An Efficient Liveness Analysis Method for Petri Nets via Maximally Good-Step Graphs

Hao Dou, *Graduate Student Member, IEEE*, MengChu Zhou, *Fellow, IEEE*, Shouguang Wang, *Senior Member, IEEE*, and Aiiad Albeshri

***Abstract*—Liveness is among the most significant properties when Petri net (PN) models of automated systems are analyzed, which ensures systems' deadlock-freeness. Traditionally, the liveness analysis methods based on reachability graphs (RGs) of PNs often suffer from state-space explosion problems. In this article, we propose a novel liveness-analysis method for PN based on maximally good-step graphs (MGs), namely, the reduced form of RGs, which can effectively alleviate such problems in liveness analysis. First, we introduce the concept of sound steps and establish an algorithm for assessing the soundness of an enabled step at the current marking from a practice point of view. Second, we propose a definition of maximal sound steps and construct an algorithm for calculating a maximal-sound-step set at each marking whose computational complexity grows polynomial with the number of places and transitions. Then, we introduce a definition for good steps and an algorithm for generating maximally good step graphs of PN; and discuss its computational complexity with respect to the net size and initial marking. Next, we for the first time answer how to evaluate the liveness of PN by using MGs. Experiments in diverse large-scale automated manufacturing systems demonstrate that the proposed method significantly reduces state space and time consumption in the liveness analysis of network systems.**

***Index Terms*—Automated manufacturing systems (AMSs), liveness analysis, maximal strongly connected components (maximal SCC), maximally good-step graphs (MGs), Petri nets (PNs).**

I. Introduction

SYSTEMS operating through discrete events and characterized by discrete states are referred to as discrete event systems (DESs) [1], [2], [3]. In our real life, many man-made systems can fall under the category of DES, including communication systems, automated manufacturing systems (AMSs) [4], [5], [6], [7], [8], and computer systems. The design of DES usually involves optimization, control, modeling, analysis, and performance evaluation. Among the crucial properties evaluated in DES performance, liveness stands out, as a paramount concern for researchers. The absence of liveness can lead to the underlying system's low-operational efficiency, economic loss, and security incidents in the real world. Consequently, ensuring this property within DES holds significant importance.

Some major mathematical tools employed in liveness analysis are Petri nets (PNs) [9], [10], and automata [2], [3]. PN are particularly prevalent since they are powerful in simulating and assessing many DES characteristics, such as resource sharing, asynchronism, concurrency, liveness, and deadlocks. If all transitions (events) can be triggered under markings (states) reachable from the initial one, then a PN is considered as "live." PN have many variants, such as ordinary, general, colored, and timed ones. This article mainly focuses on ordinary PN since all their properties can be retained in several other PN variants with minor modifications [11].

A. Literature Review

In the realm of liveness analysis for PN, two principal techniques emerge: one relies on structural objects, such as siphons [13], [14], [15], [16], [17], [18], [19], resource-transition circuits [20], and perfect activity circuits [21], while the other is grounded in reachability graphs (RGs) [10], [22]. The efficacy of these methods is evaluated through two crucial factors: 1) computational complexity and 2) structural complexity. The former techniques exhibit limited applicability, primarily suited for specific classes of PN, making them suboptimal solutions. In contrast, the latter may provide a more comprehensive view of system behavior and can be applied to a broader spectrum of PN. The challenge with reachability analysis lies in its known exponential [23] or even Ackermann-complete complexity [24], [25]. To overcome this challenge, various techniques have been proposed, including partial order methods [26], [27], [28], [29], [30], [31], symmetries, and compositional verification. Among these, partial order methods have demonstrated the greatest effectiveness in practical applications [30].

Partial order techniques can be split into two categories: 1) covering step graphs [31] and 2) partial order reduction methods [33]. The target of these techniques is to simplify the state space of net systems and allow for the verification of

Manuscript received 31 October 2023; accepted 21 February 2024. This work was supported in part by the Public Technology Research Plan of Zhejiang Province under Grant LGJ21F030001; in part by the Fundo para o Desenvolvimento das Cienciase da Tecnologia (FDCT) under Grant 0047/2021/A1; and in part by the Institutional Fund Projects provided by the Ministry of Education and King Abdulaziz University under Grant IFPIP-1480-611-1443. This article was recommended by Associate Editor C. Zhang. *(Corresponding authors: Shouguang Wang; MengChu Zhou.)*

Hao Dou is with the Macau Institute of Systems Engineering, Macau University of Science and Technology, Macau, China (e-mail: haodou1@foxmail.com).

MengChu Zhou and Shouguang Wang are with the School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China (e-mail: mengchu@gmail.com; wsg5000@hotmail.com).

Aiiad Albeshri is with the Department of Computer Science, King Abdulaziz University, Jeddah 21481, Saudi Arabia (e-mail: aaalbeshri@kau.edu.sa).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2024.3372941.

Digital Object Identifier 10.1109/TSMC.2024.3372941

system properties through partial interleavings of concurrent executions rather than all of them [30]. Partial order reduction methods, such as persistent sets [32], stubborn sets [29], and sleep sets [30], primarily concentrate on decreasing the breadth of the state space while covering step graph techniques emphasize the reduction of depth. Researchers often seek to enhance the efficiencies of the state space reduction and computation process by combining both types of methods, aiming to achieve superior outcomes compared to using either method in isolation. It is noteworthy that the combination of persistent set methods and covering step graphs merits special attention, as persistent set methods are a typical case of partial order reduction techniques [33].

In Ribet et al. [32] proposed hybrid persistent step graphs (HPSGs) that integrate covering step graphs and persistent sets to fulfill the state-space reduction in both breadth and depth. Despite being proven to maintain the deadlocks of PN, the method has limitations in handling concurrent relations among events, thus potentially missing opportunities for further state-space reduction. Subsequently, Barkaoui et al. [33] put forward a weak-persistent step graph method, introducing a novel concept of weak-persistent sets. They reclassify the persistent sets proposed by [34] as strong-persistent sets, thereby broadening the scope of persistent sets and allowing for a greater number of transitions to be fired simultaneously. The application of this method results in a more substantial state-space reduction compared to HPSG. Despite resolving the issue of the state-space explosion to some degree, can we further reduce such complicated state-space? To answer it, we have proposed new concepts of sound steps and good steps in [35] after better-comprehending concurrency and conflict relations among transitions. The generation of good steps at each state enables an extreme reduction in state space since good steps cover the scope of covering steps, strong- and weak-persistent sets. Moreover, the maximal good step graph (MGSG) method presented in [35] demonstrates superior efficiency in state-space reduction compared to existing techniques.

B. Motivations and Contributions of This Work

Despite partial order methods sharing the fundamental principle of examining partial interleaving of concurrent executions rather than exhaustively considering all possible interleavings to verify the properties of net systems, they vary in their implementation details. Specifically, each method evaluates the tradeoffs among computational complexity, running-time consumption, and the number of states it can reduce. In essence, the pursuit of the best-reduction method becomes impractical since the preservation of deadlocks in PN is a crucial requirement for partial order methods, and the detection of deadlocks has been proven to be a PSPACE-complete problem [36]. It is reasonable that some methods yield a better result in state-space reduction while requiring more complicated operations and consuming more time than others. Hence, the first motivation behind our method is exploring a more reduced state space than that generated by existing partial order methods with the lowest-possible computational complexity.

To the best of our knowledge, the majority of partial order methods are adept at detecting deadlocks of net systems, while very few preserve system liveness. Hence, investigating a compact state space to confirm the liveness of net systems is a valuable pursuit, which serves as the second motivation of our work.

Considering the power of MGSGs in alleviating the state-space explosion issues, we improve MGSG into maximally good-step graphs (MGs) and apply the improved version to explore the liveness analysis issue for PN. Through this article, we intend to bring forward the following novel and unique contributions.

1) We enhance the definitions of sound transitions, maximal sound steps, and good steps, which are significant components to construct MG of PN. Furthermore, we introduce the constraint that each sound step contains only two transitions, thus facilitating the exploration of maximal sound ones at the current marking with a low-computational effort.
2) We propose an algorithm to verify whether an enabled step is sound under a current marking, which is applied to construct the algorithms of a) calculating a maximal-sound-step set under a reachable marking and b) generating MG.
3) We illustrate that the improved MGSG, i.e., MG, can detect deadlocks of a PN. Besides, we prove that MG can preserve the liveness of net systems. Notably, we introduce, for the first time, a necessary and sufficient condition for conducting liveness analysis on a PN using MG.

Relative to RG and the existing partial order methods, we demonstrate the effectiveness of our proposed method in both state-space reduction and the liveness analysis of net systems through the modeling of several large-scale plants. However, a primary limitation of our work lies in the exclusive application of our proposed method to ordinary PN. Although such nets can provide significant modeling solutions for various problem domains, they may encounter challenges in modeling certain complex systems, unless additional diverse elements are incorporated into them [12].

C. Outline of This Article

This article can be summarized as follows: Section II reviews fundamental knowledge of graph theory and PN. Section III presents some definitions and properties associated with MG and proposes a new liveness-analysis method via MG of PN. Section IV showcases the efficiency of our proposed method in simplifying the state space and analyzing the liveness of net systems through several manufacturing-oriented PN. In Section V, this work gets a conclusion accompanied by insights into future research directions.

II. Preliminaries

This section provides an overview of the essential concepts in graph theory and PN, guided by established references, such as [32], [37], and [38].

A. Petri Nets

In the ensuing discussion, Σ represents a nonempty, finite set of notations referred to as an alphabet. Σ^* denotes the set of all finite strings of notations in Σ, including an empty string ϵ. Another set of notation strings, Σ^+, excludes ϵ. We have $\Sigma^* = \{\epsilon\} \cup \Sigma^+$. For instance, if $\Sigma = \{\alpha, \beta\}$, then $\Sigma^* = \{\epsilon, \alpha, \beta, \alpha\alpha, \alpha\beta, \beta\beta, \cdots\}$ and $\Sigma^+ = \{\alpha, \beta, \alpha\alpha, \alpha\beta, \beta\beta, \cdots\}$.

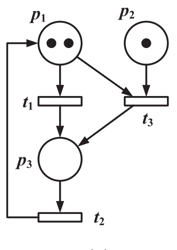
A 4-tuple $N = (P, T, F, W)$ defines a general PN. P and T represent sets of places and transitions, separately. These sets are both finite and nonempty and they do not overlap. The directed arcs connecting places to transitions or transitions to places represent the flow relation, defined as $F \subseteq (P \times T) \cup (T \times P)$. The function W: $F \to \mathbb{N}^+$ assigns a positive integer weight to each directed arc, implying that $\forall f \in F$, $W(f) \geq 1$, where $\mathbb{N}^+ = \{1, 2, 3, \cdots\}$. For any node x in N, where $x \in P \cup T$, the preset and post-set of x are defined as $^\bullet x = \{y \in P \cup T \mid (y, x \in F)\}$ and $x^\bullet = \{y \in P \cup T \mid (x, y \in F)\}$, respectively. For any set $X \subseteq P \cup T$, we have $^\bullet X = \cup_{x\in X} {}^\bullet x$ and $X^\bullet = \cup_{x\in X} x^\bullet$. A PN $N = (P, T, F, W)$ is an ordinary PN if $\forall f \in F$, $W(f)$=1. In this case, the net can be represented concisely as $N = (P, T, F)$.

A Marking M: $P \to \mathbb{N}$ is defined as a function that associates each place in the set P with a non-negative integer from $\mathbb{N}$. Typically, for brevity, a multiset $\Sigma_{p\in P} M(p)p$ is represented by a vector M, where $M(p)$ indicates the number of tokens held by p. For example, a marking $M = [3, 1, 2, 9]^T$ can be expressed as $M = 3p_1 + p_2 + 2p_3 + 9p_4$. When a PN N is equipped with its initial marking M_0, the combination is referred to as a net system or marked net, represented by the notation (N, M_0).

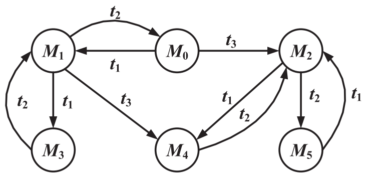
A symbol $M[t\rangle$ denotes that $t \in T$ can be fired at a marking M if $\forall p \in {}^\bullet t$, $M(p) \geq W(p, t)$, while $M[t\rangle M'$ represents that a new marking M' is immediately yielded after firing t. In such a case, we have $\forall p \in {}^\bullet t$, $M'(p) = M(p) - W(p, t) + W(t, p)$. $E(M)$ is a set contains all enabled transitions at M, and $E(M) = \emptyset$ means that M is a deadlock marking. $M[\sigma\rangle$ defines that σ can be firable under M, where σ is a sequence of transitions with $\sigma = t_1t_2, \ldots t_n$. Specifically, there exist several intermediate markings $M_1, M_2, \ldots, M_{n-1}$ such that $M[t_1\rangle M_1 \wedge M_1[t_2\rangle M_2 \wedge \ldots \wedge M_{n-1}[t_n\rangle$. If there is a marking M'' s.t. $M_{n-1}[t_n\rangle M''$, then we have $M[\sigma\rangle M''$, i.e., M'' is a reachable marking of M. $M[\epsilon\rangle M$ denotes that an empty sequence ϵ can be fired at M. The reachability set $R(N, M_0)$ of a net system (N, M_0) is used to store all markings reachable from the initial one.

An enabled step τ represents a collection of enabled transitions that can all be fired together under a reachable marking M, denoted by $M[\tau\rangle$, where $M(p) \geq \sum_{t\in\tau} W(p, t)$ $\forall p \in {}^\bullet\tau$. This implies that there are sufficient tokens to fire all transitions within step τ simultaneously at M. The notation $M[\tau\rangle M'$ signifies that the firing of τ at M results in marking M'. We define $E_s(M)$ as the set of enabled steps at M, expressed by $E_s(M) = \{\tau \subseteq E(M) | M[\tau\rangle, \text{where} M \in R(N, M_0)\}$. The cardinality of τ is denoted by $|\tau|$, meaning the number of transitions it includes.

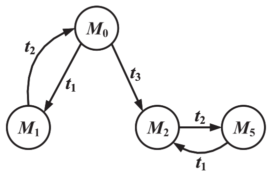
The notation $t \perp t'$ denotes that transitions t and t' are in conflict relation if: 1) $t^\bullet \cap t'^\bullet \neq {}^\bullet t \cap {}^\bullet t'$ and 2) ${}^\bullet t \cap {}^\bullet t' \neq \emptyset$. A set of transitions conflicting with t is represented by $C(t)$ $= \{t' \in T | t \perp t'\}$. The symbol $t \wr t'$ means that t and t' are independent with each other since ${}^\bullet t \cap {}^\bullet t' = \emptyset$ or ${}^\bullet t \cap {}^\bullet t' = t^\bullet \cap t'^\bullet$. Equivalent sequences can be denoted as $\sigma_1 \equiv \sigma_2$, where their Parikh vectors are identical. If $\sigma_1 \equiv \sigma_2$ and $M[\sigma_1\rangle M_1 \wedge M[\sigma_2\rangle M_2$, where $\sigma_1, \sigma_2 \in T^*$ and $M \in R(N, M_0)$, then we have $M_1 = M_2$. The set of transitions included in a sequence $\sigma \in T^*$ is denoted by $||\sigma|| = \{t \in T | \exists \sigma_1, \sigma_2 \in T^*, \sigma_1 t \sigma_2 \equiv \sigma\}$. For instance, if $\sigma = t_1t_3t_2$, then $||\sigma|| = \{t_1, t_2, t_3\}$.



(a)



(b)



(c)

Fig. 1. (a) PN (N_1, M_0), (b) its RG, and (c) subgraph of its RG.

Given a marked net (N, M_0), $t \in T$ is live at M_0 if $\forall M \in R(N, M_0)$, $\exists M' \in R(N, M)$, $M'[t\rangle$. (N, M_0) is called live if $\forall t \in T$, t is live at M_0. (N, M_0) is called dead if $\exists M \in R(N, M_0)$, $\nexists t \in T$, $M[t\rangle$. (N, M_0) is deadlock-free (or weakly live) if $\forall M \in R(N, M_0)$, $\exists t \in T$, $M[t\rangle$. (N, M_0) is k-bounded if $\exists k \in \mathbb{N}$ $\forall M \in R(N, M_0)$ $\forall p \in P$, $M(p) \leq k$.

In the following sections of this article, we concentrate on ordinary and bounded PN. For clarity, the PN mentioned refer exclusively to marked ones.

B. Graph Theory

Definition 1: A *directed graph* is a two-tuple $G = (V, E)$, where 1) V is a node-set and 2) $E \subseteq V \times V$ represents a set of directed edges of G.

Definition 2: A directed graph $G' = (V', E')$ is called a *subgraph* of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. This is denoted as $G' \subseteq G$.

Example 1: $G' = (V', E')$ shown in Fig. 1(c) is regarded as a subgraph of $G = (V, E)$ depicted in Fig. 1(b), i.e., $G' \subset G$. We have $V' = \{M_0, M_1, M_2, M_5\} \subset V$ and $E' = \{(M_0, M_1), (M_1, M_0), (M_0, M_2), (M_2, M_5), (M_5, M_2)\} \subset E$.

In a directed graph $G = (V, E)$, a path from node v_1 to v_k, denoted as $\rho(v_1, v_k)$, is a sequence of nodes such that $(v_i, v_{i+1}) \in E$, where $i \in \{1, 2, \ldots, k-1\}$. The path is referred to as a cycle if $v_0 = v_k$. If the nodes in a path are all distinct, it is called an elementary path.

Definition 3: A subgraph $G' = (V', E')$ of a directed graph $G = (V, E)$ is *strongly connected* if there is a path between each pair of nodes in the subgraph. G' is called a maximal strongly connected component (maximal SCC) of G if there

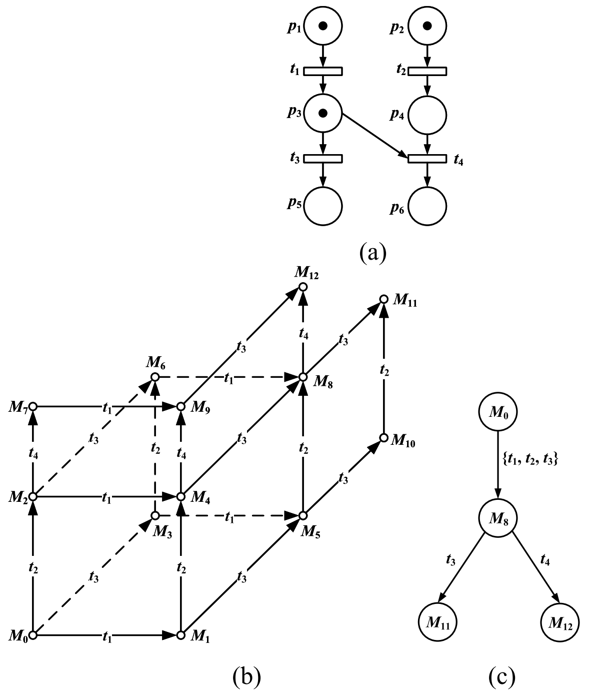


Fig. 2. (a) PN (N_2, M_0), (b) its RG, and (c) its MG.

is no other strongly connected subgraph $G'' = (V'', E'')$ of G with $V' \subsetneq V''$. The maximal SCC is denoted as $\overset{\leftrightarrow}{U}$, where $U \subseteq V$.

Specifically, a maximal SCC can consist of a single node if the subgraphs it is located in are not strongly connected.

III. Liveness Analysis Method

A. Maximally Good-Step Graphs

Definition 4 [33]: Let M be a reachable marking of a PN (N, M_0) and $\pi \subseteq E(M)$ be a set of enabled transitions. π is *persistent* at M if all transitions in π meet the conditions.

1) $E(M) \neq \emptyset \Leftrightarrow \pi \neq \emptyset$.
2) $\forall t \in \pi\ \forall \omega \in (T{-}\pi)^+$, $M[\omega\rangle \Rightarrow M[\omega t\rangle$.
3) $\forall t \in \pi\ \forall \omega \in (T{-}\pi)^+$, $M[\omega t\rangle \Rightarrow M[t\omega\rangle$.

Intuitively, Condition 1 means that a persistent set π consists of enabled transitions at M. If π is an empty set, then there is no enabled transition under M and (N, M_0) is dead. Condition 2 implies that all transitions of π cannot be forbidden by any firable sequence ω as long as it does not contain transitions in π. Condition 3 signifies that if any transition t of π is enabled under M after firing ω that does not contain transitions in π, then the sequence ω shall also be firable at M after t. For a PN (N_2, M_0) shown in Fig. 2(a), t_1, t_2, and t_3 are all enabled transitions of an initial marking M_0, i.e., $E(M_0) = \{t_1, t_2, t_3\}$. According to Definition 4, t_3 is disabled after firing t_2t_4 at M_0 (i.e., $M_0[t_2t_4\rangle$ and $\neg M_0[t_2t_4t_3\rangle$), thus $\{t_3\}$ is not persistent at M_0.

After defining persistent sets, we introduce a concept of sound steps, which plays a crucial role in reducing the state space of PN. Note that sound steps and maximal sound steps proposed in this article are slightly different from the corresponding ones in [35]. To mitigate the computation of determining sound steps in practical systems, we limit the number of transitions within a sound step τ to two, i.e., $|\tau| = 2$. This avoids constructing redundant transition sequences, consequently decreasing the overall calculation cost.

Definition 5 Let $M \in R(N, M_0)$ be a reachable marking and $\tau_s \in E_s(M)$ an enabled step at M with $|\tau_s| = 2$. A transition $t \in \tau_s$ is *sound* at M w.r.t. $t' = \tau_s\backslash\{t\}$, denoted as $t\|t'$, if:

1) $\forall\ \sigma \in (T\backslash\{t\})^+$, $(M[t'\sigma\rangle \wedge \neg M[t'\sigma t\rangle) \Rightarrow$
   a) $\exists\ t_a \in E(M)\backslash\tau_s$, $\exists\ \sigma_a \in (T\backslash\{t\})^*$, s.t. $M[t'\sigma t_a\sigma_a t\rangle$; or
   b) $\exists\ t_1 \in (E(M) \cap ||\sigma||)\backslash\tau_s$, $\exists\ \sigma_1 \in (T\backslash\{t\})^*$, s.t. $(t_1\sigma_1 \equiv \sigma) \wedge M[t_1t'\sigma_1\rangle$.
2) $\forall\ \sigma \in (T\backslash\{t\})^+$, $M[t'\sigma t\rangle \Rightarrow \exists\sigma' \in (T\backslash\{t\})^+$, s.t. $(\sigma \equiv \sigma') \wedge M[tt'\sigma'\rangle$.

Intuitively, Condition 1 indicates that even if a sequence $t'\sigma$ starting with t' disables t at M, it does not necessarily imply that t is unsound for t' unless the transitions fail to meet Condition 1(a) or 1(b). Condition 1(a) ensures that t is re-enabled after firing an inserted sequence $t_a\sigma_a$, i.e., $M[t'\sigma t_a\sigma_a t\rangle$, where t_a is an enabled transition outside of τ_s under M. Condition 1(b) states that there is an enabled transition t_1 in $||\sigma||$ located outside of τ_s such that a sequence equivalent to $t'\sigma$ starting with t_1 can be firable at M, meaning that the forbidden sequence $t'\sigma$ can also be firable via its equivalent sequence $t_1t'\sigma_1$. Condition 2 indicates that the firing of t and t' cannot interfere with each other.

Definition 6 Given a marking M, an enabled step τ_s at M is a *sound step* if: 1) $|\tau_s| = 2$ and 2) $\exists t \in \tau_s$, $(t\|t') \wedge (t'\|t)$, where $t' = \tau_s\backslash\{t\}$. An enabled step τ_{ms} with $|\tau_{ms}| \geq 2$ is a *maximal sound step* under M if: 1) $\forall\ \tau_s \subseteq \tau_{ms}$ where $|\tau_s| = 2$, τ_s is a sound step at M and 2) $\nexists\ t'' \in E(M)\backslash\tau_{ms}\ \forall t \in \tau_{ms}$, $\{t, t''\}$ is a sound step at M.

An enabled step consisting of only two transitions that are sound for each other is considered a sound step at the current marking. An enabled step containing more than two transitions is called a maximal sound step if all pairs of transitions within it are sound for each other and no enabled transition outside of it exhibits mutual soundness with any transition in the step.

Example 2: In the PN (N_2, M_0) depicted in Fig. 2(a), the transition t_3 is sound for t_2 under $M_0 = p_1{+}p_2{+}p_3$. This is because, even if the sequence t_2t_4 disables t_3 at M_0 (i.e., $M_0[t_2\ {}_4\rangle$ and $\neg M_0[t_2\ {}_4t_3\rangle$), we can find ${}_1 \in E(M_0)\backslash\{t_2, t_3\}$ such that firing ${}_1$ re-enables t_3 at M_0 (i.e., $M_0[t_2t_4\ {}_1t_3\rangle$). The maximal sound step under M_0 is $\{\ {}_1, t_2, t_3\}$ since any two different transitions within it are sound for each other.

Consider a PN (N_3, M_0) depicted in Fig. 3(a). Assuming that a transition t_0 is sound for t_1 under the marking $M_0 = [1, 2, 1, 1]^T$, we have the sequence $t_1t_2t_3t_1$ firable at M_0 while t_0 unenabled after firing this sequence at M_0 (i.e., $M_0[t_1t_2t_3t_1\rangle$ and $\neg M_0[t_1t_2t_3t_1t_0\rangle$). There exists an equivalent sequence with $t_1t_2t_3t_1$, starting with an enabled transition outside of $\{t_0, t_1\}$ at M_0 (i.e.,$t_2t_1t_3t_1$), which can be fired at M_0. Thus, the hypothesis is valid.

Considering a PN (N_4, M_0) of Fig. 4(a), t_2 is not sound for t_0 under $M_0 = [1, 1, 1, 0, 0, 0, 0]^T$. After firing $t_0t_1t_3$ under M_0, the transition t_2 can be fired (i.e., $M_0[t_0t_1t_3t_2\rangle$). However, the equivalent sequence with $t_0t_1t_3t_2$ is not firable at M_0 (i.e., $\neg M_0[t_2t_0t_1t_3\rangle$). ■

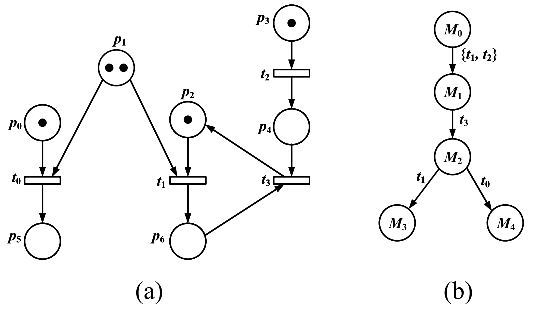


Fig. 3. (a) PN (N_3, M_0) and (b) its MG.

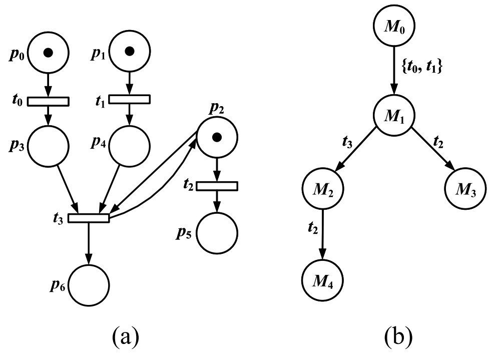


Fig. 4. (a) PN (N_4, M_0) and (b) its MG.

We introduce Algorithm 1 to verify whether an enabled transition is sound in relation to another at the present marking. In this algorithm, $\rho(x, y)$ signifies a path from node x to node y in PN, where $x, y \in P \cup T$. Each stage in Algorithm 1 aligns with a specific condition of Definition 5. In brief, steps 10–20 correspond to Condition 1.a, steps 10 and 21–25 correspond to Condition 1.b, and steps 15–19 and 27–33 correspond to Condition 2.

The Complexity of Algorithm 1: PN is considered to be a directed graph with $|P| + |T|$ nodes. Computing the conflict transitions of transition t (step 6) has complexity $O(n_c)$, where n_c is the number of conflict transitions of t. The complexity of calculating paths from one node to another (steps 8 and 13) is $O(V+E)$ in the worst case (e.g., by using Depth-first search), which is $O(|P| \cdot |T|)$ in PN. Therefore, the total complexity of Algorithm 1 is $O(n_c(|P| \cdot |T|)^2)$.

We develop Algorithm 2 to calculate a maximal-sound-step set under a marking. In such an algorithm, *IsSound* is defined as a decision function. Specifically, given an enabled step $\{t, t'\}$ of a marking M, *IsSound*(M, $\{t, t'\}$) yields a "true" result if both t and t' satisfy the soundness criteria for each other at M according to the rules delineated in Algorithm 1.

In Algorithm 2, we first check the soundness of each pair of transitions within τ (steps 5 and 6) at the marking M. If an enabled transition t is not sound for t' at the current marking, then we proceed to partition τ into two sets of transitions, one excluding t and the other excluding t' (steps 7–9). Repeat this procedure until all transitions in each divided set are sound for each other. In the worst case, no transition is mutually sound with others. Let n be the size of τ. The algorithm needs to explore at most $(n-1)$ sets under $(n-2)$ levels. As a result,

Algorithm 1: Verifying Whether an Enabled Transition t Is Sound for t' Under the Current Marking

```
Input: An original Petri net $N = (P, T, F)$, a reachable marking
       $M \in R(N, M_0)$, and two enabled transitions $t$ and $t'$ at $M$
Output: "$t$ is sound/not sound for $t'$ at $M$"
1  Let $t_c$, $t_a$, and $t_e$ be transitions;
2  Let $\sigma'$, $\sigma_e$, and $\sigma_c'$ be sequences of transitions;
3  if $t$ does not have any conflict transition then
4      Output "$t$ is sound for $t'$ at $M$" and Exit;
5  end
6  for all $t_c \in T$ s.t. $t_c \perp t$ do
7      if there is a path from $t'$ to $t_c$ then
8          for each path $\rho(t', t_c)$ do
9              $\sigma_c \leftarrow \rho(t', t_c) \backslash (P \cup \{t'\})$;
10             if $M[t'\sigma_c\rangle \wedge \neg M[t'\sigma_c t\rangle$ then
11                 $p_c \leftarrow {}^\bullet t \cap {}^\bullet t_c$;
12                 if $\exists t_a \in E(M) \backslash \{t, t', t_c\}$ s.t. there is a path from
                   $t_a$ to $p_c$ and there is no path from $t_a$ to $t'$ then
13                     Choose a path $\rho(t_a, p_c)$;
14                     $\sigma_a \leftarrow \rho(t_a, p_c) \backslash P$;
15                     if $M[t'\sigma_c\sigma_a t\rangle$ and $\exists \sigma' \equiv \sigma_c\sigma_a$ s.t. $M[tt'\sigma'\rangle$
                       then
16                         Continue;
17                     else
18                         Output "$t$ is not sound for $t'$ at $M$"
                           and Exit;
19                     end
20                 end
21                 if $\exists t_e \in (E(M) \cap ||\sigma_c||) \backslash \{t, t'\}$ s.t.
                   $(t_e\sigma_e \equiv \sigma_c) \wedge M[t_e t'\sigma_e\rangle$ then
22                     Continue;
23                 else
24                     Output "$t$ is not sound for $t'$ at $M$" and Exit;
25                 end
26             end
27             if $M[t'\sigma_c\rangle \wedge M[t'\sigma_c t\rangle$ then
28                 if $\exists \sigma_c' \equiv \sigma_c$ s.t. $M[tt'\sigma_c'\rangle$ then
29                     Continue;
30                 else
31                     Output "$t$ is not sound for $t'$ at $M$" and Exit;
32                 end
33             end
34         end
35     end
36 end
37 Output "$\tau$ is sound at $M$" and Exit;
```

the complexity of splitting τ is $O(n^2)$. The maximal size of τ is $|E(M)|$. Since the complexity of *IsSound* is $O(n_c(|P| \cdot |T|)^2)$, the overall complexity of Algorithm 2 is bounded by $O(\hat{n_c}(|E(M)| \cdot |P| \cdot |T|)^2)$, where $\hat{n_c}$ is the maximal number of conflict transitions for transition t in $E(M)$.

Definition 7 Let $M \in R(N, M_0)$ be a reachable marking, τ_{ms} a maximal sound step at M, and τ_g a subset of τ_{ms}, i.e., $\tau_g \subseteq \tau_{ms}$. The set τ_g is called a *good step* at M if it is persistent under M. A maximal good step τ_{mg} of M is a step such that there is no good step ${\tau_{mg}}'$ with $\tau_{mg} \subset {\tau_{mg}}'$.

Algorithm 3 is developed to build a PN's MG, which mainly contains two stages.

1) *Stage 1, Steps 10–28:* Compute the enabled transitions or steps that are firable at the current marking M. Specifically, step 10 is used to compute a maximal-sound-step set *MSS*(M) under M according to

Algorithm 2: Calculating a Maximal-Sound-Step Set of the Current Marking

```
   Input: A reachable marking $M \in R(N, M_0)$
   Output: A maximal-sound-step set $MSS(M)$ under $M$
1  Let $t$ and $t'$ be transitions;
2  Let $\tau$, $\tau_1$, and $\tau_2$ be sets of transitions;
3  $MSS(M) \leftarrow \{E(M)\}$;
4  $MSS'(M) \leftarrow MSS(M)$;
5  for each $\tau \in MSS'(M)$ do
6      if $\exists\, t, t' \in \tau, t \neq t'$ s.t. $\neg IsSound(M, \{t, t'\})$, then
7          $\tau_1 \leftarrow \tau \backslash \{t\}$;
8          $\tau_2 \leftarrow \tau \backslash \{t'\}$;
9          $MSS'(M) \leftarrow (MSS'(M) \backslash \{\tau\}) \cup \{\tau_1, \tau_2\}$;
10     end
11 end
12 if $\forall \tau \in MSS'(\mathrm{M})$, $|\tau| \geq 2$ then
13     $MSS(M) \leftarrow MSS'(M)$;
14 else
15     $MSS(M) \leftarrow \emptyset$;
16 end
17 Output: $MSS(M)$.
```

Algorithm 2. If $MSS(M)$ is not empty, we need to explore whether there exists a maximal good step at M (steps 11–23). The algorithm in [30] computes the smallest persistent set for each enabled transition at the current marking (step 12). Those persistent sets containing only a single enabled transition are initially paired with each other and subsequently combined with other persistent sets to create maximal persistent sets for each enabled transition (steps 13 and 14). If a maximal persistent set is included within a maximal sound step, it qualifies as the maximal good step at the current marking (by Definition 7) and can be fired (steps 15–17). In cases where certain persistent sets comprise only one transition, these sets should be merged and fired at the marking (steps 18–19). Otherwise, all maximal sound steps occur at the marking (steps 20–22). Furthermore, if there does not exist a sound step under M, we have to explore whether there is an enabled transition that is structurally independent of other transitions. If such a transition exists, it can occur at M since its firing does not impede other transitions (steps 24–25). Otherwise, all enabled transitions are fired (steps 26–28).

2) *Stage 2, Steps 29–43:* Add nodes, directed edges between nodes, and labels of directed edges to MG. The structure of this stage is similar to that of Algorithm 1 in [35]. Its detailed explanation is omitted for the sake of brevity.

In Algorithm 3, step 10 calculates a maximal-sound-step set under the current marking M, with a worst-case complexity of $O(\hat{n_c}(|E(M)| \cdot |P| \cdot |T|)^2)$. Steps 11–23 involve [30, Algorithm 1], whose computational complexity is $O(|E(M)|^2)$ in the worst case. Steps 29–43 explore the yielded markings from M by firing enabled transitions (steps 24–28) or steps (steps 15–22). To construct MG, all markings in MG should be explored until Λ contains no elements (step 6). Let n_m denote the number of markings in MG. In summary, the complexity of building MG is expressed as $O(\sum_{i\in 1,2,\ldots,n_m} \hat{n_{c_i}}(|E(M_i)| \cdot |P| \cdot |T|)^2)$, where $E(M_i)$ denotes the set of enabled transitions at the marking M_i in MG and $\hat{n_{c_i}}$ is the maximal number of conflict transitions of t in $E(M_i)$.

Algorithm 3: MG Generation of a PN

```
   Input: A PN $(N, M_0)$
   Output: An MG $\Theta = (\mathbb{M}, \mathbb{E}, \mathbb{L}, M_0)$
1  Given a root node $v_0$ and its marking $M_0$ of MG;
2  $\Lambda \leftarrow (v_0)$; /*$\Lambda$ is defined as a stack to store nodes.*/
3  $\Sigma \leftarrow \{M_0\}$; /*Each marking in $\Sigma$ corresponds to a node.*/
4  $\Delta \leftarrow \emptyset$; /*$\Delta$ is a set of directed edges between nodes.*/
5  $\Xi \leftarrow \emptyset$; /*$\Xi$ is a set of transitions and enabled steps that label directed
   edges.*/
6  while $\Lambda \neq ()$ do
7      $v \leftarrow pop(\Lambda)$; /*Remove the last node $v$ from $\Lambda$.*/
8      Define $M_v$ as the corresponding marking of the node $v$;
9      if $E(M_v) \neq \emptyset$ then
10         Compute the set $MSS(M_v)$ of maximal sound steps under $M_v$;
           /*Refer to Algorithm 2.*/;
11         if $MSS(M_v) \neq \emptyset$ then
12             Compute the persistent set for each transition $t$ of $E(M_v)$
               and let $P(M_v)$ store persistent sets of $E(M_v)$; /*Refer to
               [30, Fig. 4.2].*/
13             $\tau_g \leftarrow \bigcup_{\pi \in P(M_v), |\pi|=1} \pi$;
14             $Q(M_v) \leftarrow \{\tau_g \cup \pi | \forall \pi \in P(M_v), |\pi| \neq 1\}$;
15             if $\exists \pi \in Q(M_v)\ \forall \tau_{ms} \in MSS(M_v)$ s.t. $\pi \subseteq \tau_{ms}$ then
16                 $\Xi \leftarrow \Xi \cup \{\pi\}$; /*$\pi$ is a maximal good step at $M_v$.*/
17             end
18             if $\tau_g \neq \emptyset$ then
19                 $\Xi \leftarrow \Xi \cup \{\tau_g\}$;
20             else
21                 $\Xi \leftarrow \Xi \cup MSS(M_v)$; /*since there is no maximal
                   good step at $M_v$, all maximal sound steps of $M_v$ are
                   fired.*/
22             end
23         end
24         if $\exists t \in E(M_v) \forall t' \in T - \{t\}$ s.t. $t \wr t'$ then
25             $\Xi \leftarrow \Xi \cup \{t\}$;
26         else
27             $\Xi \leftarrow \Xi \cup E(M_v)$;
28         end
29         for each $\xi \in \Xi$ do
30             Compute the yielded marking $M_w$ s.t. $M_v[\xi\rangle M_w$;
31             if $M_w \notin \Sigma$ then
32                 Create a node $w$;
33                 $\Delta \leftarrow \Delta \cup \{(v, w)\}$;
34                 Label $(v, w)$ by $\xi$;
35                 Define $M_w$ as a marking of $w$;
36                 $\Sigma \leftarrow \Sigma \cup \{M_w\}$;
37                 $\Lambda \leftarrow push(\Lambda, w)$; /*Add $w$ into $\Lambda$ as the last node.*/
38             else
39                 Obtain the node $w$ of $M_w$;
40                 $\Delta \leftarrow \Delta \cup \{(v, w)\}$;
41                 Label $(v, w)$ by $\xi$;
42             end
43         end
44     end
45 end
46 $\mathbb{E} \leftarrow \Delta$;
47 $\mathbb{L} \leftarrow \Xi$;
48 Output: $\Theta = (\mathbb{M}, \mathbb{E}, \mathbb{L}, M_0)$.
```

The complexity of generating MG connects to the net size and the number of markings in MG. In the worst case, where there are no sound steps and no structurally independent enabled transitions at each marking, the set of reachable markings in MG is the same as RG, i.e., $\mathbb{M}(N, M_0) = R(N, M_0)$, and its complexity mirrors that of establishing RG, where the number of markings grows exponentially with the net size and initial marking. Since maximal sound steps or structurally independent enabled transitions can usually be explored at multiple markings, the size of MG, i.e., the number of reachable markings in MG, is generally smaller than that of RG, namely, $|\mathbb{M}(N, M_0)| < |R(N, M_0)|$. In other words, MG performs much better than RG at the state space required to represent them, providing a more efficient approach.

B. Liveness Analysis With MG

According to [35, Th. 2], MGSG can detect all deadlock markings in a PN. Algorithm 3 is utilized to construct MG by exploring maximal sound or good steps at each marking. Since we have modified the definitions of sound steps, maximal sound steps, and good steps proposed in [35], it is essential to prove that the enhanced version of MGSG, i.e., MG, has all deadlock markings in a PN if existing.

Theorem 1 Exploring maximal sound steps at each marking can detect all deadlock markings in PN.

Proof: The exploration of maximal sound steps refers to a generating procedure that can recursively compute maximal-sound-step sets under the initial marking and all the succeeding markings that are reachable by firing these steps. If X is a deadlock state of a PN and is reachable by firing sequence ξ at some marking M, i.e., $M[\xi\rangle$X. Let $MSS(M)$ be a maximal-sound-step set at M. We shall prove by induction on the length of ξ that X is reachable from M through the referred to generating procedure. For the sake of convenience, the length of ξ is denoted by $|\xi|$ in this proof.

1) If $|\xi| = 0$, then M is a deadlock marking X.

2) If $\xi = t$ and $\exists\ \tau_{ms} \in MSS(M)$, $t \in \tau_{ms}$, then X is not a deadlock marking since $|\tau_{ms}| \geq 2$ by Definition 6.

3) If $\xi = t$ and $\forall\ \tau_{ms} \in MSS(M)$, $t \notin \tau_{ms}$, then X is not a deadlock marking since transitions of τ_{ms} may be enabled after firing t at M.

4) If $\xi = tt'$ and $\exists\ \tau_{ms} \in MSS(M)$, then either X is not a deadlock marking (if $\{t, t'\} \subset \tau_{ms}$) or X is reachable from M through the generating procedure (if $\{t, t'\} = \tau_{ms}$).

5) Assume that Theorem 1 holds for $|\xi| = m$, where $m \in \mathbb{N}^+$. Then, we shall prove that it holds for $|\xi| = m+1$. There are two cases.

a) If $\forall\ \tau_{ms} \in MSS(M)$, transitions of τ_{ms} are not contained in ξ, then X is not a deadlock marking since at least a transition of τ_{ms} can occur after the firing of ξ at M, i.e., X can yield another marking.

b) If $\exists\ \tau_{ms} \in MSS(M)$, all transitions of τ_{ms} are contained in ξ, then there exist an equivalent sequence ξ' whose prefix is the transitions in τ_{ms}. It means that a marking M_1 is yielded after firing τ_{ms} at M, i.e., $M[\tau_{ms}\rangle M_1$, and X is reachable from M_1 by ξ_1, i.e., $M_1[\xi_1\rangle$X, where $\tau_{ms}\xi_1 \equiv \xi$. Since $|\tau_{ms}| \geq 2$, we have $\xi_1 < m$, which satisfies the induction hypothesis (Condition 5). It implies that X is reachable from M_1 and then from M through the generating procedure. ■

Theorem 2 MG returned by Algorithm 3 can preserve all existing deadlock markings in a PN.

Proof: Algorithm 3 provides the generation rule of MG. Under each reachable marking M, there exist three cases. The notation $MSS(M)$ denotes a maximal-sound-step set under M and $G(M)$ represents a set of good steps at M.

1) If $MSS(M) = \emptyset$, there are two cases.

a) If $\nexists t \in E(M)\ \forall t' \in E(M)\backslash\{t\}$, s.t. $t \wr t'$, then MG is equivalent to RG of a PN. It can detect all deadlock markings of a PN.

b) If $\exists t \in E(M)\ \forall t' \in E(M)\backslash\{t\}$, s.t. $t \wr t'$, then only t is fired at M according to step 25 of Algorithm 3. We refer to t as a structurally independent transition. In this case, the generated MG can detect all deadlock markings in a PN since the firing of structurally independent transitions cannot interfere with the firing of other transitions.

2) If $MSS(M) \neq \emptyset$, we compute the maximal persistent sets for all enabled transitions at the current marking (steps 12–14 of Algorithm 3) to determine whether the maximal good step exits at this marking. There are two possible outcomes.

a) If each maximal sound step at M shares the same persistent set, as determined in step 15 of Algorithm 3, such a set is called a maximal good step according to Definition 7. Otherwise, all structurally independent and enabled transitions at M can be amalgamated into a maximal good step, as shown in steps 18–19 of Algorithm 3. Since each transition in such a step must be sound with respect to the others within the step, and the firing of maximal sound steps can detect all deadlocks of PN, as per Theorem 1, the exploration of maximal good steps can also preserve deadlocks.

b) Alternatively, if there is no maximal good step at M, then all maximal sound steps should occur under M, as indicated in step 21 of Algorithm 3. By Theorem 1, the exploration of maximal sound steps can detect PN's all deadlock markings. ■

Next, we shall propose a new liveness analysis method. Before giving the criteria about how to check the liveness of PN via MG, a basic concept is proposed.

Definition 8 Given a directed graph $G = (V, E)$ and its maximal SCC $\overset{\leftrightarrow}{U}$ with $U \subseteq V$, $\overset{\leftrightarrow}{U}$ is called a *leaf strongly connected component* if $\forall\ u \in U$, $\nexists\ v \in V\backslash U$ such that there is a path from u to v.

For instance, there are two maximal SCC $\overset{\leftrightarrow}{U}_1$ and $\overset{\leftrightarrow}{U}_2$ in Fig. 5(c), where $U_1 = \{M_0\}$ and $U_2 = \{M_1, M_3, M_4\}$. $\overset{\leftrightarrow}{U}_1$ is not a leaf strongly connected component since $M_0 \in U_1$, $\exists\ M_1 \notin U_1$ such that there is a path from M_0 to M_1. By Definition 8, its leaf strongly connected component is $\overset{\leftrightarrow}{U}_2$.

Theorem 3 Given a PN (N, M_0), $\Theta = (\mathbb{M}, \mathbb{E}, \mathbb{L}, M_0)$ is its MG. (N, M_0) is live iff for any leaf strongly connected component of Θ, there are sets $\gamma_1, \ldots, \gamma_n$ that label the n edges in each leaf strongly connected component of Θ, such that $\bigcup_{i\in\{1,\ldots,n\}} \gamma_i = T$.

Proof: To prove that (N, M_0) is live, we shall prove that $\forall\ t \in T$, t is live at M_0. In other words, for each t in T, the condition $\forall\ M \in R(N, M_0)$, $\exists\ M' \in R(N, M)$ s.t. $M'[t\rangle$ should be satisfied. ■

(*if*) Let $\overset{\leftrightarrow}{U}$ be a leaf strongly connected component of Θ, where $U \subseteq \mathbb{M}$. Since $\overset{\leftrightarrow}{U}$ is a subgraph of MG, we can also use $\Theta_u = (U, \mathbb{E}_u, \mathbb{L}_u, M_0)$ to represent $\overset{\leftrightarrow}{U}$. According to the

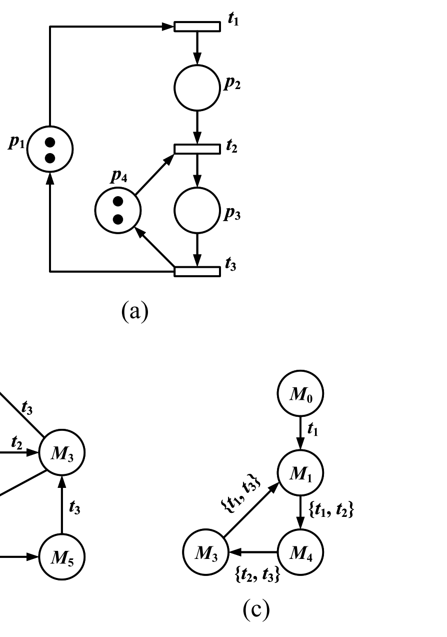


Fig. 5. (a) PN (N_5, M_0), (b) its RG, and (c) its MG.

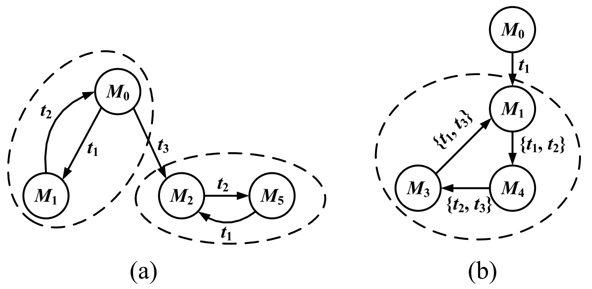


Fig. 6. (a) MG of (N_1, M_0) with circle-labeled maximal SCCs and (b) MG of (N_5, M_0) with a circle-labeled leaf strongly connected component.

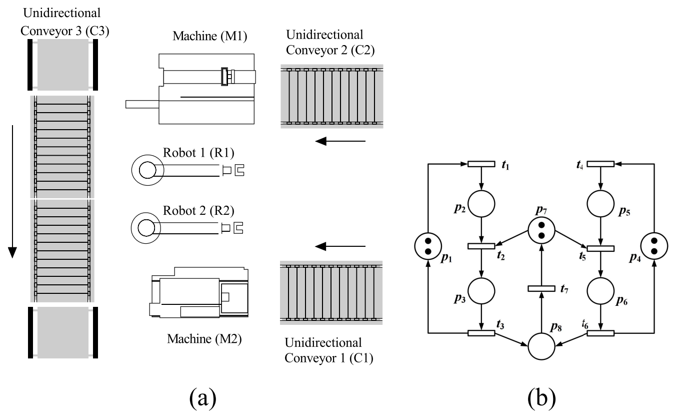

Unidirectional Conveyor 3 (C3)
Machine (M1)
Unidirectional Conveyor 2 (C2)
Robot 1 (R1)
Robot 2 (R2)
Machine (M2)
Unidirectional Conveyor 1 (C1)
(a)
t1
t4
p2
p7
p5
p1
t2
t5
p4
p3
t7
p6
t3
p8
t6
(b)

Fig. 7. (a) Layout of AMS and (b) PN model (N_6, M_0) of AMS.

generating rule of MG, for any marking M in U, $\exists\ \delta \in T^*$ such that $M_0[\delta\rangle M$. Suppose that $\forall$ $e_i \in \mathbb{E}_u$ is labeled by γ_i with $\bigcup_{i\in\mathbb{N}^+} \gamma_i = T$. We have a labeling function $\mathbb{L}_u$: $\mathbb{E}_u \to \Gamma$, where $\Gamma = \{\gamma_i \mid i\in \mathbb{N}^+\}$. Note that $\overset{\leftrightarrow}{U}$ is a leaf strongly connected component of Θ, thus there is no child node of nodes in $\overset{\leftrightarrow}{U}$ and $\forall$ $M' \in R(N, M)$, $M' \in U$. For any γ in Γ, $\exists\ \mu \in \Gamma^*$ s.t. $M[\mu\rangle M'$ and $M'[\gamma\rangle$ hold. Besides $\forall$ $t \in \gamma$, $\gamma' = \gamma - \{t\}$, $\exists\ M'' \in R(N, M')$ s.t. $M'[\gamma'\rangle M''[t\rangle$ holds. Since we have known that $\bigcup_{\gamma_i\in\Gamma, i\in\mathbb{N}^+} \gamma_i = T$, the conclusion $M''[t\rangle$ with $\exists\ M'' \in R(N, M')$ is true for each transition t of T. Similarly, this can be done for other leaf-strongly connected components in Θ. In summary, we conclude that (N, M_0) is live.

(*only if*) The precondition is that (N, M_0) is live. By contradiction, if there is a leaf strongly connected component $\overset{\leftrightarrow}{U}$ where the sets $\gamma_1, \ldots,$ and γ_n that label the n edges in $\overset{\leftrightarrow}{U}$ satisfy $\bigcup_{i\in\mathbb{N}^+} \gamma_i \neq T$ with $U \subseteq \mathbb{M}$, we must consider two cases.

Case 1: There is no edge in $\overset{\leftrightarrow}{U}$ s.t. $\bigcup_{i\in\mathbb{N}^+} \gamma_i = \emptyset$. There exists at least a deadlock node in MG. According to Theorem 1, (N, M_0) is deadlock, which is contradictory to the precondition.

Case 2: $\bigcup_{i\in\mathbb{N}^+} \gamma_i \subsetneqq T$. Let a subgraph $\Theta_u = (U, \mathbb{E}_u, \mathbb{L}_u, M_0)$ define a leaf strongly connected component $\overset{\leftrightarrow}{U}$ of Θ. Obviously, the labeling function is $\mathbb{L}_u$: $\mathbb{E}_u \to \Gamma$, where $\Gamma = \{\gamma_i \mid i\in \mathbb{N}^+\}$. In this case, $\bigcup_{\gamma_i\in\Gamma, i\in\mathbb{N}^+} \gamma_i \subsetneqq T$ means that there exist some but not all transitions in T enabled at a marking in U, i.e., $\exists\ t \in T$ $\forall$ $M \in R(N, M_0)$, $\exists\ M' \in R(N, M)$ with $M' \in U$ such that $M'[t\rangle$. In other words, $\exists\ t \in T$, t is not live at M_0, which also contradicts the precondition. ■

Example 3: Considering MG of (N_1, M_0) shown in Fig. 6(a), its leaf strongly connected component is $\overset{\leftrightarrow}{U}$, where $U = \{M_2, M_5\}$. The directed edge (M_2, M_5) is labeled by t_2 and (M_5, M_2) is labeled by t_1. By Theorem 3, (N_1, M_0) is deadlock-free since $\{t_1\} \cup \{t_2\} = \{t_1, t_2\} \neq T$. Regarding MG of ($N_5$, M_0) shown in Fig. 6(b), the leaf strongly connected component is $\overset{\leftrightarrow}{U}$, where $U = \{M_1, M_3, M_4\}$. The union of enabled steps, which label directed edges of $\overset{\leftrightarrow}{U}$, is $\{t_1, t_2\} \cup \{t_2, t_3\} \cup \{t_1, t_3\} = T$. Thus, ($N_3$, M_0) is live.

When assessing the computational complexity of analyzing the liveness of PN, it is beneficial to search for maximal SCC in MG. The complexity of computing maximal SCC in a directed graph $G = (V, E)$ has been efficiently solved as $O(V+E)$ (e.g., by *Tarjan algorithm*), which is $O(|\mathbb{M}(N, M_0)| + |\mathbb{E}(N, M_0)|)$ in MG, where $\mathbb{M}(N, M_0)$ and $\mathbb{E}(N, M_0)$ are the sets of markings and directed edges between markings in MG, respectively. Except for the worst case, MG's state space is much more compact than RG regarding the number of arcs and nodes. Roughly speaking, generating MG's maximal SCC performs better than RG from the viewpoint of computational complexity. Concrete details about the evaluation results are shown in Section IV.

IV. Evaluation Results on Liveness Analysis

We use an AMS [41] to discuss the practicability of our method. The system's layout is shown in Fig. 7(a). There exist two robots R_1 and R_2, two machines M_1 and M_2, and three unidirectional conveyors C_1–C_3. Raw materials are transported to a machine $M_1(M_2)$ via a conveyor $C_2(C_1)$, and a finished product is unloaded to C_3 from $M_1(M_2)$ by a robot $R_1(R_2)$. The PN is depicted in Fig. 7(b), where p_1, p_4, and p_7 represent M_1, M_2, and robots, separately. Tokens in p_1, p_4, and p_7 denote that each machine can handle at most two materials and each robot grabs one product at a time.

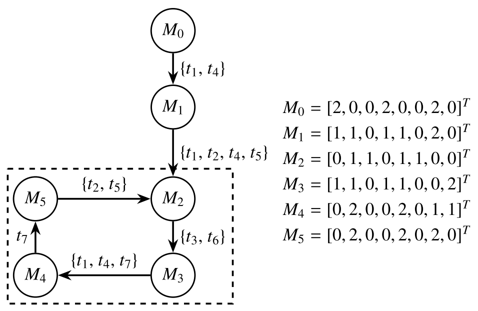

M_0
$\{t_1, t_4\}$
M_1
$\{t_1, t_2, t_4, t_5\}$
M_2
$\{t_2, t_5\}$
M_5
t_7
$\{t_3, t_6\}$
M_4
$\{t_1, t_4, t_7\}$
M_3
$M_0 = [2, 0, 0, 2, 0, 0, 2, 0]^T$
$M_1 = [1, 1, 0, 1, 1, 0, 2, 0]^T$
$M_2 = [0, 1, 1, 0, 1, 1, 0, 0]^T$
$M_3 = [1, 1, 0, 1, 1, 0, 0, 2]^T$
$M_4 = [0, 2, 0, 0, 2, 0, 1, 1]^T$
$M_5 = [0, 2, 0, 0, 2, 0, 2, 0]^T$

Fig. 8. MG of (N_6, M_0) with rectangle-labeled maximal SCC.

TABLE I
METHODS COMPARISON IN LIVENESS ANALYSIS

Methods	PG	CSG	HPSG	MGSG	MG
Deadlock exploration	✓[a]	✓	✓	✓	✓
Liveness exploration	×[b]	✓	×	×	✓
Liveness analysis method	×	×	×	×	✓

[a] "✓" means that in the corresponding paper, this aspect is discussed.
[b] "×" means that this aspect is not discussed in the corresponding paper and there is no further study in such an aspect.

For this net model, its RG consists of 61 reachable markings and 178 arcs among markings. Its MG with a rectangle-labeled maximal SCC is displayed in Fig. 8. MG comprises only 6 markings and 6 directed arcs connecting these markings, significantly smaller than RG. According to Theorem 3, (N_6, M_0) is live since all transitions $t_1 - t_7$ are encompassed in the maximal SCC. Thus, all raw materials loaded into the system can be successfully processed.

The existing partial order methods can also effectively reduce the state space of net systems, but some have not proven that they can be applied to evaluate PN's liveness. To show the differences between MG and several partial order methods, we report the comparison results in Table I, where PG is persistent graphs [30], CSG means covering step graphs [31], and HPSG represents HPSG [32].

It can be seen that all methods can be used to detect deadlocks, and CSG may check the liveness of a net system. Unfortunately, it does not give a specific method about how to analyze liveness via CSG in [31]. Besides, it is proven that CSG is E-live iff RG is E-live [31], where E represents a subset of T. E-live indicates that each transition of E is firable at each reachable marking. Clearly, the T-liveness of a PN means that the net is live.

Subsequently, we shall test these methods on several realistic systems with large sizes to evaluate their effect on state-space reduction and property analysis. Four models applicated by experiments are detailed.

1) In AMS depicted by Fig. 7(a), we can treat a single conveyor, robot, and machine as a unified entity. AMS(n) represents an expanded system model that incorporates n additional encapsulated entities into the system.
2) FMS is the abbreviation for flexible manufacturing systems. To distinguish systems depicted in [42, Fig. 1] and [43, Fig. 1], we use FMS1 and FMS2 to label them, respectively. The PN modeling FMS1 and FMS2 are shown separately in [42, Fig. 7] and [43, Fig. 2]. FMS1(n) denotes a PN model of FMS1 where there are n tokens in p_{10}–p_{14}, p_{24}, p_{x6}, p_{x8}, p_{y1}, p_{y2}, p_{y4}, p_{y5}, and p_{y8}. FMS2(n) is a PN model of FMS2, where n represents the number of tokens in p_{20}, p_{21}, and p_{22}.
3) $||_n$AMS is a model of a concurrent system containing several parallel AMS instances [41], where n means the number of parallel PN used to model AMS.

Experimental results are exhibited in Table II. Graphs, namely, RG, PG, CSG, and HPSG, are computed by a tool TINA from http://projects.laas.fr/tina//home.php. The experiments concerning MGSG and MG are performed on a PC with an Intel i7-9700K 3.6 GHz CPU and 32 GB memory under Ubuntu 18.04 operating system.

In Table II, nodes and directed edges denote the numbers of markings and arcs between markings in corresponding graphs. Time (in seconds) for RG, PG, CSG, and HPSG is reported by TINA. The time consumption of MGSG and MG represents the total real-time usage of their respective programs. Such programs run multiple times for each measurement and then calculate an average value to avoid the unstable I/O loading time. Besides, "NC" means "no computation" since we cannot obtain the corresponding information. "NR" means "no result" since the tool cannot output results due to the state-space explosion problems. Note that "AZ" seconds imply that the time consumption is lower than the negative quadratic of ten and thus is approximated to zero in TINA.

In Table II, the bolded content indicates the superior result among the compared elements. According to the results exhibited in this table, some observations can be made regarding the respective effect of PG, CSG, HPSG, MGSG, and MG on state-space reduction and liveness analysis.

1) PG works well on reducing the number of reachable markings and arcs between markings. Unfortunately, its liveness-analysis result is incorrect in some cases. For instance, $||_2$AMS and $||_3$AMS are live nets through the analysis of their RG, while PG returns that they have deadlocks.
2) CSG delivers a much less impressive state-space reduction result than the existing partial order techniques and MG. Besides, by using TINA, CSG cannot return the liveness-analysis results of net systems.
3) The MGSG method performs better in reducing the state space of systems than the established classical partial order techniques like PG, CSG, and HPSG. Nevertheless, the time required to generate MGSG may be longer than that needed to construct PG or HPSG, which depends upon the MGSG generation algorithm [35] and the input PN structure. For instance, in FMS2(n), each transition has a multitude of conflict transitions. Moreover, considering the complex processes within the system, an array of firing sequences exists between different transitions in the PN model, resulting in significant time overhead when attempting to identify sound steps for each marking. The worst-case scenario entails the exhaustive exploration of firing sequences without discovering any sound step.
4) In this study, we have revised the algorithm for identifying maximal sound steps at each marking, thus

TABLE II
EVALUATION RESULTS IN PROPERTIES ANALYSIS FOR MANUFACTURING-ORIENTED PN

Petri nets	Graphs	Nodes	Directed edges	CPU time (s)	Property	Petri nets	Graphs	Nodes	Directed edges	CPU time (s)	Property
FMS1(*n*)[a]						$\parallel_n$AMS[b]					
FMS1(1)	RG	522	1485	AZ[e]	Liveness	$\parallel_2$AMS	RG	3721	21716	6.20×10^{-2}	Liveness
	PG	25	26	AZ	NC[f]		PG	14	15	AZ	Deadlock[*]
	CSG	68	98	AZ	NC		CSG	26	53	AZ	NC
	HPSG	19	20	AZ	NC		HPSG	11	14	AZ	NC
	MGSG	19	20	1.84×10^{-4}	NC		MGSG	11	14	4.20×10^{-5}	NC
	MG	**18**	**19**	**1.67×10^{-4}**	**Liveness**		**MG**	**6**	**6**	**3.50×10^{-5}**	**Liveness**
FMS1(2)	RG	39477	187727	3.10×10^{-2}	Liveness	$\parallel_3$AMS	RG	226981	1987014	11.95	Liveness
	PG	85	95	AZ	NC		PG	18	19	AZ	Deadlock[*]
	CSG	1240	2339	1.60×10^{-2}	NC		CSG	82	201	AZ	NC
	HPSG	66	76	AZ	NC		HPSG	19	26	AZ	NC
	MGSG	66	76	1.92×10^{-4}	NC		MGSG	18	19	2.58×10^{-4}	NC
	MG	**62**	**72**	**1.57×10^{-4}**	**Liveness**		**MG**	**6**	**6**	**1.85×10^{-4}**	**Liveness**
FMS1(3)	RG	1162472	7258425	37.77	Liveness	$\parallel_4$AMS	RG	NR	NR	NR	NR
	PG	184	210	AZ	NC		PG	22	23	AZ	Deadlock[*]
	CSG	9935	21995	1.72×10^{-2}	NC		CSG	290	785	3.10×10^{-2}	NC
	HPSG	143	170	AZ	NC		HPSG	35	50	AZ	NC
	MGSG	143	170	2.87×10^{-4}	NC		MGSG	22	23	5.46×10^{-4}	NC
	MG	**138**	**165**	**2.43×10^{-4}**	**Liveness**		**MG**	**6**	**6**	**2.49×10^{-4}**	**Liveness**
FMS1(4)	RG	NR[g]	NR	NR	NR	$\parallel_5$AMS	RG	NR	NR	NR	NR
	PG	332	385	AZ	NC		PG	26	27	AZ	Deadlock[*]
	CSG	39568	98396	1.08	NC		CSG	1090	3105	6.30×10^{-2}	NC
	HPSG	275	334	AZ	NC		HPSG	67	98	AZ	NC
	MGSG	275	334	5.14×10^{-3}	NC		MGSG	26	27	7.82×10^{-4}	Liveness
	MG	**269**	**328**	**4.61×10^{-3}**	**Liveness**		**MG**	**6**	**6**	**2.81×10^{-4}**	**Liveness**
FMS1(5)	RG	NR	NR	NR	NR	$\parallel_6$AMS	RG	NR	NR	NR	NR
	PG	528	625	AZ	NC		PG	30	31	AZ	Deadlock[*]
	CSG	101749	277929	3.06	NC		CSG	4226	12353	4.53×10^{-1}	NC
	HPSG	443	555	AZ	NC		HPSG	131	194	AZ	NC
	MGSG	443	555	6.21×10^{-3}	NC		MGSG	30	31	8.01×10^{-4}	NC
	MG	**436**	**548**	**5.78×10^{-3}**	**Liveness**		**MG**	**6**	**6**	**3.17×10^{-4}**	**Liveness**
FMS2(*n*)[c]						AMS(*n*)[d]					
FMS2(1)	RG	26750	93320	4.85×10^{-1}	Deadlock	AMS(6)	RG	6227611	72943144	827.99	Liveness
	PG	**15489**	**35094**	**1.09×10^{-1}**	**Deadlock**		PG	34	41	AZ	NC
	CSG	26323	86763	5.78×10^{-1}	Deadlock		CSG	133568	498639	3.85	NC
	HPSG	15515	35512	1.56×10^{-1}	Deadlock		HPSG	19	26	AZ	NC
	MGSG	**15489**	**35094**	3.52×10^{-1}	**Deadlock**		MGSG	19	26	4.29×10^{-5}	NC
	MG	**15489**	**35094**	2.71×10^{-1}	**Deadlock**		**MG**	**12**	**12**	**2.81×10^{-5}**	**Liveness**
FMS2(2)	RG	439479	2393013	15.36	Deadlock	AMS(16)	RG	NR	NR	NR	NR
	PG	**93077**	**241726**	**1.72**	**Deadlock**		PG	74	91	AZ	NC
	CSG	364025	1814983	14.39	Deadlock		CSG	NR	NR	NR	NR
	HPSG	95002	253019	2.33	Deadlock		HPSG	39	56	AZ	NC
	MGSG	**93077**	**241726**	1.83	**Deadlock**		MGSG	39	56	7.89×10^{-5}	NC
	MG	**93077**	**241726**	1.77	**Deadlock**		**MG**	**22**	**22**	**5.10×10^{-5}**	**Liveness**
FMS2(3)	RG	3101809	20989580	451.17	Deadlock	AMS(26)	RG	NR	NR	NR	NR
	PG	**348304**	**928653**	**8.87**	**Deadlock**		PG	118	145	AZ	NC
	CSG	2787296	16678065	578.94	Deadlock		CSG	NR	NR	NR	NR
	HPSG	362552	1015015	8.91	Deadlock		HPSG	59	86	AZ	NC
	MGSG	**348304**	**928653**	11.74	**Deadlock**		MGSG	59	86	1.24×10^{-4}	NC
	MG	**348304**	**928653**	9.32	**Deadlock**		**MG**	**32**	**32**	**8.49×10^{-5}**	**Liveness**

[a] n tokens in $p_{10}-p_{14}$, p_{24}, p_{x_6}, p_{x_8}, p_{y_1}, p_{y_2}, p_{y_4}, p_{y_5}, and p_{y_8} of a PN modeling FMS in Fig. 1 of [42].
[b] n parallel Petri nets modeling AMS in Figure 9.4(a) of [41].
[c] n tokens in p_{20}, p_{21} and p_{22} of a PN modeling FMS in Fig. 1 of [43].
[d] Adding n unified entity into AMS shown in Fig. 7(a), where each unified entity corresponds to a single conveyor, robot, and machine.
[e] "AZ" seconds mean that the time consumption is lower than the negative quadratic of ten and thus is approximated to zero in TINA.
[f] "NC" means "no computation" since the tool cannot return the corresponding information.
[g] "NR" means "no result" since the tool TINA cannot output the results due to the state-space explosion problems.
[*] "*" indicates that the result is incorrect.

reducing the computational complexity compared to the algorithm presented in [35]. Thanks to these algorithmic improvements, the generation time for MG is shorter than that for MGSG. Furthermore, MG exhibits a more compact size than MGSG, owing to the methodological enhancements.

5) MG offers the most substantial reduction in the number of reachable markings and arcs between markings. Moreover, the time required for liveness analysis of PN via MG is 10 to 1000 times faster than RG, and the state space of MG is much smaller than that of RG. Under some circumstances, e.g., FMS1(4), FMS1(5), $||_4$AMS–$||_6$AMS, AMS(16), and AMS(26), state-space explosion issues prevent RG from accurately exploring the properties of PN. Besides, HPSG and MGSG cannot be employed to attain the liveness-analysis result of net systems. Fortunately, MG proves to be a useful approach for checking the liveness of net systems.

In general, MGs can significantly decrease the state space and the time consumption needed to analyze the liveness of net systems. Thus, the proposed MG method performs well when applied in verifying the properties of some large-size realistic systems.

V. Conclusion and Future Work

This work introduces a MG that represents an improved version of the previously proposed MGSGs [35]. MG serves as the basis for a new method to analyze the liveness of PN, specifically by leveraging the maximal SCC within MG. Although there exist some partial order methods [30], [31], [32] capable of reducing the state space of PN, they achieve a less significant state-space reduction than MG does. Furthermore, these techniques have not demonstrated efficacy in checking the liveness of PN. Experimental results using several manufacturing-oriented PN show that our proposed method performs better in the state-space reduction and liveness analysis than the existing partial order methods and RG.

This article concentrates on ordinary and bounded PN with precise information. Some directions for future work include: 1) Investigating the state-space reduction issues for some complex PN like general and unbounded PN [44], [45], [46], [47], [48], [49] and 2) measuring the complexity of state-space reduction for some PN with imprecise information like uncertainty, ambiguity, and missing [50]. Furthermore, we attempt to analyze some other reachability properties and synthesize liveness-enforcing controllers [51], [52], [53] by using MGs of PN.

References

[1] C. Chen, A. Raman, H. S. Hu, and R. S. Sreenivas, "On liveness enforcing supervisory policies for arbitrary Petri nets," *IEEE Trans. Autom. Control*, vol. 65, no. 12, pp. 5236–5247, Jan. 2020.

[2] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optim.*, vol. 25, no. 1, pp. 206–230, Jan. 1987.

[3] P. J. Rampage and W. M. Wonham, "The control of discrete event systems," *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, Jan. 1989.

[4] J. L. Luo, Y. X. Wan, W. M. Wu, and Z. W. Li, "Optimal Petri-net controller for avoiding collisions in a class of automated guided vehicle systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4526–4537, Nov. 2020.

[5] X. J. Wang, H. S. Hu, and M. C. Zhou, "Discrete event approach to robust control in automated manufacturing systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 1, pp. 123–135, Jan. 2022.

[6] J. C. Luo, Z. Q. Liu, S. G. Wang, and K. Y. Xing, "Robust deadlock avoidance policy for automated manufacturing system with multiple unreliable resources," *IEEE/CAA J. Automatica Sinica*, vol. 7, no. 3, pp. 812–821, May 2020.

[7] N. Du, H. S. Hu, and M. C. Zhou, "Robust deadlock avoidance and control of automated manufacturing systems with assembly operations using Petri nets," *IEEE Trans. Automat. Sci. Eng.*, vol. 17, no. 4, pp. 1961–1975, Oct. 2020.

[8] N. Q. Wu and M. C. Zhou, "Modeling and deadlock avoidance of automated manufacturing systems with multiple automated guided vehicles," *IEEE Trans. Syst., Man, Cybern., Part-B, Cybern.*, vol. 35, no. 6, pp. 1193–1202, Dec. 2005.

[9] C. A. Petri, "Kommunikation mit automaten," Ph.D. dissertation, Faculty Math. Phys., Univ. Bonn, Bonn, Germany, 1962.

[10] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.

[11] R. David and H. Alla, "Petri nets for modeling of dynamic systems: A survey," *Automatica*, vol. 30, no. 2, pp. 175–202, Feb. 1994.

[12] V. Gehlot and C. Nigro, "An introduction to systems modeling and simulation with colored Petri nets," in *Proc. Winter Simul. Conf.*, Baltimore, Maryland, USA, 2010, pp. 104–118.

[13] S. G. Wang, X. Guo, O. Karoui, M. C. Zhou, D. You, and A. Abusorrah, "A refined siphon-based deadlock prevention policy for a class of Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 53, no. 1, pp. 191–203, Jan. 2023.

[14] D. You, O. Karoui, and S. G. Wang, "Computation of minimal siphons in Petri nets using problem partitioning approaches," *IEEE/CAA J. Automatica Sinica*, vol. 9, no. 2, pp. 329–338, Feb. 2022.

[15] O. Oanea, H. Wimmel, and K. Wolf, "New algorithms for deciding the siphon-trap property," in *Proc. 31st Int. Conf. Appl. Theory Petri Nets*, Brage, Portugal, 2010, pp. 267–286.

[16] S. G. Wang, D. You, and M. C. Zhou, "A necessary and sufficient condition for a resource subset to generate a strict minimal siphon in S^{4}PR," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 4173–4279, Aug. 2017.

[17] B. Huang, M. C. Zhou, C. Wang, A. Abusorrah, and Y. Al-Turki, "Deadlock-free supervisor design for robotic manufacturing cells with uncontrollable and unobservable events," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 3, pp. 597–605, Mar. 2021.

[18] T. Nishi, Y. Watanabe, and M. Sakai, "An efficient deadlock prevention policy for noncyclic scheduling of multicluster tools," *IEEE Trans. Automat. Sci. Eng.*, vol. 15, no. 4, pp. 1677–1691, Oct. 2018.

[19] S. G. Wang et al., "Computation of an emptiable minimal siphon in a subclass of Petri nets using mixed-integer programming," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 1, pp. 219–226, Jan. 2021.

[20] L. B. Han, K. Y. Xing, M. C. Zhou, H. X. liu, and F. Wang, "Two-stage deadlock prevention policy based on resource-transition circuits," in *Proc. IEEE Int. Conf. Automat. Sci. Eng.*, Seoul, South Korea, 2012, pp. 741–746.

[21] Y. X. Feng, K. Y. Xing, M. C. Zhou, F. Tian, and H. X. Liu, "Structural liveness analysis of automated manufacturing systems modeled by S^{4}PRs," *IEEE Trans. Automat. Sci. Eng.*, vol. 16, no. 4, pp. 1952–1959, Oct. 2019.

[22] M. Notomi and T. Murata, "Hierarchical reachability graph of bounded Petri nets for concurrent-software analysis," *IEEE Trans. Softw. Eng.*, vol. 20, no. 5, pp. 325–336, May 1994.

[23] R. Lipton, "The reachability problem requires exponential space," Rep. 62, Dept. Comput. Sci., Yale Univ., New Haven, CT, USA, 1976.

[24] L. Jérôme and S. Sylvain, "Reachability in vector addition systems is primitive-recursive in fixed dimension," in *Proc. 34th Annu. ACM/IEEE Symp. Log. Comput. Sci.*, Vancouver, BC, Canada, 2019, pp. 1–13.

[25] W. Czerwiński, S. Lasota, R. Lazić, J. Leroux, and F. Mazowiecki, "The reachability problem for Petri nets is not elementary," *J. ACM*, vol. 68, no. 1, pp. 1–28, Dec. 2020.

[26] P. Godefroid, "Using partial orders to improve automatic verification methods," in *Proc. 2nd Int. Conf. Comput. Aided Verificat.*, New Brunswick, New Jersey, USA, 1990, pp. 176–185.

[27] P. Godefroid, D. Peled, and M. Staskauskas, "Using partial-order methods in the formal validation of industrial concurrent programs," *IEEE Trans. Software Eng.*, vol. 22, no. 7, pp. 496–507, Jul. 1996.

[28] D. Peled and T. Wilke, "Stutter-invariant temporal properties are expressible without the next-time operator," *Inf. Process. Lett.*, vol. 63, no. 5, pp. 243–246, Sep. 1997.

[29] A. Valmari and H. Hansen, "Can stubborn sets be optimal?" *Fundamenta Informaticae*, vol. 113, nos. 3–4, pp. 377–397, Jan. 2011.

[30] P. Godefroid, J. van Leeuwen, J. Hartmanis, G. Goos, and P. Wolper,*Partial-Order Methods for the Verification of Concurrent Systems: An Approach to the State-Explosion Problem*, vol. 1032, (Lecture Notes Computer Science). Berlin, Germany: Springer, Jan. 1996.

[31] F. Vernadat, P. Azema, and F. Michel, "Covering step graph," in *Proc. 17th Int. Conf. Appl. Theory Petri Nets*, Osaka, Japan, 1996, pp. 516–535.

[32] P. O. Ribet, F. Vernadat, and B. Berthomieu, "On combining the persistent sets method with the covering steps graph method," in *Proc. 22nd IFIP WG 6.1 Int. Conf. Formal Techn. Netw. Distrib. Syst.*, Houston, TX, USA, 2002, pp. 344–359.

[33] K. Barkaoui, H. Boucheneb, and Z. W. Li, "Exploiting local persistency for reduced state-space generation," *Innovat. Syst. Softw. Eng.*, vol. 16, no. 2, pp. 181–197, Feb. 2020.

[34] P. Godefroid and D. Pirottin, "Refining dependencies improves partial-order verification methods," in *Proc. 5th Int. Conf. Comput. Aided Verificat.*, Elounda, Greece, 1993, pp. 438–449.

[35] H. Dou, K. Barkaoui, H. Boucheneb, X. N. Jiang, and S. G. Wang, "Maximal good step graph methods for reducing the generation of the state space," *IEEE Access*, vol. 7, pp. 155805–155817, 2019.

[36] Z. H. Ding, M. C. Zhou, and S. G. Wang, "Ordinary differential equation-based deadlock detection," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 10, pp. 1035–1454, Apr. 2014.

[37] D. B. West, *Introduction to Graph Theory*. Upper Saddle River, NJ, USA: Prentice Hall, 2001.

[38] Z. W. Li and M. C. Zhou, *Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach*. London, U.K.: Springer, 2009.

[39] M. C. Zhou and F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*. New York, NY, USA: Springer, 2012.

[40] G. Frey, "Automatic implementation of Petri net based control algorithms on PLC," in *Proc. Am. Control Conf.*, Chicago, IL, USA, 2000, pp. 2819–2823.

[41] Y. F. Chen and Z. W. Li, *Optimal Supervisory Control of Automated Manufacturing Systems*. New York, NY, USA: CRC Press, 2013.

[42] M. C. Zhou, K. McDermott, and P. A. Patel, "Petri net synthesis and analysis of a flexible manufacturing system cell," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 2, pp. 523–531, Apr. 1993.

[43] Z. W. Li, M. C. Zhou, and N. Q. Wu, "A survey and comparison of Petri net-based deadlock prevention policies for flexible manufacturing systems," *IEEE Trans. Syst., Man, Cybern., Part-C, Appl. Rev.*, vol. 38, no. 2, pp. 173–188, Feb. 2008.

[44] Y. X. Feng, K. Y. Xing, M. C. Zhou, X. N. Wang, and H. X. Liu, "Robust deadlock prevention for automated manufacturing systems with unreliable resources by using general Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 10, pp. 3515–3527, Oct. 2020.

[45] C. Chen and H. S. Hu, "Extended place-invariant control in automated manufacturing systems using Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 3, pp. 1807–1822, Mar. 2022.

[46] K. Barkaoui and H. Boucheneb, "On persistency in time Petri nets," in *Proc. 16th Int. Conf. Formats*, Beijing, China, 2018, pp. 108–124.

[47] F. M. Lu, Q. T. Zeng, M. C. Zhou, Y. X. Bao, and H. Duan, "Complex reachability trees and their application to deadlock detection for unbounded Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 6, pp. 1164–1174, Jun. 2019.

[48] J. Li, X. L. Yu, and M. C. Zhou, "Analysis of unbounded Petri net with lean reachability trees," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 6, pp. 2007–2016, Jun. 2020.

[49] S. G. Wang, M. D. Gan, and M. C. Zhou, "Macro liveness graph and liveness of ω-independent unbounded nets," *Sci. China Inf. Sci.*, vol. 58, no. 3, pp. 1–10, Mar. 2015.

[50] M. Y. Cai, Y. Z. Lin, B. Han, C. J. Liu, and W. J. Zhang, "On a simple and efficient approach to probability distribution function aggregation," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 9, pp. 2444–2453, Apr. 2016.

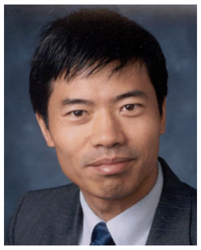
[51] W. J. Zhang, Z. M. Bi, and X. F. Zha, "A generic Petri net model for flexible manufacturing systems and its use for FMS control software testing," *Int. J. Prod. Res.*, vol. 38, no. 5, pp. 1109–1131, Nov. 2000.

[52] K. Y. Xing, F. Wang, M. C. Zhou, H. Lei, and J. C. Luo, "Deadlock characterization and control of flexible assembly systems with Petri nets," *Automatica*, vol. 87, pp. 358–364, Jan. 2018.

[53] Z. L. Zhang, G. Y. Liu, K. Barkaoui, and Z. W. Li, "Adaptive deadlock control for a class of Petri nets with unreliable resources," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 5, pp. 3113–3125, May 2022.

Hao Dou (Graduate Student Member, IEEE) received the B.S. degree in communication engineering and the M.S. degree in information and communication engineering from the School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou, China, in 2017 and 2020, respectively. She is currently pursuing the Ph.D. degree in intelligent science and systems with the Institute of Systems Engineering, Macau University of Science and Technology, Macau, China.

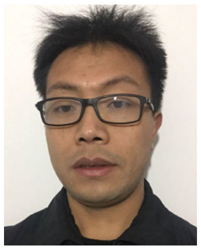
Her current research interests include Petri nets and supervisory control of discrete event systems.



MengChu Zhou (Fellow, IEEE) received the B.S. degree in control engineering from the Nanjing University of Science and Technology, Nanjing, China, in 1983, the M.S. degree in automatic control from the Beijing Institute of Technology, Beijing, China, in 1986, and the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990.

He joined the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA, in 1990, and has been a Distinguished Professor since 2013. He has over 1200 publications, including 17 books, over 800 journal papers, including over 650 IEEE TRANSACTIONS papers, 31 patents, and 32 book-chapters. His recently coauthored books include *Learning Automata and their Applications to Intelligent Systems* (IEEE Press/Wiley, Hoboken, NJ, 2024, with J. Zhang) and *Device-Edge-Cloud Continuum Paradigms, Architectures and Applications* (Springer Nature, 2023, with C. Savaglio, G. Fortino, and J. Ma). His interests are in intelligent automation, robotics, Petri nets, Internet of Things, edge/cloud computing, and big data analytics.

Dr. Zhou is a recipient of Excellence in Research Prize and Medal from NJIT, the Humboldt Research Award for U.S. Senior Scientists from Alexander von Humboldt Foundation, and the Franklin V. Taylor Memorial Award and the Norbert Wiener Award from IEEE Systems, Man, and Cybernetics Society, and the Edison Patent Award from the Research and Development Council of New Jersey. He is Fellow of International Federation of Automatic Control, American Association for the Advancement of Science, Chinese Association of Automation, and National Academy of Inventors.



Shouguang Wang (Senior Member, IEEE) received the B.S. degree in computer science from the Changsha University of Science and Technology, Changsha, China, in 2000, and the Ph.D. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 2005.

He is currently a Professor with the School of Information and Electronic Engineering, the Director of the Discrete-Event Systems Group, and the Dean of System Modeling and Control Research Institute, Zhejiang Gongshang University, Hangzhou. He was with the University of Cagliari, Cagliari, Italy, and a Visiting Professor with the New Jersey Institute of Technology, Newark, NJ, USA.

Prof. Wang serves as an Associate Editor for IEEE/CAA JOURNAL OF AUTOMATICA SINICA.

Aiiad Albeshri received the M.S. and Ph.D. degrees in information technology from the Queensland University of Technology, Brisbane, QLD, Australia, in 2007 and 2013, respectively.

He has been an Associate Professor with the Department of Computer Science, King Abdulaziz University, Jeddah, Saudi Arabia, since 2018. His current research focuses on information security, trust in cloud computing, big data, and HPC.