Path planning for automated guided vehicle systems with time constraints using timed Petri nets

Zhou He¹, Yuying Dong², Gongchang Ren¹, Chan Gu³ and Zhiwu Li⁴,⁵

Abstract
Automated guided vehicles (AGVs) are extensively used in many applications such as intelligent transportation, logistics, and industrial factories. In this paper, we address the path planning problem for an AGV system (i.e. a team of identical AGVs) with logic and time constraints using Petri nets. We propose a method to model an AGV system and its static environment by timed Petri nets. Combining the structural characteristics of Petri nets and integer linear programming technique, a path planning method is developed to ensure that all task regions are visited by AGVs in time and forbidden regions are always avoided. Finally, simulation studies are presented to show the effectiveness of the proposed path planning methodology.

Keywords
Discrete event system, Petri net, automated guided vehicle, path planning, time constraint

Introduction
With the rapid development of Industrial 4.0, automated guided vehicles (AGVs) have been extensively employed to handle complicated tasks efficiently. The path planning problem of AGV systems has been an active research topic in recent decades. Related works involve single AGV target reachability and multiple AGV obstacle avoidance.¹,² The problem aims to optimize several criteria such as travel distance and energy consumption under certain requirements: deadlock prevention, obstacle avoidance, and multi-task agents.³–⁵

An AGV system can be regarded as a discrete event system (DES) where the environment of the system is divided into several regions (or zones), and the vehicle movement from one region to another can be regarded as a discrete event.⁹–¹² Petri nets (PNs) are a graph-based mathematical tool and are widely applied for modeling,¹³–¹⁸ control,¹⁹–²⁰ and scheduling²¹–²⁵ of DES, including AGV systems, manufacturing systems, and transportation systems. By assuming that some regions have a limited capacity, a deadlock prevention strategy is developed for robot planning in order to avoid collisions.² Hsieh and Kang²⁴ present a method to convert an AGV system into a control-based PN model and develop a control design method for avoiding collisions. Costelha and Lima propose a generalized stochastic PN model to address the robot task planning.⁴ Petri net modeling for robotic assembly and trajectory planning is studied in McCarragher.⁷ In the work of Kloetzer and Mahulea,³⁵ the running environment of an AGV system is partitioned into several regions where each region and the connection between two regions are represented by a place and a transition of PNs, respectively. Then, an online supervising procedure updates the paths whenever an AGV deviates from the planned trajectory. Mahulea and Kloetzer³⁶ propose an approach to find the shortest path of an AGV system with Boolean constraints. By modeling the environment as a state machine and transforming...
the Boolean constraints into a set of linear constraints, an integer linear programming problem (ILPP) is developed to find an optimal trajectory for an AGV system such that the total travel distance is minimized. In order to cooperatively finish some complex tasks, the types of AGVs are assumed to be different in a system. Then, the task allocation problem for AGV systems is studied by using colored PNs. Recently, Luo et al. develop an optimal PN controller for AGV systems in order to avoid collisions under the condition that some events are indistinguishable and uncontrollable due to the limited sensors and actuators.

In real-world systems, tasks usually have time constraints. Therefore, it is important to take the time information into consideration when dealing with the path planning problem for AGV systems. However, there are few works dealing with path planning for AGV systems with time constraints in the framework of DESs. Tanaka and Araki tackle the single machine scheduling problem and present an exact algorithm to find an optimal path. Fanti describe an AGV system using a colored timed PN and develop a control method to avoid collisions and deadlocks. Time windows are assigned to vehicles for the control at urban intersections in order to improve the throughput. The robot rescue problem with time constraints is studied in Geng et al. and a particle swarm optimization method is used to cooperatively finish some complex tasks, the types of AGVs are assumed to be different in a system. Then, the task allocation problem for AGV systems is studied by using colored PNs. Recently, Luo et al. develop an optimal PN controller for AGV systems in order to avoid collisions under the condition that some events are indistinguishable and uncontrollable due to the limited sensors and actuators.

In real-world systems, tasks usually have time constraints. Therefore, it is important to take the time information into consideration when dealing with the path planning problem for AGV systems. However, there are few works dealing with path planning for AGV systems with time constraints in the framework of DESs. Tanaka and Araki tackle the single machine scheduling problem and present an exact algorithm to find an optimal path. Fanti describe an AGV system using a colored timed PN and develop a control method to avoid collisions and deadlocks. Time windows are assigned to vehicles for the control at urban intersections in order to improve the throughput. The robot rescue problem with time constraints is studied in Geng et al. and a particle swarm optimization method is used to cooperatively finish some complex tasks, the types of AGVs are assumed to be different in a system. Then, the task allocation problem for AGV systems is studied by using colored PNs. Recently, Luo et al. develop an optimal PN controller for AGV systems in order to avoid collisions under the condition that some events are indistinguishable and uncontrollable due to the limited sensors and actuators.

In real-world systems, tasks usually have time constraints. Therefore, it is important to take the time information into consideration when dealing with the path planning problem for AGV systems. However, there are few works dealing with path planning for AGV systems with time constraints in the framework of DESs. Tanaka and Araki tackle the single machine scheduling problem and present an exact algorithm to find an optimal path. Fanti describe an AGV system using a colored timed PN and develop a control method to avoid collisions and deadlocks. Time windows are assigned to vehicles for the control at urban intersections in order to improve the throughput. The robot rescue problem with time constraints is studied in Geng et al. and a particle swarm optimization method is used to cooperatively finish some complex tasks, the types of AGVs are assumed to be different in a system. Then, the task allocation problem for AGV systems is studied by using colored PNs. Recently, Luo et al. develop an optimal PN controller for AGV systems in order to avoid collisions under the condition that some events are indistinguishable and uncontrollable due to the limited sensors and actuators.

In real-world systems, tasks usually have time constraints. Therefore, it is important to take the time information into consideration when dealing with the path planning problem for AGV systems. However, there are few works dealing with path planning for AGV systems with time constraints in the framework of DESs. Tanaka and Araki tackle the single machine scheduling problem and present an exact algorithm to find an optimal path. Fanti describe an AGV system using a colored timed PN and develop a control method to avoid collisions and deadlocks. Time windows are assigned to vehicles for the control at urban intersections in order to improve the throughput. The robot rescue problem with time constraints is studied in Geng et al. and a particle swarm optimization method is used to cooperatively finish some complex tasks, the types of AGVs are assumed to be different in a system. Then, the task allocation problem for AGV systems is studied by using colored PNs. Recently, Luo et al. develop an optimal PN controller for AGV systems in order to avoid collisions under the condition that some events are indistinguishable and uncontrollable due to the limited sensors and actuators.

In real-world systems, tasks usually have time constraints. Therefore, it is important to take the time information into consideration when dealing with the path planning problem for AGV systems. However, there are few works dealing with path planning for AGV systems with time constraints in the framework of DESs. Tanaka and Araki tackle the single machine scheduling problem and present an exact algorithm to find an optimal path. Fanti describe an AGV system using a colored timed PN and develop a control method to avoid collisions and deadlocks. Time windows are assigned to vehicles for the control at urban intersections in order to improve the throughput. The robot rescue problem with time constraints is studied in Geng et al. and a particle swarm optimization method is used to cooperatively finish some complex tasks, the types of AGVs are assumed to be different in a system. Then, the task allocation problem for AGV systems is studied by using colored PNs. Recently, Luo et al. develop an optimal PN controller for AGV systems in order to avoid collisions under the condition that some events are indistinguishable and uncontrollable due to the limited sensors and actuators.

In real-world systems, tasks usually have time constraints. Therefore, it is important to take the time information into consideration when dealing with the path planning problem for AGV systems. However, there are few works dealing with path planning for AGV systems with time constraints in the framework of DESs. Tanaka and Araki tackle the single machine scheduling problem and present an exact algorithm to find an optimal path. Fanti describe an AGV system using a colored timed PN and develop a control method to avoid collisions and deadlocks. Time windows are assigned to vehicles for the control at urban intersections in order to improve the throughput. The robot rescue problem with time constraints is studied in Geng et al. and a particle swarm optimization method is used to cooperatively finish some complex tasks, the types of AGVs are assumed to be different in a system. Then, the task allocation problem for AGV systems is studied by using colored PNs. Recently, Luo et al. develop an optimal PN controller for AGV systems in order to avoid collisions under the condition that some events are indistinguishable and uncontrollable due to the limited sensors and actuators.
Definition 1 A timed Petri net (TPN) is a pair $N_d = (N, \theta)$, where:

- $N = (P, T, Pre, Post)$ is a PN,
- $\theta : T \to \mathbb{R}_{>0}$ is a firing delay function that assigns to each transition a non-negative real number, where $\mathbb{R}_{>0}$ denotes the set of nonnegative real numbers.

A TPN $(N, \theta)$ is said to be a timed state machine (TSM) if net $N$ is a state machine. A TSM system is a triple $G = (N, \theta, M)$, where $(N, \theta)$ is a TSM and $M$ is an initial marking.

The environment of an AGV system that is known and static is partitioned into $n$ regions labeled with $\pi_1, \pi_2, \ldots, \pi_n$. We denote the set of regions by $\Pi = \{\pi_1, \ldots, \pi_n\}$. Each region $\pi_i$ is modeled by a place $p_i$ and an event in which a vehicle moves from region $\pi_i$ to its adjacent region $\pi_j$ is modeled by a transition $t_{ij}$. The number of AGVs currently located in region $\pi_i$ can be modeled by the number of tokens in place $p_i$. We associate each transition $t_{ij}$ with a deterministic duration $\theta(t_{ij})$ to represent the AGV movement time from $\pi_i$ to $\pi_j$.

Algorithm 1 returns a TSM system $G = (N, \theta, M)$ for an AGV system and a vector $w \in \mathbb{R}^n_{>0}$ that contains the average distance for traveling between two adjacent regions.

Path requirement: Logic and time constraints

Among all the regions of an AGV system, we are interested in a set of regions that should be visited or avoided, denoted by $\Pi'$ (where $\Pi' \subseteq \Pi$). In particular, we consider two types of interest regions: task regions that should be visited and forbidden regions that should be always avoided. The path requirement for an AGV system under consideration consists of logic and time constraints for the AGVs’ trajectories and the final states. Therefore, we divide the set of interest regions $\Pi'$ into three disjoint subsets $\Pi_a', \Pi_f'$, and $\Pi_v'$ such that $\Pi' = \Pi_a' \cup \Pi_f' \cup \Pi_v'$, where $\Pi_a'$, $\Pi_f'$, and $\Pi_v'$ denote the sets of task regions that should be visited along the trajectory, task regions that should be visited at the final state, and forbidden regions that should be always avoided, respectively.

For each interest region $\pi \in \Pi'$, we define a state function $\Delta(\pi)$ as follows:

$$\Delta : \Pi' \to \{A, F, J\},$$

where

- $\Delta(\pi) = J$ if region $\pi$ should be visited along the trajectory;
- $\Delta(\pi) = F$ if region $\pi$ should be visited at the final state;
- $\Delta(\pi) = A$ if region $\pi$ should be always avoided.

Problem statement

In this paper, we deal with the path planning problem for AGV systems with logic and time constraints. In particular, we consider a team of $k$ identical AGVs moving in a known and static environment with $n$ regions. Given a path requirement $\varphi$ that contains logic
and time constraints discussed in Section “Path planning for AGV systems using PNs”, we aim to find an optimal trajectory of the AGV system such that all task regions are visited by AGVs in time and forbidden regions are always avoided, while the total travel distance is minimized. In this paper, we assume that any region could be visited multiple times by AGVs, except for the forbidden regions. On the other hand, in order to finish some complex tasks, multiple types of AGVs may be needed. This problem is also a challenge issue of path planning and will be investigated in our future work.

It is worth mentioning that an optimal solution for the path planning for AGV systems with logic constraints using PNs was developed in Mahulea and Kloetzer. Nevertheless, the time constraints for AGV systems are not discussed in Mahulea and Kloetzer.36

Path planning for AGV systems using PNs

In this section, we first introduce the method developed in Mahulea and Kloetzer36 to transform the logic constraint into a set of linear constraints. Since the path requirement for AGV systems considered in this paper is different from the one in Mahulea and Kloetzer,36 we propose some methods to transform the logic and time constraints into a set of linear constraints. Finally, we develop an ILPP to obtain an optimal trajectory for an AGV system such that the total travel distance is minimized.

Transformation of logic constraint

Logic constraint on the trajectory.36 In the following, we denote the maximum number of steps for AGVs to complete the tasks by $h$. The trajectory of an AGV system can be represented by a finite sequence of markings $M_0, M_1, \ldots, M_h$ of the TSM system $\mathcal{G}$ such that:

$$
\begin{align*}
M_i &= M_{i-1} + C \cdot \sigma_i, \\
M_{i-1} &= \text{Pre} \cdot \sigma_i \geq 0, \\
T^T \cdot \sigma_i &\leq 1, \\
&i = 1, \ldots, h,
\end{align*}
$$

(4)

where $\sigma_i = [\sigma_i(t_1), \ldots, \sigma_i(t_m)]^T \in \mathbb{N}^m$ is the firing count vector at the $i$-th step. The final constraint of equation (4) enforces that between markings $M_{i-1}$ and $M_i$ each token moves at most through one transition, that is, the team of AGVs advances maximum one region at each step.

For each interest region $\pi_j \in \Pi'$, we define an $n$-component vector $v_{\pi_j} = [v_{\pi_j}(p_1), \ldots, v_{\pi_j}(p_n)]$ with all its elements be zero except the $j$-th element which is equal to one, that is,

$$
v_{\pi_j}(p_i) = \begin{cases} 
1, & \text{if } i = j, \\
0, & \text{else}.
\end{cases}
$$

Therefore, a task region $\pi_j$ is visited at marking $M$ iff $v_{\pi_j} \cdot M \geq 1$ holds.

Since the trajectory of an AGV system is represented by the sequence of $h$ markings $M_1, M_2, \ldots, M_h$ obtained by firing a series of transitions from the initial marking $M_0$, the logic constraint on the trajectory should be considered for all the markings, that is, $M_1, \ldots, M_h$. The following constraint guarantees that an AGV will visit region $\pi_j \in \Pi'$ as required in $\varphi$ along the trajectory:

$$
\begin{align*}
&k \cdot h \geq v_{\pi_j} \cdot (\sum_{i=1}^{h} M_i), \forall \pi_j \in \Pi', \\
&1 \leq v_{\pi_j} \cdot (\sum_{i=1}^{h} M_i), \forall \pi_j \in \Pi'.
\end{align*}
$$

(5)

where $k$ is the total number of AGVs.

Logic constraint on the final state:36 The final state of an AGV system can be represented by the final marking $M_h$, where $h$ is the maximal number of steps for AGVs to complete the tasks. Therefore, the following constraint guarantees that an AGV will visit region $\pi_j \in \Pi'$ as required in $\varphi$ at the final state:

$$
\begin{align*}
&k \geq v_{\pi_j} \cdot M_h, \forall \pi_j \in \Pi', \\
&1 \leq v_{\pi_j} \cdot M_h, \forall \pi_j \in \Pi'.
\end{align*}
$$

(6)

Transformation of logic constraint under considered path requirement

Considering the path planning problem for the AGV system discussed in Section “Problem statement”, we assume that a task region $\pi_j$ should be visited by at least one AGV along the trajectory (if $\pi_j \in \Pi'$) or at the final state (if $\pi_j \in \Pi'$). For each AGV $q (q = 1, \ldots, k)$, we denote by $M_{q,i} = [M_{q,i}(p_1), \ldots, M_{q,i}(p_n)]^T$ the marking of AGV $q$ at the $i$-th step $(i = 1, \ldots, h)$, where $M_{q,i}(p_j) = 1$ if robot $q$ is located in region $\pi_j$ at the $i$-th step and $M_{q,i}(p_j) = 0$ otherwise.

Logic constraint on the trajectory under considered path requirement. For any task region $\pi_j \in \Pi'$ that should be visited by an AGV along the trajectory, the following logic constraint should be satisfied:

$$
\exists q \in \{1, \ldots, k\}, \exists i \in \{1, \ldots, h\}, M_{q,i}(p_j) = 1, \forall \pi_j \in \Pi'.
$$

(7)

As one can observe, constraint (7) is a logic “or” constraint. In the following proposition, we show how to transform constraint (7) into a set of linear constraints.

Proposition 1. The logic constraint on the trajectory (7) can be transformed into a set of linear constraints as follows:
\[
\begin{align*}
    &\begin{cases}
        r_{\pi_j} \cdot \left( \sum_{i=1}^{b} M_{q,i} \right) - h \leq z_j(q) \cdot H, \forall \pi_j \in \Pi', \\
        1 - r_{\pi_j} \cdot \left( \sum_{i=1}^{b} M_{q,i} \right) \leq z_j(q) \cdot H, \forall \pi_j \in \Pi'.
    \end{cases} \\
    &\begin{cases}
        z_j(1) + z_j(2) + \ldots + z_j(k) \leq k - 1, \\
        z_j(1), \ldots, z_j(k) \in \{0, 1\}, \\
        q = 1, \ldots, k.
    \end{cases}
\end{align*}
\]

where \( H \in \mathbb{R}_{>0} \) is a constant satisfying

\[
H > \max \left\{ r_{\pi_j} \cdot \left( \sum_{i=1}^{b} M_{q,i} \right) - |h|, |1 - r_{\pi_j} \cdot \left( \sum_{i=1}^{b} M_{q,i} \right) | \right\}.
\]

**Proof:** Constraints (8b) and (8c) impose that at least one variable \( z_j \) should be zero. By assuming that \( z_j(r) = 0 \) for \( r \in \{1, \ldots, k\} \), constraint (8a) can be simplified as follows:

\[
\begin{align*}
    &\begin{cases}
        r_{\pi_j} \cdot \left( \sum_{i=1}^{b} M_{q,i} \right) \leq h, \forall \pi_j \in \Pi', \\
        1 \leq r_{\pi_j} \cdot \left( \sum_{i=1}^{b} M_{q,i} \right), \forall \pi_j \in \Pi'.
    \end{cases}
\end{align*}
\]

According to equation (5), it guarantees that AGV \( r \) will visit the task region \( \pi_j \) along the trajectory. For any other variable \( z_j(q) = 1 \) \((q \neq r)\), we have the following condition:

\[
\begin{align*}
    &\begin{cases}
        r_{\pi_j} \cdot \left( \sum_{i=1}^{b} M_{q,i} \right) - h \leq H, \forall \pi_j \in \Pi', \\
        1 - r_{\pi_j} \cdot \left( \sum_{i=1}^{b} M_{q,i} \right) \leq H, \forall \pi_j \in \Pi'.
    \end{cases}
\end{align*}
\]

Since \( H \) is a large enough number as defined in equation (9), condition (11) becomes redundant and imposes no constraint on the trajectory of AGV \( q \) \((q \neq r)\). Therefore, the logic constraint on the trajectory (7) is implemented by Proposition 1.

**Logic constraint on the final state under the considered path requirement.** For any task region \( \pi_j \in \Pi' \) that should be visited by an AGV at the final step \( h \) (i.e., the final marking \( M_h \)), the following logic constraint should be satisfied:

\[
\exists q \in \{1, \ldots, k\}, M_{q,h(p)} = 1, \forall \pi_j \in \Pi'.
\]

Similarly, we can transform the logic constraint (12) on the final state into a set of linear constraints by the following proposition.

**Proposition 2.** The logic constraint (12) on the final state can be transformed into a set of linear constraints as follows:

\[
\begin{align*}
    &\begin{cases}
        r_{\pi_j} \cdot M_{q,h} - 1 \leq z_j(q) \cdot H, \forall \pi_j \in \Pi', \\
        1 - r_{\pi_j} \cdot M_{q,h} \leq z_j(q) \cdot H, \forall \pi_j \in \Pi', \\
        z_j(1) + z_j(2) + \ldots + z_j(k) \leq k - 1, \\
        z_j(1), \ldots, z_j(k) \in \{0, 1\}, \\
        q = 1, \ldots, k.
    \end{cases}
\end{align*}
\]

where \( H \in \mathbb{R}_{>0} \) is a constant satisfying

\[
H > |r_{\pi_j} \cdot M_{q,h} - 1|.
\]

**Proof:** This proof is analogous to the proof of Proposition 1. Constraints (13b) and (13c) impose that at least one variable \( z_j \) should be zero. If \( z_j(r) = 0 \) \((r \in \{1, \ldots, k\}\) holds, condition (13a) can be simplified as \( r_{\pi_j} \cdot M_{q,h} = 1 \), which implies that AGV \( r \) will finally be located in region \( \pi_j \). Otherwise, condition (13b) becomes redundant since \( H \) is a large enough number as defined in equation (14). Therefore, the logic constraint on the final state (12) is implemented by Proposition 2.

**Logic constraint on avoidance.** It is easy to verify that the logic constraint on a forbidden region \( \pi_j \in \Pi'' \) that should be avoided by AGVs can be implemented by the following constraint:

\[
r_{\pi_j} \cdot \left( \sum_{q=1,i=1}^{b} M_{q,i} \right) = 0, \forall \pi_j \in \Pi''.
\]

**Transformation of time constraint**

In this subsection, we propose a method to transform the time constraint of an AGV system into a set of linear constraints. For any task region \( \pi_j \in \Pi' \cup \Pi'' \) with a predefined time window \([e_{\pi_j}, l_{\pi_j}]\), at least one AGV should visit \( \pi_j \) along the trajectory (if \( \pi_j \in \Pi' \)) or at the final state (if \( \pi_j \in \Pi'' \)) in the time window \([e_{\pi_j}, l_{\pi_j}]\). In other words, place \( p_j \) of the TSM system obtained by Algorithm 1 should be marked in the predefined time window at least once. In the following, we denote by \( \tau_{\pi_j} \) the time instant when an AGV \( q \) visits a region at the \( i \)-th step, where \( q \in \{1, \ldots, k\} \) and \( i \in \{1, \ldots, h\} \).

**Time constraint on the trajectory:** For any task region \( \pi_j \in \Pi'' \) that should be visited along the trajectory with time constraint \([e_{\pi_j}, l_{\pi_j}]\), the following condition should be satisfied:

\[
\exists q \in \{1, \ldots, k\}, \exists i \in \{1, \ldots, h\}, M_{q,i(p)} = 1, \tau_{\pi_j,i} \in [e_{\pi_j}, l_{\pi_j}], \forall \pi_j \in \Pi''.
\]

We can transform the time constraint on the trajectory (16) into a set of linear constraints by the following proposition.

**Proposition 3.** The time constraint on the trajectory (16) can be transformed into a set of linear constraints as follows:
\[
\begin{aligned}
  &\{ r_{\pi_j} \cdot (\sum_{i=1}^{h} M_{q,i}) - h \leq z(q) \cdot H, \forall \pi_j \in \Pi' \} \\
  &1 - r_{\pi_j} \cdot (\sum_{i=1}^{h} M_{q,i}) \leq z(q) \cdot H, \forall \pi_j \in \Pi' \\
  &z(1) + z(2) + \ldots + z(k) \leq k - 1, \\
  &q = 1, \ldots, k, i = 1, \ldots, h, \\
  &\tau_{q,i} = \tau_{q,i-1} + \theta \cdot \sigma_{q,i}, \\
  &\tau_{q,0} = 0, \\
  &e_{\pi_j} - \tau_{q,i} - l_{\pi_j} \leq 1 - M_{q,i}(p_j) + o(q) \cdot H, \forall \pi_j \in \Pi' \\
  &\tau_{q,i} - l_{\pi_j} \leq 1 - M_{q,i}(p_j) + o(q) \cdot H, \forall \pi_j \in \Pi' \\
  &z(q) \leq o(q), \\
  &o(1) + \ldots + o(k) \leq k - 1, \\
  &o(1) + o(2) + \ldots + o(k) \leq k - 1, \\
  &H > \max\{ r_{\pi_j} \cdot (\sum_{i=1}^{h} M_{q,i}) - h, |\tau_{q,i} - e_{\pi_j}|, |1 - r_{\pi_j} \cdot (\sum_{i=1}^{h} M_{q,i})|, |\tau_{q,i} - l_{\pi_j}| \}.
\end{aligned}
\]

where \( \theta = [\theta(t_1), \ldots, \theta(t_m)] \) is the firing delay vector that represents the movement time between adjacent regions, \( \sigma_{q,i} = [\sigma_{q,i}(t_1), \ldots, \sigma_{q,i}(t_m)]^T \in \mathbb{N}^m \) is the firing count vector of AGV \( q \) at the \( i \)-th step, and \( [e_{\pi_j}, l_{\pi_j}] \) is a prescribed time window for task region \( \pi_j \). Parameter \( H \in \mathbb{R}_{\geq 0} \) is a constant satisfying

\[
H > \max\{ r_{\pi_j} \cdot (\sum_{i=1}^{h} M_{q,i}) - h, |\tau_{q,i} - e_{\pi_j}|, |1 - r_{\pi_j} \cdot (\sum_{i=1}^{h} M_{q,i})|, |\tau_{q,i} - l_{\pi_j}| \}.
\]

Proof: According to Proposition 1, constraint (17a) imposes that region \( \pi_j \) will be visited by at least one AGV along the trajectory. Constraint (17b) represents the time instant when an AGV \( q \) visits a region at each step.

For any AGV \( r \) that does not appear in the region \( \pi_j \), that is, \( z(r) = 1 \) and \( M_{r,i}(p_j) = 0 \) for \( i = 1, \ldots, h \), constraint (17c) can be simplified as follows:

\[
\begin{aligned}
  &e_{\pi_j} - \tau_{q,i} \leq H, \forall \pi_j \in \Pi' \\
  &\tau_{q,i} - l_{\pi_j} \leq H, \forall \pi_j \in \Pi' \\
  &e_{\pi_j} - \tau_{q,i} - l_{\pi_j} \leq 1 - M_{q,i}(p_j) + o(q) \cdot H, \forall \pi_j \in \Pi' \\
  &\tau_{q,i} - l_{\pi_j} \leq 1 - M_{q,i}(p_j) + o(q) \cdot H, \forall \pi_j \in \Pi' \\
  &z(q) \leq o(q), \\
  &o(1) + \ldots + o(k) \leq k - 1, \\
  &o(1) + o(2) + \ldots + o(k) \leq k - 1, \\
  &H > \max\{ r_{\pi_j} \cdot (\sum_{i=1}^{h} M_{q,i}) - l_{\pi_j}, |\tau_{q,i} - e_{\pi_j}|, |\tau_{q,i} - l_{\pi_j}| \}.
\end{aligned}
\]

where \( H \in \mathbb{R}_{\geq 0} \) is a constant satisfying

\[
H > \max\{ r_{\pi_j} \cdot (\sum_{i=1}^{h} M_{q,i}) - l_{\pi_j}, |\tau_{q,i} - e_{\pi_j}|, |\tau_{q,i} - l_{\pi_j}| \}.
\]

Proof: According to Proposition 2, constraint (23a) imposes that region \( \pi_j \) will be visited by at least one AGV at the final state. Constraint (23b) represents the time instant when an AGV \( q \) visits a region at each step. By Proposition 3, constraint (23c) enforces that at least one AGV will visit the region \( \pi_j \) at the final step \( h \) in the time window \([e_{\pi_j}, l_{\pi_j}]\). Therefore, the time constraint on the trajectory (16) is implemented by Proposition 4.

Time constraint on the final state: For any task region \( \pi_j \in \Pi' \) that should be visited at the final state with time constraint \([e_{\pi_j}, l_{\pi_j}]\), the following condition should be satisfied:

\[
\exists q \in \{1, \ldots, k\}, M_{q,h}(p_j) = 1, \tau_{q,h} \in [e_{\pi_j}, l_{\pi_j}], \forall \pi_j \in \Pi'.
\]

Similarly, we can transform the time constraint (22) on the final state into a set of linear constraints by the following proposition.

Proposition 4. The time constraint on the final state (22) can be transformed into a set of linear constraints as follows:

\[
\begin{aligned}
  &e_{\pi_j} - \tau_{q,i} - l_{\pi_j} \leq 0, \forall \pi_j \in \Pi' \\
  &\tau_{q,i} - l_{\pi_j} \leq 0, \forall \pi_j \in \Pi' \\
  &e_{\pi_j} - \tau_{q,i} - l_{\pi_j} \leq 1 - M_{q,i}(p_j) + o(q) \cdot H, \forall \pi_j \in \Pi' \\
  &\tau_{q,i} - l_{\pi_j} \leq 1 - M_{q,i}(p_j) + o(q) \cdot H, \forall \pi_j \in \Pi' \\
  &z(q) \leq o(q), \\
  &o(1) + \ldots + o(k) \leq k - 1, \\
  &o(1) + o(2) + \ldots + o(k) \leq k - 1,
\end{aligned}
\]

which guarantees that the time instant when the region \( \pi_j \) is visited by an AGV \( q \) at the \( i \)-th step will satisfy the prescribed time constraint; Situation 2) if \( o(q) = 0 \) holds, condition (20) is equivalent to (19) that is redundant. It means that AGV \( q \) will visit the region \( \pi_j \) without a specified time constraint. The final constraint of (17c) ensures that at least one AGV will visit region \( \pi_j \) in the time window \([e_{\pi_j}, l_{\pi_j}]\). Therefore, the time constraint on the trajectory (16) is implemented by Proposition 3.
Path Planning for AGV systems with an ILPP

Combining above results, we present an ILPP to solve the path planning problem for an AGV system with time constraints as follows:

\[
\min_w \sum_{q=1}^{h} \sum_{i=1}^{h} \sigma_{q,i} \cdot M_{q,i} - M_{q,i-1} + C \cdot \sigma_{q,i},
\]

\[
M_{q,i} = \text{Pre} \cdot \sigma_{q,i} + \sigma_{q,i} \geq 0, \quad 1 \leq \sigma_{q,i} \leq 1,
\]

\[
1 - \sigma_{q,i} \cdot \sum_{i=1}^{h} M_{q,i} \leq h \leq z(q) \cdot H, \quad \forall \pi_j \in \Pi',
\]

\[
e_{q,i} - \tau_{q,i} \leq [1 - M_{q,i}(p_j) + o_j(q)] \cdot H, \quad \forall \pi_j \in \Pi',
\]

\[
\tau_{q,i} - l_{q,i} \leq [1 - M_{q,i}(p_j) + o_j(q)] \cdot H, \quad \forall \pi_j \in \Pi',
\]

\[
\tau_{q,i} = \tau_{q,i-1} + \theta \cdot \sigma_{q,i},
\]

\[
\tau_{q,0} = 0,
\]

where \( H \in \mathbb{R}_{>0} \) is a constant satisfying

\[
H > \max \left\{ |v_{q,i} \cdot \sum_{i=1}^{h} M_{q,i} - h|, |v_{q,i} \cdot M_{q,i} - 1|, |1 - v_{q,i} \cdot \sum_{i=0}^{h} M_{q,i}|, |\tau_{q,i} - e_{q,i}|, |\tau_{q,i} - l_{q,i}| \right\},
\]

Note that parameter \( h \) in (25) is a predesigned number that represents the maximum number of steps for AGVs to complete the tasks, which is lower than or equal to the total number of transitions, that is, \( h \leq m \). The objective function in (25) accounts for the total travel distance of all AGVs in the system. Constraint (25a) from (4) enforces the correctness of the firing count vector. Note that constraint (25a-3) imposes that an AGV can advance maximum one region at each step. Constraints (25b) (resp., (25c)), (25d), (25f), and (25g) jointly impose the logic and time constraints on the trajectory (resp., on the final state) for the task regions. Constraint (25e) imposes the avoidance requirement for the forbidden regions. The optimal solution \( \sigma_{q,i} \) (\( q = 1, \ldots, k, i = 1, \ldots, h \)) of ILPP (25) represents the sequence of the firing count vector of the TSM system \( G \) that corresponds to the trajectory of each AGV.

**Computational complexity.** The optimization problem (25) is a standard ILPP whose computational complexity is well known as NP-hard. However, it can be efficiently solved by some off-the-shelf tools such as LINGO, MATLAB, and CPLEX. The computational burden of an ILPP is usually characterized by the number of variables and constraints. Problem (25) has \( 3kh + 2k(|\Pi'| + |\Pi'|) \) variables and \( 2kh + (3kh + 3k + 2) \cdot (|\Pi'| + |\Pi'| + |\Pi'| + 2k) \) constraints in total, where |\Pi'|, |\Pi'|, and |\Pi'| denote the numbers of regions that should be visited along the trajectory, regions that should be visited at the final state, and regions that should be always avoided, respectively.

**Simulation results.** In this section, the illustration of the developed approach for the path planning problem of AGV systems with time constraints is made on an example that is taken from Mahulea and Kloetzer. The environment of the system is partitioned by using a constrained triangular decomposition and has 30 regions as shown in Figure 1, that is, \( \Pi = \{ \pi_1, \pi_2, \ldots, \pi_{30} \} \). There are three AGVs (i.e. \( k = 3 \)) moving in the environment whose initial positions are marked by circles in Figure 1, that is, regions \( \pi_5 \) (green circle), \( \pi_{16} \) (blue circle), and \( \pi_{17} \) (yellow circle) each contain one AGV.

By Algorithm 1, we obtain a TSM system \( G \) that consists of 86 transitions and 30 places, as depicted in Figure 2. For exemplification purposes, we consider unitary distance vector \( w = 1 \) and unitary firing delay...
vector $\theta = 1$. The firing delays of transitions are omitted in Figure 2 for the sake of brevity and readability.

Consider a set of interest regions as follows:

- $\Pi^a = \{ \pi_1 \}$,

it implies that regions $\pi_{10}$, $\pi_{12}$, $\pi_{21}$, $\pi_{22}$, and $\pi_{23}$ should be visited along the trajectory, region $\pi_1$ should be avoided (red region in Figure 1), and regions $\pi_7$, $\pi_{12}$, and $\pi_{29}$ should be occupied at the final state, that is, AGVs will stop in these regions. We assume that regions $\pi_{10}$, $\pi_{12}$, and $\pi_{21}$ must be visited in the time windows $[1, 2]$, $[1, 3]$, and $[2, 4]$, respectively. Therefore, the path requirement of the AGV system can be represented as follows:

$$\varphi = (\pi_1, A), (\pi_7, F), (\pi_8, J), (\pi_{10}, J, [1, 2]), (\pi_{12}, F, [1, 3]), (\pi_{21}, J, [2, 4]), (\pi_{22}, J, (\pi_{23}, J, (\pi_{29}, F).$$

The developed approach is implemented by MATLAB with the ILPP toolbox YALMIP and Optimal Solver Gurobi on a Windows operating system with i5-7500 CPU 3.40 GHz and 4 GB memory. By solving ILPP (25) with a maximum number of steps $h = 16$ (in approximately 0.82 s), we obtain a series of firing sequences that can be translated into the trajectories of the AGVs, as depicted in Figure 1. Regions $\pi_7$, $\pi_8$, and $\pi_{10}$ are visited by the green AGV, region $\pi_{12}$ is visited by the blue AGV, and regions $\pi_{21}$, $\pi_{22}$, $\pi_{23}$, and $\pi_{29}$ are visited by the yellow AGV.
It should be noticed that all the AGVs are assumed to be identical, that is, a single type of AGVs is considered. From a practical point of view, multiple types of AGVs may be needed to cooperatively finish some complex tasks. Our future work focuses on the path planning problem with time constraints for AGV systems that contain multiple types of AGVs.

### Author’s Note
Zhou He and Gongchang Ren are also affiliated with Shaanxi Key Laboratory of Complex System Control and Intelligent Information Processing, Xi’an University of Technology, Xi’an China.

### Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

### Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the National Natural Science Foundation of China under Grant Nos. 61803246 and 62003201, the China Postdoctoral Science Foundation under Grant No. 2019M663608, the Shaanxi Provincial Natural Science Foundation under Grant No. 2020JQ-733, the Science and Technology Plan of Weiyang District under Grant No 201914, and the Shaanxi Key Laboratory of Complex System Control and Intelligent Information Processing under Grant No. SKL2020CP03, Xi’an University of Technology.

### ORCID iD
Zhiwu Li https://orcid.org/0000-0003-2486-1246

### References
1. Geng N, Gong DW and Zhang Y. Petri net model and its optimization for the problem of robot rescue path planning. In: Proceedings of the international conference on intelligent computing, 2017, pp. 551–563. Cham: Springer.
2. Kloetzer M, Mahulea C and Colom JM. Petri net approach for deadlock prevention in robot planning. In: Proceedings of IEEE 18th conference on emerging technologies & factory automation (ETFA 2013), Cagliari, Italy, 10–13 September 2013, pp. 1–4. IEEE.
3. Costella H. Modelling, analysis and execution of multirobot tasks using Petri nets. In: Proceedings of the...
international conference on autonomous agents and multiagent systems, Estoril, Portugal, 12–16 May 2008, pp. 1187–1190. Publisher: INESC-ID.

4. Costelha H and Lima P. Robot task plan representation by Petri nets: modelling, identification, analysis and execution. *Auton Robots* 2012; 33(4): 337–360.

5. Kloetzler M and Mahulea C. Accomplish multi-robot tasks via Petri net models. In: *Proceedings of the IEEE international conference on automation science and engineering*, Gothenburg, Sweden, 24–28 August 2015, pp. 304–309. IEEE.

6. Kloetzler M and Mahulea C. An assembly problem with mobile robots. In: *Proceedings of the IEEE emerging technology and factory automation*, Barcelona, Spain, 16–19 September 2014, pp. 16–19. IEEE.

7. McCarragher B.J. Petri net modelling for robotic assembly and trajectory planning. *IEEE Trans Ind Electron* 1994; 41(6): 631–640.

8. Mohanta JC, Parhi DR and Patel SK. Path planning strategy for autonomous mobile robot navigation using Petri-GA optimization. *Comput Electr Eng* 2011; 37(6): 1058–1070.

9. Luo JL, Wan YX, Wu WM, et al. Optimal Petri-net controller for avoiding collisions in a class of automated guided vehicle systems. *IEEE Trans Intell Trans Syst* 2019; 99: 1–12.

10. Singh N, Sarnagadharan P and Pal PK. AGV scheduling for automated material distribution: a case study. *J Intell Manuf* 2011; 22(2): 219–228.

11. Luo JL, Ni HJ and Zhou MC. Control program design for automated guided vehicle systems via Petri nets. *IEEE Trans Syst Man Cybern* 2015; 45(1): 44–55.

12. Gu C, Wang X and Li ZW. Synthesis of supervisory control with partial observation on normal state tree structures. *IEEE Trans Autom Sci Eng* 2019; 16(2): 984–997.

13. Liu GY, Li P, Li ZW, et al. Robust deadlock control for automated manufacturing systems with unreliable resources based on Petri net reachability graphs. *IEEE Trans Syst Man Cybern* 2019; 49(7): 1371–1385.

14. Li ZW, Wu NQ and Zhou MC. Deadlock control of automated manufacturing systems based on Petri nets-a literature review. *IEEE Trans Syst Man Cybern C* 2012; 42(4): 437–462.

15. Cong X, Wang A, Chen YF, et al. Most permissive liveness-enforcing petri net supervisors for discrete event systems via linear monitors. *ISA Trans* 2019; 92: 145–154.

16. He Z, Li ZW and Giua A. Cycle time optimization of deterministic timed weighted marked graphs by transformation. *IEEE Trans Control Syst Technol* 2017; 25(4): 1318–1330.

17. Liu GY, Li ZW, Barkaoui K, et al. Robustness of deadlock control for a class of Petri nets with unreliable resources. *Inform Sci* 2013; 235: 259–279.

18. Coman D, Bizdoaca N, Petrisor A, et al. Modeling and analysis of soccer robot tasks using generalized stochastic Petri nets. *Solid State Phenom* 2010; 166: 133–138.

19. Ma ZY, Zhu GH and Li ZW. Marking estimation in Petri nets using hierarchical basis reachability graphs. *IEEE Trans Autom Control*. Epub ahead of print March 2020. DOI: 10.1109/TAC.2020.2983088.

20. Ma ZY, Li ZW and Giua A. Characterization of admissible marking sets in Petri nets with conflicts and synchronizations. *IEEE Trans Autom Control* 2017; 62(3): 1329–1341.

21. Tong Y, Li ZW, Seatzu C, et al. Verification of state-based opacity using Petri nets. *IEEE Trans Autom Control* 2017; 62(6): 2823–2837.

22. Tong Y, Li ZW and Giua A. On the equivalence of observation structures for petri net generators. *IEEE Trans Autom Control* 2016; 61(9): 2448–2462.

23. Zhu GH, Li ZW, Wu N, et al. Fault identification of discrete event systems modeled by Petri nets with unobservable transitions. *IEEE Trans Syst Man Cybern* 2019; 49(2): 333–345.

24. Zhu GH, Li ZW and Wu N. Model-based fault identification of discrete event systems using partially observed Petri nets. *Automatica* 2018; 96: 201–212.

25. Ran N, Su HY and Wang SG. An improved approach to test diagnosability of bounded Petri nets. *IEEE CAA J Autom Syst Sci* 2017; 4(2): 297–303.

26. Luo JL, Wu W, Zhou MC, et al. Structural controller for logical expression of linear constraints on Petri nets. *IEEE Trans Autom Control* 2020; 65(1): 397–403.

27. Ma ZY, Li ZW and Giua A. Marking estimation in a class of time labeled Petri nets. *IEEE Trans Autom Control* 2020; 65(2): 493–506.

28. Pan H, Jing X, Sun W, et al. A bioinspired dynamics-based adaptive tracking control for nonlinear suspension systems. *IEEE Trans Control Syst Technol* 2017; 26(3): 903–914.

29. Pan H and Sun W. Nonlinear output feedback finite-time control for vehicle active suspension systems. *IEEE Trans Ind Inform* 2018; 15(4): 2073–2082.

30. He Z, Li ZW and Giua A. Optimization of deterministic timed weighted marked graphs. *IEEE Trans Autom Sci Eng* 2017; 14(2): 1084–1095.

31. Yang FJ, Wu NQ, Qiao Y, et al. Polynomial approach to optimal one-wafer cyclic scheduling of treelike hybrid multi-cluster tools via Petri nets. *IEEE CAA J Autom Sin* 2018; 5(1): 270–280.

32. He Z, Liu M, Ma ZY, et al. An improved approach for marking optimization of timed weighted marked graphs. *Discrete Event Dyn Syst* 2019; 29(2): 127–143.

33. He Z, Li ZW and Giua A. Performance optimization for timed weighted marked graphs under infinite server semantics. *IEEE Trans Autom Control* 2018; 63(8): 2573–2580.

34. Hsieh S and Kang MY. Developing AGVS Petri net control models from flowpath nets. *J Manuf Syst* 1998; 17(4): 237–250.

35. Kloetzler M and Mahulea C. A Petri net based approach for multi-robot path planning. *Discrete Event Dyn Syst* 2014; 24(4): 417–445.

36. Mahulea C and Kloetzler M. Robot planning based on boolean specifications using Petri net models. *IEEE Trans Autom Control* 2017; 63(7): 2218–2225.

37. Wang X, Rui F and Hu H. Task allocation policy for UGV systems using colored Petri nets. In: *Proceedings of annual American control conference (ACC)*, Milwaukee, WI, 27–29 June 2018, pp. 3050–3055. IEEE.

38. Tanaka S and Araki M. An exact algorithm for the single-machine total weighted tardiness problem with sequence-dependent setup times. *Comput Oper Res* 2013; 40(1): 344–352.

39. Fanti MP. A deadlock avoidance strategy for AGV systems modelled by coloured Petri nets. In: *Proceedings of International workshop discrete event Systems*, Zaragoza, Spain, 2002, pp. 61–66. IEEE.
40. Meng X and Cassandras CG. Optimal control of autonomous vehicles for non-stop signalized intersection crossing. In: Proceedings of IEEE conference on decision and control, Miami Beach, FL, 17–19 December 2018, pp. 6988–6993. IEEE.

41. Willms AR and Yang SX. Real-time robot path planning via a distance-propagating dynamic system with obstacle clearance. IEEE Trans Cybern 2008; 38(3): 884–893.

42. Dallal E, Colombo A, Del-Veccchio D, et al. Supervisory control for collision avoidance in vehicular networks using discrete event abstractions. Discrete Event Dyn Syst 2016; 27(1): 1–44.

43. Morita K. The legal firing sequence problem of Petri nets with state machine structure. In: Proceedings of IEEE international symposium on circuits and systems, Atlanta, GA, 15 May 1996, pp. 64–67. IEEE.

44. Silva M, Teruel E and Colom JM. Linear algebraic and linear programming techniques for the analysis of place or transition net systems. In: Advanced course on Petri Nets. Lecture on Petri Nets: basic models. Vol. 1491, Berlin, Heidelberg: Springer, 1996, pp. 309–371.

45. Cabasino MP, Giua A and Seatzu C. Identification of Petri nets from knowledge of their language. Discrete Event Dyn Syst 2007; 17(4): 447–474.

46. Berg M, Cheong O and Kreveld M. Computational geometry: algorithms and applications. Berlin, Heidelberg: Springer, 2000, pp. 333–334.

47. YALMIP solver, 2019, yalmip.github.io/allsolvers/.