Grading and Calculation of Synchronic Distance in Petri Nets for Trustworthy Modeling and analyzing

Yumeng Cheng
School of Computer Science
Shaanxi Normal University
Xi'an, China
chengym@snnu.edu.cn

Wangyang Yu (Corresponding Author)
School of Computer Science
Shaanxi Normal University
Xi'an, China
ywy191@snnu.edu.cn

Xiaojun Zhai
School of Computer Science and Electronic Engineering
University of Essex
Colchester, UK
xzhai@essex.ac.uk

Fei Hao
School of Computer Science
Shaanxi Normal University
Xi'an, China
fhao@snnu.edu.cn

Yuan Liu
School of Computer Science
Shaanxi Normal University
Xi'an, China
liu_yuan@snnu.edu.cn

***Abstract*—Synchronization plays a crucial role in computer systems, providing support for system security, data consistency, and coordination. It contributes to the establishment and application of trust, security, and dependability in distributed systems and concurrent computing to a significant extent. This article makes innovative contributions in the field of synchronic distance in Petri net. We provide refined definitions for the hierarchical classification of synchronic levels in Petri net, proposing the concepts of absolute synchronization, strong synchronization, and extended synchronization based on different conditions. Furthermore, we propose an innovative method for calculating synchronic distance. This method can automate the calculation of synchronic distance between any two transitions using computer computation, resulting in improved accuracy and reduced errors. This novel approach provides an effective tool for system security and trustworthy modeling, as accurate synchronic distance calculations allow for better evaluation of synchronic distance between different transitions, leading to the identification of potential security vulnerabilities and design flaws, thereby enhancing the credibility of decision-making and promoting the reliability of models and analysis results. To validate the proposed method, we introduce a specific example of a Petri net with concurrency, demonstrate the practicality and effectiveness of the proposed method and algorithm through analysis of this example. Our work extends the research on Petri net synchronic distance, further advancing the understanding and exploration of this field.**

***Index Terms*—Petri net, synchronization, synchronic distance, system security, concurrency, trustworthy, security, safety**

I. Introduction

Currently, Petri nets have become a widely used modeling tool in various domains [1]. As a graphical modeling language, Petri nets provide a clear representation of asynchronous and concurrent behavior in systems [2], [3], offering advantages such as intuitiveness, visualization, formalism, and verifiability. Petri nets have found extensive applications in various fields [4]–[12]. It is noteworthy that in practical scenarios, many coupled systems or distributed systems [13], collaborative interactions often occur in a synchronous manner [14], [15], ensuring the synchronicity and coordination among various components of the system remains crucial, it is also a key element in ensuring system security and trustworthiness. For instance, in the domain of Advanced Driver Assistance Systems (ADAS) [16], [17], synchronization among subsystems, including sensors, control units, actuators, and human-machine interfaces, is considered crucial. The degree of synchronization among these subsystems directly affects the safety and reliability of the entire ADAS system [18].

In light of this, scholars have embarked on research regarding the synchronization theory of Petri nets [19]. Such research has profound implications for process synchronization and communication problems in distributed systems, which are directly related to system safety and reliability. However, there has been a lack of scholars categorizing synchronic systems, which represents a gap in this field of study.

Furthermore, the study of synchronization theory cannot be separated from discussions on synchronic distance [1]. Synchronic distance refers to the measurement of patterns observed during the alternating firing process between two events or transitions in a Petri net. In terms of calculating the synchronic distance, excellent algorithms with clear procedures have emerged, Ref. [20] gives a method to compute the synchronic distance between two transitions in a fair Petri net by adopting a weighted observe-place. However, the complexity of network topology and the randomness of network operation can lead to limitations such as high computational complexity and potential errors. In such situations, this paper proposes a automated computational approach that can effectively reduce computational costs and calculate synchronic distance more

This work was in part supported by the UK Engineering and Physical Sciences Research Council through grants, EP/V034111/1, EP/X015955/1, EP/X019160/1 and EP/V000462/1, the Natural Science Foundation of Shaanxi Province, China under Grant 2021JM-205, the Fundamental Research Funds for the Central Universities, China under Grant GK202205039, and the Open Research Fund of Anhui Province Engineering Laboratory for Big Data Analysis and Early Warning Technology of Coal Mine Safety, China under Grant CSBD2022-ZD05.

accurately. Additionally, this paper introduces an innovative approach by classifying the synchronization of Petri nets based on hierarchical levels, which can assist developers in gaining a deeper understanding of system security and reliability. Consequently, this enables better system design and development practices to ensure the safety and dependability of the system. This paper aims to explore and validate the synchronization and the computational methods for synchronic distance of Petri nets, in order to better facilitate the advancement of research on synchronization mechanisms

The main contributions of this paper can be summarized as follows:

• We provide different classification definitions for the synchronic levels of Petri nets.

• We propose an innovative method for calculating synchronic distance.

• We apply the proposed classification method and algorithm to analyze a specific example.

The remaining sections of this paper are organized as follows: Section II introduces the relevant definitions of Petri nets. Section III presents the classification definitions for synchronic levels. Section IV proposes a new method for calculating synchronic distance and provides an example for illustration. Finally, Section V concludes the paper.

II. Related Concepts

For the convenience of subsequent discussion, we provide relevant definitions here.

Definition 1 [1]: An original Petri net is a four-tuple $\Sigma = (S, T, F, M_0)$, where

1) S is a finite set of places.

2) T is a finite set of transitions, which represents the action of the system, where $S \cup T \neq \emptyset, S \cap T = \emptyset$.

3) $F \subseteq (S \times T) \cup (T \times S)$ is a set of directed arcs, which represents a change in the state of the system, where $\mathrm{dom}(F) = \{x \in S \cup T \mid \exists y \in S \cup T : (x, y) \in F\}, \mathrm{cod}(F) = \{x \in S \cup T \mid \exists y \in S \cup T : (y, x) \in F\}, \mathrm{dom}(F) \cup \mathrm{cod}(F) = S \cup T$.

4) M_0 is a initial marking, which represents the initial state of the Petri net.

Definition 2 [1]: Transition firing rules of Petri net:

1) For $t \in T$, if $\forall s \in S : s \in {}^\bullet t \rightarrow M(s) \geq 1$, then $M[t >$

2) For $M[t >$, new marking M' will be obtained, where

$$M'(s) = \begin{cases} M(s) - 1, \text{if } s \in {}^\bullet t - t^\bullet \\ M(s) + 1, \text{if } s \in t^\bullet - {}^\bullet t \\ M(s), \text{ others} \end{cases}$$

Definition 3 [1]: Reachable Marking Graph.

Let $\Sigma = (S, T, F, M_0)$ is a bounded Petri net, reachable marking graph of Σ is a three-tuple $RG(\Sigma) = (R(M_0), E, P)$, where

1) $R(M_0)$ is the set of all reachable markings from M_0.

2) $E = \{(M_i, M_j) \mid M_i, M_j \in R(M_0), \exists t_k \in T : M_i[t_k > M_j\}$.

3) $P : E \rightarrow T, P(M_i, M_j) = t_k$ if and only if $M_i | t_k > M_j$.

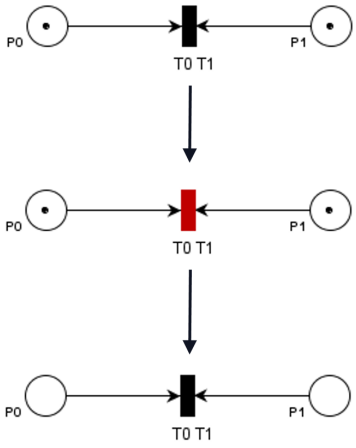

Fig. 1. The absolute synchronization relation

The reachable marking graph is a useful analysis tool for bounded Petri nets, which can be used to determine various dynamic properties of the bounded Petri nets. However, when Σ is not a bounded Petri net, it is impossible to draw the reachable marking graph of Σ because $R(M_0)$ is an infinite set. In order to express the operational behavior of a system with infinitely many states in a finite form, it is necessary to introduce a symbol ω to represent the unboundedness.

When the number of tokens in place s_j tends to infinity during the execution of the Petri net, the j-th component of the marking vector is replaced by ω to cover all such markings. In this way, the operational behavior of the Petri net can be reflected by a finite tree. This finite tree is called the coverability tree of the Petri net Σ, denoted as $CT(\Sigma)$.

Definition 4 [1]: Coverability Graph.

Let $CT(\Sigma)$ is a coverability tree of the Petri net. If the nodes in $CT(\Sigma)$ with the same marking vectors are overlapped (merged into one), the resulting directed graph is referred to as the coverability graph of Σ, denoted as $CG(\Sigma)$.

Definition 5 [1]: Synchronic Distance.

Let $\Sigma = (S, T, F, M_0)$ is a Petri net, $t_1, t_2 \in T$, the synchronic distance between t_1 and t_2 is given by the following formula:

$$\sigma(t_1, t_2) = \begin{cases} \infty, \text{if } t_1 \text{ and } t_2 \text{ do not exhibit a fair relation-ship in } \Sigma \\ \max_{t_i, t_j \in T} \{\sharp(t_j/\sigma) | \exists M \in R(M_0) : M[\sigma > \wedge \\ \sharp(t_i/\sigma) = 0 \wedge i, j \in \{1, 2\}(i \neq j)\}, \text{ others} \end{cases}$$

III. Hierarchical Classification of Petri Net Synchronic Levels

In the previous section, we provided an overview of the research status and fundamental knowledge of Petri nets. Now, we propose the concept of classifying the synchronic levels of Petri nets. By categorizing the synchronization of Petri nets, we can gain a better understanding of the system's security and

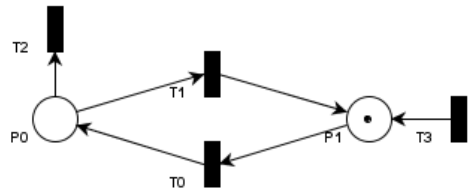

Fig. 2. The strong synchronization relation

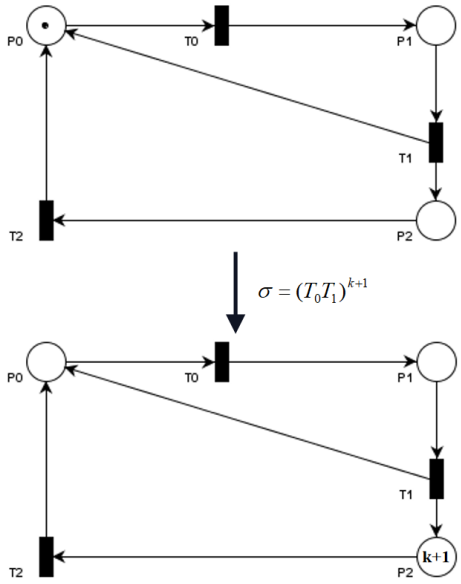

Fig. 3. Non-extended Synchronization Petri net

reliability, thereby providing enhanced assurance in system design and development.

According to Definition 5, we can comprehend the essence of synchronic distance, which refers to the maximum number of occurrences of one transition while the other transition remains unenabled between any two transitions in the Petri net. Synchronic distance effectively defines the level of synchronization between the two events represented by the transitions. We are aware of the close relationship between fairness and synchronicity in Petri nets. If two transitions are not in a fair relationship, their synchronic distance is considered infinite. Therefore, when the synchronic distance is finite, we also aim to provide appropriate definitions that delineate the differences therein. In the following, we present more refined classification definitions for synchronic distance.

Definition 6: Absolute Synchronization Relation.

Let $\Sigma = (S, T; F, M_0)$ be a Petri net, and $t_1, t_2 \in T$. $\forall M \in R(M_0)$ and $\forall \sigma \in T^* : M[\sigma >$ holds such that $\max_{t_i,t_j \in T}\{\sharp(t_j/\sigma)|\sharp(t_i/\sigma) = 0\} = 0, i, j \in \{1, 2\}$ and $i \neq j$. then t_1 and t_2 are said to be in an absolute synchronization relationship. If any two transitions in Σ are in an absolute synchronization relationship, then Σ is called an absolute synchronization Petri net.

Absolute synchronization implies a highly stringent constraint on the synchronization between transitions. As depicted in Fig. 1, it represents a partially idealized state of synchronization where two transitions, t_0 and t_1, are merged into one transition, denoting the simultaneity of the two events. In real-life scenarios, a typical example of an absolute synchronic system is the act of "clapping", where the left and right hands simultaneously complete the action of "clapping".

What we commonly refer to as "alternation" is, in fact, a specific sequential relationship, which also constitutes a form of strong synchronization. After the firing of transition t_i, transition t_j immediately acquires the firing right to occur. Subsequently, after the occurrence of t_j, t_i promptly regains the firing right. Their synchronization distance is 1. Based on the definition of the sequential relation, we provide the definition of strong synchronization as follows.

Definition 7: Strong Synchronization Relation

Let $\Sigma = (S, T; F, M_0)$ be a Petri net, with t_1 and $t_2 \in T$, $M \in R(M_0)$. If the following conditions hold ($i, j \in \{1, 2\}(i \neq j)$)

1) $M[t_i >$ but $\neg M[t_j > \wedge M'[t_j >$ but $\neg M'[t_i >$

2) $M[t_i > M' \rightarrow M'[t_j > M$

then t_i and t_j are said to be in a strong synchronization relation.

As shown in Fig. 2, the strong synchronization relation typically exists between two regular events that occur locally and alternately within a system, when token flow in, they alternately trigger each other, and they will cease to occur when token flow out of the loop. This kind of relationship is manifested when the synchronic distance between two transitions is 1, this relationship exhibits exclusivity. Generally, a Petri net does not have arbitrary pairs of transitions that simultaneously possess a strong synchronization relation. However, it is possible to have multiple groups of strong synchronization relation within a Petri net, where transitions within each group are in strong synchronization with each other pairwise.

Definition 8: Extended Synchronization Relation.

If there exists a positive integer k such that for any $M \in R(M_0)$, $\forall \sigma \in T^* : M[\sigma >$ it holds that $\max_{t_i,t_j \in T}\{\sharp(t_j/\sigma)|\sharp(t_i/\sigma) = 0\} \leq k$, $i, j \in \{1, 2\}(i \neq j)$.

then t_1 and t_2 are said to be in an extended synchronization relation. If any two transitions in Σ are in an extended synchronization relation, Σ is referred to as an extended synchronization Petri net.

We are aware that there is a certain connection between synchronicity and fairness. In a Petri net Σ, if two transitions, t_1 and t_2, are not in a fair relation, it means for any positive integer k, there exists a reachable marking M and a transition sequence σ, such that $M[\sigma >$, and $\sharp(t_i/\sigma) = 0 \wedge \sharp(t_j/\sigma) > k$ (where $i, j \in 1, 2$ and $i \neq j$), then we consider their synchronic distance to be ∞ [1]. This serves as a counterexample

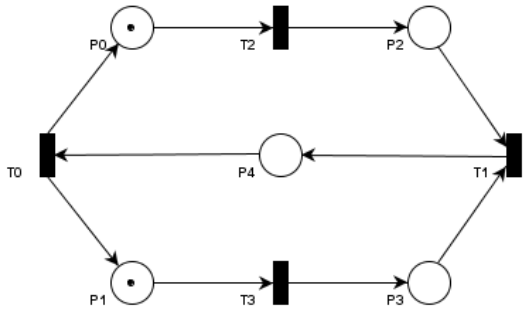

Fig. 4. A Petri net with concurrency

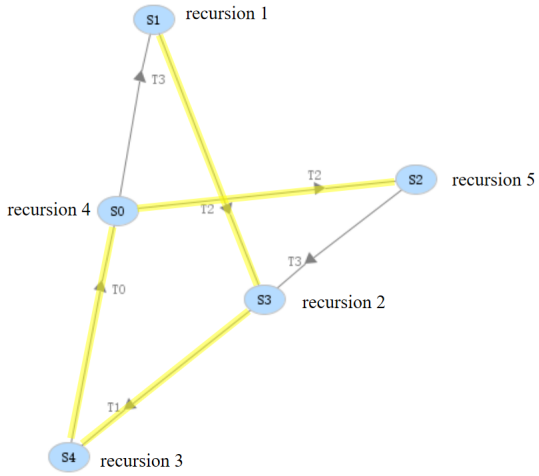

Fig. 5. Reachable marking graph with a marked path

to the extended synchronization. In Definition 8, k does not depend on a specific marking but imposes restrictions on the entire operational process of the Petri net.

To illustrate this, we provide an example with the non-extended synchronization Petri net depicted in Fig. 3. For any positive integer k, we can find a transition sequence $\sigma = (t_0t_1)^{k+1}$ such that $M_0[\sigma > M_1$. In this case, $M_1(P_2) = k+1$. Under the marking M_1, there exists a transition sequence $\sigma = t_2^{k+1}$ such that $M_1[\sigma >$, $\sharp(t_0/\sigma)$=0 $\wedge$ $\sharp(t_1/\sigma)$=0 but $\sharp(t_2/\sigma) > k$. Therefore, even though the occurrence of t_2 is limited by t_0 and t_1, t_2 is not in an extended synchronization relation with t_0 or t_1.

The following lemma is self-evident.

Lemma 1: Let $\Sigma = (S, T; F, M_0)$ be a Petri net, and $t_1, t_2 \in T$. If t_1 and t_2 are in an absolute synchronization relation or strong synchronization relation, then they must be in an extended synchronization relation.

The more common application in real-world industries is the

Algorithm 1: SDA($ALAM, cur\_node, cur\_t_i\_count, t_i, t_j$)

1. Mark cur_node as visited;
2. if $cur\_t_i\_count > max(t_i)$ **then**
  update $max(t_i)$;
3. for *each neighbor of* cur_node **do**
  if *the arc-label to the neighbor* $\neq t_j$ **then**
    if *the neighbor is already visited* **then**
      set $max(t_i)$ = ∞ and end the program;
    else
      if *the arc-label* = t_i **then**
        SDA($ALAM, neighbor, cur\_t_i\_count+$
        $+, t_i, t_j$;
)
      else
        SDA($ALAM, neighbor, cur\_t_i\_count, t_i, t_j$);
  else
    continue;
4. Mark cur_node as unvisited and return;

extended synchronization system. Extended synchronization implies that regardless of the system's state, the repeated occurrence of any event is strictly limited in frequency, yet there is a certain degree of flexibility for adjustments within the system while still being subject to synchronization constraints.

However, in natural systems such like chemical reaction processes, for example, many phenomena are asynchronous systems, which means they are not constantly in a synchronized state. This implies that the occurrences of these processes are not dependent on strict synchronization requirements but are influenced by various factors.

This distinction reflects the different demands and adaptabilities of synchronization in different fields, as well as considerations of flexibility and practical applications in system design. By distinguishing between different levels of synchronic systems, we can better ensure the safety of the system. In the industrial and manufacturing sectors, system safety is of paramount importance. By determining the level of a synchronic system, we can plan and monitor the operation and control of the system more precisely. The strict constraints of a extended synchronic system limit the repeated occurrence of events, which helps prevent unexpected situations and potential hazards. For example, in ADAS (Advanced Driver Assistance Systems), strict synchronization control ensures that the vehicle's perception and decision-making align with the actual road conditions, thereby enhancing driving safety.

IV. Algorithm for Computing Synchronic Distance

To ensure system security more effectively, we continuously seek innovative methods and technologies. In this regard, we introduce a novel approach for computing synchronic distance,

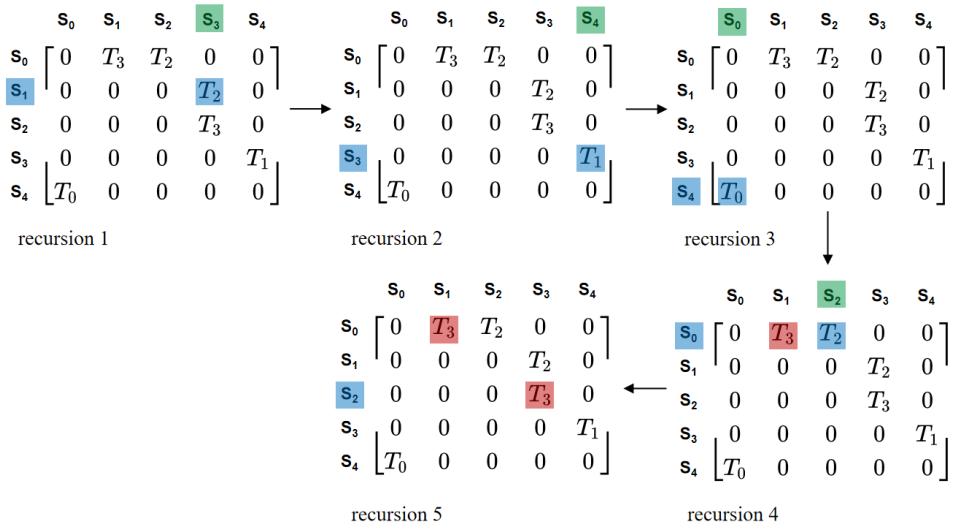

Fig. 6. The recursive invocation of SDA starting from node S_1

aiming to assess and control system synchronization more accurately.

We believe that through this innovative method of computing synchronic distance, we will provide optimized solutions for system security and efficiency. This will leads to more reliable, intelligent, and sustainable industrial processes.

The proposed method is based on automated computation using reachable marking graphs and coverability graphs. Firstly, to enable the computer to recognize reachable marking graphs and coverability graphs, we treat them as a special type of directed graph. We then store them as a unique adjacency matrix called the Arc-label Adjacency Matrix (ALAM). The uniqueness of ALAM lies in its matrix elements, which are not 0 or 1. Instead, A_{ij} represents the arc-label from marking S_i to S_j in the reachable marking graph.

Subsequently, the task of computing the synchronic distance between any two transitions t_i and t_j is delegated to an automated program. We present the innovative algorithm in pseudocode and name it the Synchronic Distance Algorithm (SDA), as shown in Algorithm 1.

A feasible algorithm is expected to have termination. Herein, we present the relevant theorem and its proof for the SDA algorithm.

Theorem 1: The Synchronic Distance Algorithm (SDA) is guaranteed to terminate.

Proof: Consider a reachable graph (or coverability graph) with N nodes and E edges. During the execution of Algorithm 1, SDA, the algorithm traverses the neighboring nodes of the current node and recursively calls itself on eligible nodes.

By observation, it can be noted that each recursive call marks an unvisited node as visited and proceeds to mark its neighbors, once all neighbors have been traversed, the current loop will terminate, returning to the previous level.

Since the number of nodes in the graph is finite, after a finite number of recursive calls, Algorithm 1 will mark all nodes that satisfy the conditions and eventually terminate the recursion. Alternatively, if a marked node is encountered, the program terminates directly.

Therefore, Algorithm 1 is guaranteed to terminate. □

For each node in the reachable marking graph (or coverability graph), invoke SDA($ALAM, node, 0, t_i, t_j$). Eventually, we obtain $max(t_i)$, which represents the maximum number of occurrences of t_i when t_j does not firing. By swapping the positions of t_i and t_j, we obtain $max(t_j)$. Finally, we take the larger value between the two to obtain $\sigma(t_i, t_j)$.

V. Evaluation

TABLE I
VALUE CHANGES DURING THE RECURSION

Recursion	cur_node	$cur\_t_2\_count$	$max(t_2)$
1	S_1	0	1
2	S_3	1	1
3	S_4	1	1
4	S_0	1	1
5	S_2	2	2

To illustrate our algorithm, we will provide an example. Fig. 4 depicts a Petri net with a concurrent relationship, and Fig. 5 shows its reachable marking graph. Using this method, we calculate the synchronic distance for transitions t_2 and t_3. We present a portion of the process where node S_1 calls SDA($ALAM, S_1, 0, t_2, t_3$), as shown in Fig. 6.

The process aims to find a path starting from node S_1 where the transition t_2 is fired the most number of times and

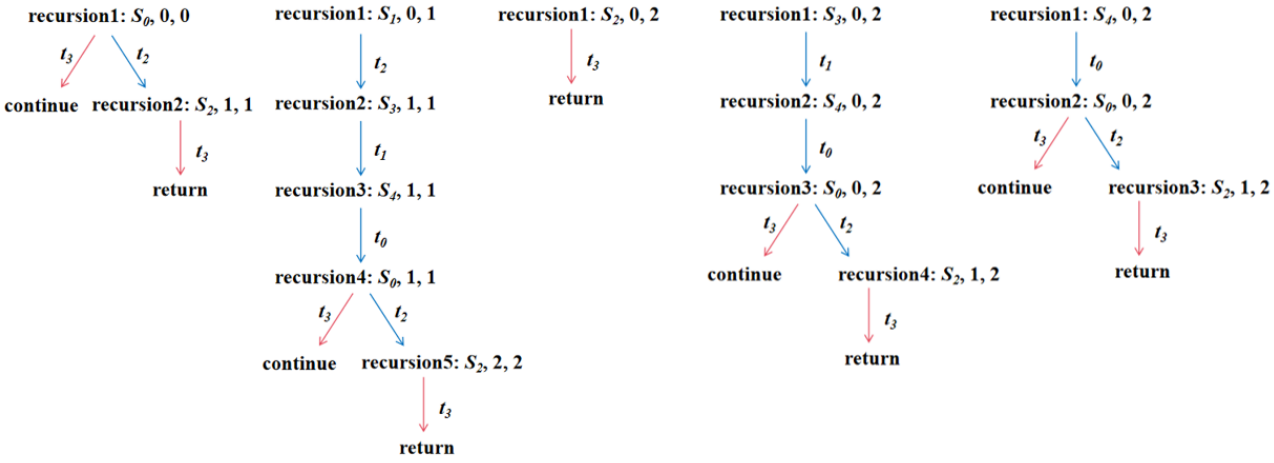

Fig. 7. Calculation Process of $max(t_2)$

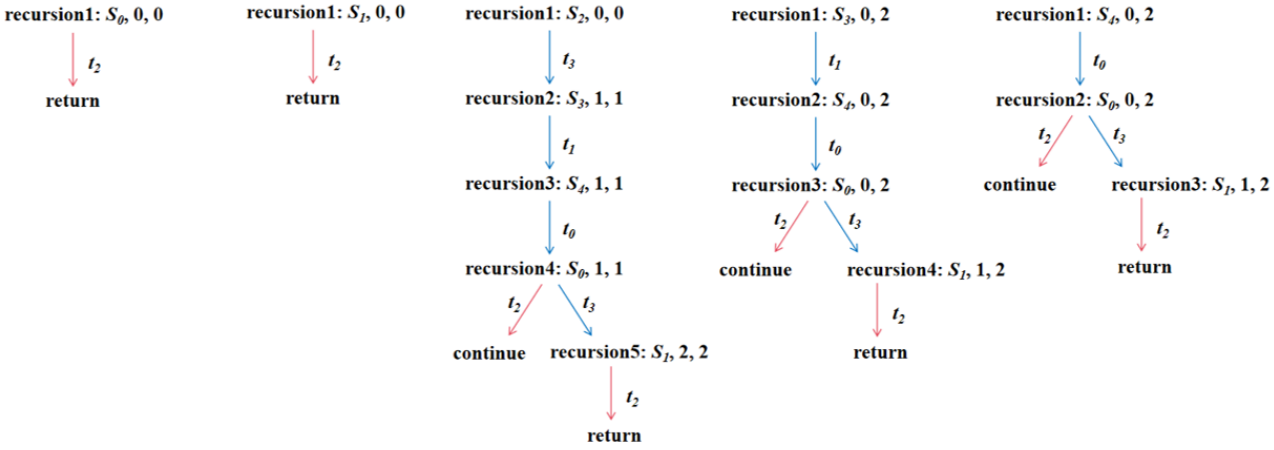

Fig. 8. Calculation Process of $max(t_3)$

the transition t_3 is not fired. In fact, for illustrative purposes, this path is precisely the one with the highest occurrence of t_2 in the entire graph, as shown in Fig. 5. We use different colored markings on ALAM to visualize this process to the reader (blue markings represent the currently visited node S_i and the t_j along the path leading to adjacent nodes. green markings represent the adjacent nodes to be recursively called, and red indicates that the arc-label is t_3, so the corresponding adjacent node will not be visited). At the fourth level of recursion, when encountering a branch labeled as t_3, the algorithm chooses to avoid it and selects the next node as S_2. The fifth level of recursion begins, and at this point, since the branch from S_2 to its only neighboring node is labeled as t_3, our algorithm initiates a return process. This continues until reaching the initial recursion, and the algorithm concludes, updating $max(t_2)$ to 2. This value represents the maximum number of occurrences of t_2 that can happen given the current knowledge that t_3 does not occur. However, this only represents a small part of the automated execution process of the algorithm. Table 1 displays the values of $cur\_t_2\_count$ and $max(t_2)$ corresponding to each level of recursive calls shown in Fig. 6 (note that $max(t_2)$ has already been updated to 1 in the previously unshown steps of the algorithm).

The key to the successful execution of the algorithm lies in avoiding paths containing t_3 during the traversal of adjacent nodes at each level of recursion. Furthermore, when the algorithm exits the current recursion, the value of $cur\_t_2\_count$ is returned to the count of the previous recursion level, ensuring the accuracy of counting the occurrences of t_2 on each traversed path.

Now, let's consider the scenario where the arc-label from S_0 to S_1 is t_2, how would the algorithm operate in this case? According to the steps of SDA, when we recursively reach the fourth level with S_0, the algorithm continues to the fifth level, which leads to the already visited node S_1. In this case,

$max(t_2)$ would be directly updated to ∞, and the algorithm would terminate entirely. This precisely represents another termination condition of the algorithm: the presence of a loop in the reachable marking graph, where only t_i (or t_j) fires.

As shown in Fig. 7 and 8, in order to provide a more comprehensive explanation of the algorithm's process, we illustrate the complete procedure of computing $\sigma(t_2, t_3)$ using a diagrammatic approach. The three variables involved in each recursion are the current node, $cur\_t_2\_count$, and $max(t_2)$(or $cur\_t_3\_count$, and $max(t_3)$). Firstly, we sequentially invoke Algorithm 1 SDA$(ALAM, S_i, 0, t_2, t_3)$ for each node in the reachable marking graph, continuously updating $max(t_2)$ during the execution. After applying SDA to all nodes in the reachable marking graph, we obtain the final value of $max(t_2)$. Subsequently, we interchange the positions of t_2 and t_3 in SDA, transforming it into SDA$(ALAM, S_i, 0, t_3, t_2)$, and repeat the aforementioned process. Similarly, we obtain the final value of $max(t_3)$. The value of $\sigma(t_2, t_3)$ can be obtained by taking the larger of the two values, which is expressed as $\sigma(t_2, t_3) = max(max(t_2), max(t_3)) = 2$. Although this procedure may appear intricate, it can be effectively delegated to computational systems due to the introduction of $ALAM$, which converts the reachable marking graph into a matrix recognizable by computers. Consequently, this approach significantly reduces manual effort and computational errors.

VI. Conclusions

The contribution of this paper lies in proposing a hierarchical definition for the synchronization of Petri nets, as well as an algorithm for automating the calculation of synchronic distance. This will lead to a further improvement in the efficiency of system design and analysis, consequently bolstering the security, reliability, and dependability of the system. Based on different levels of synchronic constraints, we provide definitions for absolute synchronization, strong synchronization, and extended synchronization. The refined classification principles aid in distinguishing various risks and security vulnerabilities within the system. Our proposed synchronic distance algorithm realizes a high degree of automation in the computation process. This assists in reducing the workload associated with synchronic distance calculation in complex network structures. However, the existing work still has certain limitations, and it is anticipated that in the future, a simulation platform can be developed to directly convert the modeled Petri net into an ALAM. This would enable the entire process to be fully automated.

References

[1] Wu Zhehui. Introduction to petri nets. *Beijing, Press of Machinery and Industry*, pages 15–21, 2006.

[2] Wangyang Yu, Yadi Wang, Lu Liu, Yisheng An, Bo Yuan, and John Panneerselvam. A multiperspective fraud detection method for multiparticipant e-commerce transactions. *IEEE Transactions on Computational Social Systems*, 2023.

[3] Wangyang Yu, Chungang Yan, Zhijun Ding, Changjun Jiang, and Mengchu Zhou. Analyzing e-commerce business process nets via incidence matrix and reduction. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(1):130–141, 2016.

[4] WangYang Yu, ChunGang Yan, ZhiJun Ding, ChangJun Jiang, and MengChu Zhou. Modeling and validating e-commerce business process based on petri nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(3):327–341, 2013.

[5] Ziyue Ma, Guanghui Zhu, Zhiwu Li, and Alessandro Giua. Computation of admissible marking sets in weighted synchronization-free petri nets by dynamic programming. *IEEE Transactions on Automatic Control*, 65(6):2662–2669, 2019.

[6] Ricardo J Rodríguez, Simona Bernardi, and Armin Zimmermann. An evaluation framework for comparative analysis of generalized stochastic petri net simulation techniques. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(8):2834–2844, 2018.

[7] Alessandro Giua, Stéphane Lafortune, and Carla Seatzu. Divergence properties of labeled petri nets and their relevance for diagnosability analysis. *IEEE Transactions on Automatic Control*, 65(7):3092–3097, 2019.

[8] Max Weissman, Daniel Ponsini, and Seung-yun Kim. Prioritized situation awareness for soccer robots using timed transition petri nets. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 134–138. IEEE, 2018.

[9] Hong Zheng, Litian Li, Bozhi Wang, and Tong Ruan. Enterprise workflow modeling based on priced timed petri nets. In *2018 5th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2018 4th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, pages 106–110. IEEE, 2018.

[10] Wolfgang Reisig. *Understanding petri nets: modeling techniques, analysis methods, case studies.* Springer, 2013.

[11] Claude Girault, Rüdiger Valk, and R Valk. Modelling and analysis techniques by example. *Petri Nets for Systems Engineering: A Guide to Modeling, Verification, and Applications*, pages 85–103, 2003.

[12] Haiyang Hu, Jiawei Yu, Zhongjin Li, Jie Chen, and Hua Hu. Modeling and analysis of cyber–physical system based on object-oriente generalized stochastic petri net. *IEEE Transactions on Reliability*, 70(3):1271–1285, 2020.

[13] Duqu Wei and Chuncao Mei. Synchronization of distributed power grids with the external loading system. *International Journal of Modern Physics B*, 32(14):1850173, 2018.

[14] Saber Jafarizadeh, Farzad Tofigh, Justin Lipman, and Mehran Abolhasan. Optimizing synchronizability in networks of coupled systems. *Automatica*, 112:108711, 2020.

[15] Chih-Wen Shih and Jui-Pin Tseng. From approximate synchronization to identical synchronization in coupled systems. *Discrete and Continuous Dynamical Systems-B*, 25(9):3677–3714, 2020.

[16] K Suganthi, M Arun Kumar, N Harish, S HariKrishnan, G Rajesh, and S Sofana Reka. Advanced driver assistance system based on iot v2v and v2i for vision enabled lane changing with futuristic drivability. *Sensors*, 23(7):3423, 2023.

[17] Kshitij Dhawan and Nadesh RK. Identification of traffic signs for advanced driving assistance systems in smart cities using deep learning. *Multimedia Tools and Applications*, pages 1–16, 2023.

[18] Qi Guo, Wangyang Yu, Fei Hao, Yuke Zhou, and Yuan Liu. Modelling and analysis of adaptive cruise control system based on synchronization theory of petri nets. *Electronics*, 11(21):3632, 2022.

[19] Marco Pocci, Isabel Demongodin, Norbert Giambiasi, and Alessandro Giua. Synchronizing sequences on a class of unbounded systems using synchronized petri nets. *Discrete Event Dynamic Systems*, 26:85–108, 2016.

[20] Li-li Wang and Xiang-wei Liu. A solution of synchronic distance in fair petri nets. *International Journal of Hybrid Information Technology*, 8(3):33–40, 2015.