IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 9, 2019, accepted September 9, 2019, date of publication September 23, 2019, date of current version October 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2942851

A Petri Net-Based Heuristic Algorithm for Short-Term Vehicle Scheduling in a Vehicle Inspection System

LINJIAN YANG[1], YISHENG AN[1], (Member, IEEE), NAIQI WU[2], (Fellow, IEEE), PEI CHEN[1], (Student Member, IEEE), HANHAN CHENG[1], (Student Member, IEEE), AND XIANGMO ZHAO[1]

[1]School of Information Engineering, Chang'an University, Xi'an 710064, China
[2]Institute of Systems Engineering, Macau University of Science and Technology, Macau 999078, China

Corresponding authors: Yisheng An (aysm@chd.edu.cn) and Naiqi Wu (nqwu@must.edu.mo)

This work was supported in part by the National Natural Science Foundation of China under Grant 61703053, in part by the Key Research and Development Project of Shaanxi Province under Grant 2019GY-070, in part by the China Postdoctoral Science Foundation under Grant 2012M521729, and in part by the Science and Technology Development Fund (FDCT), Macau, under Grant 0017/2019/A1.

ABSTRACT Vehicle inspection systems (VIS) have been extensively used by automobile manufacturers, maintenance companies, and the traffic administrative departments for the inspection of safety, reliability, and other indicators of both newly produced and in-used vehicles. How to increase their productivity by effectively operating such systems is an interesting and crucial problem. Because of the space limitation of testing field in such a system, a vehicle must complete pre-set inspection items at multiple stations in sequence. Hence, it is very challenging to sequence the vehicles to be tested to shorten the unnecessary waiting time for vehicle testing. This paper investigates the vehicle scheduling problem of such systems, where the time for vehicle movement in the system can be ignored. The system is modeled by resource-oriented Petri net, a graphical and mathematical modeling tool. Based on the model, this work analyzes the testing cycle time and develops a heuristic algorithm to efficiently solve the vehicle scheduling problem in such a system. For the first time, a heuristic algorithm for generating an optimal vehicle queue for testing is proposed. Industrial examples are used to illustrate the proposed algorithm and results show that a significant reduction in total test turnaround time can be obtained in comparison with the existing methods.

INDEX TERMS Vehicle inspection system, short-term vehicle scheduling, resource-oriented petri net, vehicle test turnaround time, heuristic algorithm.

I. INTRODUCTION

Vehicle inspection system (VIS) is a kind of key specially purposed facilities mainly used by automotive manufacturers, maintenance companies, and traffic administrative departments for the inspection of safety, reliability and other indicators of both newly produced and in-used vehicles. Statistics show that the widespread use of VISs can significantly improve the safety and economy of vehicle driving and reduce the incidence rate of traffic accidents.

However, the number of existing VISs cannot meet the increasing demands for vehicle testing due to the rapid growth of vehicles in use, leading to a longer average waiting time for vehicle testing in a VIS. Aiming to improve the inspection efficiency and shorten the potential waiting time for vehicle testing, VIS vendors have been increasingly focusing on finding a highly efficient vehicle scheduling algorithm for their vehicle inspection processes.

Given a group of vehicles waiting to be tested, they are queued up to enter a VIS and a dispatching rule is used to release vehicles into the VIS according to the order of the queue. Obviously, the simplest and most common method is to queue up the vehicles according to their arrival order, which we call the first-come first-served scheduling (FCFS) algorithm. Chen *et al.* [1] introduce the concept of variable storage and a sequence-dependent storage policy for the short-term scheduling problem of vehicle performance test system. Then, the short-term scheduling problem for different storage strategies is formulated and solved by mixed integer linear programming (MILP) methods. Later,

The associate editor coordinating the review of this manuscript and approving it for publication was Shih-Wei Lin.

Chen and Pan [2] solve the scheduling problem of reentrant flow shop and job shop by using a mixed binary integer programming method. Ding *et al.* [3] establish a model based on storage state and decomposition structure for the short-term scheduling problem of batch processes and it is then solved by using a decomposition-based genetic algorithm.

For the problem of scheduling optimization of VISs, the above methods can accurately model its key factors, such as constraints and objective functions, but the increasing number of constraints and variables make the solution space increase rapidly such that the problem is very difficult to solve. However, if the problem is simplified by imposing some assumptions, the model cannot fully reflect the actual scheduling requirements, which limits the application of mathematical programming methods for this scheduling problem to some extent [4]–[11].

In the existing VISs, the testing activities at different testing stations can be carried out simultaneously, i.e., it is characterized as concurrency. Meanwhile, as a kind of semi-automatic production system, the evolution of the distributed system states of a VIS is driven by a series of events. Hence, a VIS is also a typical discrete event system (DES) that is characterized by some phenomena such as concurrency, synchronization, and parallelism. Petri net (PN) provides a mathematical and graphical formalized tool to describe concurrent events [12], [13] such that the structure and activities of a system can be systematically modelled.

Hence, Petri nets are widely used to model, simulate, and control various types of automated production systems [14]–[24]. Baruwa *et al.* [25] build a timed-colored Petri net to model the problem of deadlock-free scheduling in flexible manufacturing systems. Based on the model's reachability analysis, a heuristic search algorithm is proposed to find an optimal or near optimal deadlock-free schedule. In wafer fabrication, Zhu *et al.* [26] construct a Petri net model for constrained single-arm cluster tools to analyze its scheduling performance and proposed a heuristic search method to solve deadlock-free scheduling problems in the system with shared resources. Maaoui *et al.* [27] propose an enhanced Petri net model for the manufacturing system scheduling problem for a better solution by using a heuristic search method. Jung and Lee [28] use a time-sharing Petri net to establish a TPN model for cluster tools with various scheduling requirements in semiconductor manufacturing. Based on a TPN model and its state equation, a new mixed integer programming algorithm is proposed to effectively solve the scheduling optimization problem. In [11], the short-term scheduling problem of a VIS is investigated by employing object Petri net (OPN) [29] models and an OPN-based scheduling strategy (PNSS) is presented and analyzed.

Essentially, compared with the traditional FCFS-based short-term vehicle scheduling algorithm, the OPN-based scheduling strategy proposed in [11] can increase productivity and shorten the test turnaround time required for a group of vehicles to be tested. However, the following deficiencies still exist:

a) From the viewpoint of testing performance, there exists a gap between the capability of the number of vehicles in the waiting area and the actual number of vehicles to be tested. In the existing VISs, there are always dozens of vehicles waiting to be tested in a queue, while, according to the scheduling method of the PNSS, only six to seven vehicles can be held in the waiting area.
b) To model the vehicle test operations, the OPN-based method employs a process-oriented model and requires additional places and transitions to represent the testing process.
c) In addition, it is impossible to distinguish whether a transition is process-enabled or resource-enabled in the OPN-based testing operation model.

In this paper, in order to model the concurrent testing activities occurring in a VIS as shown in Fig. 1 and analyze its qualitative properties, with the advantage of resource-oriented Petri nets [30]–[33], we develop an ROPN model such that the space and time constraints used for the testing operations can be well modeled. This model is used to describe the behavior and the functions of the short-term vehicle scheduling problem of a VIS to ensure behavior feasibility.

The remainder of this work is organized as follows. After briefly discussing the testing process and time constraints in a VIS in Section II, an ROPN-based model is developed in Section III. Then, temporal properties and scheduling operational architecture for vehicle testing are presented in Section IV and Section V, respectively. Based on them, a heuristic algorithm for an optimal or near-optimal short-term schedule is developed in Section VI. Then, several performance indicators are analyzed to show the results and the advantages of the proposed heuristic algorithm. Conclusions are given in Section VIII.

II. TESTING PROCESS AND TIME CONSTRAINTS

Generally, a VIS contains a number of testing stations as illustrated by an example in Fig. 1, where the VIS consists of three testing stations named as Stations 1 - 3 in accordance with the order of testing process. In a testing station, there is a testing field and several testing items should be performed. The testing items of the three stations can be pre-configured as speedometer and emission testing, axle weight and brake testing, and headlamp, slab skid, and sound level testing, respectively. Since the testing items performed at different stations are different, the testing time spent for a vehicle at different stations is also different. The test time spent for the testing items of a vehicle at different stations is given in Table 1.

In the vehicle testing operations, the vehicles to be tested can be classified into two types. If a vehicle has not been tested before, we call it a non-inspected vehicle. If a vehicle has been tested at least once and arrives for retesting, we call it an inspected vehicle. When a non-inspected vehicle is tested at a VIS for the first time, it should go through the whole

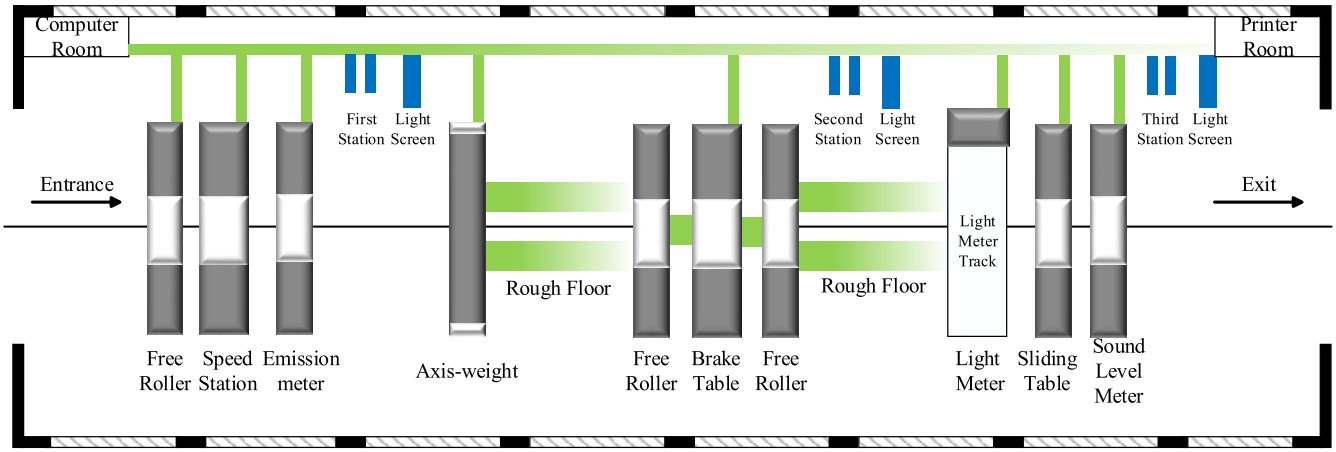


FIGURE 1. An illustrated example of vehicle inspection systems with three test stations.

TABLE 1. The inspection time for the test items of a vehicle at different stations.

Test Station	Items	Time consuming (minutes)	Total time (minutes)
Station one (S1)	speed	2	5
	emission	3	
Station two (S2)	axle weight	3	5
	brake	2	
Station three (S3)	headlamp	4	6
	slide	1	
	sound	1	

VIS and complete all test items. Sometimes, the test results of a non-inspected vehicle for some items may not meet the required standard such that the vehicle need to be repaired and then re-inspected. When such vehicles come to a VIS, they belong to the type of inspected vehicles. Therefore, in a running VIS, there are both inspected and non-inspected vehicles for testing. For an inspected vehicle, often, unlike a non-inspected one, it does not need to test all the items, but only the previously unqualified items need to be tested. This is why there are as many as seven test requirements as shown in Fig. 2, and only the first one is for non-inspected vehicles, while the other six are for the inspected vehicles.

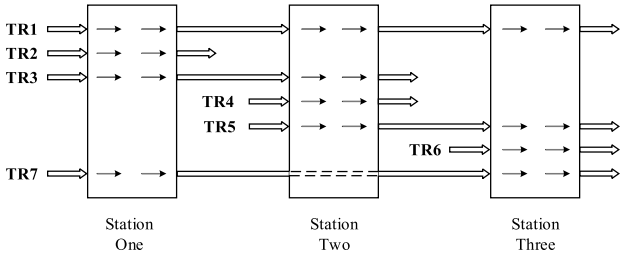


FIGURE 2. Seven optional testing requirements.

According to the layout of a VIS and its testing items shown in Fig. 1, there are seven optional testing operations. We name them as testing requirements (TR) as shown in Fig. 2. TR_1 represents a full testing process in which a non-inspected vehicle is tested sequentially at the three stations for speedometer, emission, axle weight and brake, as well as headlamps, slab skid, and sound level testing. TR_2 to TR_7 describe several different testing operations for inspected vehicles. As shown in Fig. 2, TR_2 means that an inspected vehicle only needs to test the items in station one, similarly, TR_4 and TR_6 correspond to the test of item set at Station 2 or 3 for a vehicle, respectively. TR_3 indicates that an inspected vehicle needs to test the item set at Stations 1 and 2. For the details of TR_5 and TR_7, it can be shown in a way similar to that of TR_3 and are omitted.

In addition, it should be pointed out that, in the existing VISs, the testing item set at each station can only be accomplished by using a specific testing facility. For instance, Station 1 needs to install a speedometer platform and an exhaust gas analyzer, Station 2 needs to install an axle-weight and brake test platforms, and Station 3 needs to install a sound level meter, a sideslip platform, a headlamp tester and its guide rail. The installation locations of these testing facilities cannot be changed once they are installed, and the testing operations of a VIS should be consistent with the arrangement of these facilities. Thus, the testing time required for a vehicle to accomplish the testing items for all three stations is also determined accordingly. Then, we can elaborate the constraints that the vehicle testing operations should satisfy, which is the basis for PN modeling of short-term vehicle scheduling in a VIS.

To describe the time aspect of the vehicle testing operations, we call the start or end of a testing item at a station an event, and we use $V = \{v_1, v_2, \ldots, v_K\}$ and $TS = \{Ts_1, Ts_2, \ldots, Ts_J\}$ to denote the sets of vehicles to be tested and stations, respectively, where K and J are nonnegative integers. Also, $T^{start}_{(v,Ts)}$ and $T^{end}_{(v,Ts)}$ are employed to represent the time points when vehicle $v \in V$ starts to be tested and ends its testing at station Ts ($Ts \in TS$), respectively. Then, we can present the time constraints as follows.

The testing activity of vehicle v_k at station Ts_j should satisfy the following constraint:

$$T^{end}_{(v_k,Ts_j)} > T^{start}_{(v_k,Ts_j)} \tag{1}$$

For vehicle v_k to be tested on adjacent stations Ts_j and Ts_{j+1}, the following constraint should be satisfied:

$$T^{start}_{(v_k,Ts_{j+1})} \geq T^{end}_{(v_k,Ts_j)} \tag{2}$$

For two vehicles v_k and v_{k+1} to be tested at the same station Ts_j, the following time constraint should be satisfied:

$$T^{start}_{(v_{k+1},Ts_j)} > T^{end}_{(v_k,Ts_j)} \tag{3}$$

According to the above time constraints, it can be inferred that as long as the total number (K) of vehicles to be tested and their testing order are known, the total testing time of the K vehicles can be calculated by the following formula.

$$T = \sum_{k=1}^{K} \left(T^{end}_{(v_k,Ts_J)} - T^{start}_{(v_k,Ts_1)} \right) \tag{4}$$

On this basis, in order to quantitatively analyze the efficiency of vehicle testing operation in a VIS, the test turnaround time of K vehicles is defined and calculated according to the following formula:

$$TR = T^{end}_{(v_K,Ts_J)} - T^{start}_{(v_1,Ts_1)} \tag{5}$$

With Equations (4) and (5), for a vehicle queue with K vehicles, we can further infer that the following formula holds:

$$T \geq TR \tag{6}$$

This inequality must be true, mainly due to the fact that there are a lot of concurrent testing activities in the testing operations of a VIS. The shortcoming of computing the total testing time of K vehicles is that the time consumption by the concurrent testing activities are repeatedly calculated, whereas the test turnaround time eliminates such repeated calculations. Hence, the test turnaround time is a more accurate indicator to characterize the testing efficiency. Thereafter, we use the term ''test turnaround time'' to analyze the efficiency in operating a VIS.

III. PETRI NET (PN) MODELING

A. RESOURCE-ORIENTED PETRI NET

In this work, the testing processes of a VIS are modeled by using a specially purposed Petri net (PN) called resource-oriented Petri net (ROPN) [30]. PN provides a powerful tool to model and analyze the dynamic behavior of resource allocation in various types of automated manufacturing systems. In a VIS, the resources (including testing equipment and testing field) are limited, hence, a finite capacity PN is a very suitable choice to model the system.

Definition 1: A finite capacity Petri net is a six-tuple, $PN = (P, T, I, O, M, K)$, where

1) $P = (p_1, p_2, p_3, \ldots, p_m)$ is a finite set of places;
2) $T = (t_1, t_2, t_3, \ldots, t_n)$ is a finite set of transitions;
3) $I : P \times T \to N = \{0, 1, 2, \ldots\}$ is an input function. If $I(p, t) > 0$, there is an arc from place p to transition t, and its weight is $I(p, t)$; if $I(p, t) = 0$, there is no arc from p to t;
4) $O : P \times T \to N$ is an output function. If $O(p, t) > 0$, there is an arc from transition t to place p, and its weight is $O(p, t)$; if $O(p, t) = 0$, there is no arc from t to p;
5) $M : P \to N$ represents the marking of PN, M_0 is an initial marking, and $M(p_i)$ denotes the number of tokens in p_i;
6) $K : P \to \{1, 2, \ldots\}$ is a capacity function, $K(p_i)$ is the maximum number of tokens which p_i can hold at a time.

We use ${}^\bullet t = \{p : p \in P \wedge I(p, t) > 0\}$ to denote the preset of t, which is a set of input places to t; and $t^\bullet = \{p : p \in P \wedge O(p, t) > 0\}$ the postset of t, which is a set of output places from t.

Similarly, the preset and postset of a place are expressed as ${}^\bullet p = \{t : t \in T \wedge O(p, t) > 0\}$ and $p^\bullet = \{t : t \in T \wedge I(p, t) > 0\}$, respectively.

Definition 2: Let M be a marking of a finite capacity PN, a transition $t \in T$ is said to be enabled at M, if for any $p \in P$,

$$M(p) \geq I(p, t) \tag{7}$$

$$M(p) - I(p, t) + O(p, t) \leq K(p) \tag{8}$$

are satisfied. This is denoted as $[M > t$.

Definition 3: Let M be a marking of a finite capacity PN, if transition t fires, one can get a new marking as follows

$$M'(p) = M(p) - I(p, t) + O(p, t) \tag{9}$$

By the above definitions, if $\forall p \in {}^\bullet t$ have enough tokens and $\forall p \in t^\bullet$ have enough free spaces, then t is enabled and can fire. More details of PN and the structural and behavior properties of ROPN can be found in [30] and [31].

B. MODELING TESTING OPERATIONS

The key in solving the short-term vehicle scheduling problem for a VIS is to coordinate the testing activities of large number of vehicles on multiple testing stations and to ensure the maximum number of vehicles tested per unit time, or the shortest test turnaround time for a given number of vehicles. Each test station performs several testing items, depending on different testing equipment. Meanwhile, for a non-inspected vehicle, all of the testing items must be completed sequentially. For an inspected vehicle, even if it does not need to be tested for all items, it has to go through all the stations in turn due to the layout of a VIS. In order to clearly describe the processes that vehicles move sequentially through the stations of a VIS, each testing item at a testing station, the testing field, and the testing station itself need to be modeled separately. We first develop a PN model for the testing operations at individual stations and then present how the PN model of the entire system can be obtained.

We adopt the ROPN modeling method to model the system. Besides considering the time constraints discussed in the previous section, the following space constraints should also be considered:

- Two or three testing items are set for a testing station;
- Different testing items use different testing equipment;
- Different testing items share the same testing field at a station;
- The execution order of the testing items is pre-set.

By treating a testing item as a component for modeling, we first develop a PN model for testing item j at station Ts_i. As a kind of resource, the equipment used to test the j-th testing item at station Ts_i is modeled by timed place P_{ij} with $K(P_{ij}) = 1$, where $j \in \{1, \ldots, n[i]\}$, and $n[i]$ indicates that there are $n[i]$ testing items at Ts_i. A token in P_{ij} indicates that Station Ts_i is testing a vehicle for item j. Transition X_{ij} is used to represent that a vehicle ends the testing of item j and starts the testing of item $j + 1$. By adding arcs $(P_{i1}, X_{i1}), \ldots, (X_{i(j-1)}, P_{ij}), (P_{ij}, X_{ij}), \ldots$, and $(X_{i(n[i]-1)}, P_{i(n[i])})$, the model for testing multiple items at station Ts_i is realized as shown in Fig. 3. This model describes the testing processes of items at a station.

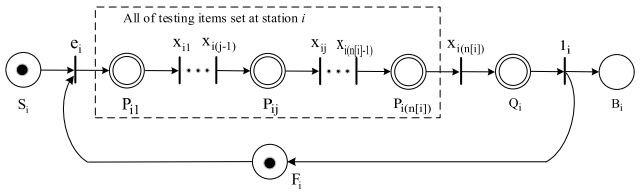


FIGURE 3. The PN model for an individual testing station Ts_i.

Based on this model, we can model the testing behavior of a station. Place F_i with $K(F_i) = 1$ is used to model the testing field at Ts_i, where $K(F_i) = 1$ indicates that at any time only a single vehicle can be accommodated at Ts_i. Transition e_i and l_i represent the actions that a vehicle enters and leaves Ts_i, respectively. Places S_i and B_i indicate that a vehicle is ready to start its testing at Ts_i and leave the station, respectively, so that $K(S_i) = K(B_i) = 1$. Two arcs (S_i, e_i) and (e_i, P_{i1}) are added such that e_i also models the start of testing a vehicle at Ts_i. Transition $X_{i(n[i])}$ connects two timed places $P_{i(n[i])}$ and Q_i. Timed place Q_i indicates that a vehicle completes all testing items at Ts_i and waits there for entering the next testing station Ts_{i+1} for testing. Arcs $(P_{i(n[i])}, X_{i(n[i])})$ and $(X_{i(n[i])}, Q_i)$ are added to represent that $X_{i(n[i])}$ also models that a vehicle ends its testing at Ts_i and is ready for its testing at the next station and waiting in Q_i for the availability of the next station. Arcs (Q_i, l_i) and (l_i, B_i) represent that l_i models that a vehicle leaves for the next station. Finally, arc (F_i, e_i) indicates that when a vehicle starts its testing, it occupies the field for the station too and the field cannot be released until the vehicle leaves the station. Thus, (l_i, F_i) implies that l_i also indicates the release of the field. In this process, the behavior of a stationTs_i is well modeled as shown in Fig. 3. The physical meaning of all places and transitions in Fig. 3 are given in Table 2.

TABLE 2. The physical meaning of places and transitions in Fig. 3.

Element	Type	Meaning
S_i / B_i	Place	A vehicle is ready to enter/leave Ts_i
Q_i	Place	A vehicle completes the test at Ts_i and waits there for entering station Ts_{i+1}
F_i	Place	The test field at station Ts_i
P_{ij}	Place	The equipment for the j-th test item at the i-th Station
e_i / l_i	Transition	A vehicle enters/leaves station Ts_i
X_{ij}	Transition	Connect different testing items for ending the testing of the j-th item and starting the testing of (j+1)-th item
$X_{i(n[i])}$	Transition	Connect the last testing item to the waiting state of station Ts_i to indicate that a vehicle ends its testing at the station

With the model of individual testing stations, the model for the whole vehicle testing process is composed of a plurality of testing station models connected in series. For the modeling of the entire process, the testing field at station Ts_i can be treated as both an output buffer of station Ts_{i-1} and an input buffer of Ts_{i+1}. Hence, the testing activities occurring at adjacent stations interact with each other. Consider that places S_i and B_i are the interface places of the model for station Ts_i, the PN models of Ts_i and Ts_{i+1} can be connected by combining place B_i of Ts_i with place S_{i+1} of Ts_{i+1} as shown in Fig. 4. On this basis, place Wa, transition S_e, arcs (Wa, S_e) and (S_e, S_1) are added to the model of a VIS, where a macro-token in place Wa is used to represent the vehicles waiting for testing, S_e means that the selected vehicle begins to enter the first station. In addition, in the last testing station Ts_J, transition $X_{J(n[J])}$ and place Q_J are removed and arc $(P_{J(n[J])}, l_J)$ is added to indicate that a vehicle exits after completing the last test item at station Ts_J.

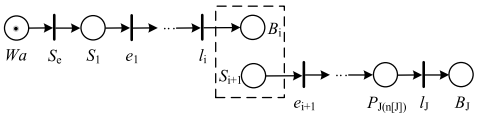


FIGURE 4. The PN model for a VIS with J testing stations.

By the above modeling, at any time and any station Ts_i, among places $P_{i1} - P_{i(n[i])}$, only one place has a token in it, which exactly describes the physical process. Thus, the structure of the PN model of a VIS is well constructed. Note that the time for a vehicle to move from testing an item to another testing item at a station and from a station to another is very short, hence the time taken for item testing decides the productivity of the system. Therefore, in the steady-state stage, if each station Ts_i ($i \in [1, J]$) is testing a vehicle, the system can achieve the maximum productivity.

It is natural that the system starts up from an idle state at which every station is idle and there are a number of vehicles waiting to be tested. Thus, the initial marking M_0 of the PN model is set as follows.

1) $M_0(Wa) = 1$;
2) $M_0(P_{ij}) = 0, j \in \{1, \ldots, n[i]\}$ and $i \in \{1, \ldots, J\}$;

3) $M_0(F_i) = 1$ and $M_0(Q_i) = 0$;
4) $M_0(S_i) = 0$.

At M_0, according to the transition enabling and firing rules [30], [31], S_e can fire such that a token enters S_1, this token can then go to $p_{11}, p_{12}, \ldots$, and $p_{1(n[1])}$, and then B_1 and S_2. At the same time, one token can enter S_1 again. In this way, after some time, there is a token (vehicle) being processed at every station and the system enters the steady state.

C. MODELING ACTIVITY TIME

For scheduling, time aspect should be taken into account. Since the time for vehicle moving in the system is very short compared with the time for testing, we ignore the time for vehicle moving. Note that all the vehicle movements are modeled by transitions in the model. Thus, in the PN model, time is associated with part of places only but not transitions and it is a p-timed PN. If time ζ is associated with place p, it requires that the token should reside in p at least ζ time units and then it can enable its output transitions.

Let the time taken for performing the j-th testing item at Ts_i be α_{ij}, the time for loading and unloading the testing equipment be μ_{ij} and μ'_{ij}, the dwell time of a vehicle at station Ts_i be τ_i, and the waiting time for a vehicle at station Ts_i be ω_i, then the time associated with the places is shown in Table 3.

TABLE 3. Time associated with places.

Symbols	Type	Meaning
α_{ij}	$P_{ij} \in P$	Time taken for testing the j-th item at Station Ts_i
μ_{ij} / μ'_{ij}	$P_{ij} \in P$	Time for loading or unloading an equipment
τ_i	Ts_i, subnet	Time for a vehicle dwelling at station Ts_i
ω_i	$Q_i \in P$	Waiting time for a vehicle at station Ts_i
	$S_i, B_i \in P$	No time is associated

IV. TEMPORAL PROPERTIES

In order to optimize the vehicle testing operations in a VIS, the vehicles should be scheduled in such a way that the overall system testing efficiency is optimal, that is, each testing station performs test concurrently. Therefore, in order to schedule the system properly, it is necessary to sequence the vehicles at each station first. The waiting time of a vehicle at a station is introduced here to parameterize the vehicle scheduling problem. From the perspective of the whole system, the moment when a vehicle is scheduled to leave a station and enter into the next station can be determined by the waiting time of the vehicle in the station. Also, to optimally operate a VIS, all stations should operate in a paced way.

Based on the PN model proposed in the previous section, with the workload being different for different stations, we analyze how to set a reasonable waiting time at Ts_i and analyze the cycle time of Ts_i. Without loss of generality, for any station Ts_i, assume that there are $n[i]$ ($n[i] >= 2$) testing items. In order to complete the j-th testing item at Ts_i, the following activities should be performed sequentially (firing the transitions sequentially):

$\sigma_1 = \langle X_{i(j-1)} \rightarrow P_{ij}$ (testing the j-th item at Ts_i with α_{ij} time units + μ_{ij} time units for starting $+\mu'_{ij}$ time units for unloading) $\rightarrow X_{ij} \rangle$.

It is worth noting that the execution of all testing items at Ts_i is performed on the same testing field modeled by place F_i as shown in Fig. 3, which means that a tested vehicle does not need to move for switching from one testing item to another and the switch is realized by changing the testing equipment by an operator. Therefore, the time required to perform the j-th testing item at Ts_i is calculated according to

$$\xi_{ij} = \alpha_{ij} + \mu_{ij} + \mu'_{ij}, \quad j \in N_{n[i]} \tag{10}$$

In (10), ξ_{ij} represents the testing time required to complete the j-th testing item at Ts_i. Consider that the vehicle testing in a VIS is production-oriented, so that the time of loading (starting) and unloading (ending) the testing equipment can also be incorporated into the testing time α_{ij} for simplicity. Then, Equation (10) can be rewritten as

$$\xi_{ij} = \tau_{ij} \tag{11}$$

where τ_{ij} is the dwelling time of a vehicle for the j-th item at Ts_i, obviously $\tau_{ij} > \alpha_{ij}$.

Similarly, in order to complete all the testing items at station Ts_i, the following activities should be performed sequentially:

$\sigma_2 = \langle$ firing transition e_i to enter station $Ts_i \rightarrow$ a token takes τ_{i1} time units for testing at $P_{i1} \rightarrow$ firing transitions X_{i1} to complete the first test item at Ts_i and switch to the next test item $\rightarrow, \ldots, \rightarrow$ a token takes τ_{ij} time units for testing at place $P_{ij} \rightarrow$ firing transition X_{ij} to complete the j-th item $\rightarrow, \ldots, \rightarrow$ a token takes $\tau_{i(n[i])}$ time units for testing at place $P_{i(n[i])} \rightarrow$ firing transition $X_{i(n[i])}$ to complete the testing of the $n[i]$-th item $\rightarrow$ a vehicle takes ω_i time units for waiting at place $Q_i \rightarrow$ firing transition l_i to leave station $Ts_i \rangle$. In this way, the time for a vehicle to perform inspection at Ts_i is:

$$\xi_i = \sum_{j=1}^{n[i]} \tau_{ij} + \omega_i \tag{12}$$

In (12), ξ_i can be divided into two parts ξ_{i1}, ξ_{i2}:

1) $\xi_{i1} = \sum_{j=1}^{n[i]} \tau_{ij}$ gives the actual testing time required for all testing items at Ts_i;

2) $\xi_{i2} = \omega_i$ indicates the waiting time at Ts_i.

In the existing testing process, the value of ξ_{i1} is relatively fixed. Since the testing operation of a VIS is a serial production-oriented testing process, under the steady state, the productivity of each station must be the same, which means that the vehicles must have the same dwelling time at each station.

Let $\eta_i = \xi_{i1}$ and $\prod = max\{\eta_1, \eta_2, \ldots, \eta_J\}$, then $\prod$ is called the cycle time of a VIS, i.e., in every cycle with $\prod$ time units, a vehicle completes its testing at any station Ts_i. In the corresponding PN model, a token which represents a

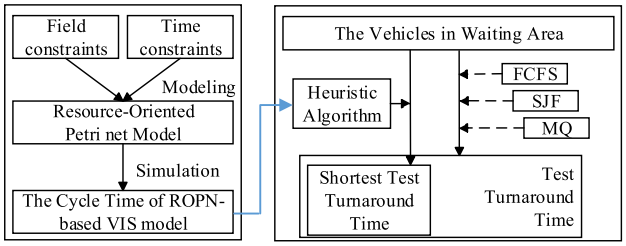


FIGURE 5. An ROPN-based vehicle testing operation architecture.

vehicle is deposited into a specific place which is a fusion of S_{i+1} and B_i as shown in Fig. 4. Therefore, in the steady state stage, the VIS should take $\prod$ as cycle time to schedule the vehicles.

If the workload of a station is equal to the cycle time of a VIS ($\eta_k = \prod$), then this station is called the bottleneck station of the VIS. In order to ensure that the non-bottleneck station can also be scheduled by $\prod$, we can let each vehicle wait for $\xi_{i2} = \omega_i = \prod - \eta_i$ time units in place Q_i. In this way, we can summarize the content of this section with a proposition, that is, when solving the optimal scheduling of the production-oriented VIS, the dwelling time of the bottleneck station should be used as cycle time for scheduling the system.

V. OPERATIONAL ARCHITECTURE FOR SCHEDULING A VIS

With the above ROPN-based vehicle testing operation model and temporal properties, we can further infer that, as long as the cycle time of a VIS, the total number of vehicles to be tested, their arrival order, and the testing items of each vehicle are known, one can obtain the total test turnaround time of them in some way other than just by simulating the testing operations proposed in [11]. Therefore, we propose an operational architecture for vehicle scheduling, which is compatible with the inference as shown in Fig. 5. In this architecture, the vehicle testing operations in a VIS are divided into two stages. The first stage is to analyze the cycle time of a VIS. Under the steady state, if a schedule can achieve the cycle time as the one obtained by analysis, the VIS operates in a most efficient way.

At the second stage, algorithms are given to generate a vehicle queue (i.e., a sequence of the vehicles) in the waiting area. The algorithm can be an FCFS-based, or SJF-based, or MQ-based algorithm, or PNSS, where SJF and MQ are the abbreviation of short-job-first and multiple-queues; while PNSS is the acronym of PN-based scheduling strategy proposed in [11]. Besides the above algorithms, in this work, a heuristic algorithm is proposed to generate the near-optimal vehicle queue and the dispatcher of a VIS can take the head vehicle of the queue one by one and schedule the vehicles to the VIS. Thus, when the system is running for a period of time to enter the steady state, every cycle, a vehicle is tested and leaves the last station.

In addition, a simulation-based test turnaround time calculation method is proposed in this section, which can be used to verify the reliability of the heuristic algorithm from one aspect. The specific process is as follow. The vehicle queue obtained by the heuristic algorithm is treated as a macro-token and put into place *Wa* in Fig. 4. During the simulation, when the head vehicle is released to the VIS for testing, a token is deposited into the PN model for the testing operations, while the token leaves the PN model when the testing operations for the last vehicle is completed. We record the time points when the token enters and leaves the PN model. Then, the test turnaround time can be calculated by Formula (5). In this way, the vehicle queue generated by using the heuristic algorithm can be verified and taken as a solution for short-term vehicle scheduling.

VI. HEURISTIC ALGORITHM FOR NEAR-OPTIMAL SCHEDULING

In [11], a concept of waiting area for a VIS is proposed, which is a parking area where the vehicles wait for entering the VIS. By this method, it enumerates all possible sequences for the vehicles in the waiting area to obtain an optimal schedule by using a PN model. However, to make the problem solvable, it limits the space of the waiting area such that it can accommodate six to seven vehicles only. When there are more vehicles waiting to be tested, the time taken to calculate the optimal vehicle sequence would be very long. For example, if 10 vehicles are waiting to be inspected, there are 10! possible sequences. To generate these 10! sequences and determine the optimal one from them, it takes about 151.2s. Although this time is relatively shorter than that for completing the testing the 10 vehicles, when the number of vehicles for testing further increases, the time to obtain the optimal schedule would be much longer, since the time for that would increase exponentially with the number of vehicles for testing. However, there are often more than 10 vehicles waiting for testing in practice. In response to this application requirement, based on the improvement of PNSS, a heuristic algorithm for near-optimal scheduling of vehicles for a VIS is presented. Essentially, this algorithm provides a queuing scheme for the vehicles that are not queued in the waiting area based on the testing requirements of each vehicle. In this work, we still use the concept of the waiting area and it is denoted by place *Wa* in the PN model in Fig. 4, but we do not impose any capacity constraint on it. Before we introduce this algorithm in detail, the interaction between adjacent vehicles during the test has to be investigated first.

A. INTERACTION BETWEEN TWO ADJACENT VEHICLES

Take the vehicle scheduling problem for a typical three-station VIS as an example, we show that the time spent for testing is affected by the order of the vehicles entering a VIS. That means the interaction between adjacent vehicles has a great impact on the test turnaround time of these vehicles.

Suppose that the testing requirements of two vehicles v_{i} and v_{j} are TR_4 and TR_6, respectively, and the cycle time of the VIS is $\prod$. Thus, we can find that there is an optimal vehicle sequence. We analyze these two testing requests in two cases:

Case 1: It schedules vehicle v_{i} first and then v_{j}. When v_{i} is being tested at Ts_2, vehicle v_{j} has to wait at Ts_1 without being test until v_{i} completes the testing items at Ts_2. Then, after v_{i} ends its testing at Ts_2, v_{j} can starts its first required test item at Ts_3. By this schedule, the total test turnaround time is two times of $\prod$.

Case 2: It schedules vehicle v_{j} first and then v_{i}. While vehicle v_{j} is being tested at Ts_3, vehicle v_{i} is being tested at Ts_2 simultaneously, such that the total test turnaround time of these two vehicles is reduced to $\prod$.

The interaction between the vehicles in these two cases is referred as to testing conflict and testing compatibility as defined below:

Definition 4: Let V be a vehicle set and TS be a station set. The testing state of a vehicle is defined as a two-tuple $vts = (v_{\mathrm{i}}, Ts_{\mathrm{k}})$, where $v_{\mathrm{i}} \in V$ and $Ts_{\mathrm{k}} \in TS$. If $vts = (v_{\mathrm{i}}, Ts_{\mathrm{k}}) = 1$, it means that it is true that v_{i} is being tested at Ts_{k}; otherwise, $vts = (v_{\mathrm{i}}, Ts_{\mathrm{k}}) = 0$.

With Definition 4, we present the following definitions.

Definition 5: Let v_{i}, $v_{\mathrm{j}} \in V$ be two vehicles and Ts_{k}, $Ts_{\mathrm{k+1}} \in TS$ be two adjacent testing stations. Vehicles v_{i} and v_{j} are in conflict for stationTs_{k}, if and only if

$$\left((v_i, Ts_k) \wedge \neg \left(v_j, Ts_{k+1}\right)\right) \vee \left(\left(v_j, Ts_k\right) \wedge \neg \left(v_i, Ts_{k+1}\right)\right) = 1.$$

Definition 6: Let v_{i}, $v_{\mathrm{j}} \in V$ be two vehicles and Ts_{k}, $Ts_{\mathrm{k+1}} \in TS$ be two adjacent testing stations. Vehicles v_{i} and v_{j} are compatible for stations Ts_{k} and $Ts_{\mathrm{k+1}}$, if and only if

$$\left((v_i, Ts_{k+1}) \wedge \left(v_j, Ts_k\right)\right) \vee \left(\left(v_j, Ts_{k+1}\right) \wedge (v_i, Ts_k)\right) = 1.$$

Definition 7: Suppose that the cycle time of a VIS is $\prod$, if the testing requirement of vehicle v is TR_i (i $\in \{1, 2,..., 7\}$) and the number of testing stations required to complete the testing of TR_i is $Len(\mathrm{TR}_i)$, then the test turnaround time of v is $\prod \times Len(\mathrm{TR}_i)$.

Lemma 1: Let the number of testing stations in a VIS is J, and the queue of vehicles waiting for testing is $v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_K$. There is no conflict or compatibility relationship between vehicle v_i and vehicle v_{i+J} and the subsequent vehicles.

Proof: Let $TS = \{Ts_1, Ts_2, \ldots, Ts_{\mathrm{J}}\}$ be a set of testing stations and $V = \{v_1, v_2, \ldots, v_{\mathrm{K}}\}$ be a set of vehicles. There are $(J - 1)$ vehicles between v_i and vehicles v_{i+J}. Obviously, when v_i is being tested at station Ts_{J}, regardless of whether vehicles v_{i+1} to $v_{(i+J-1)}$ are tested or waiting at stations from $Ts_{\mathrm{J-1}}$ to Ts_1 one by one, in general, these stations are occupied. By the ROPN model shown in Fig. 3, the token in place F_i is take away. Hence, vehicle v_{i+J} cannot be scheduled to enter the first station. Under the condition that the cycle time of a VIS is determined, vehicle v_{i+J} can enter the first station if and only if the vehicle v_i completes the testing at station Ts_{J} and leaves the field of the station. Therefore, there is no conflict or compatibility relationship between vehicle v_i and vehicle v_{i+J} and the subsequent vehicles. ■

Algorithm 1 Calculation of Compatible Number of Two Vehicles' Test Requirements

input: *Test requirements of two vehicles*:TR_A, TR_B

output: *The compatible number*

```
1.  function Compatiblenumber (TR_A, TR_B)
2.  //Calculate the Compatible number of TR_A and TR_B
3.  Compatible_number← 0
4.  i, j
5.   for each i ←TR_A.length-1: Min(TR_A.length,
6.   TR_B.length)
7.    for each j ← 0: TR_A.length−1−i
8.      if TR_A[i+j]<=TR_B[j] then
9.       exit
10.     end if
11.   end for
12.    if (j==TR_A.length-i) then
13.     Compatible_number ++
14.   end if
15.  end for
16.  returnCompatible_number
17. end function
```

If v_{i} and v_{j} are compatible with respect to $(v_{\mathrm{i}}, Ts_{\mathrm{k+1}})$ and $(v_{\mathrm{j}}, Ts_{\mathrm{k}})$ or $(v_{\mathrm{j}}, Ts_{\mathrm{k+1}})$ and $(v_{\mathrm{i}}, Ts_{\mathrm{k}})$, we say that there is a compatible pair. For testing requirements of two vehicles, there may be a number of compatible pairs, which is called the compatible number. Obviously, based on the above discussion, a compatible pair implies that the adjacent stations can perform the testing tasks concurrently, that is to say, two vehicles are tested during the same cycle. Hence, we can infer that the greater the compatible number of the testing requirements of two vehicles is, the higher the testing efficiency is. Given two testing requirements, we propose Algorithm 1 to calculate the compatible number for any two testing requirements of the two vehicles.

With Algorithm 1, it shows how to sequence two vehicles with the shortest test turnaround time. Based on Algorithm 1, we present how to calculate the total test turnaround time of two vehicles.

Lemma 2: Suppose that the cycle time of a VIS is $\prod$ and two vehicles v_{i} and v_{j} are waiting for testing, their test requirements are TR_A and TR_B, respectively. From the time when the first vehicle enters the VIS to the time when the second vehicle leaves the system, the time duration can be calculated as:

$$\left(Len\left(TR_A\right) + Len\left(TR_B\right) - Compatiblenumber\left(TR_A, TR_B\right)\right) \times \prod \quad (13)$$

Proof: As known that there are up to seven possible testing requirements for a vehicle, there are totally 49 combinations of testing requirements for two vehicles. By enumerating these 49 cases, the result of formula (13) can be shown to be consistent with the simulation result based on the ROPN model, which proves that formula (13) is valid. ■

TABLE 4. Test requirements, compatible numbers and test turnaround time of two vehicles.

Testing Requirements		Compatible number	Test Turnaround Time
First Vehicle	Second Vehicle		
Ts_1	Ts_1	0	$(1+1-0)\times\Pi$
Ts_2	Ts_3	0	$(1+1-0)\times\Pi$
Ts_3	Ts_2	1	$(1+1-1)\times\Pi$
Ts_1	$Ts_1; Ts_2$	0	$(1+2-0)\times\Pi$
$Ts_1; Ts_2$	Ts_1	1	$(2+1-1)\times\Pi$
$Ts_1; Ts_2$	$Ts_2; Ts_3$	0	$(2+2-0)\times\Pi$
$Ts_2; Ts_3$	$Ts_1; Ts_2$	2	$(2+2-2)\times\Pi$
$Ts_1; Ts_2; Ts_3$	$Ts_2; Ts_3$	1	$(3+2-1)\times\Pi$
$Ts_2; Ts_3$	$Ts_1; Ts_2; Ts_3$	2	$(2+3-2)\times\Pi$

Here, we randomly selected nine of them to analyze their testing compatibility. Table 4 lists the testing requirements, the compatible number, and the total test turnaround time. From Table 4, we can observe that, by adjusting the order of the vehicles for testing, the compatible number changes and the total test turnaround time changes as well.

B. INTERACTION AMONG THREE ADJACENT VEHICLES

Under the condition that the testing requirements of two vehicles are known, suppose that a new vehicle with its testing requirement being known is added to be tested. How can the total test turnaround time for these three vehicles be calculated? Obviously, according to Lemma 1, in a three-station VIS, after adding the third vehicle, the testing operations of the third vehicle may be affected by the testing state of the second vehicle that is released into the system before the third one.

The situation between the second vehicle and third one is similar to the one discussed in the previous subsection. Table 5 gives three different sequences for three vehicles and the test turnaround time for them is obtained by simulation based on the PN model.

TABLE 5. Testing requirements and test turnaround time for different sequences of three vehicles.

Testing Requirements			Test turnaround time	
First Vehicle	Second Vehicle	Third Vehicle	First Two Vehicles	Three Vehicles
Ts_3	Ts_2	Ts_1	Π	Π
Ts_1	Ts_2	Ts_3	$2\,\Pi$	$3\,\Pi$
$Ts_2; Ts_3$	$Ts_1; Ts_3$	$Ts_1; Ts_2$	$3\,\Pi$	$3\,\Pi$

In Table 5, for the sequence in the first row, there is no conflict for the testing of the three vehicles, but they are compatible. Therefore, the test turnaround time for the first two vehicles is same as the one for the three vehicles. However, for the sequence given in the second row, although the second vehicle needs not to be tested at the first station, it is blocked by the testing of the first one at the first station such that it cannot go to the second station. Also, the third vehicle was blocked by the second one at the second station such that it cannot go to the third station. Consequently, it spends one more cycle time to complete the testing than the sequence given in the first row. For the situation given in the third row, since the testing of the third vehicle at the second station is compatible with the testing of the second vehicle at the third station, and the testing of the second vehicle at the first station and the testing of the first vehicle at the second station is also compatible, the test turnaround time of the first two vehicles is same as that for the three vehicles.

From the above analysis, we have the following finding. When the test turnaround time of two vehicles is known and a third vehicle is to be added, to minimize the test turnaround time of the three vehicles, the first two vehicles cannot be simply viewed as a whole to put the newly arriving vehicle at the tail in the vehicle queue. The main reason is that the scheduling of the third vehicle may be affected by the testing status of the first vehicle. That is to say, when the number of stations is three, it is known from Lemma 1 that the third vehicle and the first vehicle are affected each other, depending on whether they are conflict or compatible. Therefore, a new method is necessary for sequencing three vehicles to minimize their test turnaround time.

Based on the above discussion, we propose Algorithm 2 to directly calculate the test turnaround time for three vehicles.

Algorithm 2 Calculate Test Turnaround Time for Three Vehicles

input : *Three vehicles' testing requirements:* TR_A, TR_B *and* TR_C

output: *Test turnaround time of Three Vehicles*

```
D ← n
function Test_tr_time (TR_A, TR_B, TR_C)
  count // integer variables
   Initial_Array (TR_A,0)
   Initial_Array (TR_B,1)
   Initial_Array (TR_C,2)
   for each i← 0: D.length
      ifD[i] = 1 then count ++
      end if
   end for
 return (count* Π)
  end function
  function Initial_Array(Array E, k)
  // Discover non-zero values of array E and modify
    array D
   for each i ← 0: E.length
    if E[i] ≠ 0 then D[i+k] = 1
    end if
   end for
 end function
```

By Algorithm 2, we can get how different sequences of three vehicles affect the test turnaround time, which gives a clue for us to develop a heuristic for scheduling a VIS. There are totally 343 combinations of testing requirements when there are three vehicles. The results obtained by Algorithm 2 are consistent with those obtained by simulation using the ROPN based VIS model, which ensures the correctness of the algorithm.

After the test turnaround time of three vehicles can be obtained by using Algorithm 2, it is found that there is a specific phenomenon that can be analyzed as follows. With three vehicles, the test turnaround time for the first two vehicles can be obtained according to Lemma 2. However, the test turnaround time of the last two vehicles cannot be calculated by applying Lemma 2. It raises a question on how the requirement relation of three vehicles affects the test turnaround time. Does the requirement relation between the first two vehicles have greater effect on it or the one between the last two vehicles has greater effect? In order to answer this question, we introduce two concepts: the nominal test turnaround time and the substantial test turnaround time.

Definition 8: Suppose that three vehicles are waiting for testing, the nominal test turnaround time refers to the calculation of the test turnaround time, regardless of the impact of the third vehicle on the first two vehicles, and the nominal test turnaround time of any two vehicles can be obtained by using (13).

Definition 9: Suppose that three vehicles are waiting for testing, the substantial test turnaround time means that when calculating the test turnaround time for two vehicles v_i and v_{i+1}, the influence of vehicle v_{i-1} on the latter is taken into consideration. The substantial test turnaround time of vehicles v_i and v_{i+1} is calculated as follow, three vehicles v_{i-1}, v_i and v_{i+1} are taken as a whole to calculate the test turnaround time by using Algorithm 2, and then subtract the nominal test turnaround time of two vehicles v_i and v_{i+1} obtained by using (13).

Here, we use an example to illustrate the difference between these two concepts. Assume that there are three vehicles v_a, v_b and v_c waiting in a queue for testing and their testing requirements are TR_1: (Ts_1; Ts_2; Ts_3), TR_3: (Ts_1; Ts_2), and TR_2: (Ts_1), respectively. The results of the nominal and substantial test turnaround time are shown as follows.

Case 1.1: For vehicles v_a and v_b, their nominal test turnaround time is equal to the substantial test turnaround time, and both are $3\prod$.

Case 1.2: For vehicles v_b and v_c, their nominal test turnaround time is $3\prod$, but the substantial test turnaround time is 0.

Similarly, if these vehicles' testing requirements are changed to TR_1: (Ts_1; Ts_2; Ts_3), TR_3: (Ts_1; Ts_2) and TR_6: (Ts_3), respectively. The results of the nominal and substantial test turnaround time in this scenario are shown as follows.

Case 2.1: The nominal test turnaround time and the substantial test turnaround time of vehicles v_a and v_b are still the same, both are $3\prod$.

Case 2.2: For vehicles v_b and v_c, their nominal test turnaround time is $3\prod$, whereas their substantial test turnaround time becomes $\prod$.

This example tells us that when two vehicles v_a and v_b with the testing requirements TR_1: (Ts_1; Ts_2; Ts_3) and TR_3: (Ts_1; Ts_2) are selected to put into the vehicle queue, the system prefers to select the vehicle with testing requirement TR_2: (Ts_1) to follow. In the heuristic algorithm in the next section, under the premise that vehicles v_1 to v_i have been added to the queue waiting for testing, the system selects the next vehicle to enter the queue according to the principle of the shortest substantial test turnaround time.

C. THE HEURISTIC ALGORITHM

When there are more vehicles for testing, from the perspective of system optimization, we need to generate an optimal vehicle queue, i.e., a releasing order for testing to obtain a shortest test turnaround time. The process of generating the vehicle queue can be regarded as a process of continuously appending new vehicles based on a two-vehicle-queue. When the number of vehicles in the waiting area is greater than 10, it is inefficient to enumerate all possible vehicle queues for finding an optimal schedule. Thus, it is better to find a satisfactory solution by an efficient method. In this section, we propose a heuristic algorithm to generate the near-optimal vehicle queue for K vehicles when the cycle time of a VIS is determined by the ROPN-based model and simulation, so as to ensure that the near-optimal vehicle sequence can achieve the shortest test turnaround time.

Assume that the number of vehicles waiting to be tested is K and a vehicle is randomly selected as the first vehicle to enter the VIS, actually, this vehicle is the head of the vehicle queue. Then, the second vehicle is selected from the rest (K-1) vehicles. These (K-1) vehicles form (K-1) vehicle pairs with the first vehicle one by one, and the test turnaround time of these vehicle pairs are calculated according to Lemma 1. The vehicle with the shortest test turnaround time can be selected, and serves as the second vehicle entering the queue. If more than one vehicle has the shortest test turnaround time with the first vehicle, arbitrarily select one of them as the second vehicle to enter the vehicle queue.

When the second vehicle is selected, the third vehicle can be selected from the rest (K-2) vehicles similarly. The slightly difference from the method used for selecting the second vehicle is that, from now on, the substantial test turnaround time is taken as the key index parameter. In order to find a suitable one as the third vehicle to enter the queue, the second vehicle and each of the reminding (K-2) vehicles forms a vehicle pair and totally (K-2) vehicle pairs are formed. Calculate the substantial test turnaround time for each pair and select the vehicle whose vehicle pair has the shortest substantial test turnaround time. Then, this vehicle is the third one in the queue. Similarly, if there are multiple candidates, arbitrarily select one from them as the third vehicle.

For selecting the fourth and later vehicles to enter the vehicle queue, it can be done just as selecting the third vehicle

to enter the queue. By doing so, an overview of the proposed heuristic algorithm is shown in Fig. 6. for generating an optimal sequence of vehicles.

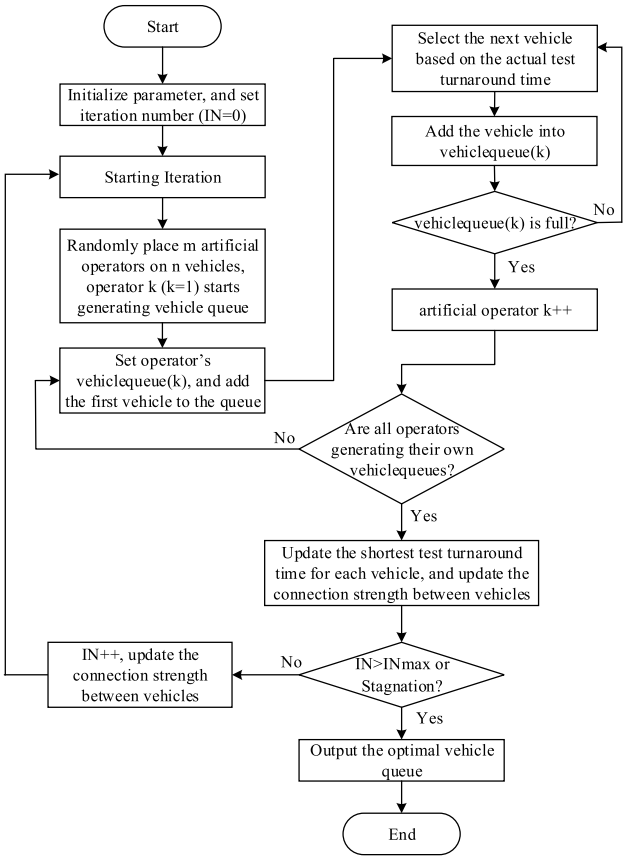


FIGURE 6. Overview of the heuristic algorithm.

In this algorithm, artificial operators are used to perform the queuing operation for K vehicles. Three nested loop structures are set in the diagram for generating the optimal vehicle queue, the innermost loop is used by an artificial operator to find a vehicle queue, the middle layer loop is used to guarantee that each artificial operator can find a vehicle queue, the outermost loop determines the total number of iterations for optimization. In addition, we introduce a key parameter called connection strength. The connection strength is for any two vehicles, reflecting the correlation between the test requirements of the two vehicles. The connection strength is the heuristic factor in the algorithm, when all the artificial operators find their vehicle queue, each operator uses the obtained local optimal vehicle queue to increase the connection strength, meanwhile, the connection strength is reduced according to a certain ratio after each outermost loop, ensuring that it does not prematurely converge because the connection strength is too large. Based on the above discussion and the diagram in Fig. 6, we present a detailed description of the proposed heuristic algorithm given as Algorithm 3.

With the above algorithm, to some extent, we have solved the short-term vehicle scheduling problem for a VIS and it is especially effective for the testing system of a vehicle

Algorithm 3 Heuristic Algorithm for Near Optimal Vehicle Sequence

input: *the number of vehicles to be inspected is n (testing requirements for each vehicle are known) , the number of artificial operators is m, the maximum iteration number* IN_{max}, *vehicles V*

output: *the near-optimal vehicle queue*

1. **function** *SEQUENCEOPTIMIZATION* (*m, n, V, INmax*)
2. *Set the number of testing items for the VIS*
3. $s \leftarrow 1$ *//vehicle queue index*
4. *Time[][] //This two-dimensional array is used to store the test turnaround time of any two vehicles.*
5. **for**($i=1$; $i <=7$; $i ++$)
6. **for**($j=1$; $j<=7$; $j + +$)
7. $Time[v_i][v_j]$ *is calculated according to Lemma 2*
8. **end for**
9. **end for**
10. *set an initial value of connection strength* $cs_{ij} \leftarrow c$ *and* $cs_{ij} \leftarrow 0$ *for each vehicle pair* (v_i, v_j)
11. *place the m artificial operators on the n vehicles, set its* $vehiclequeue_k(s)$ *for each operator*
12. **for** *number of iterations IN*$\leftarrow$ *0:* IN_{max}
13. **if** *stagnation behavior* **then** *print Optimal vehicle queue //the queue with the shortest test turnaround time*
14. **else** *empty all* $vehiclequeue_k(s)$
15. **for** *each operator* $k \leftarrow 1: m$
16. *place the first vehicle of the k-th operator in* $vehiclequeue_k(s)$
17. **while** ($vehiclequeue_k(s)$ *is not full*) **do** *//this step is repeated (n-1) times*
18. *choose the vehicle j to move to, based on the substantial test turnaround time and the connection strength on* $(v_{vehiclequeuek(s)}, v_j)$
19. $s \leftarrow s+1$
20. *move the k-th operator to the vehicle j, insert vehicle j in* $vehiclequeue_k(s)$
21. **end while**
22. **end for**
23. **for** *every pair* (v_i, v_j)
24. **for** *each operator* $k \leftarrow 1: m$
25. *compute the test turnaround time* T_k *of the k-th operator by accumulating the* $Time[v_i][v_j]$ *between each two vehicles*
26. *update the optimal vehicle queue found*
27. **if** $(v_i, v_j) \in$ set of *vehicle queue described by* $vehiclequeue_k(s)$

$$\Delta cs_{i,j}^{k} = Q/T_k$$
$$\Delta cs_{ij} = \Delta cs_{ij} + \Delta cs_{ij}^{k}$$

28. **end if**
29. **end for**
30. $cs_{ij} = \rho^{*} cs_{ij} + \Delta cs_{ij}$ *//Update connection strength*

Algorithm 3 *(Continued.)*Heuristic Algorithm for Near Optimal Vehicle Sequence

```
    end for
    end if
    Δcs_ij ← 0
    IN←IN+1
    end for
    output Optimal vehicle queue
end function
```

TABLE 6. Time complexity of the heuristic algorithm.

Steps	Content	Time complexity
1	Initialization parameter	$O(n^2+m)$
2	Set the Vehicle queue list for each operator	$O(m)$
3	Each operator independently constructs a solution	$O(n^2m)$
4	Route update	$O(n^2m)$
5	Update of connection strength	$O(n^2)$
6	Determine whether the end condition of the algorithm is reached	$O(mn)$
7	Output the calculation result	$O(1)$

manufacturer to improve the operation efficiency, which also shows that the proposed PN model has a strong ability to simulate vehicle scheduling problems.

For the above proposed heuristic algorithm, the computational complexity is analyzed and summarized in the Table 6.

As shown in Table 6, when m operators traverse n nodes and go through IN cycles, the computational complexity by T(n) given below, which is polynomial.

$$T(n) = O\left(IN \cdot n^2 \cdot m\right).$$

VII. INDUSTRIAL CASE STUDY

In this section, several experiments are carried out to illustrate the applications of the proposed algorithm and its efficiency. In the experiments, the PN-based heuristic algorithm for short-term vehicle scheduling of a VIS is implemented by using Visual Studio 2017 and CPN tools on a machine configured with Intel Core i7-8700 CPU and 8GB memory. The data used for simulation and experiment are extracted from the test log files provided by Shaanxi Automobile Group automotive testing center. The layout of its VIS has been shown in Fig.1. The setting of test items and stations as well as time consuming at each item are listed in Table 1.

A. AN INDUSTRIAL CASE

In such a VIS, before the system starts to work, there are tens of vehicles waiting for testing. Here, we use a case with 30 vehicles to show the proposed method. 30 vehicles' information and their testing requirements are extracted from the log file and taken as a complete data segment, and these data are recorded in the order with which the vehicles are actually tested. According to the order of arrival, their testing requirements are listed as follows:

$TR_7, TR_3, TR_5, TR_1, TR_2, TR_1, TR_1, TR_6, TR_1, TR_1, TR_1, TR_1, TR_6, TR_2, TR_1, TR_1, TR_3, TR_1, TR_7, TR_5, TR_1, TR_1, TR_1, TR_4, TR_3, TR_1, TR_1, TR_2, TR_4, TR_6$

For this case problem, first, by using the heuristic algorithm, an optimal vehicle queue is generated. In the experiment, the number of artificial operators is set to 200, the maximum number of iterations is 2000, the cycle time of the VIS is six minutes, and the initial iteration number is IN=0. The initial connection strength between every two vehicles is set as $\tau = 1e-4$.

The data for these 30 vehicles to be tested is shown in Fig. 7(a), and any vehicle can be scheduled to be either of the head of a queue or the tail of a queue, or at any immediate position. 200 artificial operators are randomly assigned to these 30 vehicles, each operator has its own vehiclequeue_k list and puts the first vehicle in the list. Each operator selects the next vehicle to move based on the substantial test turnaround time and the connection strength on $(v_{vehiclequeuek(s)}, v_j)$, and adds the selected vehicle v_j to the vehiclequeue_k list until all the operators' vehiclequeue_k

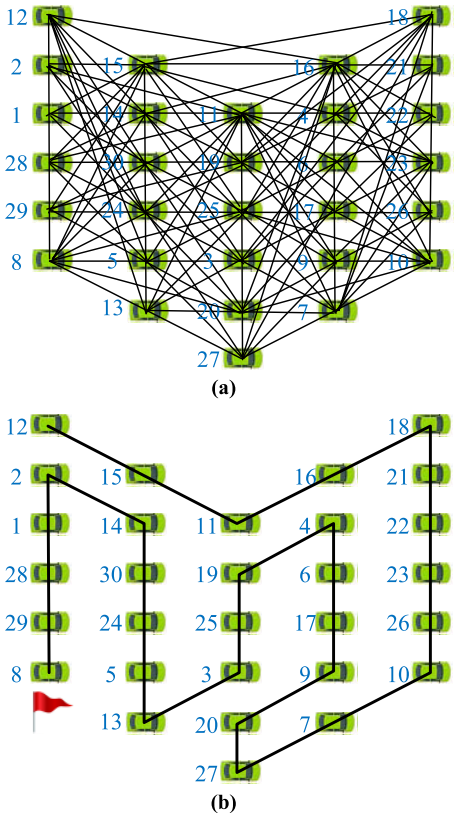


FIGURE 7. A schematic diagram for generating the optimal vehicle queue for the testing operations.

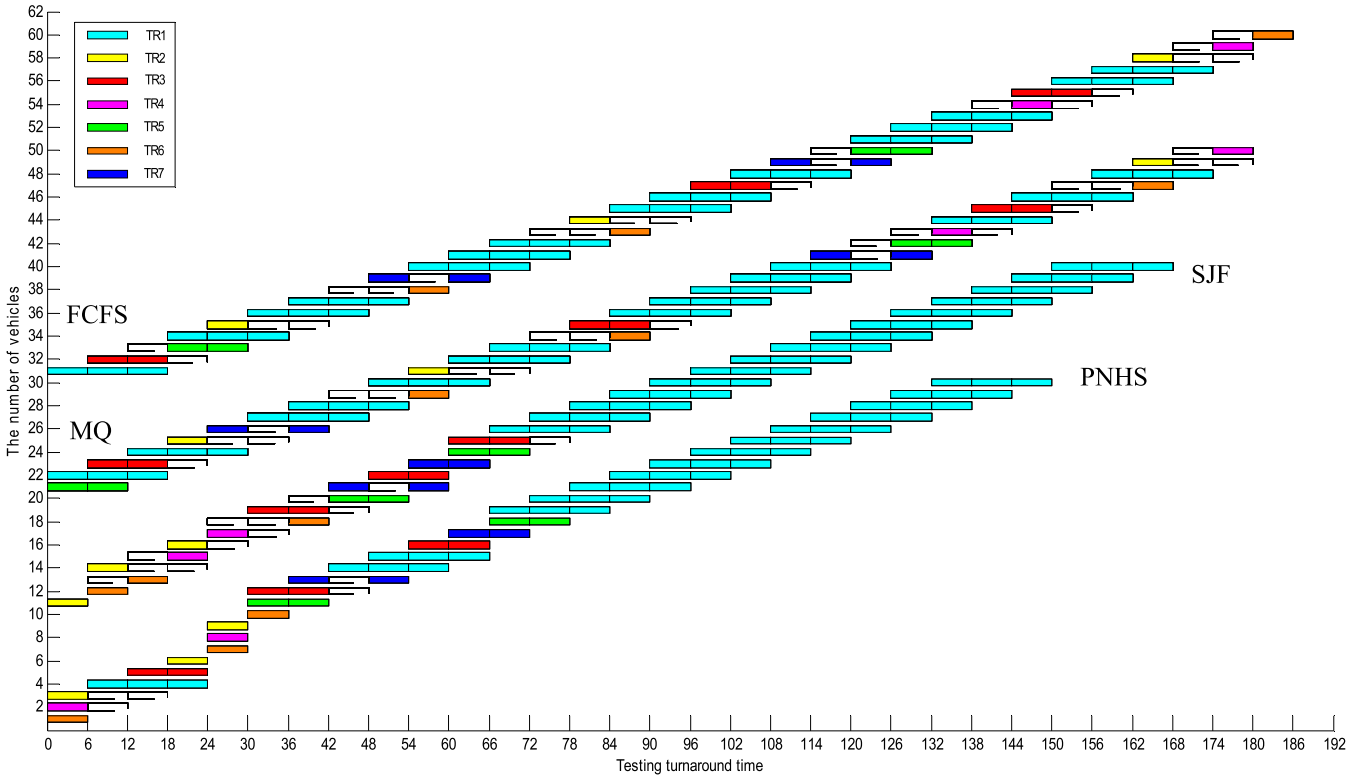


FIGURE 8. Gantt chart of test turnaround time for 30 vehicles using different scheduling strategies.

lists are full. Then, calculate the test turnaround time of each vehicle queue for each operator and update the optimal vehicle queue. The connection strength is updated and then empty all vehiclequeue$_k$ lists. Thus, an iteration is completed. According to the above process, the current number of iterations is updated. When the number of iterations reaches IN_{max} (in this experiment it is set to 2000), the vehicle queue with the shortest test turnaround time is then obtained. It is the optimized vehicle sequence for the testing operations, which is 8→29→28→1→2→14→30→24→5→13→3→25→19→4→6→17→9→20→27→7→10→26→23→22→21→18→16→11→15→12, as shown in Fig. 7(b), where Vehicle 8 marked with a small red flag is the head of the optimal vehicle queue and the tail of the queue is Vehicle 12.

B. ANALYSIS OF THE EFFICIENCY OF THE PROPOSED HEURISTIC ALGORITHM

At present, the main scheduling algorithms used in existing VISs are: First-Come-First-Served (FCFS) strategy, Multi-Queue scheduling (MQ) strategy, and Short-Job-First (SJF) strategy. By FCFS strategy, the vehicles are scheduled by the order of arrival. By MQ strategy, the vehicles to be tested are divided into some groups with each group forming an independent queue, and different queues are set with different priorities based on testing requirements. By SJF strategy, the less the number of testing items requested by a vehicle is, the higher priority is for this vehicle.

In Fig. 8, the test turnaround time for testing these 30 vehicles obtained by these four scheduling strategies is presented, respectively. Also, the Gantt charts of vehicle scheduling operations obtained by using FCFS-, MQ-, SJF-and PNHS-based algorithm are illustrated in Fig. 8. The test turnaround time obtained by the four methods is 186, 180, 168, and 150 minutes, respectively, which shows that the proposed heuristic algorithm outperforms the others.

In order to further verify the efficiency of the proposed PN-based heuristic scheduling (PNHS) algorithm, 60 vehicles with different testing requirements are randomly selected from the test log files for comparison. Then, a number of experiments are done for different number of vehicles. The first experiment is done for the first 30 vehicles, and then, by adding five vehicles into the first test, the second experiment is done for the first 35 vehicles. In this way, for each next experiment, based on the previous experiment, five more vehicles are added. In this way, the last experiment is done for 60 vehicles. For these experiments, the test turnaround time obtained by FCFS, MQ, SJF and PNHS is depicted in Fig. 9.

As shown in Fig. 9, the test turnaround time for scheduling 30 vehicles by the four scheduling strategies is 186, 180, 168, and 144 minutes, respectively. When the number of vehicles is up to 60, the test turnaround time by the four scheduling strategies increases to 372, 354, 318, and 306 minutes, respectively. Compared with the SJF, MQ, and FCFS, the proposed PNHS algorithm shortens the testing time by 12, 48, and 66 minutes, respectively. Generally, when a group

TABLE 7. Simulation results of the 10 experiments and parameters by using the PNHS.

The experiment NO	1	2	3	4	5	6	7	8	9	10	Average
Number of vehicles	30	30	30	30	30	30	30	30	30	30	30
Time spent (sec)	12.55	12.68	12.87	12.95	12.38	12.45	12.64	13.01	13.00	12.92	12.75
Number of Iterations to Reach the Optimal Vehicle Queue	34	28	39	35	25	37	30	33	40	29	33
Test turnaround time	144	144	144	144	144	144	144	144	144	144	144

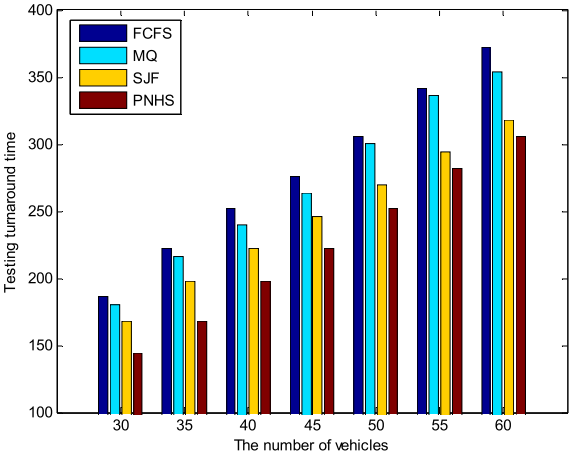


FIGURE 9. Performance comparison of four scheduling strategies.

of vehicles is scheduled as a vehicle queue obtained by the PNHS, the test turnaround time is always shorter than the other three strategies. The above experimental results show the applicability and effectiveness of the PNHS algorithm, which can play a key role in solving the short-term vehicle scheduling problem in the existing VISs.

C. ANALYSIS OF THE RELIABILITY OF THE PROPOSED HEURISTIC ALGORITHM

The proposed heuristic algorithm has a lot of features such as distributed computing, positive feedback, and heuristic search. At the same time, we should also know that the premature convergence of an algorithm to the local optimal solution and the inability to obtain the optimal solution may occur and seriously affect the reliability of the algorithm. Hence, it is necessary to discuss this issue due to that reliability of an algorithm is crucial for its applicability in the practical VISs.

For the purpose of verifying the reliability of the PNHS algorithm, 50 groups of testing requirements are randomly selected from the log file for testing and each group contains 30 vehicles. With the PNHS algorithm, 100 times of simulation have been conducted on each group. The simulation results and parameters including test turnaround time, time for finding the schedule, and the number of iterations to reach the optimal solution for the first time, and so on have been recorded.

One of 50 groups of vehicles' testing requirements are given as:

TR_1, TR_3, TR_1, TR_5, TR_2, TR_1, TR_1, TR_6, TR_7, TR_1,
TR_1, TR_1, TR_6, TR_2, TR_1, TR_1, TR_3, TR_1, TR_7, TR_5,
TR_1, TR_1, TR_1, TR_4, TR_3, TR_1, TR_1, TR_2, TR_4, TR_6.

By using the conventional FCFS scheduling strategy in Shaanxi Automobile Group's VIS, the historical test turnaround time is about 186 minutes. Here, we use the proposed method in this work to do the experiments. Among the 100 experimental results, we randomly select ten of them as listed in Table 7. The statistical data indicates that the reliability of the proposed algorithm can meet the requirements of an actual VIS for vehicle scheduling. For the testing requirements of the same group of vehicles, after 100 rounds of independent experiments, we find that, every time, the test turnaround time is same, i.e., 144 minutes, which shortens 42 minutes compared with the conventional FCFS scheduling strategy. Also, the algorithm has high computational efficiency with the average computing time for finding a solution being 12.75 seconds. At the same time, in the experiment, based on the PNHS algorithm, the number of iterations that reach the optimal solution for the first time is relatively stable, which is between 25 to 40 iterations.

D. IMPACT OF THE RE-INSPECTION RATE ON THE PROPOSED ALGORITHM

In this section, we analyze the effect of PNHS algorithm on the operating efficiency of a VIS under different re-inspection rate. The re-inspection rate indicates how many inspected vehicles are in all vehicles (including non-inspected and inspected vehicles) to be tested. Same as [11], we also perform experiments according to the re-inspection rate of 0, 25%, 50%, 75%, and 100%, respectively. The data set used in this experiment contains 150 groups of testing requirements, and each group consists of 30 vehicles. The data set is extracted from the test file log mentioned above. The number of experimental data groups corresponding to different re-inspection rates are 76, 52, 15, 5, and 2, respectively.

1) NO RE-INSPECTION VEHICLE

In this case, the 76 data sets are selected and required to be simulated at the PN-based VIS model. When the

TABLE 8. Distribution of vehicle position changes.

Re-inspection rate (Number of data sets)	Change of vehicle position before and after optimization									
	<-20	-20~-16	-15~-11	-10~-6	-5~-1	0~5	6~10	11~15	16~20	>20
25% (52)	0%	0%	6.7%	10%	30%	43.3	3.3%	0%	6.7%	0%
50% (15)	0%	0%	18.3%	15%	13.3%	31.7%	5%	6.7%	6.7%	3.3%
75% (5)	0%	3.3%	16.6%	13.3%	30%	16.7%	6.7%	6.7%	0%	6.7%

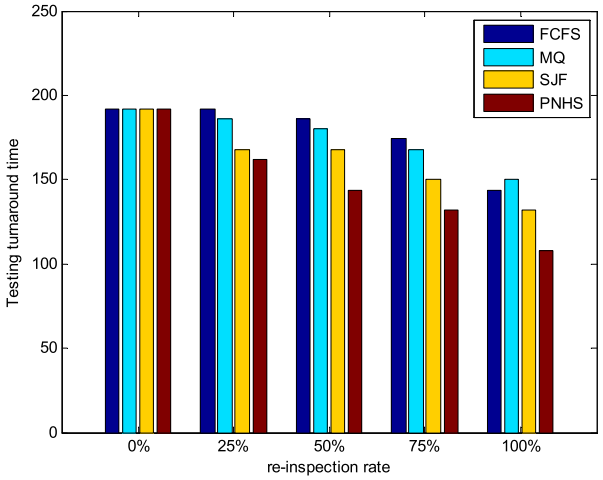


FIGURE 10. Performance comparison of four scheduling algorithms at different re-inspection rates.

re-inspection rate is zero, which means that all of vehicles in this group are non-inspected vehicles and have to complete all of test items for the first time. Due to all vehicles require the full inspection, the performance of MQ, SJF and PNHS algorithms is degraded into that of FCFS. Hence, the average test turnaround time by using different algorithm is same, i.e., 192 minutes.

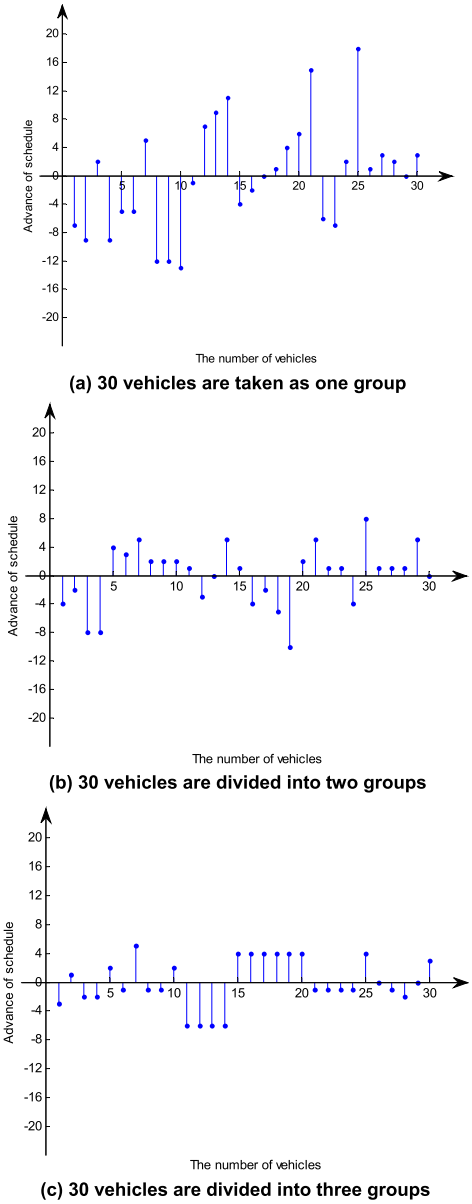
2) THE RE-INSPECTION RATE IS ABOUT 25%

In this case, only about 25% of the vehicles need to be re-inspected, and the rest of the vehicles are non-inspected vehicles and have to conduct a full inspection. By using FCFS, MQ, SJF and PNHS algorithms, the average test turnaround time are 192, 186, 168, and 162 minutes, respectively.

3) THE RE-INSPECTION RATE IS ABOUT 50%, 75%, AND 100%

In these three cases, more than 50% vehicles need to be re-inspected. Fig. 10 shows that the average test turnaround time obtained by adopting the four algorithms decreases with the increase of re-inspection rate. In addition, it is worth noting that, in the practical testing process, the re-inspection rate is less than 50%.

In general, especially compared with the FCFS strategy, a most common used scheduling strategy in existing VISs, when the re-inspection rates are about 0, 25%, and 50%,



(a) 30 vehicles are taken as one group

(b) 30 vehicles are divided into two groups

(c) 30 vehicles are divided into three groups

FIGURE 11. Statistical diagrams of vehicle position changes.

the test turnaround time obtained by the PNHS strategy was reduced by 0%, 15.63% and 22.58%, respectively. It can be concluded that for a given number of vehicles to be tested, the greater the re-inspection rate is, the more productivity increase can be achieved by the proposed PNHS algorithm.

TABLE 9. Position changes of individual vehicles before and after optimization.

Groups	Testing sequence of 30 vehicles	Algorithm	Test turnaround time	Position Fluctuation Variance
One	$TR_1, TR_6, TR_1, TR_1, TR_6, TR_1, TR_4, TR_1, TR_1, TR_1, TR_1, TR_3, TR_1, TR_2, TR_1, TR_1, TR1,$ $TR_1, TR_1, TR_1, TR_7, TR_1, TR_1, TR_1, TR_5, TR_1, TR_1, TR_1, TR_1, TR_1$	FCFS	192	57.7
	$TR_6, TR_4, TR_2, TR_1, TR_3, TR_7, TR_5, TR_1, TR_6, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1,$ $TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1$	PNHS	174	
Two	$TR_1, TR_6, TR_1, TR_1, TR_6, TR_1, TR_4, TR_1, TR_1, TR_1, TR_1, TR_3, TR_1, TR_2, TR_1,$ $TR_1, TR_1, TR_1, TR_1, TR_1, TR_7, TR_1, TR_1, TR_1, TR_5, TR_1, TR_1, TR_1, TR_1, TR_1$	FCFS	192	18.8
	$TR_6, TR_4, TR_1, TR_6, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_3, TR_2,$ $TR_7, TR_5, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1$	PNHS	174	
three	$TR_1, TR_6, TR_1, TR_1, TR_6, TR_1, TR_4, TR_1, TR_1, TR_1,$ $TR_1, TR_3, TR_1, TR_2, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1,$ $TR_7, TR_1, TR_1, TR_1, TR_5, TR_1, TR_1, TR_1, TR_1, TR_1$	FCFS	192	6.9
	$TR_6, TR_4, TR_6, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1$ $TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_3, TR_1, TR_2$ $TR_5, TR_7, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1, TR_1$	PNHS	180	

E. ANALYSIS OF POSITION CHANGE IN A VEHICLE QUEUE

The optimized vehicle sequence based on the PNHS algorithm for short-term vehicle scheduling breaks the traditional mode of testing operations, which schedules the vehicles according to the order of arrival of vehicles. In order to improve the testing efficiency of the overall testing operations and reduce the unnecessary waiting time, some vehicles may be advanced or postponed with respect to their arrival. We refer as to the change in the arrival positions of the vehicles as the change of positions. For individual vehicles, the intensity of changes is one of the important indicators to measure the fairness of a scheduling algorithm. Therefore, under the premise of ensuring the shortest test turnaround time and the highest testing efficiency, the change of the vehicle positions in a queue should be kept as small as possible.

In probability theory, variance is a variable that measures the magnitude of a set of data fluctuations. In practical problems, it is critical to study the variance of the samples, that is, the degree of deviation of the samples. The mean and variance of the data set are calculated as follows:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{14}$$

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (X_i - \mu)^2 \tag{15}$$

where μ is the mean, n is the number of vehicles, X_i is the change of positions of the vehicles in the obtained vehicle queue, and σ^2 is the variance.

In this section, we take a vehicle group consisting of 30 vehicles with a 25% re-inspection rate as an example. By calculating the variance of the position change of each vehicle in the queue and use it to illustrate the fluctuation of vehicle positions before or after applying the PNHS algorithm. In Table 8, some interesting results are illustrated. If we take these 30 vehicles as a group, with FCFS-based and PNHS algorithms, the test turnaround time is 192 and 174 minutes, respectively. Although the PNHS algorithm can shorten the test turnaround time by 18 minutes, the variance of the position change is 57.7 after the vehicle sequence is optimized.

We further carried out experiments using data sets with re-inspection rates of 25%, 50% and 75%. The experimental results are shown in Table 8. In the case with the re-inspection rate being 25%, about 73.3% of vehicles have position changes within -5 and 5, where "-5" and "5" represent postponing and advancing five positions. Only 6.7% of vehicles postpone more than 10 positions. If the re-inspection rate is about 50% and 75%, the vehicles have a position change between -5 and 5 are 45% and 46.7%, respectively.

Actually, if all these vehicles are newly produced vehicle and tested by a team of inspectors, they only care about how to shorten the total test turnaround time. Hence, even if the PNHS algorithm may cause large changes in vehicle positions, especially as the number of vehicles waiting to be tested continues to increase. They still prefer to adopt the PNHS algorithm for short-term vehicle scheduling. However, if the PNHS algorithm is used by a VIS which mainly tests the in-used vehicles, the fairness in vehicle testing and how to decrease the variance of position change should be considered and solved.

Here, we provide a simple method to solve this problem, that is, the original group containing 30 vehicles is further divided into two or three smaller groups, and vehicle order in each small group are independently optimized by using the PNHS algorithm, as shown in Fig. 11. Then, the optimized vehicle queues for the individual groups are released to the system for processing separately. In Table 9, the obtained experimental results by doing so are exhibited. We can observe that when these vehicles are divided into two groups, the test turnaround time is shortened by 18 minutes compared with FCFS, and the variance of the position fluctuation is decreased to 18.8. When these vehicles are divided into three groups, the variance of the position fluctuation is further reduced to 6.9. However, compared with FCFS, the

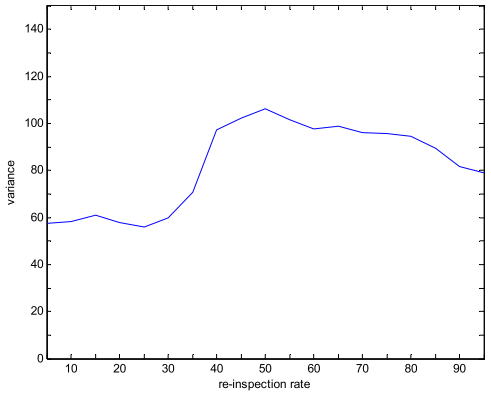


FIGURE 12. Average variance of vehicle position changes for different re-inspection rate.

test turnaround time is also reduced by 12 minutes. This implies that, by dividing the vehicles into smaller groups, the position change can be reduced by paying some cost in the productivity. Thus, when using the PNHS algorithm to generate optimal vehicle queue for vehicle scheduling, it may need to make a trade-off between production efficiency and fairness, which can be done by dividing the vehicles into some groups in using the proposed approach.

At last, we also analyze the impact of the re-inspection rate on the change of vehicle position before and after optimization. For this purpose, we conduct a specialized experiment. The vehicle queues with different re-inspection rate, such as 5%, 10%, 15%, 20%, ..., 95%, 100% are randomly selected. For a re-inspection rate, 10 experiments with 30 vehicles are carried out by using 10 sets of data. For each experiment, the variance of the position fluctuation in the optimized vehicle queues based on the PNHS algorithm is calculated and the average value is shown in Fig. 12. Generally, there is an increasing trend when the re-inspection rate increases. When the re-inspection rate is 50%, the variance reaches is the peak value 106.12. Then, as the re-inspection rate increases, the variance gradually decreases. This phenomenon can be explained as follows. With low re-inspection rate, the number of full inspection vehicles is relatively large. The testing requirements of the full-inspection vehicles are consistent and the testing time is fixed, resulting in that the vehicles in the optimized sequence are generally concentrated. Also, for a vehicle queue with low re-inspection rate, the magnitude of the vehicle testing adjustments is small, i.e., the fluctuation of the vehicle position change is small.

VIII. CONCLUSION

In this work, we investigate the scheduling problem of vehicle inspection systems, where the time for vehicle's movement can be ignored. Aiming to ensure the feasibility of optimal scheduling of such a system, we propose a vehicle testing operational architecture, which basically consists of two stages. In the first stage, the cycle time of a VIS is studied based on the modeling and analysis of ROPN-based model. In the second stage, an ROPN-based heuristic algorithm is proposed for generating an optimal vehicle queue which can be used to schedule the vehicles in a VIS. Then, we use a lot of experiments to demonstrate that the proposed PNHS algorithm can achieve much better production efficiency compared with the conventional FCFS-based scheduling algorithm, and other two algorithms such as MQ-, SJF-based algorithms.

At present, most companies that provide automotive inspection services install more than two VISs in parallel in the facility. Hence, how efficiently schedule such application scenarios is very interesting and challenging. In addition, when there is a high priority vehicle that needs to be added to the already optimized vehicle sequence, it is also a problem to solve. In the future, more work should be done on these issues.

REFERENCES

[1] C. L. Chen, D. C. Yuan, H. H. Shao, and P. Sun, "Modeling and short-term scheduling of synthetic vehicle performance test line," *Control Theory Appl.*, vol. 19, no. 5, pp. 681–688, 2002.
[2] J.-S. Chen and J. C.-H. Pan, "Integer programming models for the re-entrant shop scheduling problems," *Eng. Optim.*, vol. 38, no. 5, pp. 577–592, 2006.
[3] R. Ding, Q. Q. Li, and T. Liang, "Short-term scheduling formulation with decomposition structure for multi-purpose batch plants," *J. Shangdong Univ.*, vol. 45, no. 1, 73-79, 2010.
[4] I. A. Chaudhry and A. A. Khan, "A research survey: Review of flexible job shop scheduling techniques," *Int. Trans. Oper. Res.*, vol. 23, no. 3, pp. 551–591, May 2016.
[5] C. A. Méndez, G. P. Henning, and J. Cerdá, "An MILP continuous-time approach to short-term scheduling of resource-constrained multi-stage flowshop batch facilities," *Comput. Chem. Eng.*, vol. 25, nos. 4–6, pp. 701–711, May 2001.
[6] C.-W. Hui and A. Gupta, "A novel MILP formulation for short-term scheduling of multistage multi-product batch plants," *Comput. Chem. Eng.*, vol. 24, nos. 2–7, pp. 1611–1617, Jul. 2000.
[7] L. F. L. Moro and J. M. Pinto, "Mixed-integer programming approach for short-term crude oil scheduling," *Ind. Eng. Chem. Res.*, vol. 43, no. 1, pp. 85–94, 2004.
[8] S. Moon and A. N. Hrymak, "Mixed-integer linear programming model for short-term scheduling of a special class of multipurpose batch plants," *Ind. Eng. Chem. Res.*, vol. 38, no. 5, pp. 2144–2150, Mar. 1999.
[9] N. Wu, F. Chu, C. Chu, and M. Zhou, "Short-term schedulability analysis of crude oil operations in refinery with oil residency time constraint using Petri nets," *IEEE Trans. Syst., Man, Cybern., C (Appl. Rev.)*, vol. 38, no. 6, pp. 765–778, Nov. 2008.
[10] N. Wu, C. Chu, F. Chu, and M. Zhou, "Schedulability analysis of short-term scheduling for crude oil operations in refinery with oil residency time and charging-tank-switch-overlap constraints," *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 1, pp. 190–204, Jan. 2011.
[11] Y. An, N. Wu, and P. Chen, "Short-term scheduling of vehicle testing system using object petri net," *IEEE Access*, vol. 6, pp. 61317–61330, 2018.
[12] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.
[13] C. Girault and R. Valk, *Petri Nets for Systems Engineering: A Guide to Modeling, Verification, and Applications*. Berlin, Germany: Springer, 2013.
[14] N. Q. Wu and M. C. Zhou, "Modeling, analysis and control of dual-arm cluster tools with residency time constraint and activity time variation based on Petri nets," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 2, pp. 446–454, Apr. 2012.
[15] N. Wu and M. Zhou, "Schedulability analysis and optimal scheduling of dual-arm cluster tools with residency time constraint and activity time variation," *IEEE Trans. Autom. Sci. Eng*, vol. 9, no. 1, pp. 203–209, Jan. 2012.
[16] N. Wu, F. Chu, C. Chu, and M. Zhou, "Petri net modeling and cycle-time analysis of dual-arm cluster tools with wafer revisiting," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 1, pp. 196–207, Jan. 2013.

[17] N. Wu, M. Zhou, L. Bai, and Z. Li, "Short-term scheduling of crude oil operations in refinery with high-fusion-point oil and two transportation pipelines," *Enterprise Inf. Syst.*, vol. 10, no. 6, 581-610, 2016.
[18] Y. Chen, Z. Li, A. Al-Ahmari, N. Wu, and T. Qu, "Deadlock recovery for flexible manufacturing systems modeled with petri nets," *Inf. Sci.*, vol. 381, pp. 290–303, Mar. 2017.
[19] F. Yang, N. Wu, Y. Qao, M. Zhou, and Z. Li, "Scheduling of single-arm cluster tools for an atomic layer deposition process with residency time constraints," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 3, pp. 502–516, Mar. 2017.
[20] S. Zhang, N. Wu, Z. Li, T. Qu, and C. D. Li, "Petri net-based approach to short-term scheduling of crude oil operations with less tank requirement," *Inf. Sci.*, vol. 417, pp. 247–261, Nov. 2017.
[21] Y. S. An, X. M. Zhao, and R. H. Li, "Modeling and simulation of vehicle testing system based on color ed Petri nets," *Comput. Integr. Manuf. Syst.*, vol. 18, no. 9, pp. 1991–2002, Sep. 2012.
[22] L. Bai, N. Wu, Z. Li, and M. Zhou, "Optimal one-wafer cyclic scheduling and buffer space configuration for single-arm multicluster tools with linear topology," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 10, pp. 1456–1467, Oct. 2016.
[23] N. Wu, M. Zhou, and Z. Li, "Short-term scheduling of crude-oil operations: Enhancement of crude-oil operations scheduling using a Petri net-based control-theoretic approach," *IEEE Robot. Autom. Mag.*, vol. 22, no. 2, pp. 64–76, Jun. 2015.
[24] N. Wu, Z. Li, and T. Qu, "Energy efficiency optimization in scheduling crude oil operations of refinery based on linear programming," *J. Cleaner Prod.*, vol. 166, pp. 49–57, Nov. 2017.
[25] O. T. Baruwa, M. A. Piera, and A. Guasch, "Deadlock-free scheduling method for flexible manufacturing systems based on timed colored Petri nets and anytime heuristic search," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 5, pp. 831–846, May 2015.
[26] Q. Zhu, M. Zhou, Y. Qiao, and N. Wu, "Petri net modeling and scheduling of a close-down process for time-constrained single-arm cluster tools," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 3, pp. 389–400, Mar. 2018.
[27] A. Maaoui, A. Abdellatif, A. J. Telmoudi, S. Gattoufi, and L. Nabli, "Multi-site manufacturing system scheduling using Petri nets," in *Proc. 15th Int. Multi-Conf. Syst., Signals Devices (SSD)*, Mar. 2018, pp. 469–474.
[28] C. Jung and T.-E. Lee, "An efficient mixed integer programming model based on timed Petri nets for diverse complex cluster tool scheduling problems," *IEEE Trans. Semicond. Manuf.*, vol. 25, no. 2, pp. 186–199, May 2012.
[29] R. Valk, "Object Petri nets: Using the nets-within-nets paradigm," in *Advanced Course on Petri Nets* (Lectures on Concurrency and Petri Nets), vol. 3098. Berlin, Germany: Springer-Verlag, 2004, pp. 819–848.
[30] N. Wu, M. Zhou, and Z. Li, "Resource-oriented Petri net for deadlock avoidance in flexible assembly systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 1, pp. 56–69, Jan. 2008.
[31] N. Wu and M. Zhou, "Deadlock resolution in automated manufacturing systems with robots," *IEEE Trans. Autom. Sci. Eng.*, vol. 4, no. 3, pp. 474–480, Jul. 2007.
[32] Y. Qiao, N. Wu, F. Yang, M. Zhou, and Q. Zhu, "Wafer sojourn time fluctuation analysis of time-constrained dual-arm cluster tools with wafer revisiting and activity time variation," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 4, pp. 622–636, Apr. 2018.
[33] Q. Zhu, N. Wu, Y. Qiao, and M. Zhou, "Optimal scheduling of complex multi-cluster tools based on timed resource-oriented Petri nets," *IEEE Access*, vol. 4, pp. 2096–2109, 2016.

LINJIAN YANG received the B.S. degree from the Shandong University of Finance and Economics, Shandong, China, in 2001, and the M.S. degree from Yunnan University, Yunnan, China, in 2004. He is currently pursuing the Ph.D. degree with Chang'an University, Xi'an. His research interests include intelligent transportation systems, discrete event systems, and petri net theory and applications.

YISHENG AN (M'16) received the B.S. degree in computer engineering from Shaanxi Normal University, Xi'an, China, in 1995, and the M.S. and Ph.D. degrees in systems engineering from Xi'an Jiaotong University, Xi'an, in 2001 and 2007, respectively. From January 2010 to June 2010, he was a Visiting Scholar with the Department of Computer Engineering, Eastern Washington University, Spokane, WA, USA. He is currently a Professor with the Department of Computer Science and Engineering, School of Information Engineering, Chang'an University, Xi'an. He has authored or coauthored over 60 peer-reviewed journal articles. His research interests include discrete event systems, petri net theory and applications, and intelligent transportation systems. He has served as a Reviewer for a number of journals.

NAIQI WU (M'04–SM'05–F'18) received the B.S. degree in electrical engineering from the Anhui University of Technology, Huainan, China, in 1982, and the M.S. and Ph.D. degrees in systems engineering from Xi'an Jiaotong University, Xi'an, China, in 1985 and 1988, respectively. From 1988 to 1995 and from 1995 to 1998, he was with the Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China, and Shantou University, Shantou, China, respectively. He moved to the Guangdong University of Technology, Guangzhou, China, in 1998. He joined the Macau University of Science and Technology, Taipa, Macau, in 2013, where he is currently a Professor with the Institute of Systems Engineering. He is the author or coauthor of one book, five book chapters, and over 140 peer-reviewed journal articles. His research interests include production planning and scheduling, manufacturing system modeling and control, discrete event systems, petri net theory and applications, intelligent transportation systems, and energy systems. He was an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C, the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, and the Editor-in-Chief of the *Industrial Engineering Journal*. He is also an Associate Editor of *Information Sciences* and the IEEE/CAA JOURNAL OF AUTOMATICA SINICA.

PEI CHEN (S'17) received the B.S. degree from the Xi'an University of Posts and Telecommunications, Shaanxi, China, in 2016. She is currently pursuing the M.S. degree with Chang'an University, Xi'an. Her research interests include discrete event systems, and petri net theory and applications.

HANHAN CHENG received the B.S. degree from the Shandong University of Agricultures, Shandong, China, in 2017. She is currently pursuing the M.S. degree with Chang'an University, Xi'an. Her research interest includes petri net theory and applications.

XIANGMO ZHAO was with Chang'an University for more than 30 years, where he is currently the Vice President and the Director of the Science and Technology Innovation Team of Multi-Sources Traffic Information Sensing and Fusion, Ministry of Education, and a Professor with the School of Information Engineering. He is also the Academic Leader of the State-Level Key Discipline of the Traffic Information Engineering and Control at Chang'an University. He has published more than 200 peer-reviewed articles. His research interests include the Internet of Vehicles, testing of intelligent vehicles, intelligent transportation systems, and nondestructive testing for road infrastructures. He also serves as a member for the State Council's Discipline Evaluation Committee on Transportation Engineering. He received the National SciTech Progress Awards twice for his contribution on promoting the development of indoor vehicle testing technology in China.