be obtained from the initial marking. From each new marking, more new markings can be reached until repeated nodes are encountered "*old*" or no transitions are enabled "*dead-end*". This process yields tree representation of the evaluation of the system which is called as *reachability tree*. In the reachability tree each node represents a marking and the firing of a transition transforming one marking to another is represented by arcs.

A *reachability graph* of a PN is a directed graph $G = (\mathrm{R}(G, \boldsymbol{m}_0), E)$, where $e \in E$ represents a directed arc from a class of markings to the other class of markings. The reachability graph demonstrates a better performance than the reachability tree [11].

Example 1:

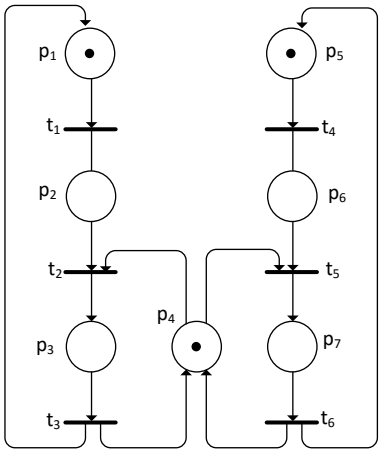

Figure 1. A simple Petri net model

For the Petri net system in Figure 1, the initial marking is $\boldsymbol{m}_0 = [1\ 0\ 0\ 1\ 1\ 0\ 0]^{\mathrm{T}}$.

The set of places is $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$ and the set of transitions is $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$, respectively. Input and output matrices of this Petri net model are as follows:

$$N = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad O = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Note that, P-semiflows of this net are $\boldsymbol{y_1} = [1\ 1\ 1\ 0\ 0\ 0\ 0]^{\mathrm{T}}$, $\boldsymbol{y_2} = [0\ 0\ 1\ 1\ 0\ 0\ 1]^{\mathrm{T}}$ and $\boldsymbol{y_3} = [0\ 0\ 0\ 0\ 1\ 1\ 1]^{\mathrm{T}}$. The reachability graph of the net is given in Figure 2.

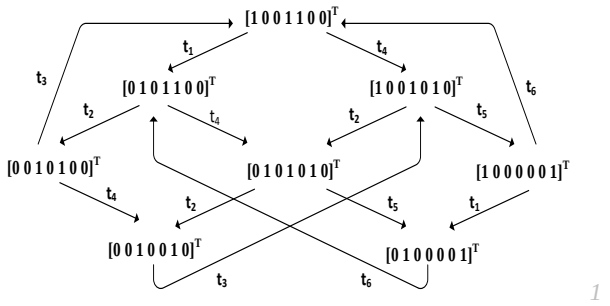

Figure 2. Reachability graph of the Petri net in Figure 1.

By using the reachability tree, it is possible to obtain all reachable markings and to get corresponding transition sequences to reach these markings. For example, it is clear from the tree that firing the transition sequence $\boldsymbol{g} = t_4 t_5$ from $\boldsymbol{m}_0$ drives the system to the marking $[1\ 0\ 0\ 0\ 0\ 0\ 1]^T$.

Unfortunatelly, for Petri nets with great number of reachable markings, using reachability graph to find the paths driving the system from the initial marking to desired marking is an exhausting way.

3. Shortest Path Algorithms

In this work, the main goal is to develop efficient techniques to solve shortest path problem of general Petri nets. Firstly, we give a formal shortest path definition for Petri nets.

Definition 1: Let $(G, \boldsymbol{m}_0)$ be a Petri net system and $\boldsymbol{m}_d \in \mathrm{R}(G, \boldsymbol{m}_0)$ be a desired marking. The transition-marking sequence leading $\boldsymbol{m}_0$ to $\boldsymbol{m}_d$ while including minimum number of transition firing is called as *shortest path.*

In order to obtain the shortest path for a Petri net system, an exhausting approach may be composing reachability tree as considered in many works in the literature [9,10,11] . But this is time consuming method especially for big sized Petri nets. In this work, two algorithms, namely Forward Algorithm and Backward Algorithm, will be introduced to determine the shortest path driving the system from $\boldsymbol{m}_0$ to $\boldsymbol{m}_d$ without constructing a reachability tree.

3.1. Forward Algorithm

In the Forward Algorithm, for a Petri net system $(G, \boldsymbol{m}_0)$ the shortest transition sequence driving the system from $\boldsymbol{m}_0$ to a desired marking $\boldsymbol{m}_d \in \mathrm{R}(G, \boldsymbol{m}_0)$ is attained by obtaining the fired transitions one by one starting from $\boldsymbol{m}_0$ towards $\boldsymbol{m}_d$.

Let $\boldsymbol{\sigma}_g$ be firing count vector of transition sequence **g** driving the system from the given initial marking to the desired target marking, i..e. $\boldsymbol{m}_0 \xrightarrow{g} \boldsymbol{m}_d$. This vector can be calculated by solving the state equation $\boldsymbol{m}_d = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}_g$ for $\boldsymbol{\sigma}_g$ with given $\boldsymbol{m}_0$ and $\boldsymbol{m}_d$ by using an integer programming method.

Integer Linear Programming (ILP) deals with problems that same as linear programming (LP) with the one additional restriction that the variables must have integer values. Compared to LP , ILP has additional constraints. An ILP begins with an LP, and adds the requirement that some or all of the variables take on integer values. This addition increases the number of problems that can be modeled, while making the models more difficult to solve. For solving ILP, the first step is to solve its LP relaxation, which is obtained by dropping the requirement that all variables must be integers. If all variables are integers in the solution of this relaxation, the solution of ILP has been found. Otherwise, methods like Branch and Bound (such as, Dakin's and Dijkstra etc,) can be used to find an integer solution (see [12] for details) which may increase the computational time slightly.

In the Forward algorithm the first step is to find firing count vector $\boldsymbol{\sigma}_g$ driving the system from $\boldsymbol{m}_0$ to $\boldsymbol{m}_d$ by the following ILP:

$$\begin{aligned} &\textit{Minimize} && \boldsymbol{\sigma}_g \\ &\textit{Subject\_to} && \boldsymbol{m}_d = \boldsymbol{m} + \boldsymbol{C} \cdot \boldsymbol{\sigma}_g \\ & && \boldsymbol{\sigma}_g \geq 0 \\ & && \left[\boldsymbol{\sigma}_g\right]_i \in Z^+ \qquad \forall i \in \left\{1, 2, \mathrm{K}\ , \left|\boldsymbol{\sigma}_g\right|\right\} \end{aligned} \tag{4}$$

After determining $\boldsymbol{\sigma}_g$ which provides the information of transitions and minimum number of times they should be fired to drive the system from $\boldsymbol{m}_0$ to $\boldsymbol{m}_d$, it is time to find their firing order. Here, it is proposed to build up a branch structure in order to find this order. Forward Algorithm generates node couples $(\boldsymbol{m}, {}^{m\rangle}\boldsymbol{\sigma}_g)$ which consists of the marking $\boldsymbol{m}$ and ${}^{m\rangle}\boldsymbol{\sigma}_g$ which is the set of ordered transition sequence driving the system from $\boldsymbol{m}_0$ to $\boldsymbol{m}$. This process starts with the *root* couple of $(\boldsymbol{m}_0, \varnothing)$ and labeling it as "*new*". From each *new* node couple $(\boldsymbol{m}, {}^{m\rangle}\boldsymbol{\sigma}_g)$, more *new* node couples are obtained by firing the enabled transitions which also belong to $\boldsymbol{\sigma}_g$ but not fired to reach the marking $\boldsymbol{m}$ of the considered node couple as the same number of times as in $\boldsymbol{\sigma}_g$. The set of these enabled transitions at $\boldsymbol{m}$, i.e. ${}^{m\rangle}T_{en}$, is obtained by ${}^{m\rangle}T_{en} = (E(G, \boldsymbol{m}) \cap {}^{\rangle}\boldsymbol{\sigma}_g) \setminus {}^{m\rangle}\boldsymbol{\sigma}_g$, where ${}^{\rangle}\boldsymbol{\sigma}_g$ is the set of transitions that includes the transitions as the same number of times as in $\boldsymbol{\sigma}_g$. When, a node couple of $\boldsymbol{m}_d$ is obtained, the algorithm terminates. Note that if the algorithm continues to process after obtaining $\boldsymbol{m}_d$, more than one shortest path can be obtained.

Forward Algorithm

Input: $(G, \boldsymbol{m}_0), \boldsymbol{m}_d$

Solve

Minimize $\boldsymbol{\sigma}_g$

$$\boldsymbol{m}_d = \boldsymbol{m}_0 + \boldsymbol{C}\cdot\boldsymbol{\sigma}_g$$

$$\boldsymbol{\sigma}_g \geq 0$$

$$\left[\boldsymbol{\sigma}_g\right]_i \in Z^+ \qquad \forall i \in \left\{1,2,\mathrm{K}\,,\left|\boldsymbol{\sigma}_g\right|\right\}$$

Label $(\boldsymbol{m}, {}^{m}\!\boldsymbol{\sigma}_g)$ as "*new*"

do-while ∃ "*new*" node couple

Select a "*new*" $(\boldsymbol{m}, {}^{m}\!\boldsymbol{\sigma}_g)$

Delete the label of $(\boldsymbol{m}, {}^{m}\!\boldsymbol{\sigma}_g)$

${}^{m}\!T_{en} = (E(G,\boldsymbol{m}) \cap \boldsymbol{\sigma}_g) \setminus {}^{m}\!\boldsymbol{\sigma}_g$

for i=1: $\left|{}^{m}\!T_{en}\right|$

$\boldsymbol{m}' = \boldsymbol{m} + \boldsymbol{C}\cdot\left[{}^{m}\!T_{en}\right]_i$

${}^{m'}\!\boldsymbol{\sigma}_g = {}^{m}\!\boldsymbol{\sigma}_g \cup \left[{}^{m}\!T_{en}\right]_i$

Introduce $(\boldsymbol{m}', {}^{m'}\!\boldsymbol{\sigma}_g)$ as a "*new*" node couple.

if $\boldsymbol{m}' = \boldsymbol{m}_d$

Return $(\boldsymbol{m}', {}^{m'}\!\boldsymbol{\sigma}_g)$

Exit

end-if

end-for

end-while

Example 2: Let us apply Forward Algorithm to the Petri net in Example 1. Let the initial and desired markings are $\boldsymbol{m}_0$ = [1 0 0 1 1 0 0]$^{\mathrm{T}}$ and $\boldsymbol{m}_d$ =[0 1 0 0 0 0 1]$^{\mathrm{T}}$, respectively. The solution of ILP (3) is obtained via its LP relaxation as $\boldsymbol{\sigma}_g$ =[1 0 0 1 0 1]$^{\mathrm{T}}$, thus each of transitions t_1, t_4 and t_5 should be fired at once in order to drive the system from $\boldsymbol{m}_0$ to $\boldsymbol{m}_d$. The first node couple of the branch structure is $(\boldsymbol{m}_0, \varnothing)$. The set of enabled transitions which also belong to $\boldsymbol{\sigma}_g$ but not fired to reach the marking $\boldsymbol{m}$ of the considered node couple as the same number of times as in $\boldsymbol{\sigma}_g$ is ${}^{m_0}\!T_{en} = (\{t_1,t_4\} \cap \{t_1,t_4,t_5\}) \setminus \varnothing = \{t_1,t_4\}$. From $\boldsymbol{m}_0$, firing the transition t_1 yields $\boldsymbol{m}_1$ =[0 1 0 1 1 0 0]$^{\mathrm{T}}$, while firing the transition t_4 yields $\boldsymbol{m}_2$ =[1 0 0 1 0 1 0]$^{\mathrm{T}}$. Hence, the *new* node couples are $(\boldsymbol{m}_1, \{t_1\})$ for which ${}^{m_1}\!T_{en} = \{t_4\}$ and $(\boldsymbol{m}_2, \{t_4\})$ for which ${}^{m_2}\!T_{en} = \{t_5\}$ (see Figure 3). Firing transition t_4 from $\boldsymbol{m}_1$ of $(\boldsymbol{m}_1, \{t_1\})$ yields $\boldsymbol{m}_3$ =[0 1 0 1 1 0 0]$^{\mathrm{T}}$ and firing transition t_5 from $\boldsymbol{m}_2$ of $(\boldsymbol{m}_2, \{t_4\})$ yields $\boldsymbol{m}_4$ =[1 0 0 0 0 0 1]$^{\mathrm{T}}$. Hence, the *new* node couples are $(\boldsymbol{m}_3, \{t_1,t_4\})$ for which ${}^{m}\!T_{en} = \{t_5\}$ and $(\boldsymbol{m}_4, \{t_4,t_5\})$ for which ${}^{m}\!T_{en} = \{t_1\}$ (see Figure 4). By firing the transition t_1 from $\boldsymbol{m}_4$ of $(\boldsymbol{m}_4, \{t_4,t_5\})$, $\boldsymbol{m}_d$ is reached. Finally, the last node couple of the algorithm is obtained as ([0 1 0 0 0 0 1]$^{\mathrm{T}}$, $\{t_4,t_5,t_1\}$) (which represents the path $\boldsymbol{m}_0 \xrightarrow{t_4 t_5 t_1} \boldsymbol{m}_d$) and the algorithm terminates, with the following shortest path: $\boldsymbol{m}_0$ = [1 0 0 1 1 0 0]$^{\mathrm{T}} \xrightarrow{t_4}$ [1 0 0 1 0 1 0]$^{\mathrm{T}} \xrightarrow{t_5}$ [1 0 0 0 0 1]$^{\mathrm{T}} \xrightarrow{t_1}$ $\boldsymbol{m}_d$ = [0 1 0 0 0 0 1]$^{\mathrm{T}}$ (see Figure 5).

Note that if the algorithm continues to evaluate, after $\boldsymbol{m}_d$ is reached, an alternative shortest path would be obtained (see Figure 6).

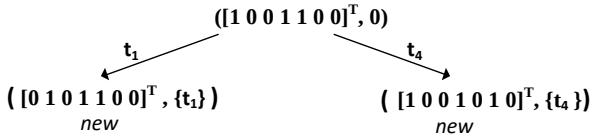

Figure 3. The first step of the Forward Algorithm

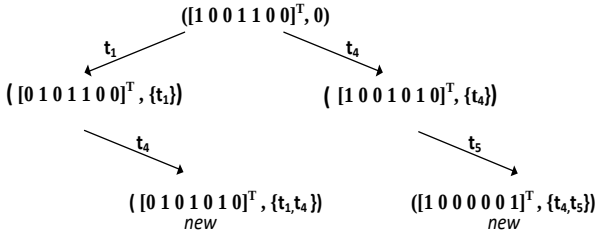

Figure 4. The second step of the Forward Algorithm

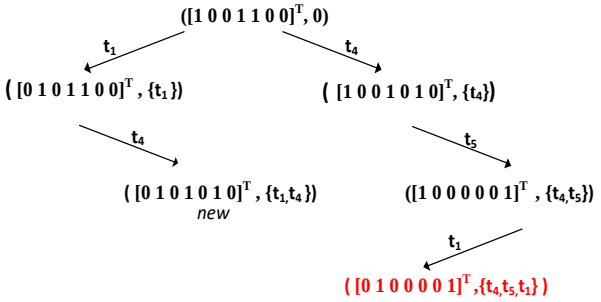

Figure 5. The final step of the Forward Algorithm

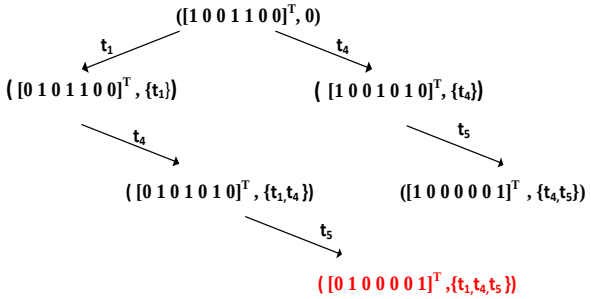

Figure 6. An alternative shortest path for Example 2

3.2. Backward Algorithm

In the Backward Algorithm, for a Petri net system $(G, \boldsymbol{m}_0)$ the transition sequence driving the system from $\boldsymbol{m}_0$ to $\boldsymbol{m}_d \in \mathrm{R}(G, \boldsymbol{m}_0)$ is attained by obtaining the fired transitions one by one starting from $\boldsymbol{m}_d$ back to $\boldsymbol{m}_0$.

If a transition $t_j \in T$ is fired at a marking $\boldsymbol{m}$ yielding $\boldsymbol{m}'$, element-wise representation of the state equation can be written as

$$\boldsymbol{m}'(p_i) = \boldsymbol{m}(p_i) + \boldsymbol{O}(p_i, t_j) - \boldsymbol{N}(p_i, t_j) \;\; \forall p_i \in P \qquad (5)$$

yielding

$$\boldsymbol{m}'(p_i) - \boldsymbol{O}(p_i, t_j) = \boldsymbol{m}(p_i) - \boldsymbol{N}(p_i, t_j) \;\; \forall p_i \in P \qquad (6)$$

Necessary condition for transition t_j to be enabled at a marking $\boldsymbol{m}$ yielding $\boldsymbol{m}'$ can be reformulated in terms of $\boldsymbol{m}'$ by combining equation (5) and enabling condition (1) together:

$$\boldsymbol{m}'(p_i) - \boldsymbol{O}(p_i, t_j) \geq 0 \;\; \forall p_i \in {}^{\bullet}t_j \qquad (7)$$

where ${}^{\bullet}t_j$ denotes the set of all places from which there exists a directed arc to t_j, i.e., ${}^{\bullet}t_j = \{ p \,|\, N(\mathrm{p}, t_j) \geq 1 \}$.

From a marking $\boldsymbol{m} \in \mathrm{R}(G, \boldsymbol{m}_0)$, firing any transition $t_j \in T$ which satisfies the condition (6) may yield $\boldsymbol{m}'$. Corresponding $\boldsymbol{m}$ is obtained by solving the following equation system:

$$\begin{aligned} \boldsymbol{m} &= \boldsymbol{m}' - \boldsymbol{O}(:, t_j) - \boldsymbol{N}(:, t_j) \quad \text{(a)} \\ \boldsymbol{y}^T \boldsymbol{m} &= \boldsymbol{y}^T \boldsymbol{m}' \qquad\qquad\;\; \text{(b)} \end{aligned} \qquad (8)$$

where equation (7)(a) is the state equation and equation (7)(b) corresponds to reachability of $\boldsymbol{m}$ from $\boldsymbol{m}_0$, i.e. $\boldsymbol{m} \in \mathrm{R}(G, \boldsymbol{m}_0)$ [2].

Backward Algorithm is based on generating couples $(\boldsymbol{m}, {}^{m}\gamma_g)$ which consists of a marking $\boldsymbol{m}$ and ${}^{m}\gamma_g$ which is the set of ordered transition sequence driving the system from $\boldsymbol{m}$ to $\boldsymbol{m}_d$. The marking of each node couple $(\boldsymbol{m}, {}^{m}\gamma_g)$ is definitely obtained by firing transitions $t \in \vec{T}^m$ from preceding marking $\boldsymbol{m}'$, such that $\boldsymbol{m}' \xrightarrow{t} \boldsymbol{m}$. Transitions of $\vec{T}^m$ are obtained according to the condition (7) and the preceding markings are obtained by equation system in (8). Each node couple obtained by this way is introduced as *new*. The most promising *new* node couple, thus including closest marking to $\boldsymbol{m}_0$, is introduced as *path*, while the other *new* node couples are deleted. The algorithm starts with the node couple of $(\boldsymbol{m}_d, \varnothing)$ and introducing it as *path*. If the *path* includes $\boldsymbol{m}_0$ the algorithm terminates, otherwise preceding *new* node couples are obtained.

Backward Algorithm

Input: $(G, \boldsymbol{m}_0), \boldsymbol{m}_d$

Label $(\boldsymbol{m}_d, \varnothing)$ as *path*

do-while $\boldsymbol{m} \neq \boldsymbol{m}_0$

New= $\varnothing$

Take the node couple *path* $(\boldsymbol{m}', {}^{m}\gamma_g)$

$\vec{T}^m = \{ t \,|\, \boldsymbol{m}'(p_i) - \boldsymbol{O}(p_i, t) \geq 0 \quad \forall p_i \in {}^{\bullet}t \}$

for i=1: $|\vec{T}^m|$

Solve the following equation system for $\boldsymbol{m}$

$$\boldsymbol{m} = \boldsymbol{m}' - \boldsymbol{O}(:, [\vec{T}^m]_i) - \boldsymbol{N}(:, [\vec{T}^m]_i)$$

$$\boldsymbol{y}^T \boldsymbol{m} = \boldsymbol{y}^T \boldsymbol{m}'$$

${}^{m}\gamma_g = [\vec{T}^m]_i \; {}^{m'}\gamma_g$

Introduce $(\boldsymbol{m}, {}^{m}\gamma_g)$ as a *new* node couple.

end-for

Select the most promising node couple $(\boldsymbol{m}, {}^{m}\gamma_g)$ through the *new* node couples via:

$$\text{st. } |\boldsymbol{m} - \boldsymbol{m}_0| = \min_{\bar{\boldsymbol{m}} \in New} \{ |\bar{\boldsymbol{m}} - \boldsymbol{m}_0| \}$$

Introduce $(\boldsymbol{m}, {}^{m}\gamma_g)$ as *path*

Delete *new* nodes

if $\boldsymbol{m} = \boldsymbol{m}_0$

Return $(\boldsymbol{m}_0, {}^{m_0}\gamma_g)$

Exit

end-if

end-while

Example 3: Let us consider the PN system given in Example 2, again with the same initial and desired markings, i.e., $\boldsymbol{m}_0 = [1\ 0\ 0\ 1\ 1\ 0\ 0]^T$ and $\boldsymbol{m}_d = [0\ 1\ 0\ 0\ 0\ 0\ 1]^T$. Backward algorithm starts with the node couple $(\boldsymbol{m}_d, \varnothing)$ and introducing it as *path*. Transitions firing which may yield $\boldsymbol{m}_d$ are t_1 and t_5 and the corresponding preceding markings are $\boldsymbol{m}_1 = [1\ 0\ 0\ 1\ 0\ 1\ 0]^T$ (i.e., $\boldsymbol{m}_1 \xrightarrow{t_1} \boldsymbol{m}_d$) and $\boldsymbol{m}_2 = [0\ 1\ 0\ 1\ 0\ 1\ 0]^T$ (i.e., $\boldsymbol{m}_2 \xrightarrow{t_5} \boldsymbol{m}_d$), respectively. Hence, *new* node couples are $(\boldsymbol{m}_1, t_1)$ and $(\boldsymbol{m}_2, t_5)$ (see Figure 7). Through these *new* node couples the most promising one is $(\boldsymbol{m}_2, t_5)$ which is introduced as *path* and the other *new* node couple is deleted. Transitions firing which may yield $\boldsymbol{m}_2$ of $(\boldsymbol{m}_2, t_5)$ are t_4 and t_2 with the corresponding preceding markings are $\boldsymbol{m}_3 = [0\ 1\ 0\ 1\ 1\ 0\ 0]^T$ and $\boldsymbol{m}_4 = [1\ 0\ 0\ 1\ 0\ 1\ 0]^T$, respectively. Hence the *new* node couples are obtained as $(\boldsymbol{m}_3, t_4 t_5)$ and $(\boldsymbol{m}_4, t_2 t_5)$ (see Figure 8). Among them, $(\boldsymbol{m}_4, t_2 t_5)$ is the most promising one through these *new* node couples and introduced as *path* while the other one is deleted (see Figure 9). The transition, firing which yields $\boldsymbol{m}_4$ of $(\boldsymbol{m}_4, t_2 t_5)$ is t_4 with the preceding marking $\boldsymbol{m}_0 = [0\ 1\ 0\ 1\ 1\ 0\ 0]^T$. Hence the last node couple is $(\boldsymbol{m}_0, t_4 t_2 t_5)$ presenting the following shortest path: $\boldsymbol{m}_0$ = [0

$1\ 0\ 1\ 1\ 0\ 0]^T \xrightarrow{t_4} [1\ 0\ 0\ 1\ 0\ 1\ 0]^T \xrightarrow{t_2} [0\ 1\ 0\ 1\ 0\ 1\ 0]^T \xrightarrow{t_5}$ $\boldsymbol{m}_d = [0\ 1\ 0\ 0\ 0\ 0\ 1]^T$ (See Figure 10)

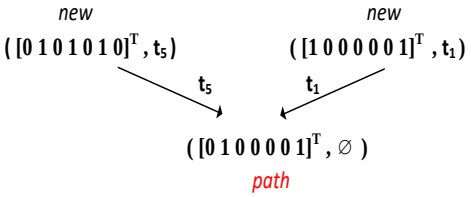

Figure 7. The second step of the Backward Algorithm

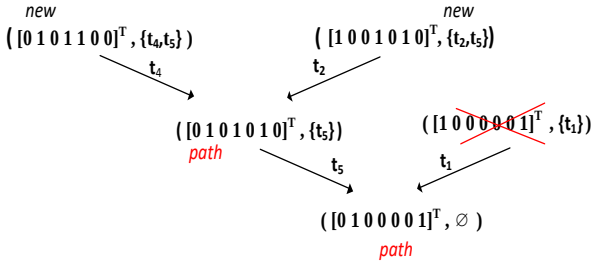

Figure 8. The fourth step of the Backward Algorithm

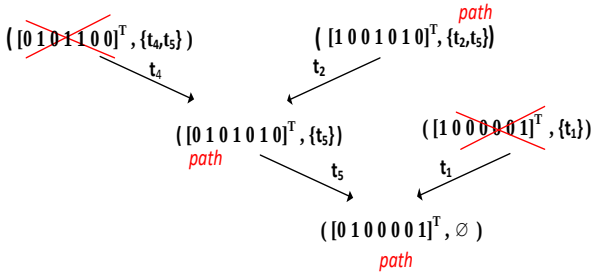

Figure 9. The third step of the Backward Algorithm

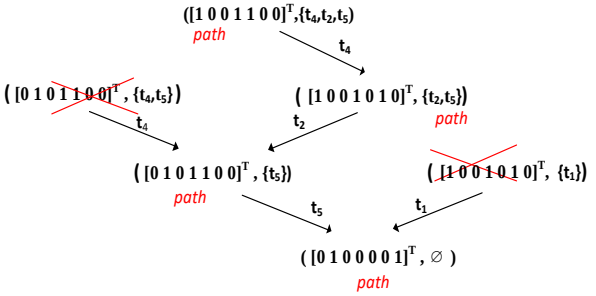

Figure 10. The final step of the Backward Algorithm

4. Case Study

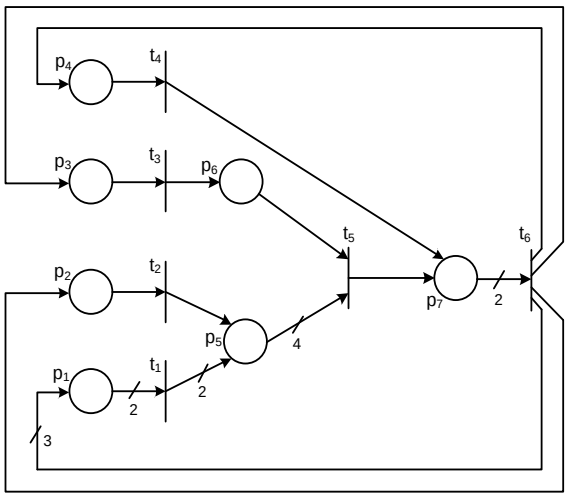

Figure 11. A Petri net model of a table factory system

Let us consider the Petri net system sketched in Figure 11 (taken from [13]) which models a table factory system.This system, consists of two different machines to make table-legs, a new one which produces two legs at a time, and the old one, which makes legs one by one (t_1 and t_2), a machine to produce the table boards (t_3), a machine to assemble four legs and a board (t_5), a big painting line which paints two tables at once (t_6). More unpainted tables are sent (t_4) from another factory. The places p_1, p_2, p_3 and p_4 are work orders; while p_5, p_6 and p_7 are devoted to the storage of table-legs, boards and unpainted tables, respectively (see [17] for details). Suppose in the initial marking $m_0(p_1)=m_0(p_4)=2$, $m_0(p_2)=m_0(p_3)=m_0(p_5)=1$ and $m_0(p_6)=m_0(p_7)=0$. For the final state, work orders are desired as $m_d(p_1)=m_d(p_2)=m_d(p_3)=m_d(p_4)=1$; while the number of stored table legs is 6, i.e. $m_d(p_5)=6$. It is also desired to keep the number of stored boards $m_d(p_7)=1$ and store 1 unpainted table, i.e., $m_d(p_6)=1$. For this net $R(G,\boldsymbol{m}_0)$ has 294 marking vectors and constructing the reachability tree of the net takes 33.2 sec. by using the Matlab toolbox "pntool" (on a PC with Intel(R) Core(TM) i7 2640M CPU @ 2.8GHz).Searching the shortest path from $\boldsymbol{m}_0$ to $\boldsymbol{m}_d$ through this tree takes 4.1 sec. additionally. For the same shortest path, LP relaxation of ILP results in integer solution and the proposed Forward Algorithm terminates with the node couple $([1\ 1\ 1\ 1\ 6\ 1\ 0]^T, \{t_4, t_2, t_4, t_4, t_6, t_1, t_3\})$ in 3.4 sec; while the Backward Algorithm terminates with $([1\ 1\ 1\ 1\ 6\ 1\ 0]^T, \{t_4, t_3, t_4, t_4, t_6, t_1, t_2\})$ in 3.5 sec. (see Figure 12). The proposed algorithms reduce the computational time about 5% of the time consumed by conventional reachability tree methods.

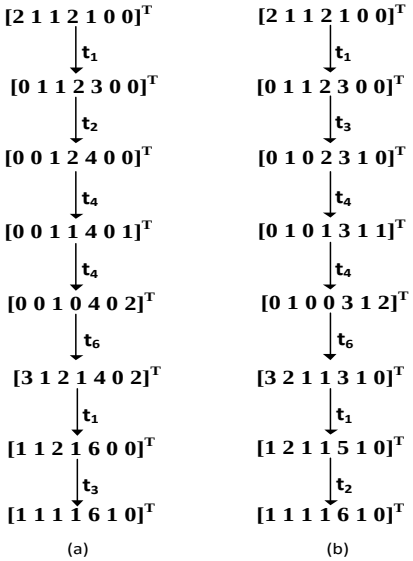

Figure 12. Shortest paths obtaind by (a) Forward Algorithm and (b) Backward Algorithm

5. Conclusions

In this work, shortest path problem is considered from the view of Petri nets and two algorithms are developed to obtain the shortest path leading the system from the given initial state to the desired state with the minimum number of transition firing (operation). The first algorithm, namely Forward Algorithm, uses ILP approach and makes the process start from the initial marking towards the desired marking. The second algorithm, namely Backward Algorithm uses most promising marking vector to generate the shortest path and makes the process start from the desired marking back to the initial marking. In the case that LP relaxation of the ILP results in integer solution without applying other methods (such as Daikin's and Dijkstra) Forward Algorithm finds the shortest path in minimum time. Otherwise, Backward Algorithm obtains the shortest path in a little less time compared to Forward Algorithm. The proposed algorithms do not deal with the reachability tree or reachability graph of the net under analysis and use memory only for storing the obtained path as distinct from the approaches based on reachability tree. Moreover, developed algorithms can be applied to general Petri nets without any restriction. Simulation results demonstrate that the proposed algorithms reduces the computational time and complexity significantly.

6. Acknowledgement

This work was supported by Anadolu University through Research Project 1501F019.

7. References

[1] Desel, J., "Shortest paths in reachability graphs". *Journal of Computer and System Sciences*, 51:314–323,1995.

[2] Desrochers, A. A. and Al-Jaar, R. Y. , "Applications of Petri nets in Manufacturing Systems", The Institute of Electrical Electrical and Electronics Engineers Inc., New York, 1995.

[3] Kostin, A., "Reachability analysis in t-invariant- less Petri nets. *IEEE Transactions on Automatic Control*, 48, Issue: 6:10-19 1024, 2003.

[4] Murata, T., "Petri nets: properties, analysis and applications" *Proceedings of the IEEE*, 77 No:4:541– 580, 1989.

[5] Proth, J. and Xie, X. , "Petri Nets: A Tool for Design and Management of Manufacturing Systems. John Wiley & Sons, West Sussex, 1996.

[6] Ru, Y. and C.N.Hadjicostis, "Reachability analysis for a class of petri nets , In *Proceedings of IEEE Conference on Decision and Control*, 2009, pp. 1261–1266, Shanghai.

[7] S. Wang; M. Gan; M. Zhou; D. You, "A reduced reachability tree for a class of unbounded petri nets," in *IEEE/CAA Journal of Automatica*, 2, 4:345-352, 2015.

[8] Y. Haoxiong and H. Yang, "Congested traffic based on ant colony algorithm for shortest path algorithm," in *International Conference on Logistics, Informatics and Service Sciences (LISS)*,2015 , pp.1-3, July.

[9] Y. Zheng; K. Hou; W. Liao and L.Yang., "The Shortest Path Algorithm Based on Petri Net," in 19th International Conference on Industrial Engineering and Engineering Management, 2013 , pp.221-229, April.

[10] Apaydin-Özkan, H., "On Achieving Reachability Paths of Petri nets," in 9th International Conference on Electrical and Electronics Engineering (ELECO), 2015, pp.724-728, November.

[11] X. Ye, J. Z. and Song, X., "On reachability graphs of Petri nets", *Computers and Electrical Engineering*, 29, 2:263-272, 2003.

[12] M.Conforti, G.Cornuejols, G. Zambelli and Giacomo., "Integer Programming", Springer International Publishing, 2014.

[13] E. Teruel, J. M. Colom, M. Silva, "Choice-free Petri nets: A model for deterministic concurrent systems with bulk services and arrivals", *IEEE Transactions on Systems, Man, and Cybernetics*, **1**(27): 73–83, 1997

Hanife Apaydın Özkan received the B.S. degree in electrical and electronics engineering from the Osmangazi University, in 2002, the M.S.and the Ph.D. degree in automatic control from Anadolu University, in 2005 and 2010, respectively and all in Eskişehir, Turkey. She was a researcher with the Department of Computer Science and Systems Engineering of the University of Zaragoza, Spain, from Sep. 2008 to Jan 2010. She is currently an Assistant Professor at the Department of Electrical and Electronics Engineering of the Anadolu University, Turkey. Her current research interests include control of discrete event systems, Petri net theory and application, control and scheduling of home appliances, smart home energy management.