On Achieving Reachability Paths of Petri nets

Hanife Apaydin Özkan

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

Abstract

Petri net is a powerful graphical and mathematical tool for analysis and design of discrete event systems. This paper focuses on the reachability path problems in Petri nets. Thereby, two algorithms are developed to create the set of minimal paths and the shortest path to lead the system from the given initial state to a desired state. Both of them are enlightened by dynamic programming approach; that is to say, they are backward techniques. 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.

1. Introduction

Petri nets are frequently used for modeling and analysis of Discrete Event Systems (DESs) 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. In terms of Petri nets this type of problems are called as reachability path problems, such as minimal path problem and shortest path problem. Obtaining the paths that do not go through the same state multiple times is called as minimal path problem. For minimizing the cost, it is aimed to reach the desired state by minimum number of operations. For Petri nets, this problem is called as shortest path problem which 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 reachability path problems are strictly connected, since the reachability analysis techniques may also be utilized to solve reachability path problems. Two main reachability analysis techniques of Petri nets are reachability tree approach which is obtained by enumerating all reachable markings and the matrix equation approach. But, constructing the reachability tree and investigating the paths to a desired state would be rather difficult task. The main drawback in the use of matrix equation approach is that the solution of the fundamental equation does not give any information about the order of firings.

Many works have been presented to contribute the reachability analysis and the reachability path problems. The author in [7] deals shortest path problem directly and the length of the shortest 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. Although, these works upgrade the reachability analysis techniques for some subclasses, their complexity and expansions are still open.

In this paper, two algorithms are developed to solve minimal paths and shortest path problems. Both of them are enlightened by dynamic programming approach, thus they are backward techniques. Given an initial state and a desired state, the first algorithm obtains the shortest path while the second one results in all minimal paths driving the system from the initial state to the desired state.

The main contributions of this paper are as follow: 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.

2. Petri nets

This section provides 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 \(\mathcal{N} = (P,T,N,O)\), where \(P\) is the set of places, \(T\) is the set of transitions, \(N : P \times T \rightarrow \mathcal{N}\) is the input matrix that specifies the weights of arcs directed from places to transitions, \(O : P \times T \rightarrow \mathcal{N}\) is the output matrix that specifies the weights of arcs directed from transitions to places. Here, \(\mathcal{N}\) is the set of non-negative integer numbers. \(M : P \rightarrow \mathcal{N}\) is a marking vector (or marking), \(M(p)\) indicates the number of tokens assigned by marking \(M\) to place \(p \in P\), and the initial marking of the system is denoted by \(m_0\).

A transition \(t \in T\) is enabled if and only if

\[
M(p) \geq N(p,t) \quad \forall p \in P.
\]

which is called as enableability condition. The set of transitions which are enabled at a marking \(M\) is denoted by \(E(G,M)\).

A transition sequence \(g\) is defined as firing sequence of enabled transitions \(t_1t_2\ldots t_k\), where \(t_1, t_2, \ldots, t_k \in T\). A marking \(M'\) is said to be reachable from \(M\) if there exists a transition sequence \(g = t_k t_{k-1} \ldots t_1\) where \(1 \leq i \leq |T|, k = 1, 2, \ldots\) which can be fired starting from \(M\) (i.e., the first transition of the sequence fires at \(M\)) and yielding \(M'\) (i.e., the final transition of the sequence yields \(M'\)) according to the following so-called state equation:

\[
M' = M + Cg_g
\]
where, $C := O - N$ denotes the incidence matrix, $\sigma_g : T \rightarrow \mathbb{Z}$ denotes the firing count vector whose $j^{th}$ element indicates how many times $t_j$ is fired in the transition sequence $g$. $M \stackrel{g}{\rightarrow} M'$ denotes that the marking $M'$ is reachable from a marking $M$ by firing the transition sequence sequence $g$. The set denoted by $R(G,M)$ is the set of all markings reachable from $M$. The set of all reachable markings from $m_0$ is called reachability set of Petri net and denoted by $R(G,m_0)$.

If $\exists y \geq 0$ such that $y \cdot C = 0$ then, $y$ is called as P-semiflow of the net and every marking $M$ reachable from $m_0$ satisfies:

$$y \cdot M = y \cdot m_0$$  \hspace{1cm} (3)

This provides a token balance law which is the necessary reachability condition.

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 $G = (R(G,m_0), E)$, 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 [8].

Example 1: Let us consider the PN in Fig. 1 with $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$, $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$ and $m_0 = [1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0]^T$. P-semiflows of this net are $P_1 = [1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0]^T$, $P_2 = [0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0]^T$ and $P_3 = [0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1]^T$.

3. Reachability Path Problems: Shortest Path & Minimal Paths

In this work, the main goal is to develop efficient techniques to solve reachability path problems of general Petri nets. Firstly, we give more formal definitions for the reachability path problems considered in this work.

Definition 3.1 (Shortest Path). Let $(N, m_0)$ be a Petri net system and $M_d \in R(G,m_0)$ a marking. The transition-marking sequence leading $m_0$ to $M_d$ while including minimum number of transition firing is called as shortest path.

Definition 3.2 (Minimal Path). Let $(N, m_0)$ be a Petri net and $M_d \in R(G,m_0)$ a marking. The transition-marking sequence leading $m_0$ to $M_d$ without passing through the same marking multiple times is called as minimal path.

In order to solve reachability path problems above, an exhausting approach may be composing reachability tree. But this is time consuming method especially for big sized Petri nets. The proposed backward method is introduced in the following part.

3.1. Backward Method

According to the proposed backward method, for a Petri net system $(N, m_0)$ the transition sequence driving the system from $m_0$ to $M' \in R(G,M)$ will be attained by obtaining the fired transitions one by one starting from $M_d$ and going back to $m_0$.

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

$$M'(p_i) = M(p_i) + O(p_i,t_j) - N(p_i,t_j), \forall p_i \in P$$  \hspace{1cm} (4)

yielding

$$M'(p_i) - O(p_i,t_j) = M(p_i) - N(p_i,t_j), \forall p_i \in P$$  \hspace{1cm} (5)

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

$$M'(p_i) - O(p_i,t_j) \geq 0, \forall p_i \in \bullet t_j$$  \hspace{1cm} (6)

where $\bullet t_j$ denotes the set of all places from which there exists a directed arc to $t_j$, i.e., $\bullet t_j = \{p_i | N(p_i,t_j) \geq 1, t_j \in T\}$. From a marking $M \in R(G,m_0)$, firing any transition $t_j \in T$ which satisfies the condition in (1) may yield $M'$. Corresponding $M$ is obtained by solving the following equation system:

![Figure 1. An example Petri net.](image)

![Figure 2. Reachability graph.](image)
Algorithm 1 Shortest Path

Input: \((\mathcal{N}, m_0), M_d\)

Path = \{(M_0,0)\}

\[ M' = M_d \]

do-while Path\_{Path\_i} \neq m_0

Candidates = θ

for \(j = 1 : |T|\)

if \(\mathcal{M}'(p_j) = O(t_j, t_j) \geq 0, \ \forall p_j \in \mathcal{S}_j\)

\[ M = M' - O(t_j, t_j) + N(t_j) \]

if \(B_t M' \neq B_t M' \notin Path \wedge M' \notin Forbidden\)

Candidates = Candidates \cup \{(M,t)\}

end-if

end-if

end-for

if Candidates ≠ θ

\textcolor{red}{Index(Candidates)}

Path = Path \cup Candidates

\[ M' = Path_{Path\_i} \]

else

Forbidden = Forbidden \cup M'

Path = Path_{Path\_i}

\[ M' = Path_{Path\_i} \]

end-while

Output: Path

Backward approach can also be used to find all minimal paths from \(m_0\) to \(M_d\). Algorithm 2 carries out the corresponding procedure. In the algorithm, Pathset is composed of the set of uncompleted paths starting from \(M_d\) and being constructed by going backward to \(m_0\). As in Algorithm 1, each path is represented by the ordered set of couples of state \(M \in R(G,m_0)\) and the fired transition \(t \in T\) from \(M\). Recall that, the node of \(M\) for this operation (i.e. \(M \rightarrow M'\)) is \((M,t)\)

Pathset\_{Path\_i} is the \(k^{th}\) \((k \in \{1,2\})\) component of the \(j^{th}\) node of the \(j^{th}\) path in Pathset. In Algorithm 2, an arbitrarily chosen path is removed from Pathset and assigned to a temporary path \(TPath\). For the 1st component of the last node of \(TPath\), i.e. \(M' = TPath_{Path\_i}\), all enabled transitions to reach \(M'\) are obtained. Then for each of these transitions the preceding marking, \(M\), from which the transition is fired and \(M'\) is reached, is calculated. In the case that \(M\) is reachable from \(m_0\) and it is not previously added to \(TPath\), then the node of \(M\) is added to \(TPath\). If \(TPath\) is completed, i.e. \(M = m_0\), \(TPath\) is added to the set of completed paths, FPathSet, otherwise it is added to the set of uncompleted paths, PathSet.

This process is continued until all candidate paths is completed by obtaining all nodes on the considered path from \(m_0\) to \(M_d\), that is Pathset is emptied.

Algorithm 2 Minimal Paths

Input: \((\mathcal{N}, m_0), M_d\)

PathSet = \{(M_0,0)\}

FPathSet = \emptyset

do-while PathSet \neq \emptyset

choose \(i \in \{1,2, \ldots, \left|\text{PathSet}\right|\}\)

\[ TPath = \text{PathSet}_i \]

PathSet = PathSet \backslash PathSet_i

\[ M = TPath_{Path\_i} \]

for \(j = 1 : |T|\)

if \(\mathcal{M}'(p_j) = O(t_j, t_j) \geq 0, \ \forall p_j \in \mathcal{S}_j\)

\[ M' = M - O(t_j, t_j) + N(t_j) \]

if \(B_t M' \neq B_t M' \notin TPath\)

\[ M' = m_0 \]

FPathSet = FPathSet \cup \{(TPath \cup (M', t_j))\}

end-if

PathSet = PathSet \cup \{(TPath \cup (M', t_j))\}

end-if

end-for

end-while

Output: FPathSet

Example 2: Let us consider the PN system considered in Example 1 again with desired target marking \(M_d = M_2\). In order to obtain the shortest path from \(m_0\) to \(M_d\) Algorithm 1 is executed, which processes as follow: At \(M_d\) the transitions satisfying the condition in (6) are \(t_2\) and \(t_4\). For transition \(t_4\), the preceding marking is obtained as \(M_2 (i.e. M_2 \overset{t_4}{\rightarrow} M_d)\).
for transition $t_2$, the preceding marking is obtained as $M_3$ (i.e. $M_3 \xrightarrow{t_2} M_d$). Since both $M_2$ and $M_3$ are reachable from $m_0$ (they satisfy the condition in (7b) and they have not already been added to the Path, Candidates = \{(M_2, t_4), (M_3, t_2)\}). The index function is called to choose the closest one to $m_0$ through the set Candidates. Since their distance from $m_0$ are same, either of them is added to Path, i.e. Path=\{(M_2, 0), (M_3, t_2)\}. Then the procedure is repeated for the marking of the last node of Path, i.e. $M_2$. The previously firable transitions and corresponding preceding markings forms new candidate nodes. Hence, this once Candidates=\{(M_1, t_4), (M_6, t_6)\}. The minimum distance node through the candidate nodes is $(M_1, t_4)$ and added to Path. Since $m_0 = M_1$ the Path is completed as Path=\{(M_2, 0), (M_3, t_4), (m_0, t_1)\} and the algorithm terminates. That is, the shortest path driving the system from $m_0$ to $M_d$ is obtained as $m_0 \xrightarrow{t_1} M_2 \xrightarrow{t_4} M_d$.

For the same system when Algorithm 2 is executed, the set of minimal paths, that is Pathset is obtained as follow:

$$\text{Pathset} = \{(M_3, 0), (M_2, t_4), (M_1, t_1)\},$$

$$\{(M_2, 0), (M_2, t_4), (M_8, t_6), (M_6, t_1), (M_3, t_5), (M_7, t_5), (M_4, t_4), (M_2, t_2), (m_0, t_1)\},$$

$$\{(M_2, 0), (M_3, t_2), (M_7, t_1), (M_4, t_4), (M_2, t_2), (m_0, t_1)\},$$

$$\{(M_2, 0), (M_3, t_2), (M_7, t_1), (M_4, t_4), (M_2, t_2), (M_8, t_6), (M_6, t_1), (M_3, t_5), (M_7, t_5), (M_2, t_2), (m_0, t_1)\}.$$  (8)

**Example 3:** Let us consider the Petri net system in Fig.3 taken from [3], which models a multiprocessor system. In the PN model, place $p_1$ contains tokens which represent the active processors whereas tokens inside $p_2$ represent the available buses; transition $t_1$ represents the sending of a request of access and place $p_3$ contains the requests which have not been still served. Tokens inside $p_2$ represent processors which are accessing to a memory whereas tokens inside $p_3$ represent processors which are waiting for the memory occupied by tokens inside $p_3$. The firing of transition $t_2$ models the end of the access to the memory which is requested by tokens inside $p_2$; the firing of $t_4$ instead, represents the end of the access to a memory there are no more requests. Transitions $t_2$ and $t_3$ model the two possible choices of memory: firing $t_3$ means to choose the memory which is at the moment utilized by processor in $p_3$ whereas firing $t_2$ means the choice of any other memory.

Suppose that at the initial state there are two active processors and two available buses (i.e. $m_0 = [2, 0, 0, 0]^T$). It is desired to reach the state where there are one active processor, one bus while one processor is accessing to the memory, i.e. $M_d = [1, 1, 0, 1]^T$. By using Algorithm 1, the shortest path to reach $M_d = [1, 1, 0, 1]^T$ from $m_0 = [2, 0, 0, 0]^T$ is obtained as $m_0 \xrightarrow{t_3} [1, 1, 0, 1]^T \xrightarrow{t_2} [1, 1, 0, 0]^T$.

All possible paths from $m_0 = [2, 0, 0, 0]^T$ to $M_d = [0, 1, 1, 0]^T$ can be obtained by Algorithm 2 which yields 4 minimal paths. One of these paths is:

$$[2, 2, 0, 0]^T \xrightarrow{t_3} [1, 2, 1, 0]^T \xrightarrow{t_2} [0, 2, 2, 0]^T \xrightarrow{t_3} [0, 1, 1, 1]^T \xrightarrow{t_2} [0, 1, 1, 1]^T \xrightarrow{t_2} [1, 1, 0, 1]^T.$$  

Note that both of the proposed algorithms can be used to obtain paths leading the systems back to $m_0$ from $m_0$, if it is possible. For example, in order to find the shortest path from $m_0$ to $m_0$, Algorithm 1 is executed with $M_d = m_0$, and it turns out the shortest path as $m_0 \xrightarrow{t_2} [1, 2, 1, 0]^T \xrightarrow{t_2} [1, 1, 1, 0]^T \xrightarrow{t_2} [0, 1, 1, 0]^T$. The previously firable transitions and the algorithm terminates.

**4. Conclusion**

In this work, two algorithms are developed to contribute reachability problems of Petri nets. Both of them deal with the paths from $m_0$ to $m_f$ while Minimal Paths Algorithm obtains all the paths which do not pass through the same marking and Shortest Path Algorithm obtains the shortest path which includes minimum number of transition to reach $m_f$. The proposed algorithms are backwards techniques and do not deal with the reachability tree or 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, the algorithms can be applied to general Petri nets without any restriction.

Future research directions include using integer and mixed-integer programming techniques for reachability path problems and comparing complexity and computational time with the present methods.

**5. References**


