A Solution of Synchronic Distance in Fair Petri Nets

Li-li Wang\(^1\) and Xiang-wei Liu\(^1\)

\(^1\)School of Mathematics and Physics
Anhui University of Science and Technology
Huainan, Anhui, China
wanglili198301@163.com

Abstract

This paper gives a solution of the synchronic distance between any two transitions in a fair Petri net. In order to describe the synchronic distance between two transitions more accurately, the concept of weighted synchronic distance is adopted. The solution method is firstly to add a weighted observe-place between two transitions, and a net system with a weighted observe-place is constructed by assigning a suitable weight function for an arc between a transition and an observe-place. Then the initial tokens of a weighted observe-place are obtained by constructing an augmented coverability tree of a net system with a weighted observe-place. The synchronic distance between two transitions is finally yielded by constructing a coverability tree of a net system with a weighted observe-place, and the corresponding solution algorithms are also given.

Keywords: Fair Petri net, weighted synchronic distance, weighted observe-place, A net system with a weighted observe-place, Suitable weight function

1. Introduction

As a model for the description of the asynchronous and concurrent phenomenon, Petri nets have been widely used in various fields [1-3], and have shown their unique advantage especially in concurrent systems. However, the synchronization between information in some systems often occurs, such as the synchronization between information sending, transferring, and receiving. In order to exactly depict the synchronization, the concept of synchronic distance is presented to describe the synchronization of the actual systems in the synchrony theory [4]. The synchronic distance is not only an analyzing metric to describe the synchronic relationship between two events, but also a tool to analyze the dynamic behavior of systems. Many literatures have shown that it is very useful for system design, analysis, modeling and optimization, one of the most successful application areas is the workflow domain [5-8].

Since the computation of synchronic distance involves both structure and initial markings of net systems, it undoubtedly brings some difficulties to the solution of synchronic distance. The computation of synchronic distance in some special subclass of Petri nets are relatively simple and feasible so far [8-9]. However, the computation of synchronic distance in a normal Petri net is still very difficult.

The paper gives the solution methods of the synchronic distance between two transitions by a observe-place. Its principle can be found in details in the reference [10]. In the Petri net shown in Figure 1, according to the principle of observe-place, we obtain \( \sigma(t_i,t_j) = \infty \) \((\sigma(t_i,t_j))\) indicates the synchronic distance between \( t_i \) and \( t_j \). However, it is easy to find that \( t_j \) cannot fire more than twice without firing \( t_i \). Undoubtedly, there is some dependence between the two transition firings, so \( \sigma(t_i,t_j) = 2 \). How to unify these two results? When the synchronic distance is infinite, we can use the weighted synchronic distance to distinguish irregular and
irregular among them, which can be solved by adding a weight to an arc between a observe-place and a transition.

From the result in the previous researches, the paper discusses the computation of the synchronic distance between any two transitions in a fair Petri net by adopting the notion of the weighted synchronic distance. By assigning a weight to the arc between an observe-place and a transition, the purpose is to make the number of tokens in an observe-place in some cases will not infinitely increase as the number of the cycle process section loops.

2. A Net System with a Weighted Observe-place and an Augumented Cover Ability Tree

The basic concepts and conclusions of Petri nets can be found in details in the reference [11]. For the convenience of the later discussion, the related basic concepts, terminology, and notations are introduced in this section.

**Definition 2.1** [11]. Let $\Sigma = (S,T,F,M_i)$ be a Petri net, $\forall t_1, t_2 \in T$, the weighted synchronic distance between two transitions $t_1$ and $t_2$ is defined by

$$\sigma(t_1,t_2) = \begin{cases} \infty & \text{if } t_1 \text{ and } t_2 \text{ are not in a fair relation in } \Sigma \\
\max_{t \in \Sigma} \left| \frac{\#(t_1)}{\#(t_2)} - r(t_1) \frac{\#(t_1)}{\#(t_2)} \right| & \text{otherwise}
\end{cases}$$

in which $r$ is a primitive repetitive vector of $\Sigma$, and $r(t_1) \neq 0, r(t_2) \neq 0$.

For a fair Petri net, if it has primitive repetitive vectors, then that is unique. Therefore when computing the synchronic distance between two transitions by adopting the weighted synchronic distance, a weight on an arc connecting an observe-place and a transition is also unique. In order to distinguish observe-place in the paper from that in literature [12], the observe-place here is called the weighted observe-place.

**Definition 2.2.** Let $N = (S,T,F)$ be a Petri net, $X$ is a unique primitive repetitive vector of $N$, $t_1, t_2 \in T$ the weighted observe-place $s_y$ satisfies $s_y = \{t_1\}$, $s_y^* = \{t_2\}$, a weight function meeting the following two conditions is said to be a suitable weight function:

1) if exist the primitive repetitive vector $X$ such that $X(i) \neq X(j) \neq 0$, then $w(t_1,s_y) = X(j)$, $w(s_y,t_1) = X(i)$;

2) if not exist the primitive repetitive vector $X$ such that $X(i) \neq X(j) \neq 0$, then $w(t_1,s_y) = w(s_y,t_1) = 1$.
Definition 2.3. Let $\Sigma = (S, T; F, M_0)$ be a Petri net, $R(M_0)$ is the set of reachable marking in a net $\Sigma$, $t_1, t_2 \in T$, $s_{12}$ is a weighted observe-place between $t_1$ and $t_2$, $\Sigma_{s_{12}} = (S \cup s_{12}, T; F \cup F', W, M_{s_{12}})$ is said to be a net system with a weighted observe-place of $\Sigma$. SWNet, if

1. $s_{12} = \{t_1\}, s^*_{12} = \{t_2\}$ ;
2. $F' = (t_1, s_{12}) \cup (s_{12}, t_2)$ ;
3. $W : F' \rightarrow \{1, 2, 3, \cdots\}$ and $W$ is a suitable weight function;
4. $M_{s_{12}} = M_0 \circ M_{s_{12}}$, in which $M_{s_{12}} : s_{12} \rightarrow N$ ($N$ is a positive integer) is the initial markings of $s_{12}$, and for $\forall \sigma \in T^* : M_0[\sigma] \rightarrow M_{s_{12}}[\sigma]$.

Definition 2.4. Let $\Sigma = (S, T; F, M_0)$ be a fair Petri net, in which $|S| = m$, $|T| = n$, $\Sigma_{s_{12}} = (S \cup s_{12}, T; F \cup F', W, M_{s_{12}})$ is a net system with a weighted observe-place of $\Sigma$, $s_{12}$ is an weighted observe-place between $t_i$ and $t_j$, in which $|S \cup s_{12}| = m + 1$, an augmented cover ability tree $\text{EXT}(\Sigma)$ for a net $\Sigma$, is a tree satisfying the following conditions:

1) The augmented cover ability tree $\text{EXT}(\Sigma)$ for a net $\Sigma$, and the cover ability tree for a net $\Sigma$ is isomorphic.
2) Each nodes in $\text{EXT}(\Sigma)$ is denoted by multiple marking vectors (or one marking vector), and each marking vectors is $m + 1$ dimension, each nodes in $\text{EXT}(\Sigma)$ is divided into two levels (or one level).

It has one marking vector alone in the above level, which is reachable marking $M_i$ from $M_{s_{12}}$;

It has multiple marking vectors (or zero marking vector) in the under level.

For the convenience of the later discussion, the node with multiple marking vectors in $\text{EXT}(\Sigma)$ is said to be an augmented node, and the node with one marking vector in $\text{EXT}(\Sigma)$ is said to be a general node.

Theorem 2.1. Let $\Sigma = (S, T; F, M_0)$ be a fair Petri net, $\Sigma_{s_{12}} = (S \cup s_{12}, T; F \cup F', W, M_{s_{12}})$ is a net system with a weighted observe-place of $\Sigma$, $R(M_{s_{12}})$ is the set of reachable markings in the net $\Sigma_{s_{12}}$, $\forall i, t_i \in T (i \neq j)$, the syncronic distance between two transitions $t_i$ and $t_j$ can be obtained by the following:

$$\sigma_q = \max \{M_i(s_q) | M_i \in R(M_{s_{12}})\}$$

Proof: the literature [6] points out that the greatest number of the tokens difference in $s_{q}$ is the synchronous distance between two transitions $t_i$ and $t_j$, i.e.,

$$sd_q = \max \{M_i(s_q) | M_i \in R(M_{s_{12}})\} - \min \{M_i(s_q) | M_i \in R(M_{s_{12}})\}$$

Definition 3 indicates that the number of tokens in a weighted observe-place $s_q$ is the minimum number of tokens. It means that as long as the transition $t_i$ fires several times, the tokens in $s_q$ will be emptied at last, so $\exists M_i \in M_{s_{12}}$ such that the numbers of tokens in $s_q$ is zero, i.e., $\min \{M_i(s_q) | M_i \in R(M_{s_{12}})\} = 0$, therefore $\sigma_q = \max \{M_i(s_q) | M_i \in R(M_{s_{12}})\}$.
3. An Algorithm of Computing Synchronous Distance

The computation of the synchronous distance between two transitions in a fair Petri net is divided into four steps: 1. A cover ability tree of the original net system $\Sigma$ is first constructed; 2. A net system with a weighted observe-place $\Sigma$ is obtained by allocating a suitable weight function on an arc connecting a weighted observe-place and a transition; 3. Next we allocate initial tokens for the weighted observe-place $s$ during constructing the augmented cover ability tree for $\Sigma$; 4. After we observe the maximum tokens in the weighted observe-place during constructing the cover ability tree for a net system with a weighted observe-place $\Sigma$. The synchronous distance between two transitions is finally obtained.

3.1. The Algorithm Allocating the Initial Tokens to the Weighted Observe-place

The algorithm principle: It adopts a variable $Initinal(s_{12})$ to record the initial tokens in the weighted observe-place $s_{12}$. If the present markings is $M_i$ and $t_i$ or $t_i$ can fire at the marking $f(M_i)$, the node $M_i$ is expressed as an augmented node in an augmented cover ability tree. If the present markings is $M_i$ and $t_i$ or $t_i$ can not fire at the marking $f(M_i)$, the node $M_i$ is expressed as a normal node in an augmented cover ability tree.

Algorithm 3.1. The algorithm of allocating the initial tokens to the weighted observe-place $s_{12}$

Input: a net system with a weighted observe-place $\Sigma = (S \cup s_{12}, T; F \cup F', W, M_{s0})$ and the coverability tree $CT(\Sigma)$ for the original net system

Output: the number of initial tokens in $s_{12}$

Algorithm steps:

Step 1: $Initinal(s_{12}) = 0$

Step 2: $Q = \{M_{s0}\}$ and label $M_{s0}$ as the root, and tag it "new", in which $M_{s0}(s_{12}) = 0$.

$M_{s0}(s_i) = M_0(s_i)$ $(i = 1, \ldots, m)$, $f(M_{s0}) = M_0$

Step 3: While "new" marking exist in $Q$ do

Select any marking labeled "new", let be $M_i$

Step 4: If $f(M_i)$ is a leaf node in $CT(\Sigma)$ then

Tag $M_i$ "old", $M_i$ is a normal node without the lower node, and go to Step 3

Step 5: Else for $\forall t_i \in T$ satisfy $f(M_i)[t_i] > 0$ and $t_i \neq t_1, t_2$, then $M_i$ is a normal node without the lower node.

5.1 Obtain the marking $M'_i$ that results from computing $M_i[t_1] > M'_i$

5.2 On the path from $M_{s0}$ to $M'_i$ if there exists a marking $M''_i$ such that $f(M''_i) < f(M'_i)$, then find a component of the vector $j$ such that $f(M''_i) < f(M'_i)$, then replace the $j$-th component of the vector$M'_i$ by $\omega$

5.3 $Q = Q \cup M'_i$, draw an arc with label $t_i$ from $M_i$ to $M'_i$ and erase $M_i$ "new", tag $M'_i$ "new", and go to Step 3

Step 6: If $\exists t_i \in T$ and $t_i = t_2$ satisfy $f(M_i)[t_i] > 0$

$M_i$ is an augmented node, and the number of lower nodes depend on the number of firing transitions at $f(M_i)$.
6.1 For each $t_i \in T$ satisfy $f(M_s)[t_i] > 0$
6.2 If $t_i = t_2$ and $M_s[t_2] > t_3$
   then
   If $M_s(s_{12}) < k_1$ then $Initinal(s_{12}) = Initinal(s_{12}) + k_1 - M_s(s_{12})$, one of
   marking vectors for the lower nodes of $M_s$ is denoted by $\tilde{M}_{s-2}$,
   $\tilde{M}_{s-2}(s_{12}) = k_1$, $\tilde{M}_{s-2}(s) = M_s(s)$, ($i = 1, \cdots, m$)
   else $\tilde{M}_{s-2} = M_s$ (if $M_s(s_{12}) \geq k_1$) else $\tilde{M}_{s-2} = M_s$(if $t_i \neq t_2$)
6.3 Compute $\tilde{M}_{s-1}[t_i] > M'_s$
6.4 On the path from $M_{s0}$ to $M'_s$ if there exists a marking $M'_s$ such that $f(M'_s) < f(M'_s)$
   then
6.5 Find a component of the vector $j$ such that $f(M'_s) < f(M'_s)$, then replace the $j$-th component of
   the vector $M'_s$ by $\omega$
6.6 $Q = Q \cup M'_s$, draw an arc with label $t_i$ from $\tilde{M}_{s-1}$ to $M'_s$ and erase $\tilde{M}_{s-1}$ ”new”, tag
   $M'_s$ ”new”, and go to Step3
   Step7: $M_{s0}(s_{12}) = Initial(s_{12})$

3.2. The Algorithm for Computation of the Synchronic Distance

Using the algorithm 1 we can obtain initial tokens in the weighted observe-place for the net
system with a weighted observe-place, then the maximum tokens in the weighted observe-
place can be obtained during the running of the net system by constructing the augmented
cover ability tree for $\sum_s$, which is the synchronic distance between two transitions.

Algorithm 3.2. The synchronic distance between two transitions $t_1$ and $t_2$ in a fair Petri net
Input: A net system with a weighted observe-place $\sum_s = (S \cup s_{12}, T; F \cup F', W, M_{s0})$
with the initial marking
Output: The synchronic distance $\sigma(t_1, t_2)$ between two transitions $t_1$ and $t_2$
Algorithm steps:
Step0: $Max(s_{12}) = Initial(s_{12})$
Step1: $Q = \{M_{s0}\}$ and label $M_{s0}$ as the root, and tag it “new”
Step2: While “new” marking exist in $Q$ do Select any “new” marking, let be $M_s$
Step3: If on the path from $M_{s0}$ to $M_s$ if there exists a marking equal it then
   Tag $M_s$ ”old”, and go to Step2
Step4: If $\forall t \in T: -M_s[t] >$ then Tag $M_s$ ”old”, and go to Step2
Step5: For any $t \in T$ satisfy $M_s[t] >$ do
5.1 Obtain the marking $M'_s$ that results from computing $M_s[t] > M'_s$
5.2 On the path from $M_{s0}$ to $M'_s$ if there exists a marking $M'_s$ such that $M'_s < M'_s$
   then
   Find a component of the vector $j$ such that $M'_s < M'_s$, then replace the $j$-th component of
   the vector $M'_s$ by $\omega$
5.3 If $M'_s(s_{12}) > Max(s_{12})$ then $Max(s_{12}) = M'_s(s_{12})$
5.4 $Q=Q\cup M'_s$, draw an arc with label $t_1$ from $M_s$ to $M'_s$, erase $M_s$ "new", tag $M'_s$ "new", and go to Step2

Step6: $\sigma(t_1,t_2) = \text{Max}(s_{12})$

We will demonstrate algorithm 1 and the algorithm 2 in the following example.

Example 3.1. In order to obtain the synchronic distance between two transitions $t_2$ and $t_3$ in a Petri net $\Sigma=(S,T,F,M_0)$ as shown in fig. 1. The results are shown in Fig.2-Fig.5. by adopting the algorithm 1 and the algorithm 2. So that there yields the synchronic distance between two transitions $t_2$ and $t_3$, $\sigma(t_2,t_3) = 2$.

4. Conclusions

In order to describe the synchronic distance between two transitions accurately, the paper gives a method to compute the synchronic distance between two transitions in a fair Petri net by adopting a weighted observe-place. Although the formula and the algorithm for the synchronic distance computation are presented for a fair Petri net in the paper, they do not lose the generality. Because the synchronic distance between two transitions is infinite in a fair Petri net if they are not in fair relation. For any given Petri net $\Sigma$, it all can be decomposed into several fair branches, for any two transitions $t_1$ and $t_2$, if they are not in the same fair branch, then the synchronic distance between them is $\infty$, otherwise, the synchronic distance between them are obtained by the paper. The algorithm is finally demonstrated by some examples.

Acknowledgements

This work is partially supported by the National Natural Science Foundation of China under Grant (No.61272153 and No.61340003), the Natural Science Foundation of Educational Government of Anhui Province of China (No.KJ2011A086), the Natural Science Foundation of Anhui Province of China (No. 1208085MF105). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References


**Author**

Li-li Wan (1983), female, lecturer, master degree, the main research area is the theory and application of Petri nets and research and design of algorithms