Received October 9, 2017, accepted November 7, 2017, date of publication November 13, 2017, date of current version December 22, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2772829

A Compositional Analysis Method for Petri-Net Models

JIE DING[1,2], XIAO CHEN[1,3], (Member, IEEE), AND RUI WANG[1,2]
[1]School of Information Engineering, Yangzhou University, Yangzhou 225127, China
[2]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[3]School of Computer Science and Communication Engineering, Zhenjiang 212013, China

Corresponding author: Xiao Chen (xiaochen@ujs.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61472343 and Grant 61702233, in part by the Natural Science Foundation of Jiangsu, China, under Grant BK20160543, and in part by the Foundation of Jiangsu University under Grant 14JDG122 and Grant BM20082061507. The work of J. Ding was supported by the Blue Project for Young Research Leader of Jiangsu, China.

ABSTRACT Compositional analysis aims to reveal the underlying structures of a large-scale system or network by analyzing its constituent components and their relationships. Today's mathematical modeling languages, such as Petri nets, are useful for describing distributed systems and complex networks. However, the flat model architecture of Petri nets makes it difficult for them to depict the compositional structures of a large-scale model. Therefore, an enhanced compositionality feature has become a significant demand in large-scale modeling with Petri nets. This paper explores the underlying compositional structures of a given Petri net model by using a proposed sorting algorithm. The algorithm analyses compositional structures by sorting an incidence matrix that is generated from the Petri net model. Finally, the proposed sorting algorithm is applied to a traffic network model that was built with Petri nets to analyze its compositional structures, which represent different traffic lines, with the aim of optimizing the traffic network.

INDEX TERMS Compositionality, Petri-nets, incidence matrix, sorting.

I. INTRODUCTION

In mathematics and semantics, compositionality is the principle that the meaning of a complex expression is determined by the meanings of its constituent expressions and their relationships. Compositionality is a key feature of some popular mathematical languages that are applied for modelling compositional architectures. The Petri-net, which was initially defined by Carl Adam Petri in 1939, is one such popular mathematical modelling language.

The Petri net is a widespread modelling language for describing a family of related Discrete Event Dynamic Systems (DEDS) formalisms. According to Silva's description [1], Petri nets can be represented as bipartite graphs, in which *state variables* and *state transformers* are named *places (P)* and *transitions (T)*, respectively. *Places* are represented by circles, and *Transactions* are represented by bars; they are connected by *pre-incidence (input)* and *post-incidence (output)* functions (*Pre* and *Post*). *Pre (Post)* defines a connection from places to transitions (transitions to places) [1]. According to Silva's description [1], a net structure can be considered as a quaternion $N = \langle P, T, Pre, Post\rangle$. A Petri-net system is represented by a net structure, in which *Pre* and *Post* can be defined as two $|P| \times |T|$ sized incidence matrices that describe the relationships between *Ps* and *Ts*.

Petri nets offer a graphical notation for stepwise process modelling and have an exact mathematical definition of their execution semantics, with a well-developed mathematical theory for process analysis. Therefore, formality is a core feature of Petri nets compared with other modelling languages or industry standards. Moreover, locality and structuration can be considered two additional core features of Petri nets based on the syntax. In a Petri net, the description is given in ''local'' terms: local states (places) and local state changes (transitions). This is a cornerstone for constructing net models [2]. It achieves the refinement of a place or a transition (i.e., to make the model more detailed), or the composition of two modular components by identifying shared transitions (merging two or more into one) or places (fusion operation).

However, the recent growth of technologies has promoted systems to more comprehensive and large scales [3]–[5]. For large-scale networks or systems with complex structures and cooperations, Petri nets' locality feature produces a model with a dizzying layout due to the great number of places and transitions. As a result, it is difficult to distinguish its complex inner structures from the layout. This is a drawback of the ''compositionality '' of Petri nets in large scale models. New applications require further improvements by exploring the new feature of Petri nets in compositionality. This paper aims to enhance the compositional feature of Petri nets by analysing the structure of Petri net models.

To address this issue, a decomposition technique is applied to decompose a large Petri net model into several sub-models; thereafter, the unconcerned sub-model can be abstracted to a single component in a high-level model. To decompose a Petri net model, the first step is to find its potential inner structures. According to our literature review [6], process algebra models benefit from the compositional feature, which can be numerically represented by an activity matrix that is composed of a set of transition vectors that correspond to each activity in the model. We are inspired by the activity matrix of process algebra and analyse the potential compositional structures from the ''incidence matrix'' of a given Petri net model. A Petri net's incidence matrix generally represents the relationships between places and transitions.

This paper aims to enhance the compositionality of Petri net models by applying a proposed sorting algorithm that implements a traversal sorting process on the incidence matrix of a Petri net model. The algorithm will finally identify all places that are involved in the same underlying structure of a Petri net model. Each structure can be considered a sub-model. The general Petri net model, which represents the whole system, is a union of these sub-models combined with the concurrent transitions. Thus, a large-scale Petri net model can be decomposed into sub-models that corresponds to each compositional structure identified by the algorithm.

Section 1 briefly introduces the concept of ''compositionality'', which is a core feature of mathematical modelling language, and the reason for its importance in the modelling of large-scale networks or systems based on Petri nets. Section 2 discusses the related work and our main contributions. Section 3 presents the background of Petri-nets and their compositional features. Section 4 defines the proposed compositional analysis algorithm. Section 5 applies the algorithm to a traffic network scenario to facilitate route planning. Section 6 concludes the research and discusses future work.

II. RELATED WORK

In today's computing industry, the scale of systems gives rise to many new challenges: they will be used by different stakeholders across multiple organizations, which may have conflicting needs and purposes; they will be constructed with complex dependencies and emergent properties; and they will be evolving continuously. These systems are called as Ultra-Large-Scale Systems (ULSSs). Hence, new improvements of modelling techniques (e.g., Petri nets) are required to facilitate the design of these large-scale systems. The Petri net, as a typical mathematical formalism, has been widely applied in many fields, such as performance modelling and evaluation of concurrent systems, communication networks, and even intelligent transportation systems. However, with the increased system scale, Petri nets have a slight deficiency in representing complex system structures and interactions. Some researchers have begun investigating these issues.

Blake and Trivedi [7] developed hierarchical composition by generating a two-level hierarchical model. In Blake's solution, each subsystem is modelled by a Markov chain and the system reliability is represented by a series of independent Markov components. Based on this result, Petri nets have been used to define hierarchical Markov models due to their features in locality and structuration. Other researchers have also made efforts to improve in the scalability of Petri nets. For example, Martin *et al.* [8] derives a new product-form solution for Stochastic Petri nets (SPNs) with signals using the application of RACT. Ma *et al.* [9] defines a specification called OR-AND Generalized Mutual Exclusion Constraints (GMEC) and applies it to a framework of supervisory control using Petri nets to realize the disjunction of conjunctive GMECs. Moreover, in Hayden's [10] research, due to the condition of weak compatibility in Inter-organizational workflow nets (IWF-nets), a related algorithm is investigated to take advantage of practical applications of IWF-nets, as in synchronous/asynchronous communication systems and process interactive systems. In [11], the researchers propose a new method to solve sparse systems of linear algebraic equations via sequential composition of their clans based on a weight graph and gain an extra computational speed-up. To overcome the complexity of analysing and controlling a Hybrid Petri Net (HPN) with large dimensions, Goldar *et al.* [12] decomposes the HPN and adopts an algorithm for the hierarchical control of subnets using a coordinator. By using the hierarchical control, the overall optimization problem (HPN) is divided into several optimization problems (subnets) that can be solved in parallel.

For large-scale system modelling, decomposition is another typical modelling technique in Petri nets, in which a general model is decomposed into several sub-models, which are then analysed in parallel [13]–[15]. Our research has a main goal of performing a compositional analysis on a Petri net model to obtain its potential compositional structures and thereby facilitate model decomposition. Hence, a sorting algorithm based on the incidence matrix of a Petri net model is designed to group places within the same structure in terms of the concurrent transitions in the model. This algorithm can enhance the Petri net's ability to model large-scale systems.

Compositionality is a core feature in stochastic process algebra (SPA), such as Performance Evaluation Process Algebra (PEPA), which is a compact symbolic representation of stochastic process algebra based on the underlying continuous-time Markov chain. In contrast to Petri nets, PEPA provides a compositional feature for modelling systems with complex interactions between components. Chen and Wang [16], [17] applies PEPA to the construction of complex vehicular network models, in which the compositionality of PEPA is represented in modelling cooperative system activities and subsystems. In our preceding work, we performed some pre-research in exploring the relationship between Petri nets and PEPA. From the pre-research [18], [19], we found some special types of Petri net models that can be converted into PEPA models by sorting their incidence matrices with the initial labels. Furthermore, a large scale Petri net includes a series of simple sub-nets through concurrent transitions, and a given Petri net can be converted into a corresponding incidence matrix that is used to explore the compositional structure. It is proved that such algebraic conversion and numerical method can facilitate the sorting process [19].

According to our recent literature review, some other researchers (such as Zhou *et al.* [20]) have obtained the decomposed structure of a fuzzy Petri net and made it more adaptive in engineering applications. Their work introduces an algorithm that uses the index function and incidence matrix to decompose a Petri net model into various completed inference paths, such as a sub-fuzzy Petri net model. Compared with Zhou's research, this paper proposes a more complete sorting process by defining a novel algorithm for sorting concurrent transitions in the incidence matrix of a Petri net model to obtain all potential compositional structures. The main contribution is this sorting algorithm, which is designed to decompose a complex Petri net model by sorting its incidence matrix rather than generating duplicate places or inference paths in contrast to the algorithm proposed in [20].

In addition to Petri nets, compositional modelling has been investigated earlier by Hillston [21]. Hillston proposed a novel compositional modelling language that has a key feature in compositionality. In [21], the compositional formal modelling approach is introduced by specifying various components in model definitions. Furthermore, Wang *et al.* [22] explores a compositional modelling and verification framework by extending Hybrid CSP that is process algebra-like formal modelling language; Atefnoaei *et al.* [23] uses a compositional verification rule for the issue of increased complexity in modelling systems and success chance of model-checking these systems; and Fu and Shen [24] presents a an innovative framework of fuzzy compositional modelling (FCM) by considering the processing of knowledge and data with uncertain environment, and the proposed FCM approach is able to represent and reason various inexact data and information.

The main contributions can be highlighted as: First, we first propose the idea of exploring compositional architectures of a Petri-net model through its corresponding incidence matrix, which has capability of enhancing the compositionality of Petri-net-based modelling approach; Second, we propose a sorting algorithm to classify places from an unordered incidence matrix to obtain each component of the corresponding Petri-net model.

III. PETRI-NETS AND COMPOSITIONAL STRUCTURES

This section presents the formal definition of Petri nets (Definition 1) and several core concepts, which will be used for exploring the compositional structure of Petri nets (Definitions 2, 3 and 4). For further illustration, we introduce different types of Petri-net models and the notion of compositional structures of Petri-net models.

A. INTRODUCTION TO PETRI NET

Definition 1 (Petri Net): A classic Petri net is composed of a 5-tuple $PN = (P, T, I^-, I^+, M_0)$, namely place P, transition T, input function I^-, output function I^- and token M_0. The definition of a Petri net can be given:

- $P = \{p_1, \cdots, p_n\}$ is a finite and non-empty set of places;
- $T = \{t_1, \cdots, t_m\}$ is a finite and non-empty set of transitions;
- $P \bigcap T = \emptyset$;
- $I^-, I^+ : P \times T \longrightarrow N_0$ are the backward and forward incidence functions, respectively;
- $M_0 : P \longrightarrow N_0$ is the initial marking.

Definition 2 (Pre Set): We use pre(t) represent a pre-set of transition t (input places of transition t), if

$$pre(t) = \{p \in P \mid I^-(p, t) > 0\}. \tag{1}$$

Definition 3 (Post Set): We use pre(t) represent a post-set of transition t (output places of transition t), if

$$post(t) = \{p \in P \mid I^+(p, t) > 0\}. \tag{2}$$

Definition 4 (Incidence Matrix): For a Petri net $PN = (P, T, I^-, I^+, M_0)$. Then, the backward incidence matrix $C^- = (c_{ij}{}^-) \in N_0{}^{\#P \times \#T}$ is defined by

$$c_{ij}{}^- = I^-(p_i, t_j), \quad \forall p_i \in P,\ t_j \in T, \tag{3}$$

the forward incidence matrix $C^+ = (c_{ij}{}^+) \in N_0{}^{\#P \times \#T}$ is defined by

$$c_{ij}{}^+ = I^+(p_i, t_j), \quad \forall p_i \in P,\ t_j \in T, \tag{4}$$

and the incidence matrix of PN is defined as

$$C = C^+ - C^-. \tag{5}$$

From the definition, it is clear that the structure of Petri nets can be represented by a corresponding incidence matrix $C_{\#P \times \#T} = [C_{ij}]$ accurately, in which $i \in [1, \#P], j \in [1, \#T]$ ($\#P$ denotes the number of places, $\#T$ denotes the number of transitions). In the matrix, each row represents a place, and each column represents a transition. For instance, the row i is a place that is named P_i, and the column j represents a transition t_j. Each matrix element C_{ij} represents whether a link exists between the place P_i and the transition t_j. Moreover, C_{ij} also represents the link direction if it exists. More details of incidence matrix are introduced in [2] and [7].

B. PETRI-NET SYNTAX

According to the investigation, several classification methods are used to analyse structures of Petri-net models, including state machines, marked graphs, FC-nets, EFC-nets, SPL-nets and ESPL-nets. These methods aim to analyse the potential structures of a Petri-net model and classify Petri nets into three main types, which are defined as follows:

1) The structure of the Petri net is closed (Fig. 1.a).

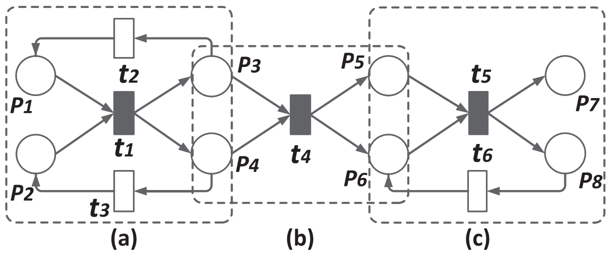


FIGURE 1. Three types of petri nets.

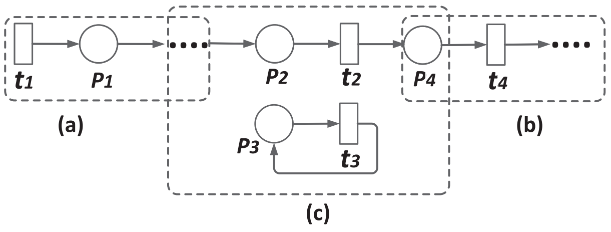


FIGURE 2. Detailed schemes in each type of petri-net.

2) The structure of the Petri net is non-closed (Fig. 1.b).
3) The structure of the Petri net is closed mix with non-closed (Fig. 1.c).

Each of the preceding three types can be further classified by considering different situations of the places and transitions. Thus, each type is divided into three specific situations:

1) Existing a transition that is lack of input place (Fig. 2.a).
2) Existing a place that is lack of input transition (Fig. 2.b).
3) Existing a place returning to itself through a transition but no arc between the place/transition or any other places/transitions (Fig. 2.c).

C. COMPOSITIONAL STRUCTURES

For large-scale system modelling, Petri net models always have a complex structure due to the concurrent transitions. As a result, it is difficult to analyse the compositional structure of a Petri-net model. Thus, this research aims to explore all potential compositional structures in a given Petri-net model through a new sorting algorithm. The algorithm uses an incidence matrix that is generated from a Petri-net model. The matrix, which represents all places and transitions of a Petri-net model, is sorted by the algorithm to find all potential structures.

A complex Petri-net model can be considered as a set of sub-models that are combined through concurrent transitions, such that for each sub-model, there is no further concurrent transition. In contrast to other compositional analysis methods, this algorithm is able to find all possible compositional structures by sorting the concurrent transitions of the incidence matrix. To illustrate the algorithm, a Petri-net model (Fig. 3) and its corresponding incidence matrix (Table 1) will be used as an example.

From Table 1, it is clear that transition t_1 is a concurrent transition as it has two input places and two output places.

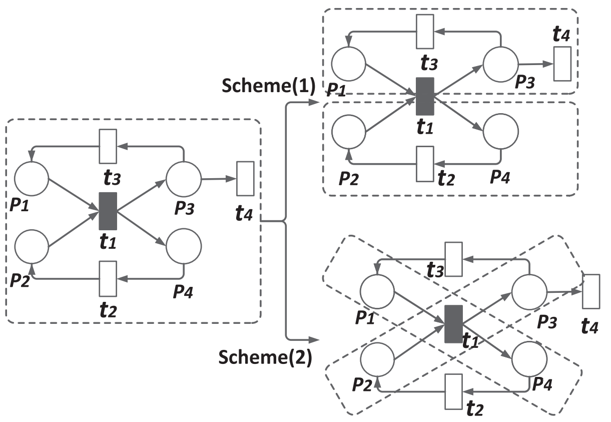


FIGURE 3. The corresponding Petri net of Table 1 and its two possible compositional structure.

TABLE 1. Incidence matrix of a petri-net model.

$P \backslash T$	t_1	t_2	t_3	t_4
P_1	-1	0	1	0
P_2	-1	1	0	0
P_3	1	0	-1	-1
P_4	1	-1	0	0

TABLE 2. Sub-incidence matrix of scheme 1.

$P \backslash T$	t_1	t_3	t_4	$P \backslash T$	t_1	t_2
P_1	-1	1	0	P_2	-1	1
P_3	1	-1	-1	P_4	1	-1

TABLE 3. Sub-incidence matrix of scheme 2.

$P \backslash T$	t_1	t_2	t_3	$P \backslash T$	t_1	t_2	t_3	t_4
P_1	-1	0	1	P_2	-1	1	0	0
P_4	1	-1	0	P_3	1	0	-1	-1

Thus, it is obvious that the incidence matrix shown in Table 1 can be considered a combined model through the two sub-models that are represented with two separated incidence matrices by the concurrent transition t_1; see Table 2. Thereafter, all places $\{P_1, P_2, P_3, P_4\}$ can be classified into two different schemes: one scheme includes $\{P_1, P_3\}$ and $\{P_2, P_4\}$ as shown in Table 2; the other scheme includes $\{P_1, P_4\}$ and $\{P_2, P_3\}$ as shown in Table 3.

The two schemes indicate all possible compositional structures of the Petri-net model represented by the incidence matrix in Table 1. Hence, the corresponding Petri-net models for the two schemes can be represented as two different architectures, which are shown in the dotted area of Fig. 3. The final goal is to find all possible compositional schemes with an analysis algorithm; these schemes are displayed in Fig. 3.

IV. COMPOSITIONAL ANALYSIS ALGORITHM

Compositional analysis is conducted by a sorting algorithm. The algorithm will first traverse transitions of the incidence matrix obtained from a Petri net model to find all potential closed-and-independent subsystems and remove them from

TABLE 4. Notations defined in Algorithms 1-3.

Variables	Explanation
C	Represents the incidence matrix.
$\#P$; $\#T$	Represent the number of rows (places) and columns (transitions) of C respectively.
P_D	A set and stores places those constitute independent closed loop subsystems.
C_{simp}	Represents the simplified matrix of C after removing rows and columns corresponding to independent closed loop subsystems from C.
$\#P_{simp}$; $\#T_{simp}$	Represent the number of rows and columns of C_{simp} respectively.
I_{Act}	A set and stores all triples $(-1, t_i, pre(t_i))$ where t_i is an individual transition.
C_{Act}	A set and stores all triples $(-1, t_i, pre(t_i))$ where t_i is a concurrent transition.
S_{post}	A set and stores all triples $(1, t_i, post(t_i))$ of all transitions.
$\#I_{Act}$; $\#C_{Act}$; $\#S_{post}$	Represent the number of subsets of I_Act, C_{Act} and S_{post} respectively.
CS_{temp}	A set and stores the results of the operation cartesian product of C_{Act} and S_{post}.
IS_{temp}	A set and stores the results of the operation cartesian product of I_{Act} and S_{post}.
CS_{differ}	A set and stores different places of CS_{temp}.
P_{post}	A set and stores the post-places those do not satisfy the condition end to end.
$CS_{subDiffer}$	A set and stores places which does not belong to CS_{differ}.
I_{subAct}	A set and stores subsets $I_{Act}(i)$ which meet the condition: the place $P(k) \in CS_{subDiffer}(j)$ and $P(k) \in I_{Act}(i)$.
$S_{subPost}$	A set and stores subsets $S_{post}(i)$ which meet the condition: the place $P(k) \in CS_{subDiffer}(j)$ and $P(k) \in S_{post}(i)$.
P_{Rest}	A set and stores the all final results of sub compositional structures of the system model.

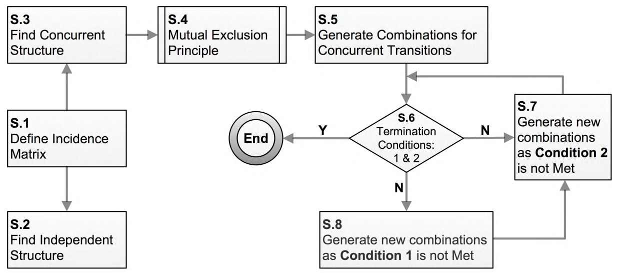

S.3
Find Concurrent Structure
S.4
Mutual Exclusion Principle
S.5
Generate Combinations for Concurrent Transitions
S.1
Define Incidence Matrix
End
Y
S.6
Termination Conditions: 1 & 2
N
S.7
Generate new combinations as Condition 2 is not Met
N
S.2
Find Independent Structure
S.8
Generate new combinations as Condition 1 is not Met

FIGURE 4. Framework of proposed sorting algorithm.

the matrix, then a simplified matrix is obtained in Stage 1. Thereafter, in Stage 2, the simplified matrix is traversed to find all independent or concurrent transitions and their corresponding *pre* and *post* sets. Finally in Stage 3, all compositional structures of a Petri net model will be given based on the results of Stage 1 and 2. Each stage is defined as an independent algorithm shown in Section 4.2. The detail sorting process is introduced in the following Section 4.1.

A. ALGORITHM DESCRIPTION

To examine the potential compositional structure from a PN model, a sorting algorithm is proposed in the section as well as its design framework. The logical architecture of the algorithm is demonstrated in Fig. 4.

The sorting algorithm is described as following steps:

S.1 Define the incidence matrix C of a given PN model; then specify rows (places) of C with names P_i, and columns (transitions) with names t_i, $i \in [1, 2, \ldots, n]$.

S.2 Find the independent structure that is represented as a closed loop in the PN model, when the condition $(C_{ij} = 0 \land C_{ik} = 0 \land C_{lj} = 0)$ is met.

S.3 Find the column j of C that includes concurrent transitions t_j under the condition (i.e., $C_{ij} = -1$ and the number of -1 in column j is over 1).

S.4 Define a mutual exclusion principle, which indicates that input places represented by -1 in the obtained column j of the incidence matrix cannot exist in a single structure. In other words, the places of $Pre(t_j)$ cannot be in the same structure when t_j is a concurrent transition.

S.5 Generate a Cartesian product between the *Pre* and *Post* set of the concurrent transitions t_j on the basis of S.4, and obtain a set of combinations in the form of $(Pre(t_j), Post(t_j))$. These combinations will be sorted under the conditions given in S.6.

S.6 Set two conditions for algorithm termination: 1) includes all rows (places) of C; 2) assemble all combinations $(Pre(t_j), Post(t_j))$ head-to-tail.

S.7 After the S.5, if both conditions of the S.6 are met, sorting is done; if only the condition 1) is met, S.7 needs be implemented by creating combinations for each place of the set $Post(t)$ corresponding to its related independent transition t_k. This operation continues until condition 2) of S.6 is met.

S.8 After the S.5, if the condition 1) is still not met except the situation in S.2, new combinations must be generated by combining the excluded rows (places) with its connected transitions until all rows (places) are involved in the combinations. Thereafter, S.6 will be implemented if the condition 2) is not met.

With the sorting algorithm, compositional analysis of a given PN model (e.g., Fig. 3) can be conducted by sorting out the two structures shown in Fig. 3. The algorithm can extend QoS modelling capability of PN formalism by analyzing compositional structures based the incidence matrix. For example, the compositional approach can be applied for modelling of a public traffic network in order to analyze all possible bus lines based on the incidence matrix derived from the PN model.

B. FORMAL DESCRIPTION OF THE ALGORITHM

According to the algorithm description, this section formally defines the proposed algorithm. It the formal definition, the whole process is defined in terms of three algorithms, which are introduced in the following sections. Table 4 gives all notations used in the algorithms. Before defining the algorithm, a set of concepts that are used in the algorithm must be predefined.

Definition 5 (Cartesian Product, Permutation and Combination): Let $CPPC(L)$ be a set of combinations between sets $pre(l)$ and $post(l)$. The combination of $pre(l)$ and $post(l)$ is denoted by $CPPC(L)$, $L = (pre(l),\ post(l))$, which has a mathematical form:

$$\begin{aligned}
&CPPC(L)\\
&\ = \{\{(pre(l)[1], post(l)[1]),\ \cdots\ (pre(l)[i], post(l)[j]),\ \cdots\}\\
&\qquad\qquad \cdots\\
&\quad \{(pre(l)[1], post(l)[j]),\ \cdots\ (pre(l)[i], post(l)[1]),\ \cdots\}\\
&\quad \cdots\}
\end{aligned}$$

where,

- $pre(l)[1] \cup \cdots \cup pre(l)[i] \cup \cdots = pre(l)$.
- $post(l)[1] \cup \cdots \cup post(l)[i] \cup \cdots = post(l)$.
- If $i \neq j$ and $i, j \in [1, \#pre(l)]$, then $pre(l)[i] \neq pre(l)[j]$.
- If $i \neq j$ and $i, j \in [1, \#post(l)]$, then $post(l)[i] \neq post(l)[j]$.

To explain the definition details, an example is applied by assuming $pre(l) = \{P_1, P_2, P_3\}$ and $post(l) = \{P_4, P_5, P_6\}$, such as $L = (pre(l),\ post(l))$. Thus, we have

$$CPPC(L) = \left\{\begin{array}{lll}
\{(P_1, P_4) & (P_2, P_5) & (P_3, P_6)\},\\
\{(P_1, P_4) & (P_2, P_6) & (P_3, P_5)\},\\
\{(P_1, P_5) & (P_2, P_4) & (P_3, P_6)\},\\
\{(P_1, P_5) & (P_2, P_6) & (P_3, P_4)\},\\
\{(P_1, P_6) & (P_2, P_4) & (P_3, P_5)\},\\
\{(P_1, P_6) & (P_2, P_5) & (P_3, P_4)\}.
\end{array}\right\}$$

Definition 6 (End to End): For several triples: (t_i, P_a, P_b), (t_j, P_b, P_c), (t_m, P_d, P_a), (t_n, P_e, P_f), $\cdots$; If $pre(t_i) = post(t_m)$ (i.e. place P_a) and $post(t_i) = pre(t_j)$ (i.e. place P_b), then, triples (t_i, P_a, P_b), (t_j, P_b, P_c) and (t_m, P_d, P_a) meet the condition end-to-end. Meanwhile, we use the symbol $ETE(L)$ to represent such condition, where

$$L = \{(t_i, P_a, P_b), (t_j, P_b, P_c), (t_m, P_d, P_a)\}.$$

Definition 7 (Combine ETE): For a set L and $L = \{(t_i, P_a, P_b),(t_j, P_b, P_c),(t_m, P_d, P_a)\}$. If L meets the condition ETE, operations can be taken on L. The symbol $Comb_{ETE}(L)$ represents this operation, such as:

$$Comb_{ETE}(L) = (P_d, P_a, P_b, P_c).$$

1) ALGORITHM 1: SORT OUT INDEPENDENT CLOSED-LOOP SUB-SYSTEMS

Algorithm 1 includes two main functions: the first function traverses all elements of C and determines whether independent closed-loop subsystems exist in the given Petri-net model; thereafter, if they do exist, the second function separates the independent closed-loop subsystems from C. The algorithm is given as Algorithm 1.

Algorithm 1 Independent Close-Loop Sub-Systems Sorting Algorithm

```
Input: $C$, $\#P$, $\#T$
Output: $P_D$
1:  for each $i \in [1, \#P]$ and $j \in [1, \#T]$ do
2:      if $C_{ij} = 0$ then
3:          for each $k \in [1, \#P]$ and $l \in [1, \#T]$ do
4:              if $C_{kj} = 0$ and $C_{il} = 0$ then
5:                  Preliminary division:
6:                  $P_D = P_D \bigcup \{P_i\}$.
7:              end if
8:          end for
9:      end if
10: end for
```

2) ALGORITHM 2: CLASSIFY INCIDENCE MATRIX BY EXAMINING CONCURRENT TRANSITIONS

Algorithm 1 sorts out places and transitions that constitute independent closed-loop sub-systems, removes the corresponding rows and columns from C and obtains a simplified incidence matrix C_{simp}. Thereafter, Algorithm 2 aims to sort the matrix C_{simp} based on concurrent transitions, and ultimately includes individual transitions, concurrent transitions and their corresponding pre- and post-places into the sets C_{Act}, I_{Act} and S_{post}. Algorithm 2 can be specified as follows:

Algorithm 2 Concurrent Transition Classification Algorithm

```
Input: $C_{simp}$, $\#P_{simp}$, $\#T_{simp}$
Output: $C_{Act}$, $I_{Act}$, $S_{post}$
1:  for each $j \in [1,\ \#T_{simp}]$ do
2:      $count = 0$
3:      for $i$ from 1 to $\#P_{simp}$ do
4:          if $C_{simp}(ij) = -1$ then
5:              $count \longleftarrow count + 1$
6:              $I_{Act} = I_{Act} \bigcup \{(-1, t_j, P_i)\}$
7:          end if
8:          if $C_{simp}(ij) = 1$ then
9:              $S_{post} = S_{post} \bigcup \{(1, t_j, P_i)\}$
10:         end if
11:     end for
12:     if $count \geq 2$ then
13:         $C_{Act} = C_{Act} \bigcup \{(-1, t_j, \text{pre}(t_j))\}$, then
14:         $I_{Act} = I_{Act} \backslash \{(-1, t_j, \text{pre}(t_j))\}$
15:     end if
16: end for
```

3) ALGORITHM 3: OBTAIN ALL COMPOSITIONAL STRUCTURES FROM A PETRI-NET MODEL

Algorithm 3 obtains all possible compositional structures of a given Petri-net model by dealing with the sets obtained from Algorithm 2. The algorithm has the following steps:

- Steps 2-4 calculate Cartesian products and perform permutation and combination operations on sets C_{Act} and S_{post}; the results are stored in the set CS_{temp}.

Algorithm 3 Obtain All Compositional Structure

```
Input: $C_{Act}$, $I_{Act}$, $S_{post}$;   Output: $P_{Rest}$
1:  for $i \in [1, \#C_{Act}],\ j \in [1, \#S_{post}]$ do
2:      if $(t_i \in C_{Act}(i)) = (t_j \in S_{post}(j))$ then $CS_{temp} = CS_{temp} \bigcup CPPC(C_{Act}(i), S_{post}(j))$
3:      end if
4:      for $m \in [1, \#CS_{temp}],\ n \in [1, \#CS_{differ}]$ do
5:          if $CS_{temp}(m) \neq CS_{differ}(n)$ then $count \leftarrow count + 1$; $CS_{differ} = CS_{differ} \bigcup CS_{temp}(m)$
6:          end if
7:          if $count = \#P_{simp}$ then
8:              if $CS_{temp}(m)$ meets Definition 6 then $P_{Rest} = Rest \bigcup Comb_{ETE}(CS_{temp}(m))$
9:              end if
10:             if !($CS_{temp}(m)$ meets Definition 6) then Determine set of post places $P_{post}$
11:                 for $l \in [1, \#I_{Act}],\ k \in [1, \#S_{post}]$ do
12:                     Find $S_{post}(k)$, where $t_k \in S_{post}(k)$ & $t_k \in I_{Act}(l)$; Cal $CPPC(S_{post}(k), I_{Act}(l)) \rightarrow IS_{temp}$
13:                 end for
14:                 Cal $Comb_{ETE}(IS_{temp})$, and save in $CS_{temp}$
15:             end if and jump to step 13
16:         end if and jump to the end
17:         if $count \neq \#P_{simp}$ then
18:             for $k \in [1, \#C_{simp}]$ & $P_k \in C_{simp}$ & $P_k \notin CS_{differ}$ do Deposit $P_k$ into $CS_{subDiffer}$
19:             end for
20:             for $x \in [1, \#CS_{subDiffer}],\ y \in [1, \#I_{Act}]$ do
21:                 if $CS_{subDiffer}(x) \in I_{Act}(y)$ then Deposit $I_{act}(y)$ into $I_{subAct}$
22:                 end if
23:             end for
24:             for $n \in [1, \#I_{subAct}],\ l \in [1, \#S_{post}]$ do Find $S_{post}(l)$, where $(t_l \in I_{subAct}$ & $t_l \in S_{post}(l))$;
25:                 Do $CPPC(S_{post}(l), I_{subAct}(n))$
26:             end for
27:             for $n \in [1, \#CS_{subDiffer}],\ l \in [1, \#S_{post}]$ do
28:                 if $CS_{subDiffer}(n) \in S_{post}(l)$ then $S_{subPost} = S_{subPost} \bigcup \{S_{post}(l)\}$
29:                     for $x \in [1, \#S_{subPost}]$ & $y \in [1, \#I_{Act}]$ do Do $CPPC(S_{subPost}(x)$, and $I_{Act}(y))$,
30:                         where $t_y \in I_{Act}(y)$ & $t_y \in S_{subPost}(x)$
31:                     end for
32:                 end if
33:             end for
34:         end if and jump to step 12
35:     end for
36: end for
```

- Steps 5-7 find all non-repeated sub-sets of CS_{temp} and store them in the set CS_{differ}. Meanwhile, a variable *count* is used to record the total number of different places in the subsets of CS_{temp}.
- Steps 8-10 determine whether the sub-sets of CS_{temp} meet the end-to-end condition; if so, combine the sub-sets and store the results in P_{Rest}.
- Steps 12-14 perform the Cartesian product, permutation and combination operations on I_{Act} and S_{post}; if the sub-sets of I_{Act} and S_{post} contain the same transitions, the results are stored in CS_{temp}.
- Step 18 traverses C_{simp} to find all places that are included in C_{simp} but not in CS_{differ} and then stores these places in $CS_{subDiffer}$.
- Steps 27-33 traverse $CS_{subDiffer}$ and S_{post}: if the places in $CS_{subDiffer}$ are included in the sub-sets of S_{post}, then the corresponding sub-sets into $S_{subPost}$. Thereafter, $S_{subPost}$ and I_{Act} are traversed to find the sub-sets are stored in $S_{subPost}$ and I_{Act} that includes the same transitions. Finally, perform the Cartesian product, permutation and combination operations on these sub-sets.

In this section, the algorithm is formally defined as three sub-algorithms. To demonstrate the process defined in these algorithms, a real-world model scenario is adopted to verify the reliability of the proposed sorting algorithm and examine its usability for real-world applications.

Section 5 uses Petri-nets to define a traffic network of a local city in China. The sorting algorithm is used to examine the corresponding incidence matrix from the Petri-net-based traffic network model to sort out all potential compositional structures that represent traffic routes in the given traffic network.

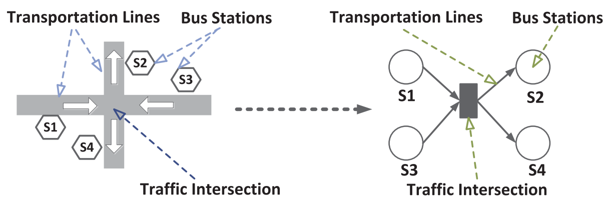


FIGURE 5. Modelling of lines, intersections and stops.

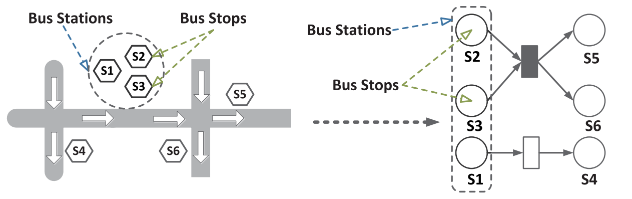


FIGURE 6. Modelling of bus stations with multi-stops.

V. ALGORITHM APPLICATION

In this section, a sample model of an actual urban traffic network is defined with Petri nets to evaluate the proposed algorithm. Based on the traffic model, the corresponding incidence matrix is derived from its Petri-net description according to Definition 4. Thereafter, Algorithms 1-3 are used to analyse the incidence matrix to decompose the overall Petri-net model to obtain its sub-structures by classifying the places that represent bus stops in the traffic network in the Petri-net model.

According to the analysis results, the algorithm is beneficial in the deployment of bus lines on such a traffic network. For example, the algorithm can be used to divide the whole network to find all potential bus lines; thereafter, further analysis can be conducted to optimize the classifications to achieve the bus line planning.

A. PETRI-NET MODELS AND INCIDENCE MATRIX

A real-world traffic network, seeing the left part of Fig. 7, is obtained to generate a target traffic model prototype. Subsequently, the prototype is simplified to the traffic network scenario that is depicted in the right part of Fig. 7. The traffic network is then defined as a Petri-net model, as shown in Fig. 8, in which the Petri-net notations are defined as follows:

1) Use places to represent the transportation stops.
2) Use transitions to represent the traffic intersections.
3) Use arcs to represent the transportation lines.

In addition, it is worth mentioning three key points in defining such a traffic network model:

1) Only one-way traffic lines are considered when building the traffic network model. Specifically, each line is regarded as a single route from the start station to the destination; the return trip is not considered. This is to avoid the duplicate behaviours that appear in models that simulate the return trip as a similar process to the onward trip.
2) A start or destination station will be considered as an individual and independent bus stop in the model if the station includes more than one stop (e.g., multiple stops at a start or a destination station); see Fig. 5.
3) All other bus stops outside the stations and traffic intersections can be modelled as corresponding relationships between bus stops; see Fig. 6.

As shown in Fig. 8, the Petri-net model is defined to represent the simplified traffic network prototype (the right part of Fig. 7); the corresponding incidence matrix is shown in Table 5.

According to Fig. 7 and Fig. 8, four bus lines are selected for experiments: L_1, L_2, L_3 and L_4. These lines include 27 bus stops (namely, 27 places in Fig. 8) and 16 intersections (namely, seven intersections and nine single crossroads) from the map of Fig. 7 (the right part). For the four selected bus lines in Fig. 8, the places L_{1_1}, L_{2_1}, L_{3_1} and L_{4_1} are the start bus stops, and the places L_{1_5}, L_{2_3}, L_{3_13} and L_{4_6} are the destination bus stops. Moreover, the series of directional arcs between the start and destination stops represent specific routes in the traffic network.

In Table 5, each row of the incidence matrix represents a place (namely, a bus stop) and the relationships between places can be specified as follows:

- $L_{1\_1} \rightarrow L_{1\_5}$ corresponds to $P_1 \rightarrow P_5$;
- $L_{2\_1} \rightarrow L_{2\_3}$ corresponds to $P_6 \rightarrow P_8$;
- $L_{3\_1} \rightarrow L_{3\_13}$ corresponds to $P_9 \rightarrow P_{21}$;
- $L_{4\_1} \rightarrow L_{4\_6}$ corresponds to $P_{22} \rightarrow P_{27}$.

In addition, the columns in Table 5 include 16 transitions (t_1 to t_{16}) that represent all of the intersections.

B. COMPOSITIONAL ANALYSIS ON A TRAFFIC NETWORK MODEL

1) ANALYSIS PRINCIPLES

In this subsection, the proposed algorithms will be used to manipulate the incidence matrix shown in Table 5 to find all possible compositional structures by classifying places in the Petri-net model. This compositional analysis aims to facilitate the planning of bus lines based on the given traffic network.

The analysis results will help maintain the deployment of bus lines. For example, the utilization of each bus line can be improved by determining the scheme for each line based on the compositional analysis. Moreover, our solution can improve the efficiency of the traffic network by considering the selection of different bus line schemes and measuring the traffic congestion.

In the analysis, if it is assumed that the distribution and the total number of bus stops are fixed, then the optimal utilization of bus lines can be guaranteed when the number of bus stops on each line follows the mean value. In this situation, the traffic load can be efficiently reduced due to the non-uniform distribution of bus lines (i.e., bus stops). Here, the mean value is set as:

$$\bar{N} = \lfloor S_{stops}/S_{lines} \rfloor, \tag{6}$$

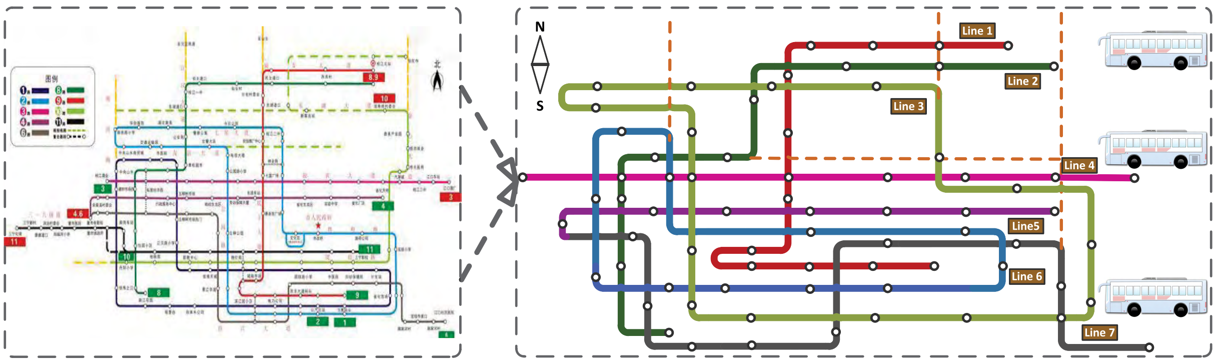


FIGURE 7. Real-world traffic network and the simplified traffic model prototype.

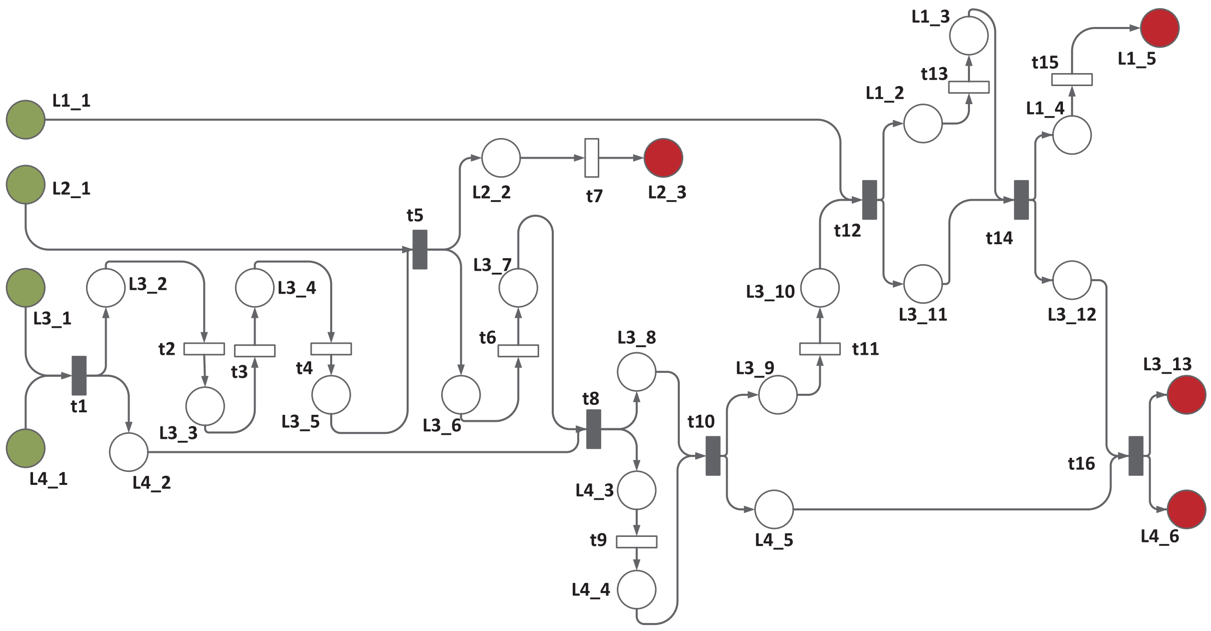


FIGURE 8. Petri-net based traffic network model.

where

- $\bar{N}$ represents the mean value;
- S_{stop} denotes the total number of bus stops;
- S_{lines} means the total number of bus lines.

2) ANALYSIS RESULTS

Based on the incidence matrix shown in Table 5, all classified bus lines are obtained using the proposed algorithms and listed in Table 6. Furthermore, based on the properties of the Cartesian product, permutation and combination operations, a formula is defined for calculating the upper bound of the classification results, which is defined as:

$$UB = \prod (N(pre(t_i))!), \tag{7}$$

where

- UB represents the upper bound;
- $t_i \in T$ and t_i is a concurrent transition;
- $N_{pre(t_i)}$ represent the number of places in pre(t_i).

According to the algorithms and the corresponding incidence matrix (Table 5), it is easy to get the sets: I_{Act}, S_{post} and C_{Act}. Here, just two concurrent transitions are given as well as its related corresponding operation $CPPC$ (i.e. the sub-sets of $CPPC(C_{Act})$):

$C_{Act} = \{(-1, t_5, pre(t_5)), \cdots, (-1, t_{12}, pre(t_{12})), \cdots\}$.

Thereafter, the sets $CPPC(t_1)$ and $CPPC(t_5)$ can be obtained from the follows:

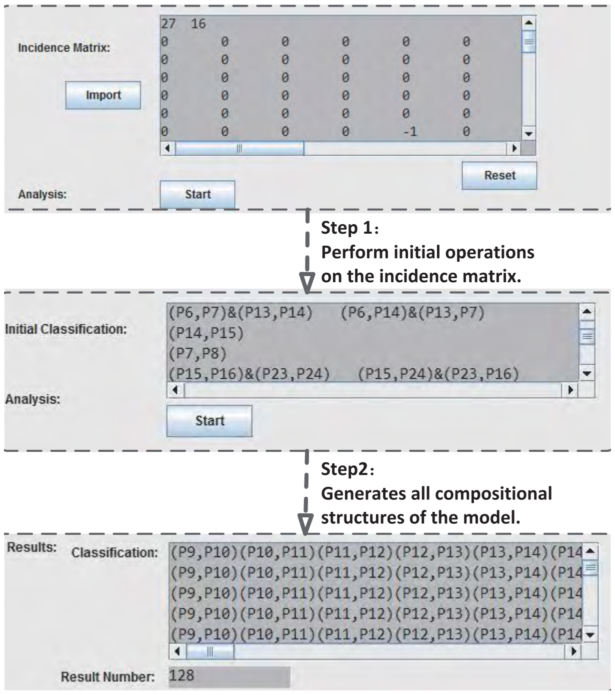
$$CPPC(t_5) = \left\{ \begin{array}{l} \{(P_6, P_7),\ (P_{13}, p_{14})\}, \\ \{(P_6, P_{14}),\ (P_{13}, P_7)\}. \end{array} \right\}$$

$$CPPC(t_5) = \left\{ \begin{array}{l} \{(P_1, P_2),\ (P_{18}, p_{19})\}, \\ \{(P_1, P_{19}),\ (P_{18}, P_2)\}. \end{array} \right\}$$

To facilitate measurements, an analysis platform is applied to implement to classification process. Fig. 9 demonstrates the basic steps for the classification of the incidence matrix on the analysing platform. Step 1 conducts initial operations on the incidence matrix to obtain the sets(I_Act, C_Act,

TABLE 5. Incidence matrix of the petri-net model defined in Fig. 8.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}
P_1	0	0	0	0	0	0	0	0	0	0	0	−1	0	0	0	0
P_2	0	0	0	0	0	0	0	0	0	0	0	1	−1	0	0	0
P_3	0	0	0	0	0	0	0	0	0	0	0	0	1	−1	0	0
P_4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	−1	0
P_5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
P_6	0	0	0	0	−1	0	0	0	0	0	0	0	0	0	0	0
P_7	0	0	0	0	1	0	−1	0	0	0	0	0	0	0	0	0
P_8	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
P_9	−1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P_{10}	1	−1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P_{11}	0	1	−1	0	0	0	0	0	0	0	0	0	0	0	0	0
P_{12}	0	0	1	−1	0	0	0	0	0	0	0	0	0	0	0	0
P_{13}	0	0	0	1	−1	0	0	0	0	0	0	0	0	0	0	0
P_{14}	0	0	0	0	1	−1	0	0	0	0	0	0	0	0	0	0
P_{15}	0	0	0	0	0	1	0	−1	0	0	0	0	0	0	0	0
P_{16}	0	0	0	0	0	0	0	1	0	−1	0	0	0	0	0	0
P_{17}	0	0	0	0	0	0	0	0	0	1	−1	0	0	0	0	0
P_{18}	0	0	0	0	0	0	0	0	0	0	1	−1	0	0	0	0
P_{19}	0	0	0	0	0	0	0	0	0	0	0	1	0	−1	0	0
P_{20}	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	−1
P_{21}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
P_{22}	−1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P_{23}	1	0	0	0	0	0	0	−1	0	0	0	0	0	0	0	0
P_{24}	0	0	0	0	0	0	0	1	−1	0	0	0	0	0	0	0
P_{25}	0	0	0	0	0	0	0	0	1	−1	0	0	0	0	0	0
P_{26}	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	−1
P_{27}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1



```
27  16
0    0    0    0    0    0
0    0    0    0    0    0
0    0    0    0    0    0
0    0    0    0    0    0
0    0    0    0    0    0
0    0    0    0   -1    0
```

```
(P6,P7)&(P13,P14)   (P6,P14)&(P13,P7)
(P14,P15)
(P7,P8)
(P15,P16)&(P23,P24)   (P15,P24)&(P23,P16)
```

```
(P9,P10)(P10,P11)(P11,P12)(P12,P13)(P13,P14)(P14
(P9,P10)(P10,P11)(P11,P12)(P12,P13)(P13,P14)(P14
(P9,P10)(P10,P11)(P11,P12)(P12,P13)(P13,P14)(P14
(P9,P10)(P10,P11)(P11,P12)(P12,P13)(P13,P14)(P14
(P9,P10)(P10,P11)(P11,P12)(P12,P13)(P13,P14)(P14
```

```
128
```

FIGURE 9. Specific operations of the algorithms achieved through programming and the final classification results.

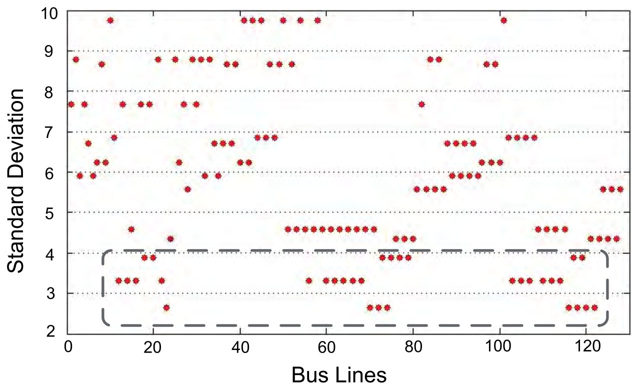


FIGURE 10. Initial results from formulas 6-8 and optimized results.

S_post), and then generates the final results from *CPPC*. Step 2 displays all sets obtained from the *CPPC* operation, which include the final 128 possible structures of the model.

According to formula (7), it can be concluded that the upper bound on the number of classification results is 128. This value is consistent with the experimental results from the analyser. Furthermore, the classification results have been optimized to find the best bus line deployment. To complete the optimization, the value of $\bar{N}$ should be set to 6 based on formula (6). Thus, the optimized results shown in Fig. 10 can be generated with formula (8).

$$S = \sqrt{\sum (N_i - \bar{N})^2}, \tag{8}$$

where

- N_i represents the number of bus stops in each line;
- S represents the corresponding standard deviation.

Fig. 10 shows the plot of bus stops on each line that was generated in the preceding analysis. This figure describes the average number of bus stops $\bar{N}$ (vertical axis) on each bus line

TABLE 6. Optimized qualifying compositional structures of the application in Fig. 8.

	Line1	Line2	Line3	Line4
Scheme 1	P: {9,10,11,12,13,7,8}	P: {6,14,15,16,17,18,19,20,21}	P: {22,23,24,25,26,27}	P: {1,2,3,4,5}
Scheme 2	P: {9,10,11,12,13,7,8}	P: {6,14,15,16,17,18,19,4,5}	P: {22,23,24,25,26,27}	P: {1,2,3,20,21}
Scheme 3	P: {22,10,11,12,13,7,8}	P: {9,23,24,25,17,18,19,4,5}	P: {6,14,15,16,26,27}	P: {1,2,3,20,21}
Scheme 4	P: {22,10,11,12,13,7,8}	P: {9,23,16,17,18,2,3,20,27}	P: {6,14,15,24,25,26,21}	P: {1,19,4,5}
Scheme 5	P: {22,10,11,12,13,7,8}	P: {9,23,16,17,18,2,3,4,5}	P: {6,14,15,24,25,26,21}	P: {1,19,20,27}
Scheme 6	P: {9,10,11,12,13,7,8}	P: {22,23,24,25,17,18,19,4,5}	P: {6,14,15,16,26,27}	P: {1,2,3,20,21}
Scheme 7	P: {9,10,11,12,13,7,8}	P: {22,23,16,17,18,19,4,5}	P: {6,14,15,24,25,26,27}	P: {1,2,3,20,21}
Scheme 8	P: {9,10,11,12,13,7,8}	P: {6,14,15,16,17,18,19,20,27}	P: {22,23,24,25,26,21}	P: {1,2,3,4,5}
Scheme 9	P: {9,10,11,12,13,7,8}	P: {6,14,15,16,17,18,19,4,5}	P: {22,23,24,25,26,21}	P: {1,2,3,20,27}
Scheme 10	P: {9,10,11,12,13,7,8}	P: {22,23,24,25,17,18,19,20,21}	P: {6,14,15,16,26,27}	P: {1,2,3,4,5}
Scheme 11	P: {9,10,11,12,13,7,8}	P: {22,23,24,25,17,18,19,20,27}	P: {6,14,15,16,26,21}	P: {1,2,3,4,5}
Scheme 12	P: {9,10,11,12,13,7,8}	P: {22,23,24,25,17,18,19,4,5}	P: {6,14,15,16,26,21}	P: {1,2,3,20,27}
Scheme 13	P: {9,10,11,12,13,7,8}	P: {22,23,16,17,18,19,20,21}	P: {6,14,15,24,25,26,27}	P: {1,2,3,P,5}
Scheme 14	P: {9,10,11,12,13,7,8}	P: {22,23,16,17,18,19,20,27}	P: {6,14,15,24,25,26,21}	P: {1,2,3,4,5}
Scheme 15	P: {9,10,11,12,13,7,8}	P: {22,23,16,17,18,19,4,5}	P: {6,14,15,24,25,26,21}	P: {1,2,3,20,27}
Scheme 16	P: {9,10,11,12,13,7,8}	P: {22,23,16,17,18,2,3,4,5}	P: {6,14,15,24,25,26,27}	P: {1,19,20,21}
Scheme 17	P: {9,10,11,12,13,7,8}	P: {22,23,16,17,18,2,3,4,5}	P: {6,14,15,24,25,26,21}	P: {1,19,20,27}
Scheme 18	P: {9,10,11,12,13,7,8}	P: {22,23,16,17,18,2,3,20,21}	P: {6,14,15,24,25,26,27}	P: {1,19,4,5}
Scheme 19	P: {9,10,11,12,13,7,8}	P: {22,23,16,17,18,2,3,20,27}	P: {6,14,15,24,25,26,21}	P: {1,19,4,5}
Scheme 20	P: {9,23,24,25,26,27}	P: {6,14,15,16,17,18,19,20,21}	P: {22,10,11,12,13,7,8}	P: {1,2,3,4,5}
Scheme 21	P: {9,23,24,25,26,21}	P: {6,14,15,16,17,18,19,20,27}	P: {22,10,11,12,13,7,8}	P: {1,2,3,4,5}
Scheme 22	P: {9,23,24,25,26,27}	P: {6,14,15,16,17,18,19,4,5}	P: {22,10,11,12,13,7,8}	P: {1,2,3,20,21}
Scheme 23	P: {9,23,24,25,26,21}	P: {6,14,15,16,17,18,19,4,5}	P: {22,10,11,12,13,7,8}	P: {1,2,3,20,27}
Scheme 24	P: {22,10,11,12,13,7,8}	P: {9,23,24,25,17,18,19,20,21}	P: {6,14,15,16,26,27}	P: {1,2,3,4,5}
Scheme 25	P: {22,10,11,12,13,7,8}	P: {9,23,24,25,17,18,19,20,27}	P: {6,14,15,16,26,21}	P: {1,2,3,4,5}
Scheme 26	P: {22,10,11,12,13,7,8}	P: {9,23,24,25,17,18,19,4,5}	P: {6,14,15,16,26,21}	P: {1,2,3,20,27}
Scheme 27	P: {22,10,11,12,13,7,8}	P: {9,23,16,17,18,19,20,21}	P: {6,14,15,24,25,26,27}	P: {1,2,3,4,5}
Scheme 28	P: {22,10,11,12,13,7,8}	P: {9,23,16,17,18,19,20,27}	P: {6,14,15,24,25,26,21}	P: {1,2,3,4,5}
Scheme 29	P: {9,23,16,17,18,19,4,5}	P: {22,10,11,12,13,7,8}	P: {6,14,15,24,25,26,27}	P: {1,2,3,20,21}
Scheme 30	P: {9,23,16,17,18,19,4,5}	P: {22,10,11,12,13,7,8}	P: {6,14,15,24,25,26,21}	P: {1,2,3,20,27}
Scheme 31	P: {22,10,11,12,13,7,8}	P: {9,23,16,17,18,2,3,4,5}	P: {6,14,15,24,25,26,27}	P: {1,19,20,21}
Scheme 32	P: {22,10,11,12,13,7,8}	P: {9,23,16,17,18,2,3,20,21}	P: {6,14,15,24,25,26,27}	P: {1,19,4,5}

- $\bar{N}$ represents the mean value;
- S_{stop} denotes the total number of bus stops;
- S_{lines} means the total number of bus lines.

(horizontal axis), and the standard deviation of the number of bus stops. Hence, the optimal bus stop distribution has the smallest standard deviation. The specific bus line deployment and the included bus stops can be found in Table 6.

VI. CONCLUSION

Compositionality is a key feature that is required by today's large-scale modelling tasks. However, not all popular modelling techniques have this feature. This research aims to explore this feature of a popular modelling technique – Petri nets by proposing a sorting algorithm. The algorithm can sort all possible compositional structures of a Petri-net model by analysing its corresponding incidence matrix, which is inspired by a feature of stochastic process algebra; it also enhances the compositional analysis ability of Petri nets, especially for large-scale networks and concurrent systems. To demonstrate the use of the algorithms, a traffic network that is modelled with Petri nets is used for experiments on planning bus lines by sorting all possible bus lines between the given bus stops. The experimental results clearly demonstrate the usability of the proposed algorithms, as they classify all potential lines and generate the optimal lines by specifying the selected bus stops for each line. Thus, the compositional analysis improves the efficiency of the traffic system and reduces the traffic load, while also bridging the gap between Petri nets and stochastic process algebras.

In future work, further research will be conducted to improve the efficiency of the sorting process, and restore the model structures of general Petri-net models from their corresponding incidence matrices to achieve a conversion between Petri nets and process algebras.

REFERENCES

[1] M. Silva, "Half a century after Carl Adam Petri's Ph.D. thesis: A perspective on the field," *Annu. Rev. Control*, vol. 37, no. 2, pp. 191–219, 2013.

[2] C. R. Vazquez and M. Silva, "Stochastic continuous Petri nets: An approximation of Markovian net models," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 42, no. 3, pp. 641–653, May 2012.

[3] G. Liu, M. Zhou, and C. Jiang, "Petri net models and collaborativeness for parallel processes with resource sharing and message passing," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 4, 2017, Art. no. 113, doi: 10.1145/2810001.

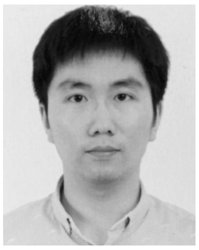
[4] P. Wang, L. Ma, R. M. P. Goverde, and Q. Wang, "Rescheduling trains using Petri nets and heuristic search," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 3, pp. 726–735, Mar. 2017.

[5] J. Huang, Y. Zhu, B. Cheng, C. Lin, and J. Chen, "A PetriNet-based approach for supporting traceability in cyber-physical manufacturing systems," *Sensors*, vol. 16, no. 3, p. 382, 2016.

[6] J. Ding and J. Hillston, "Numerically representing stochastic process algebra models," *Comput. J.*, vol. 55, no. 11, pp. 1383–1397, 2012.
[7] J. T. Blake and K. S. Trivedi, "Reliability analysis of interconnection networks using hierarchical composition," *IEEE Trans. Rel.*, vol. 38, no. 1, pp. 111–120, Apr. 1989.
[8] A. Marin, S. Balsamo, and P. G. Harrison, "Analysis of stochastic Petri nets with signals," *Perform. Eval.*, vol. 69, no. 11, pp. 551–572, 2012.
[9] Z. Ma, Z. Li, and A. Giua, "Design of optimal Petri net controllers for disjunctive generalized mutual exclusion constraints," *IEEE Trans. Autom. Control*, vol. 60, no. 7, pp. 1774–1785, Jul. 2015.
[10] R. A. Hayden, "Scalable performance analysis of massively parallel stochastic systems," Ph.D. dissertation, Dept. Comput. Imperial College London, London, U.K., 2011.
[11] D. Zaitsev, "Sequential composition of linear systems' clans," *Inf. Sci.*, vol. 363, pp. 292–307, Oct. 2016.
[12] S. N. Goldar, A. Doustmohammadi, and S. B. Ajabshir, "Decomposition of first-order hybrid Petri nets for hierarchical control of manufacturing systems," in *Proc. ICCIA*, Dec. 2013, pp. 311–316.
[13] J. Ye, M. Zhou, Z. Li, and A. Al-Ahmari, "Structural decomposition and decentralized control of Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published, doi: 10.1109/TSMC.2017.2703950.
[14] K. Marussy, A. Klenik, V. Molnár, A. Vörös, I. Majzik, and M. Telek, "Efficient decomposition algorithm for stationary analysis of complex stochastic Petri net models," in *Proc. 37th Int. Conf. Appl. Theory Petri Nets Concurrency*, 2016, pp. 281–300.
[15] I. Grobelna, R. Wisniewski, M. Grobelny, and M. Wisniewska, "Design and verification of real-life processes with application of Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 11, pp. 2856–2869, Nov. 2016.
[16] X. Chen and L. Wang, "Exploring fog computing-based adaptive vehicular data scheduling policies through a compositional formal method—PEPA," *IEEE Commun. Lett.*, vol. 21, no. 4, pp. 745–748, Apr. 2017.
[17] X. Chen and L. Wang, "A cloud-based trust management framework for vehicular social networks," *IEEE Access*, vol. 5, pp. 2967–2980, 2017.
[18] J. Ding and J. Hillston, "Structural analysis for stochastic process algebra models," in *Proc. AMAST*, Lac-Beauport, QC, Canada, Jun. 2010, pp. 1–27.
[19] J. Ding, J. Hillston, and D. Laurenson, "Structural and fluid analysis for large scale PEPA models, with applications to content adaptation systems," Ph.D. dissertation, Univ. Edinburgh, Scotland, U.K., 2010.
[20] K.-Q. Zhou, A. M. Zain, and L.-P. Mo, "A decomposition algorithm of fuzzy Petri net using an index function and incidence matrix," *Expert Syst. Appl.*, vol. 42, no. 8, pp. 3980–3990, 2015.
[21] J. Hillston, *A Compositional Approach to Performance Modelling*. Cambridge, U.K.: Cambridge Univ. Press, 1996.
[22] S. Wang, N. Zhan, and L. Zhang, "A compositional modelling and verification framework for stochastic hybrid systems," *Formal Aspects Comput.*, vol. 29, no. 4, pp. 751–775, 2017.
[23] L. Aştefănoaei, S. Bensalem, and M. Bozga, "A compositional approach to the verification of hybrid systems," in *Theory and Practice of Formal Methods* (Lecture Notes in Computer Science), vol. 9660. Cham, Switzerland: Springer, 2016, pp. 88–103.
[24] X. Fu and Q. Shen, "Fuzzy compositional modeling," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 4, pp. 823–840, Aug. 2010.

JIE DING received the B.S. degree in mathematical education from Yangzhou University, Yangzhou, China, in 2001, the M.S. degree in mathematical statistics from Southeast University, Nanjing, China, in 2004, and the Ph.D. degree in communication from The Edinburgh University, Edinburgh, U.K., in 2010. He is currently an Associate Professor with the School of Information Engineering, Yangzhou University. His research interests include performance modeling for communication systems.

XIAO CHEN received the M.Sc. and Ph.D. degrees in computing science from Newcastle University in 2009 and 2013, respectively. He is currently an Associate Professor with Jiangsu University, China. His research interests include formal modeling and performance analysis for large scale systems, e.g., smart systems, cloud systems, and cyber-physical systems. He currently involved in the research of Internet of Vehicles using a stochastic formal method for performance analysis.



RUI WANG received the B.Sc. degree in software engineering from Yangzhou University, Yangzhou, China, in 2011, where he is currently pursuing the M.Sc. degree with the School of Information Engineering. His research interests include performance modeling and evaluation, IoT, and ITS.