A control of collision and deadlock avoidance for automated guided vehicles with a fault-tolerance capability

Citation for published version (APA):
Li, Q., Pogromskiy, A. Y., Adriaansen, A. C., & Udding, J. T. (2016). A control of collision and deadlock avoidance for automated guided vehicles with a fault-tolerance capability. International Journal of Advanced Robotic Systems, 13(2), [64]. https://doi.org/10.5772/62685

DOI:
10.5772/62685

Document status and date:
Published: 07/04/2016

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication
A Control of Collision and Deadlock Avoidance for Automated Guided Vehicles with a Fault-tolerance Capability

Qin Li1*, Alexander Pogromsky2,3, Teun Adriaansen4 and Jan Tijmen Udding2,ª

1 Statoil Research Centre, Porsgrunn, Norway
2 Department of Mechanical Engineering, Eindhoven University of Technology, The Netherlands
3 Department of Control Systems and Industrial Robotics, Saint Petersburg National Research University of Information Technologies, Mechanics and Optics, Russia
4 ASML, The Netherlands
ªRetired
*Corresponding author(s) E-mail: qinli01@gmail.com

Received 01 September 2015; Accepted 20 February 2016

Abstract

Based on a novel discrete-event zone-control model, in our previous papers [1, 2], we presented a time-efficient traffic control for automated guided vehicle (AGV) systems to exclude inter-vehicle collisions and system deadlocks, together with a case study on container terminals. The traffic control allows each vehicle in an AGV system to freely choose its routes for any finite sequence of zone-to-zone transportation tasks and the routes can be constructed in an online fashion. In this paper, we extended our previous results with two practical goals: (1) to increase the utilization of the workspace area by reducing the minimal-allowed area of each zone; (2) to avoid vehicle collisions and deadlocks with the occurrence of vehicle breakdowns. To achieve the first goal, we include one extra vehicle event that allows each vehicle to probe further ahead while it is moving on the guide-path. This leads to an extension of our previous discrete-event model and traffic control rules, which are presented in the first part of the paper. The second part of the paper concerns the second goal, for which an emergency traffic control scheme is designed as supplementary to the normal traffic control rules. As in our previous papers, the improved model and traffic control are applied to a simulation of quayside container trans-shipment at container terminals; our simulation results are compared with those from two interesting works in the literature.

Keywords Automated Guided Vehicle, Deadlock Avoidance, Fault-tolerant Traffic Control, Container Terminal

1. Introduction

Automated guided vehicles (AGVs) are used as a transportation means in many industrial fields. Taking advantage of their flexibility in routing and scaling, AGV systems were traditionally employed in automated manufacturing systems (AMSs) as material handling systems (MHSs) to transport pieces or parts among various workstations.
Recently, AGV systems have extended their popularity to other applications, such as material transportation in warehouses and container transshipment at container terminals. See, e.g., [3, 4] for comprehensive surveys of the research on the design of AGV systems.

As there are normally multiple vehicles in AGV systems, traffic control is necessary to resolve motion conflicts among the vehicles. In particular, vehicle collisions and deadlocks should be prevented to avoid substantial production interruptions. Furthermore, for an AGV system with a large number of vehicles in some applications, effective and efficient traffic control is also important to achieve desirable system performance (see, e.g., [5] for a list of performance measures).

Traditionally and still currently, AGVs move along a predefined guide-path (wired, magnetic, etc.) system connecting key locations in the workspace, which eases the motion control and guidance of the vehicles, yet sets limitations on the flexibility and scalability of the system. The literature and industrial practice have witnessed a growing interest in developing free-ranging AGV systems, in which the vehicles can choose any possible path to move between two locations. This type of system is expected by some researchers to provide shorter transportation distances (via free routing), better utilization of the workspace and more flexible deadlock and congestion avoidance mechanisms. However, performance comparisons between free-ranging AGV systems and those with a fixed guide-path are very rarely seen [6]; besides, it seems that convincing conclusions have not yet been drawn on which type would perform better with dense vehicles (i.e., a large number of vehicles in a limited area). In this paper, we choose to focus on a study of AGV systems with a fixed guide-path.

Several types of flow topologies/configurations have been proposed and investigated for the guide-path of the AGV systems. Each of them has its pros and cons and should be chosen based on specific applications. All the flow configurations contain a collection of lanes, or sequences or path segments (either unidirectional or bidirectional), connecting machines, work stations, pick-up and delivery points or other fixed infrastructures. In what is called conventional configuration, lanes can intersect each other like in a normal guide-path. In single loop configuration, AGVs travel in a fixed (usually unidirectional) loop. The advantage is that the traffic is easy to control; however, each vehicle has to travel the complete loop to visit the same point again and a vehicle breakdown can block the entire system. The tandem configuration consists of non-overlapping single loops and uses transfer stations to move materials from one loop to another. As with the single loop layout, it also requires little or no traffic control (if only one vehicle serves one loop); hence, some researchers concluded that tandem layout performs better in large systems (see, e.g., [7]). However, any single loop is a bottleneck to the whole system; besides, extra space and equipment are required for the transfer stations. Lastly, similar to a tandem configuration, a segmented flow configuration contains one or more zones, each of which is divided into non-overlapping segments served by a single vehicle. Some papers also showed good performance (see, e.g., [8]) by using this type of guide-path, but it shares the same drawbacks as with the tandem configuration. The reader is referred to [3, 4] for more details on the classification and design of the guide-path. This paper discusses the traffic control of AGV systems with conventional guide-path (as aforementioned, the traffic control of the other types of guide-path configurations is heavily reduced or even fully removed).

The most popular traffic control strategy for AGV systems is called zone-control (see [9]), in which lanes in the guide-path are composed of a number of zones. By this strategy, inter-vehicle collision avoidance becomes straightforward by only allowing one vehicle to visit one zone at a time. A key traffic control issue is then to keep the vehicles away from deadlocks that may be caused by the vehicles’ competition for the occupation of the zones.

Historically, the study on deadlock problems started in the development of operating systems [10, 11] and database systems [12] in computer science. In general, three strategies have been employed to address deadlocks: deadlock prevention, deadlock avoidance and deadlock detection and resolution [12]. The goal of deadlock prevention is to design rules such that any deadlock is precluded before all processes set off. In a deadlock avoidance strategy, online control is adopted such that a resource is allocated to a process only if a deadlock will not be introduced. Lastly, a deadlock detection and resolution strategy allows deadlock to happen. When a deadlock is detected, certain online rules are used to resolve the deadlock and recover the system. During the past two decades, deadlock problems have been attracting intensive research effort for the flexible manufacturing systems (FMSs), in which AGV systems play an important part. In particular, [13-18] are among many other works that investigate deadlock avoidance using Petri net formalism. The proposed methods either exclude the occurrence of deadlocks by preventing some necessary conditions, or detect and resolve deadlocks when they actually take place [19], [20] and [21] proposed deadlock avoidance policies based on the graphic characterization of deadlocks and so-called restricted deadlocks. In [22], a variation of the Banker’s algorithm was developed, which prevents deadlocks by allowing only “ordered” vehicle mission trips. The conservativeness of this algorithm is somewhat reduced in [23] and [24] at the cost of higher complexity. There are some other methods existing in the literature. [25] provides a generalized algebraic deadlock avoidance policy that ensures the absence of deadlocks by restricting the operation of the system in a subspace satisfying a set of linear inequalities. Using a graph theoretic approach, [26] presented a deadlock avoidance algorithm that consists of
a quasi two-step resource allocation policy and a cyclic deadlock detection procedure. [27] offers a logic-algebraic approach for deadlock prevention by solving a constraint satisfaction problem. In addition, many works have been dedicated to guaranteeing deadlock avoidance by using smart routing algorithms (sometimes called conflict-free routing) [28-31, 24]. Note also that recent years have witnessed an increasing research interest in traffic control problems for the AGV systems at automated container terminals, as the volume of goods transported via sea ports has soared in booming international trade. [32] presented several procedures to detect and recover from deadlocks caused by the interaction between the AGV system and other container handling equipment systems. [33] renders a path reservation method to prevent in advance the deadlocks and collisions of AGVs. [34] suggested a traffic control approach based on the concept of "semaphore" borrowed from computer science. In addition, inspired by systems, the potential room for performance improvement conservative with acceptable complexities, or too computationally expensive under the pursuit of least restriction. Very recently, there have been some research activities aiming at maximally permissive deadlock avoidance algorithms with complexity concerns by using the resource allocation system theory [37] and Petri net formalism [38]. However, it seems that there is still quite some distance from the current status of the development to the practical applications of these optimal algorithms for large-scale AGV systems (with tens or even hundreds of vehicles). To the authors’ best knowledge, the deadlock avoidance algorithms proposed in [35, 36, 1, 2] have the smallest computational complexities, which only quadratically increase with the number of vehicles. Note that the small complexities of these algorithms partially come from the limited applicability to guide-path systems with only unidirectional lanes. However, for large-scale AGV systems, the potential room for performance improvement using bidirectional guide-path systems shrinks with increasing vehicle interferences [39, 29] and comes with a sharp rise of complexity of traffic management. Furthermore, one also notes that the traffic control with bidirectional guide-path in general depends on the scheduling and routing of the vehicles, whereas the traffic control algorithm proposed in our previous work [2] has been proven able to avoid deadlocks with no restrictions to routing of the vehicles, as long as each vehicle visits a finite sequence of zones from its initial location to a depot (where it can stay without blocking other vehicles). This free-routing property is definitely a big advantage in achieving high system performance. Note also that among the algorithms proposed in [35, 36, 2], only the one in [2] is provably correct in guaranteeing the absence of deadlocks with the help of a formal event-based modelling. (The problems with the algorithms in [35] have been pointed out in [21]; in [36] the authors did not obtain a complete solution for what they call "multi-cycle deadlocks").

Failures can degrade the performance of or even paralyse an industrial system. Fault-tolerance is, therefore, one of the key properties that make an industrial system practically useful. In the literature of deadlock avoidance, several studies have been made on fault-tolerant deadlock avoidance for some types of automated manufacturing systems modelled by Petri nets [40, 41]. In addition, some fault-tolerant routing algorithms have been developed for computer networks and for communication subsystems on integrated circuits [42, 43]. For deadlock avoidance for AGV systems, however, there is hardly any existing work that has the management of vehicle faults as a focus. In principle, for the AGV systems with free-ranging vehicles, vehicle breakdowns may be handled by letting the other (healthy) vehicles view the faulty vehicles as obstacles that can be steered clear of with their (i.e., the healthy vehicles’) built-in obstacle avoidance algorithms. However, detailed discussion and formal proofs for collision and deadlock avoidance in this direction are rarely seen in the literature. For the AGV systems with guide-paths, vehicle breakdowns can be catastrophic for certain types of configuration of guide-path. For example, in the single loop configuration, where AGVs travel in a fixed (usually unidirectional) loop, a vehicle breakdown can block the whole system; in the tandem configuration, which consists of non-overlapping single loops, each single loop is a bottleneck to the entire system; the same problem can happen to the segmented flow configuration with each of non-overlapping segments served by a single vehicle. For other types of guide-path, the AGV system might still be able to function after some vehicle breakdowns; however, a partial or complete re-dispatching, re-scheduling and/or re-rerouting may be necessary to ensure acceptable system performance. With most existing deadlock avoidance approaches, the adjustment has to be responsible for the deadlock avoidance of the AGV system as well.

In this paper, we follow the line of development in [1, 2] to design traffic control algorithms for AGV systems that guarantee the avoidance of collision and deadlocks with full freedom of routing. We have two specific goals in mind: (1) increase the utilization of the workspace area by reducing the minimally allowed area of each zone; and (2) design an emergency traffic control scheme to avoid vehicle collisions and deadlocks with the presence of vehicle breakdowns. Both these goals are motivated by practical concerns from general applications. The key to the realization of the first goal is adding one extra event to the event-based model previously presented in [2]. The triggering of
this event allows a vehicle to probe the availability of one more zone ahead on its path than before. For applications with a large number of zones, this can lead to substantial saving of space. The second goal is clearly designed to improve the robustness of traffic control to emergency situations, which is rarely addressed in literature, at least in a formal manner. Note that the achievement of the first goal also gives some leeway to that of the second, for which we enlarge the first zone of each lane so that it contains some extra space that can be split out, if needed, into a buffer zone when a vehicle breakdown occurs.

The paper is organized as follows: the extended discrete-event zone-control model is introduced in Section 2. The traffic control rules that ensure the avoidance of collision and deadlocks in a fault-free situation are presented in Section 3. A performance study of an AGV system incorporating the proposed traffic control is mentioned in Section 4. The emergency traffic control rules that guarantee the avoidance of collision and deadlocks with vehicle breakdowns are presented in Section 5. Concluding remarks are given in Section 6. All the technical proofs are included in the appendix of the paper.

2. Zone-control Modelling

In this section, we present the zone-control model that defines the structure of the guide-path\(^1\) and the event-driven vehicle behaviour. In particular, at the end of Section 2.2, we will show how our first goal of increasing the utilization of the workspace area is achieved by this extension to our previous work [2].

2.1 Building blocks of the zone-based guide-path

The guide-path contains lanes, crossings and depots. Lanes are composed of zones and each depot is a (special) zone. For any assigned transportation task, a vehicle does nothing but move from one zone to another. Some of the content in this section can be found in [2], but is still presented here for completeness.

2.1.1 Lane and zone

A lane is a finite sequence of zones. Vehicles moving in a lane must visit the zones according to their order in the sequence. In the general discussion that follows, we assume that there are \( M, M \in \mathbb{N} \) (\( \mathbb{N} \) denotes the set of natural numbers), lanes in the network and the lane \( i \) has \( m_i \) \( m_i \in \mathbb{N} \) zones. In addition, we use \( c_j \) to denote the \( k \) th zone of the lane \( i \). In particular, we call the first zone and the last zone of a lane the starting zone (SZ) and ending zone (EZ) of the lane, respectively. In practice, a zone should be large enough to accommodate at least one vehicle.

A depot is a zone that can accommodate any number of vehicles. We emphasize that a depot is not a zone of any lane; or, in other words, any zone on each lane is a non-depot zone. Each depot is affiliated with at least one entry-lane and one exit-lane. Physically, an entry-lane (resp. exit-lane) of a depot is a lane that allows a vehicle to move inwards (or outwards) from the depot.

We use \( c \) to denote the set of all zones in the guide-path.

2.1.2 Crossing

A crossing is physically a junction that joins multiple lanes. Specifically, crossing \( i \) is affiliated with a set of in-lanes, denoted by \( \mathcal{J}_i \) and a set of out-lanes, denoted by \( \mathcal{O}_i \). Each lane in \( \mathcal{J}_i \) has a (non-empty) set of neighbouring lanes, which is a subset of \( \mathcal{O}_i \). Vehicles can move from an in-lane of a crossing onto any of its neighbouring lanes by passing the crossing. A zone pair \((c_1, c_2)\) is thus called a crossing-passing zone pair of crossing \( i \) if \( c_1 \) is the EZ of a lane \( j \) in \( \mathcal{J}_i \) and \( c_2 \) is the SZ of a neighbouring lane of the lane \( j \). We denote the collection of all the crossing-passing zone pairs of crossing \( i \) by \( \mathcal{R}_i \). (Hence \( \mathcal{R}_i \) characterizes all possible ways of passing the crossing \( i \)). For each zone pair \((c_1, c_2) \in \mathcal{R}_i \), there is a subset (maybe empty) of \( \mathcal{R}_i \), denoted by \( \mathcal{X}(c_1, c_2) \), in which the zone pairs are called the conflicting crossing-passing zone pairs of \((c_1, c_2)\) at the crossing \( i \). Physically, if \((c_1, c_2) \in \mathcal{X}(c_1, c_2)\), then a vehicle passing the crossing \( i \) by moving from \( c_1 \) to \( c_2 \) can collide with another vehicle passing the same crossing by moving from \( c_1 \) to \( c_2 \). For this reason, for any crossing \( i \), it is considered that \((c_1, c_2) \in \mathcal{X}(c_1, c_2)\) if and only if \((c_1, c_2) \in \mathcal{X}(c_1, c_2)\) for each crossing \( i \), we call the set of EZs of all its in-lanes at-crossing zones of the crossing \( i \). A zone is called an off-crossing zone if it is not an at-crossing zone of any crossing.

2.1.3 Assumptions on the guide-path

Upon the layout of the guide-path, we impose the following guide-path layout assumptions:

\[ (A1) \ c_i^1 \neq c_k^2 \text{ if either } i \neq k \text{ or } k \neq k. \]

\[ (A2) \text{ Each lane is either an in-lane of a unique crossing or an entry-lane of a unique depot, but cannot be both.} \]

\[ (A3) \text{ Each lane can be either an exit-lane of at most one depot or an out-lane of at most one crossing, but cannot be both the exit-lane of a depot and the out-lane of a crossing.} \]

\[ (A4) \text{ Each lane has at least two zones, with the following exceptions (where it can have only one zone): (i) it is an entry- or exit-lane of a depot, (ii) it is an out-lane of a crossing and there is at most one in-lane of the crossing that has it as a neighbouring lane and the in-lane is different from it and (iii) it is neither an out-lane of a crossing nor an exit-lane of a depot.} \]

---

\(^1\) We called it road network previously in [1, 2], but want to align with the literature here.
Figure 1 illustrates an example of such a guide-path.

In the above assumptions, (A1) says that each lane neither self-intersects nor intersects with any other lane; (A2) implies that each lane connects to a depot or another lane (via a crossing); (A4) is key to the deadlock avoidance by the traffic control presented later. In addition, note that (A3) does not exclude the case that a lane is neither an exit-lane of a depot nor an out-lane of a crossing (e.g., the lane depicted at the top left corner of Figure 1).

2.1.4 Neighbouring zone

It is important to know, when a vehicle is in a zone, which zones the vehicle can move into next. For this purpose we define the concept neighbouring zone, which intuitively follows the definitions of lanes and depots. The set of neighbouring zones of each depot consists of the SZs of all its exit-lanes. The neighbouring zone of a non-EZ zone \( c_i \), \( i \in \{1,2,\ldots,M\} \), \( k \in \{1,2,\ldots,m_k-1\} \), is \( c_{i\downarrow k}^{-} \). The neighbouring zone of the EZ of any entry-lane of a depot is the depot. The neighboring zones of the EZ of an in-lane of a crossing are the SZs of its neighbouring lanes. We use \( Y_c \) to denote the set of neighbouring zones of zone \( c \). In addition, for any \( c \in \mathcal{C} \), define \( \Upsilon_c = \{ c' \in \mathcal{C} : c \in Y_{c'} \} \).

2.2 Vehicle state and event

The state of a vehicle indicates its current location and intention of movement with respect to the guide-path. Each vehicle can have three types of states, as follows:

1. in \( c_1 \) with the next zone \( c_2 \), denoted by \( (c_1,c_1,c_2) \);
2. moving from \( c_1 \) to \( c_2 \), denoted by \( (c_1,c_2,c_2) \);
3. moving from \( c_1 \) to \( c_2 \) via \( c_2 \), denoted by \( (c_1,c_2,c_2) \);

where \( c_1 \) can be any zone in \( \mathcal{C} \), \( c_2 \) is one of the neighbouring zones of \( c_1 \) (i.e., \( c_2 \in Y_{c_1} \)) and \( c_3 \) is a neighbouring zone of \( c_2 \) but different from \( c_1 \). In all three cases, \( c_1 \) and \( c_2 \) are called, respectively, the current zone and the next zone of the vehicle.

From an implementation point of view, the body of a vehicle is supposed to be entirely inside its current zone in case (1) and may span across its current zone and the next zone in cases (2) and (3).

To ease the presentation, we define three sets that collect all possible vehicle states in each of the above three types:

\[
\begin{align*}
S_1 &= \{(c_1,c_1,c_2) \in \mathcal{C}^3 : c_2 \in Y_{c_1}\}, \\
S_2 &= \{(c_1,c_2,c_2) \in \mathcal{C}^3 : c_2 \in Y_{c_1}\}, \\
S_3 &= \{(c_1,c_2,c_3) \in \mathcal{C}^3 : c_2 \in Y_{c_1}, c_3 \in Y_{c_2}, \{c_3\} \}.
\end{align*}
\]

In addition, we let \( S \) be the union of \( S_1, S_2 \) and \( S_3 \). Note that, by Lemma 2 (see Section A.1), \( c_2, c_3 \) and \( c_4 \) are distinct in the definitions of \( S_1, S_2 \) and \( S_3 \) and thus the three sets are disjoint. For any \( s \in S \), we will sometimes use \( s_1, s_2, s_3 \) to denote its first, second and third elements respectively.

Each vehicle has an initial state representing its beginning location and intention of movement. When the system is running, the state of each vehicle will change only at the occurrence of events. In other words, an event is responsible for a state transition that maps a vehicle state to another. There are three events that vehicles trigger for their normal operations: “leave”, “arrive” and “go ahead”. To simplify the presentation, we use LEA, ARR and GOA to denote the three events respectively. At any point in time, each vehicle can trigger at most one event; the following rules prescribe how the three events change the state (denoted by \( s \)) of a vehicle:

\[
\begin{align*}
&\text{a. if } s=(c_1,c_1,c_2) \in S_1, \text{ then the only feasible event for } s \text{ is an LEA, which changes } s \text{ to } (c_1,c_2,c_2) \in S_2; \\
&\text{b. if } s=(c_1,c_2,c_2) \in S_2, \text{ then both an ARR and a GOA are feasible for } s: \text{ (i) an ARR changes } s \text{ to } (c_2,c_2,c_2) \in S_1, \text{ where } c_2 \text{ can be any zone in } Y_{c_2}; \text{ (ii) a GOA changes } s \text{ to } (c_1,c_2,c_2) \in S_3, \text{ where } c_3 \text{ can be any zone in } Y_{c_3}; \\
&\text{c. if } s=(c_1,c_2,c_2) \in S_3, \text{ then the only feasible event for } s \text{ is an ARR, which changes } s \text{ to } (c_2,c_2,c_2) \in S_3. 
\end{align*}
\]

Note that the ARR and GOA in (b) are the events that involve new zones in the state transitions and thus directly relate to the implementation of the routes of the vehicles.

---

2 According to Lemma 1 in the Section A.1, we have defined the neighboring zones for any zone in \( \mathcal{C} \).

3 We will introduce some emergency events later when presenting the emergency traffic control of the system with vehicle breakdowns. Without causing any confusion, we mean normal events when we say events hereafter.
The state transitions by the three events are illustrated in Figure 2. In a less formal and much simpler way, we sometimes say that:

- in (a), the vehicle leaves $c_1$ by the LEA;
- in (b), the vehicle goes ahead to pass $c_2$ by the GOA;
- in (b) and (c), the vehicle arrives at $c_3$ by the ARRs.

![Figure 2. The state transitions by the three events](image)

Here we give a simple example of how each vehicle is supposed to utilize these events to change its state while moving on the guide-path. Consider a vehicle that has the state $(c_1,c_2,c_3)$ (i.e., it is in the zone $c_1$) and wants to be in $c_3 \in Y_c$. This can be achieved by first leaving $c_1$, then going ahead to pass $c_2$, arrive at $c_3$, and finally arrive at $c_4$ (i.e., by the event sequence $\text{LEA} \rightarrow \text{GOA} \rightarrow \text{ARR} \rightarrow \text{ARR}^*$).

We should emphasize that, although an event is feasible for a particular vehicle state, it does not mean that the event is always allowed to occur for the state. The traffic rules that will be presented in the following section are used to permit or inhibit events such that vehicle collisions are avoided and system deadlock can never happen. An interesting point to mention here is that an ARR is always allowed to occur by the traffic rules. Thus, the freedom in selecting the zone $c_3$ with the ARR in case (b) allows a vehicle to alter its route even if it has been predefined, or establish it online in an incremental fashion: a vehicle only needs to pick $c_3$ the succeeding zone on its route, before it arrives at $c_3$. The selection of the zone $c_3$ with the GOA event in (b) is also free; the only difference is that a GOA event can be prohibited by the traffic rules. Later on, we will see that there are no restrictions imposed on a route assignment before each transportation trip either. This free-routing property is a distinct advantage of the proposed traffic control.

From another perspective, considering the zones as resources that vehicles need to take up for their transportation trips, the above events, together with the traffic control, can also be seen to dictate the resource allocation for the overall system (in section 2.3, we will see that this is directly related to the states of zones). Specifically, in (a) the vehicle is allocated $c_1$ by the LEA, yet has not released $c_2$ by the GOA in (b) the vehicle is allocated $c_3$ yet has released neither $c_3$ nor $c_2$, the ARRs in both (b) and (c) release the zone $c_4$. Because of the last point, it is required that the vehicle has already physically moved completely out of $c_1$ and is inside $c_2$ when the ARRs are triggered. In addition, for high system performance it is good to have the ARRs triggered (so that $c_1$ is released) immediately after the vehicle fully enters $c_2$. From case (c), we see that this implies that the vehicle is not physically occupying the zone $c_3$ when its state is $(c_1,c_2,c_3)$, as the zone $c_3$ is large enough to accommodate the entire body of the vehicle.

One may notice that, compared with the discrete-event model presented in our previous work [2] (or in [1] in a different form) the driving idea behind all the modifications is the addition of the event GOA, which extends the behaviour of the vehicles. In fact, previously only two types of vehicle states were defined, i.e., the states in $S_1$ and $S_2$ and the event-driven state transitions shown in the dashed box in Figure 2.

Let us now explain the improvement brought about by the present and generalized model by using a walk-through of a typical scenario in which a vehicle tries to pass a zone. Consider a vehicle trying to pass a zone $c_3$ with its current state $(c_1,c_2,c_3)$ and suppose it knows the next zone (after passing $c_3$) on its route is $c_4 \in Y_c$. What will happen with the previous model is the following: the vehicle will have to trigger an ARR to arrive at $c_2$ first ($c_1$ will then be released). As stated above, this requires the body of the vehicle to be fully inside $c_2$. The vehicle will then try to trigger an LEA to leave $c_2$ (so that its state can be changed to $(c_2,c_3,c_4)$). If the LEA is permitted (by the traffic control), the vehicle will keep its speed and move on, passing $c_2$; otherwise, it has to break and make a full stop with its body still completely inside $c_2$. Clearly, we need the length of zone $c_2$ to be at least $L_{\text{vehicle}} + L_{\text{break}}$ in case the LEA is denied, where $L_{\text{vehicle}}$ is the length of the vehicle and $L_{\text{break}}$ is the distance for the vehicle to make a full stop from its maximum speed. Now, for the same scenario, let us follow the vehicle’s behaviour in the framework of the present model. First, the vehicle will try to trigger a GOA, which changes its state to $(c_1,c_2,c_3)$. Importantly, note that the vehicle can do this even before physically entering the zone $c_2$. If the GOA is permitted (the zone $c_2$ is then allocated to the vehicle), it will move on, passing $c_2$; otherwise, it will need to break and fully stop inside $c_2$. It is easily seen that in this case the length of $c_2$ will need only to be larger than $\max(L_{\text{vehicle}}, L_{\text{break}})$. For a guide-path with a large number of zones, this improvement can be substantial if the breaking distance $L_{\text{break}}$ is not negligible with respect to $L_{\text{vehicle}}$ (e.g., with the container-carrying vehicles at container terminals). With a closer look, one may notice that the saving of required space as the vehicle passes $c_2$ is essentially due to the early can-pass probing via the trial of the GOA.

---

4 Of course, this is not the only event sequence that can be used to realize the goal.
2.3 Zone states

Before closing this section, we introduce some properties of the zones that are directly related to the states of vehicles. Each zone can have two states: "occupied" and "available". In particular, a depot is always available. A non-depot zone is occupied if it is allocated to a vehicle; it should not be allocated to another one before being released. Put more formally, a non-depot zone is said to be occupied by a vehicle, with state $s$, when (1) it is the current zone of the vehicle if $s \in S_1$; or (2) it appears in the expression of $s$ if $s \in S_2 \cup S_3$. A non-depot zone is said to be occupied if it is occupied by at least one vehicle. A non-depot zone is available if it is not occupied. It is obvious that the states of zones change only at the occurrence of vehicle events. It is part of the responsibility of the traffic control to ensure that a zone is occupied by at most one vehicle, so that vehicle collisions are avoided.

To ease our presentation, we sometimes use colours to represent different statuses (not sets) of the zones. Specifically,

a. Any zone $c$ is white $\iff c$ is available;

b. Any non-depot zone $c$ is black $\iff c$ is occupied by only one vehicle with the state either $(c,c,c') \in S_1$, $(c',c,c) \in S_2$ or $(c,c',c') \in S_3$.

c. Any non-depot zone $c$ is grey $\iff c$ is occupied by only one vehicle with the state either $(c,c,c') \in S_1$, $(c,c',c') \in S_2$ or $(c',c,c') \in S_3$.

where, in (b) and (c), $c'$ and $c''$ can be any zones that make sense.

One can see that a zone is black if it is occupied by a vehicle that is either in the zone or will be able to be in the zone after triggering one or two ARRs (recall that ARRs are always allowed). A zone is grey if it is occupied by a vehicle that will be able to release the zone after triggering a couple of ARRs.

It is easy to show that, if each non-depot zone is occupied by at most one vehicle, then any zone must be either white (when available), black or grey (when occupied).

Finally, we say that a vehicle starts occupying a zone by an event if the zone was not occupied by the vehicle before the state transition, caused by the event, but is occupied by the vehicle afterwards. It can be shown that such an event can be either an LEA or a GOA but not an ARR (see Lemma 4 in Section A.1).

3. Traffic Control

In this section, we will propose a traffic control system that guarantees the absence of (inter-vehicle) collisions and deadlocks. One may notice that the traffic control here is slightly more complex than that previously proposed in [1, 2], mainly due to the added event GOA and vehicle states in the set $S_3$. Basically, the traffic control permits a vehicle to trigger an ARR whenever it wishes to do so, but requires it to check with some specified traffic rules when it intends to trigger either an LEA or a GOA.

Since there are different traffic rules for different situations, we now introduce a classification of the LEA and GOA events. An LEA (or GOA) is called an off-crossing LEA (or off-crossing GOA) if it causes a vehicle to leave (GOA: go ahead to pass) an off-crossing zone; is called an at-crossing LEA (or at-crossing GOA) at a crossing if it causes a vehicle to leave (GOA: go ahead to pass) an at-crossing zone of the crossing. An event is called a crossing-passing event at a crossing if it is either an at-crossing LEA or an at-crossing GOA at the crossing. An LEA or GOA is called a depot-passing event at a depot if it causes a vehicle to either leave or go ahead to pass the depot. Clearly, a depot-passing event cannot be a crossing-passing event.

3.1 Inter-vehicle collision avoidance

The requirement for collision avoidance can be roughly stated as the following: first, any non-depot zone cannot be occupied by two or more vehicles at the same time. Second, it must be excluded that two vehicles are passing a crossing in a conflicting way or will be doing so due to the lack of further restrictions. More specifically, let $s$ and $s'$ be the states of two vehicles at the same time instant; the latter point is achieved for the crossing $k$ if the following cases are prevented:

(X1) $s, s' \in S_2$ with $(s, s') \in \mathcal{L}_k(s', s')$;

(X2) $s \in S_2$, $s' \in S_3$ with either $(s, s') \in \mathcal{L}_k(s', s')$ or $(s, s') \in \mathcal{L}_k(s', s', s')$;

(X3) $s, s' \in S_3$ with one of the possibilities: $(s, s') \in \mathcal{L}_k(s', s')$, $(s, s') \in \mathcal{L}_k(s', s', s')$, $(s, s') \in \mathcal{L}_k(s', s', s', s')$.

Case (X1) in fact reflects the motivation for defining the notation of conflicting crossing-passing zone pairs $\mathcal{L}_k$ (see Section 2). The reason for the inclusion of (X2) and (X3) is that a vehicle with the state $(s, s, s') \in S_3$ is always allowed to trigger an ARR and change its state to $(s, s, s') \in S_3$. In all three cases above, we say that the state $s$ conflicts with the state $s'$ at crossing $k$. In summary, the AGV system is said to be free of collisions if both the following two conditions are satisfied:

(C1) Each non-depot zone is occupied by at most one vehicle.

(C2) The state of each vehicle does not conflict with that of any other at any crossing.

First, we need to prevent two vehicles leaving or going ahead to pass the same depot and start occupying the SZ of the same exit-lane of the depot simultaneously. For this purpose, the following Rule 1 is imposed:
Rule 1: For each depot, at most one vehicle is allowed to trigger a depot-passing event at the depot at any time.

Similarly, we also need to avoid two vehicles triggering at-crossing LEAs or GOAs at the same crossing to start occupying the SZ of the same out-lane of the crossing simultaneously. This gives ground to the following Rule 2:

Rule 2: For each crossing, at most one vehicle is allowed to trigger a crossing-passing event at the crossing at any time.

Rule 1 (or Rule 2) can be realized by requiring each vehicle to hold exclusively a local depot token (Rule 2: local crossing token) whenever triggering a depot-passing (Rule 2: crossing-passing) event at each depot (Rule 2: crossing). An example of such token assignment algorithms can be found in [1].

In addition, Rule 3 and Rule 4 below are used to make sure that a vehicle cannot start occupying a zone if the zone is occupied by another vehicle and/or doing so will give rise to two vehicles having conflicting states at a crossing.

Rule 3 (on off-crossing LEA and GOA): If a vehicle is in the state \( s = (c_1, c_2) \in S_r \), where \( c_1 \) is an off-crossing zone, then an (off-crossing) LEA is allowed to be triggered and to change the state of the vehicle to \( (c_1, c_2) \) if \( c_2 \) is available; if \( s = (c_1, c_2) \in S_r \), where \( c_2 \) is an off-crossing zone, then an (off-crossing) GOA is allowed to be triggered and to change the state of the vehicle to \( (c_1, c_2) \) if \( c_2 \) is available.

Rule 4 (on at-crossing LEA and GOA): If a vehicle is with the state \( s = (c_1, c_2) \in S_r \), where \( c_1 \) is an at-crossing zone of crossing \( i \), then an (at-crossing) LEA is allowed to be triggered and to change the state of the vehicle to \( s' = (c_1, c_2) \) if (1) \( c_2 \) is available and (2) there is no other vehicle having its state with \( s' \) at crossing \( i \); if \( s = (c_1, c_2) \in S_r \), where \( c_2 \) is an at-crossing zone of crossing \( i \), then an (at-crossing) GOA is allowed to be triggered and to change the state to \( s' = (c_1, c_2) \) if (1) \( c_2 \) is available and (2) there is no other vehicle with its state conflicting with \( s' \) at crossing \( i \).

It can be proved that traffic rules guarantee that an AGV system is free of collisions if the two conditions (C1) and (C2) are satisfied initially. We refer the interested reader to Section A.2.1 for a formal proof of the collision avoidance.

3.2 Deadlock avoidance

Abstracting from many applications, we may consider that each vehicle in an AGV system does nothing but visit a series of zones to transport things from assigned pick-up points to delivery points. In the framework of our event-driven model, to realize this a vehicle must trigger a sequence of events. We say that a vehicle is operational if it has the intention to trigger a new event in finite time; it is supposed that if a vehicle is not operational (and not faulty) it must be in a depot. Without loss of generality, we also assume that each vehicle is initially operational and terminates its operation after visiting a finite sequence of zones. It follows that each vehicle can terminate its operation after triggering finite events. In fact, if a vehicle triggers infinite events, one easily sees from Figure 2 that there must be infinite LEAs and/or infinite GOAs, which, by Lemma 4 (see Section A.1), implies that the vehicle would occupy an infinite sequence of zones.

We say that an operational vehicle is blocked if all feasible events for the state of the vehicle are not allowed to occur in finite time. Now we show that, if, at any time, at least one operational vehicle is not blocked, then all vehicles can successfully terminate their operations in finite time. By contradiction, suppose that there are some vehicles remaining operational all the time. Then, these vehicles would be able to trigger an infinite number of events, as at any time one of them is able to trigger an event in the finite future. This, however, implies that these vehicles can terminate their operations, since, as assumed, each operational vehicle can terminate its operation after triggering finite events.

This observation motivates us to give the definition of deadlock of the AGV system as follows: the AGV system has a deadlock if all operational vehicles are blocked. From the argument above, it is clear that every vehicle can terminate its operation if the AGV system is free of deadlocks (at all times).

Note that there is a special type of deadlock scenario, i.e., cyclic deadlocks: a group of operational vehicles \( v_1, v_2, \ldots, v_n \), \( m \geq 2 \), are respectively in \( m \) distinct zones \( c_1, c_2, \ldots, c_m \), where \( c_j = c_{j+1} \) is the next zone of \( c_j \) for \( j = 1, 2, \ldots, m-1 \) and \( c_n \) is the next zone of \( c_1 \). In this case, by Rules 3 and 4, none of the vehicles is allowed to trigger the only feasible event LEA. In fact, this deadlock corresponds to a black cycle in the operation graph, which is defined below.

The operation graph is a time-dependent directed graph with the vertex set \( \mathcal{C} \) and the arc set \( \mathcal{E} \), where \( \mathcal{C} \) as defined before, is the set of all zones and \( \mathcal{E} \) is the collection of every such zone pair \( (c_1, c_2) \) that satisfies either of the following two conditions:

1. \( c_1 \) is an off-crossing zone and \( c_2 \in Y_{c_1} \);
2. \( c_1 \) is an at-crossing zone and \( c_1, c_2 \) are respectively the current and next zones of an operational vehicle.

A cycle in the operation graph is a sequence of vertices \( c_1 c_2 \ldots c_n, n \geq 2 \), such that \( c_{j+1} = c_{j} \), \( c_j \), \( j = 1, 2, \ldots, n-1 \), (\( c_n \)), called the arcs of the cycle, all belong to \( \mathcal{E} \). A black cycle in the operation graph is a cycle with the property that the zones corresponding to the vertices in the cycle are all black.

---

5 We will sometimes interchange the words “zone” and “vertex” where no confusions arise.
It can be shown that an AGV system is free of collisions and deadlocks if initially the conditions (C1) and (C2) hold and the operation graph has no black cycle; and the occurrence of the LEAs and GOAs are governed by Rules 1, 2*, 3 and 4', where the Rules 2' and 4' are shown as below:

**Rule 2':** There is at most one vehicle that can trigger a crossing-passing event (at all crossings) at any time.

**Rule 4':** Rule 4 together with the additional requirements that the LEA cannot lead to a black cycle in the operation graph containing c2 and the GOA cannot lead to a black cycle containing c3.

Compared with Rule 2, which only imposes a local restriction on crossing-passing events, the aim of imposing the more stringent Rule 2’ is two-fold. Firstly, it prevents the scenario in which simultaneous occurrences of crossing-passing events at different crossings cause a black cycle but does not otherwise. Secondly, it significantly reduces the complexity of the black cycle checking, demanded in Rule 4’, for triggering a crossing-passing event, since otherwise the vehicle must be aware of all the other vehicles, possibly physically far away, that may also trigger crossing-passing events at the same time.

Rule 2’ can be realized by using a global crossing token. It is required that each vehicle request the token from a central control station when it intends to trigger a crossing-passing event. After the vehicle obtains the token, it checks with Rule 4’ while holding the token exclusively and releases the token after the checking (no matter whether the event is allowed to be triggered or not). Of course, we need to ensure that every vehicle that intends to trigger a crossing-passing event will be issued the token. The token assignment algorithm presented in [1], which is based on an index list of the “starving” vehicles waiting for the token, can be used for this purpose. The checking of potential black cycles can be implemented using an algorithm similar to Algorithm 1 in [1] and has the time complexity only quadratically dependent on the number of vehicles in the system.

A formal justification of the deadlock avoidance is given in Section A.2.2.

### 4. Performance of AGV Systems Using the Traffic Control

So far, we have concluded that each of the vehicles in an AGV system can finish visiting any finite sequence of zones and park in a depot where it terminates its operation without colliding with any other vehicle, as long as certain mild conditions are satisfied when the system starts and the traffic rules are followed. The overall transportation performance of the system, however, will also depend on many other factors, such as layout of guide-path, number of AGVs, scheduling of transportation tasks, routing algorithms and, last but not least, contingency (fault) handling, all of which in general can influence the traffic of the AGV system. Compared with most of the existing traffic control schemes in the literature, the proposed traffic control rules cause the latter factors to have the least impact on the collision and deadlock avoidance. To be specific, the layout design and task scheduling are completely decoupled from deadlock and collision avoidance; the limitation on the number of AGVs is only imposed by the conditions (C1) and (C2), in that too many vehicles would inevitably force a zone to accommodate more than one vehicle or the appearance of a cyclic deadlock. In addition, we have seen that the freedom rendered by the traffic control for routing is huge: a route can be fixed to a vehicle before it starts operation, partially altered after the vehicle sets off, or even built online in an incremental fashion. The minimum requirement is that a vehicle must fix all three zones in the state to which it needs to reach to trigger an immediate event.

In the rest of this section, the performance of an AGV system incorporating the traffic control presented in the previous section is evaluated in a simulation case study for automated container terminals; the result is compared with those described in two existing papers in the literature.

#### 4.1 Simulation setup

We focus on the quayside container transshipment, where the quay cranes (QCs) and yard stackers (YSs) are, respectively, in charge of the container pick-up and drop-off handling in the quay area (QA) and yard area (YA); and a team of AGVs is used to shuttle between these two areas to serve the cranes with containers (see Figure 3 for an illustration of the container transshipment operation). More specifically, in discharging a vessel, a handful of QCs take the containers off the vessel and put them in associated container buffers. These containers will be later on picked up by the AGVs and transported across the transportation area (TA) to the container buffers of scheduled YSs. The containers will then be put in container stacks by the YSs. Conversely, when loading a vessel, the containers are collected from certain stacks by the YSs and transported to designated QCs by the AGVs.

---

6 Also called automated stacking cranes (ASCs) in the literature.
To test the performance of our designed AGV system, we carry out a simulation comparison with some existing designs of AGV systems for automated container terminals. In particular, we compare our results with those in [44] and [45]. The main reason for choosing these two works is that they provide complete and detailed information of the simulation settings, including workspace dimensions, cycle times of cranes and vehicle specifications. In both the papers, the performance of two types of vehicles, i.e., vehicles with and without the capacity to lift containers by themselves, are simulated and compared. It was found that the vehicles that can lift containers by themselves (named automated lift vehicles (ALVs) in the two papers) in general achieve higher container transport efficiency. This is mainly because the container cranes do not need to hold a container while waiting for a vehicle to pick it up. In this comparison study, we only simulate the transportation with ALVs (also called AGVs hereafter). In addition, since our goal here is a fair comparison of system performance, we do not include vehicle faults in the simulations.

4.2 Guide path and routing algorithm

The AGVs are of the size 10m × 4m. For the guide-path over the TA, we apply the layout design with four roads and big crossings presented in Section 5.3 in our previous paper [1] (illustrated therein by Figure 8). It was shown therein that this layout renders the best performance among its peers. The layout designs for the QA and YA follow the same idea given also in [1], i.e., using a zone (in a lane) as a container buffer under each QC or YS and connecting the guide-path over TA with these QC and YS buffers by a couple of lanes and crossings. The specific layouts over QA and YA that we use for the two comparisons are based on the respective simulation settings (e.g., numbers of QC and YS buffers, distance between adjacent QCs or YSs, etc.) in [44] and [45].

The routing algorithm for the AGVs used in the simulation is the one presented in Section 4.2 in [1]. The algorithm takes into account both travelling distance and some online traffic information for decision making. Briefly speaking, by the routing algorithm the route of each vehicle extends online from one waypoint to another, where a waypoint can be either an at-crossing zone or a depot: before arriving at a waypoint, the vehicle needs to determine the next waypoint along with the lane leading to it.

4.3 Comparison of simulation results

4.3.1 The first comparison result

In the paper [44], four QCs are used in discharging one vessel and the containers are stored in 16 container stacks. The AGVs move along a loop guide-path to serve the QCs and YSs so that the traffic control reduces to the minimum and there are no routing issues. (See Figure 2 in [44] for an illustration of this layout.) Moreover, multiple lanes are laid in the QA, on which four container buffers are placed under each QC, such that more than one vehicle can pick up containers under one QC without conflicting with each other. The numerical details of this setting are summarized in Figure 4. The task dispatching rules in [44] can be roughly put as follows: empty vehicles are dispatched to QCs according to the nearest-vehicle-first principle and the YS with the fewest number of serving vehicles has the priority to get a new serving vehicle, i.e., a vehicle carrying a container that is to be put in a stack.

For a fair comparison, the same number of buffers and cycle time distributions of QCs and YSs are used in our simulations. However, we set the maximum deceleration of each vehicle to be 2.5m/s$^2$ rather than 0.5m/s$^2$, which is used in [44]. According to the remarks at the end of Section 2.2, our zone-control model allows the length of each zone to be lower bounded by the maximum of the length of a vehicle and the distance for a vehicle to make a full stop from its maximum speed. Thus, by letting the deceleration be 2.5m/s$^2$, we can put the length of non-SZs and SZs on any lane at 14m and 28m respectively (see Chapter 7.5 in [46] for details). In this way, the guide-path over the TA can be designed the same as that shown in Figure 8 in [1]. We believe that the difference in the maximum deceleration of a vehicle does not do our approach much favour in a comparison of the discharging time of a vessel. This judgement is based on the following reason: in [44], because of the loop layout of the guide-path, a vehicle basically needs to make a full stop only at a handling crane. Hence, the time that a vehicle takes to make a full stop from its
maximum speed only occupies a very small part of its total travel time.

In Figure 5, we depict the simulation results obtained by the two methods regarding the discharging time of a vessel with 2,000 containers (note that in [44] simulation results are not given for fewer than 16 vehicles). The two points below can be concluded from the simulation results:

- The minimum number of vehicles to minimize the discharging time, i.e., to maximize the QCs' capacity, is about 17 by our design, while it is about 23 in [44].
- By our approach, the discharging time with small vehicle fleets is substantially shorter than that achieved in [44].

In addition, the use of the Manhattan-like guide-path in our approach makes the distance that a vehicle needs to travel for a QC-YS or YS-QC container transportation task much smaller than that in [44], where a vehicle has to travel the whole loop for each transportation. Nevertheless, as mentioned before, the loop-following strategy in [44] greatly simplifies the traffic control and routing for the AGVs, thus is easier to be implemented.

4.3.2 The second comparison result

In [47, 45], a more flexible guide-path, composed of intersecting straight lanes, is used so that the travelling distance from a QC to YS, or vice versa, is drastically decreased from that in the loop layout mentioned in Section 4.3.1. However, the traffic control becomes much more complicated. The authors proposed a conflict-free routing algorithm to guarantee the absence of collisions and deadlocks. More specifically, after selecting a route for each vehicle, the route is divided into a series of occupation areas and the entry and exit time of each area is precisely scheduled to avoid conflicting with those vehicles whose movement has already been scheduled. See Figure 3 in [45] for a schematic illustration of this scheduling approach.

The dispatching strategy employed by Bae et al. is based on an inventory-based dispatching method (see [48]). This strategy basically says that each time a vehicle needs to be assigned to a QC or YS, the QC or YS with the smallest number of assigned vehicles is selected.

In [45], simulations of discharging and loading of one vessel with 3,600 containers are performed with various types of QCs. The numerical specifications of the simulation setting are given in Figure 6. Only the results with the tandem-lift QC will be used here for the comparison. For a fair comparison, the same number of buffers and cycle time distributions of QCs and YSs are used in our simulations. Moreover, we use the same dispatching algorithm as in [45].

Figure 7 shows the average number of discharged containers per QC per hour obtained by simulating the two methods. Note that the total number of vehicles used in each simulation is equal to four (i.e., the number of the QCs) times the corresponding number on the X axis. The following conclusions can be made from the simulation results:

- The minimum number of vehicles required to saturate the productivity of the QCs is the same for both methods.
- Our approach makes the QCs reach the maximum possible throughput (62 containers / hour), while the one in [45] does not.
- Our approach gives slightly better performance with small fleets of vehicles.

Note that in [45] no fixed guide-path is used so that the AGVs are more flexible in choosing their routes and less space may be required to accommodate the AGV system. However, the conflict-free routing algorithm used in [45] is not robust to the changes of schedules or faults of the AGVs and QCs/YSs. This is because the conflict resolution of the AGVs is achieved by precisely pre-planning the time.
windows over which a vehicle visits certain areas in the workspace. If there are changes to the loading/discharging schedule or disturbances/malfunctions with some vehicles or container handling cranes, then it is possible that the routes of many AGVs may have to be rescheduled, which is very time-consuming, not only for performance guarantee but for conflict (collision and deadlock) avoidance as well. In contrast, our traffic control (for collision and deadlock avoidance) is completely decoupled from the routing, which is based on real-time traffic information and is thus quite robust to schedule changes and other sorts of disturbances.

5. Fault-tolerant Traffic Control Scheme with Vehicle Breakdowns

The rest of the paper is devoted to a presentation of an emergency traffic control scheme that ensures the absence of collisions and deadlocks with the occurrence of a vehicle breakdown. By a vehicle breakdown, we mean that a vehicle cannot move after the failure happens (the vehicle will be called faulty vehicle hereafter). It is, however, assumed that the breakdown can be communicated by the faulty vehicle to a central control station as well as all the other vehicles (called healthy vehicles hereafter) after it takes place.

To handle the faulty vehicle there are normally several options: (1) remove it from the guide-path; (2) get it repaired on site; or (3) leave it as is (due to some physical constraints). We will discuss in Section 5.1 how collisions and deadlocks can be avoided if the decision is made to remove the faulty vehicle. If we choose to repair the faulty vehicle on site or leave it as is, the system performance may be significantly affected by the faulty vehicle, due to one or more of the following possibilities:

- There are some vehicles on the same lane and behind (thus blocked by) the faulty vehicle.
- There are some vehicles that have been assigned a route that involves the lane where the faulty vehicle stands.
- The faulty vehicle stands right at the exit of a depot and the vehicles in the depot cannot get out.
- The faulty vehicle stands in the crossing and conflicts with other vehicles that want to pass the crossing.

Therefore, it would be wise or even necessary to adjust the behaviour of the other (healthy) vehicles to mitigate the...
impact of the contingency. One such solution will be given in Section 5.2.

Unlike in normal traffic control, where the vehicles actively trigger events to follow their routes, they may be commanded to trigger specific events in an emergency traffic control scheme. The states of some zones may also be reset by the control station instead of varying based on the occurrence of vehicle events (as described in Section 2.3).

To make the presentation smooth, we put some supporting arguments, labelled by superscripts A, B, C... in Section B (in the appendix) of the paper.

5.1 Fault-tolerant traffic control scheme with vehicle removal

There can be many solutions for this case; we only show a simple one here: temporarily halt the system, remove the faulty vehicle, then resume the system. To be specific, do the following:

Step 1: Let each healthy vehicle only trigger a minimal number of ARRs to be in some zone and stop moving (no matter what its state is when it is commanded to do so).

Step 2: Remove the faulty vehicle and reset the states of the zones occupied by it to be available.

Step 3: Resume the system.

For Step 1, it is easy to see from Figure 2 that any vehicle needs to trigger at most two ARRs to be in some zone, which is in fact the last zone currently occupied by the vehicle. In Step 3, by resuming the system, we suppose that each healthy vehicle has already been assigned a partial or complete remaining route, thus has a next zone to move to.

The proof for collision and deadlock avoidance using this procedure is given in Section C.1 in the appendix. The main idea is to show that the triggering of ARRs cannot lead to collisions and black cycles, nor do the removal of the faulty vehicle and the reset of the zone state.

5.2 Fault-tolerant traffic control scheme without vehicle removal

Let $s$ be the state of the faulty vehicle when it breaks down. We consider here all possibilities for $s$. However, to simplify the overall emergency control scheme, the following state transition is enforced: if $s = (c_1, c_2, c_3) \in S_1$, then its state is altered to $s = (c_1', c_2', c_3) \in S_2$, i.e., it gives up the occupation of the zone $c_3^A$. Note that this emergency event can cause neither vehicle collisions nor deadlocks, as it simply makes one zone available on the guide-path after not being so. Now, the simplification allows us to consider only the following possible cases for $s$:

(F1) $s = (c_1, c_2, c_3) \in S_2$, where $c_1 \in D$;

(F2) $s = (c_1, c_2, c_3) \in S_2$, where $c_1 \in C \setminus D$;

(F3) $s = (c_1, c_2, c_3) \in S_2$, where $c_1 \in D \cup C \setminus D$;

(F4) $s = (c_1, c_2, c_3) \in S_2$, where $c_2 \in D \cup C \setminus D$;

(F5) $s = (c_1, c_2, c_3) \in S_2$, where $c_1$ and $c_2$ are on the same lane;

(F6) $s = (c_1, c_2, c_3) \in S_2$, where $c_1$ and $c_2$ are on different lanes.

In (F1) the broken vehicle can be seen parking in the depot $c_1$ and does not interfere with the rest of the AGVs. Hence, no actions need to be taken. In (F2) or (F5), the vehicle failure happens when it is in a zone on a lane or moving from one zone to another on the same lane; and thus the breakdown is named an on-lane breakdown. The breakdown in (F3) or (F4) is called an at-depot breakdown as the fault occurs while the vehicle is entering or exiting a depot. In (F6), the vehicle breaks down while it is passing a crossing, i.e., $(c_1, c_2) \in K$, for a crossing $i^A$; thereby the breakdown in this case is named an at-crossing breakdown.

Throughout this section, we strengthen the guide-path layout assumptions in Section 2.1.3 by adding the following items:

(A5) Each crossing (or depot) has at least two out-lanes (depot: exit-lanes).

(A3') Each lane is either an out-lane of a crossing or an exit-lane of a depot$^6$.

The additional two assumptions are quite natural. First, (A5) is needed since otherwise, if the faulty vehicle is on an out-lane of a crossing, the healthy vehicles planning to go through that lane would have no other choice but to pass the crossing. Second, without (A3'), if the faulty vehicle is on a lane which is neither an out-lane of a crossing nor an exit-lane of a depot, then the healthy vehicle on the same lane behind the faulty one would get stuck, as it would be blocked by the faulty vehicle in one direction and has no connection to a crossing or a depot in the other.

To be able to handle all the possible breakdown situations, we equip the SZ of each lane with a buffer space, such that the size of the SZ is twice that of the rest zones on the same lane. The idea behind this is that, if needed, the SZ of a lane can be split into two zones to prevent deadlocks in the system.

In the rest of this section, we will address all three types of breakdown:

5.2.1 Fault-tolerant scheme for on-lane breakdowns

Suppose that a vehicle breaks down when it is in the zone $c_1'$ or moving from $c_1'$ to $c_1'^2$ (recall that $c_1'$ denotes the $i$th zone on lane $i$). Note that, by (A3'), lane $i$ must be either an out-lane of a crossing or an exit-lane of a depot. The two cases will be discussed separately.

Case 1: lane $i$ is an out-lane of a crossing $k$: By the guide-path layout assumption (A2), lane $i$ is also either an in-lane of a crossing $k'$ (see Figure 8) or an entry-lane of a depot $d'$. Define $B_{\text{blocked}}$ to be the set of vehicles that occupy any zones in $\Omega = [c_1', c_2', ..., c_{n-1}]$ at the moment they are informed...
about the breakdown. Apparently, $V_{\text{locked}}$ does not contain the broken vehicle. In addition, define $V_{\text{change\_sz}}$ to include each vehicle that is not broken and is in an at-crossing zone of the crossing $k$ and has the SZ of lane $i$ as the next zone when it acknowledges the breakdown. Roughly speaking, $V_{\text{locked}}$ indeed contains those vehicles that are behind and thus obstructed by the broken vehicle on lane $i$; the vehicles in $V_{\text{change\_sz}}$ are those that planned to go onto lane $i$ but have to make detours because of the vehicle failure. Figure 8 depicts an example scenario of an on-lane breakdown.

Step 1: The central control station temporarily suspends the assignment of the global crossing token so that no vehicles can be granted the token and trigger crossing-passing events. (Note that Rule 1 and Rule 3 in the normal traffic control on the non-crossing-passing events still apply to all vehicles.)

Step 2: All the vehicles in $V_{\text{locked}}$ are only allowed to trigger ARRs and will be in zones in $\Omega$. (On the other hand, one can show that, after Step 2 and before the emergency traffic control scheme finishes, any vehicle occupying a zone $c \in \Omega$ must be a vehicle in $V_{\text{locked}}$ and in the zone $c$.)

Step 3: Split lane $i$ at the broken vehicle and reverse the direction of the part of it behind the broken vehicle; then connect the two new lanes to the rest of the guide-path. (Note that, by doing this, lane $i$ does not exist any more. See Figure 9.) Formally put,

a. let $c_{i-1}^j, c_i^j, c_{i+1}^j, \ldots, c_{i+u}^j$ be a new lane (say with index $i^*$), i.e., set $c_{i-1}^j = c_{i-1}^1, c_i^j = c_{i+1}^2, \ldots, c_{i+u}^j = c_{i+u}^1$ and further let it be a new in-lane of the crossing $k$; we can choose any out-lanes of the crossing $k$ other than lane $i$ as the neighbouring lanes of lane $i$.

b. let $c_1^j, c_2^j, \ldots, c_{j-1}^j$ be a new lane (say with index $i^*$), i.e., set $c_1^j = c_1^1, c_2^j = c_2^2, \ldots, c_{j-1}^j = c_{j-1}^1$ and further let it be a new in-lane of crossing $k$ or a new entry-lane of depot $d$; the neighbouring lanes of lane $i^*$ can be set equal to those of lane $i$.

If there is a vehicle in $c_1^j, i=1,2,\ldots,j-1$ (as shown above it must be a vehicle in $V_{\text{locked}}$), let it pick its new next zone freely in $V_{\text{change\_sz}}$ which is well defined after the formation of the new lane $i^*$. The state of the vehicle shall change accordingly.

Step 4: Change the next zone of the vehicles in $V_{\text{change\_sz}}$ to be the SZ of an out-lane of crossing $k$. If this introduces black cycles in the operation graph, then remove the black cycles by splitting the SZ into two new zones (this will be explained in detail below).

Step 5: Resume the token assignment procedure. The system shall run as normal again.

We now describe the black cycle elimination procedure in Step 4. Let lane $j$ be the lane of which the SZ is selected to be the new next zone of the vehicles in $V_{\text{change\_sz}}$ (see Figure 10). First, note that if a black cycle appears by this procedure, then the cycle must contain the zone $c_1^j$; thus $c_1^j$ is black. This means that there is a vehicle $v_{\text{m}}^c$ that takes one of three possible states: $(c_1^j, c_1^j, c_1^j) \in \mathcal{S}_1$, $(c_1^j, c_1^j, c_1^j) \in \mathcal{S}_2$, or $(c_1^j, c_1^j, c_1^j) \in \mathcal{S}_3$. The main idea of the black cycle elimination is to split the zone $c_1^j$ to be two new zones, then set the state of $v_{\text{m}}^c$ to be moving from one to the other. This implies that one of the two new zones will be black, but the other will be grey. As a result, the black cycles (containing $c_1^j$) are removed. More specifically, we carry out the following steps:

1. Let $v_{\text{m}}^c$ only trigger ARRs so that it will be in $c_1^j$ after triggering either 0, 1, or 2 events (depending on which of the above three possible states the vehicle has).

2. Split the zone $c_1^j$ into two new zones $c_1^j_{\text{a}}$ and $c_1^j_{\text{b}}$. redefine the lane $j$ to be the zone sequence $c_1^j_{\text{a}}c_1^j_{\text{b}}c_1^j_{\text{a}}c_1^j_{\text{b}} \ldots$. The state of the vehicle $v_{\text{m}}^c$ to be $(c_1^j_{\text{a}}c_1^j_{\text{b}}c_1^j_{\text{a}}c_1^j_{\text{b}}c_1^j_{\text{a}}c_1^j_{\text{b}})$. Note that, as the zone $c_1^j$ has been replaced with the two new zones $c_1^j_{\text{a}}$ and $c_1^j_{\text{b}}$, the route of the vehicle $v_{\text{m}}^c$ should be changed accordingly.

Steps 3 and 4 are illustrated in Figures 9 and 10, where Figure 9 corresponds to a case in which the procedure in Step 4 does not introduce a black cycle; while Figure 10 shows a situation in which a black cycle is introduced and the buffer zone is used.
Case 2: lane $i$ is an exit-lane of a depot $d$ : In this case, the $V_{\text{blocked}}$ is defined the same as before, but $V_{\text{change,ne}}$ is now changed to be the set of vehicles that are in the depot and have the SZ of lane $i$ as the next zone when they are notified of the breakdown. The breakown-tolerant scheme is similar to the one just presented and the only difference is mentioned here: in Step 1, the assignment of the local depot token of depot $d$ is suspended. In Step 3(a), let $c_{j-1}c_{j-2}...c_{j}$ be a new entry-lane of the depot $d$ (thus lane $i$ is not an exit-lane of the depot any more); the neighbouring lane selection is not applicable. In Step 4, the next zone of each vehicle in $V_{\text{change,ne}}$ is changed to be the SZ of an exit-lane of depot $d$ (otherwise we will not introduce any black cycles, as a depot is always white).

It may have been noted that, by the end of the emergency traffic control steps for both Case 1 and Case 2, the routes of the vehicles in $V_{\text{blocked}}$ and $V_{\text{change,ne}}$ are in general not complete, as each of them is in a zone and simply has a next zone. The routes of the other healthy vehicles may need to be adjusted as well, since they could include lane $i$, which however does exist any more. We claim that, as achieved in the fault-free case (see Section 3), one is assured full freedom in the route re-assignment/adjustment and the system will be free of collisions and deadlocks under the normal traffic control after it resumes. The proof is shown in Section C.2.

5.2.2 Fault-tolerant scheme for at-depot breakdowns

The tolerance scheme for the breakdown case (F3), i.e., when the state of the broken vehicle is $s=(c_1,c_2,c_3) \in S_2$ with $c_1 \in D(c_1 \in E \cup D)$, is simple. Note that $c_1$ must be the SZ of a lane $i$, which is an exit-lane of the depot $c_1$. We only need to reroute the vehicles in the depot that have the zone $c_1$ as their next zone (denoted by the set $V_{\text{blocked}}$) and disconnect lane $i$ from the depot (so that no vehicle will try to enter it). More specifically, we perform the following:

1. Change the next zone of each vehicle in $V_{\text{blocked}}$ to be the $SZ$ of an exit-lane of the depot other than lane $i$ (such a lane does exist due to the assumption (A5)).

2. Remove lane $i$ from the exit-lane list of the depot $c_1$.

In the breakdown case (F4), the state of the broken vehicle is $s=(c_1,c_2,c_3) \in S_2$ with $c_2 \in D(c_1 \in E \cup D)$. Note that $c_1$ is in fact the EZ of a lane $i$, which is an entry-lane of the depot $c_2$, i.e., $c_1 = c_{i-1}$, where $m_i$ denotes the number of zones on lane $i$. The breakdown-tolerant scheme for this case is the same as for the on-lane breakdowns stated in Section 5.2.1 (simply let $j=m_i$ in the statement of the scheme).

5.2.3 Fault tolerance scheme for at-crossing breakdowns

To deal with at-crossing breakdowns, we further strengthen the guide-path layout assumption with

(A6) Each lane cannot be both an in-lane and out-lane of the same crossing.

(A7) The out-lanes of each crossing or the exit-lanes of each depot cannot all be the in-lanes of one crossing.

Suppose that a vehicle breaks down when its state is $(c_1,c_2,c_3) \in S_2$, where $c_1$ is the EZ of an in-lane, say lane $i_1$, of crossing $k$ and $c_2$ is the SZ of an out-lane, say lane $i_2$, of the same crossing. Define the set $L_{\text{blocked}}$ to include lane $i_1$ and every other in-lane of crossing $k$ from the EZ, of which there is no way to pass the crossing without conflicting with the broken vehicle, i.e., any lane $j$ such that $(c_{j-1},c_j) \in \mathcal{R}_k(c_1,c_2)$ for any $(c_{j-1},c_j) \in \mathcal{R}_k$. Now we define $V_{\text{blocked}}$ to be the set of vehicles that occupy any non-EZs of lane $i_1$ and any zones on the other lanes in $L_{\text{blocked}}$ at the moment they are informed of the vehicle failure (all these definitions are illustrated in Figure 11). Using the same idea as in Section 5.2.1, the vehicles in $V_{\text{blocked}}$ can get out of the lanes in $L_{\text{blocked}}$ (in order to avoid being stuck by the broken vehicle) by reversing their moving directions, which is enabled by changing the directions of the corresponding lanes (see Step 3 below). In addition, define $V_{\text{change,ne}}$ to include those vehicles, each of which is (1) not an element of $V_{\text{blocked}}$ and (2) is in a zone and has the SZ of a lane in $L_{\text{blocked}} \cup [i_2]$ as the next zone when it acknowledges the breakdown. It is easy to see that the vehicles in $V_{\text{change,ne}}$ need to be rerouted (see Step 4 below) due to the direction changes of the lanes in $L_{\text{blocked}}$ and the non-entry situation of lane $i_2$ (caused by the
occupation of the zone $c_2$ by the broken vehicle). Finally, from a guide-path layout point of view, we need to remove the potential future conflicts with the broken vehicle by removing the lane $i_2$ from the out-lanes of crossing $k$ and reset the neighbouring lane relationship at the cross (see Step 5 below).

The fault-tolerant scheme can be described as the following:

**Step 1**: The central control station temporarily suspends the assignment of the global crossing token and all the depot tokens.

**Step 2**: All the vehicles in $V_{\text{blocked}}$ are only allowed to trigger ARR and will be in zones on the lanes in $L_{\text{blocked}}^n$. (On the other hand, as with Step 2 for the on-lane breakdown case in Section 5.2.1, one can show that, after Step 2 and before the emergency traffic control scheme finishes, any vehicle occupying any zone $c$ on any lane in $L_{\text{blocked}} \setminus [i]$ or any non-EZ $c$ on the lane $i_2$ must be a vehicle in $V_{\text{blocked}}$ and in the zone $c$.)

**Step 3**: Reverse the moving directions of the vehicles in $V_{\text{blocked}}$ by reversing the directions of the lanes in $L_{\text{blocked}}$ (except $i_2$, which will be partially reversed):

Let lane $j$ be a lane in $L_{\text{blocked}} \setminus [i]$, then remove it from the set of in-lanes of crossing $k$. In addition,

a. if lane $j$ is an out-lane of a crossing $k'$, then let $c_{j}'=c_{j-1}'=\cdots=c_{1}'$ be a new in-lane (with the index $j'$) of the crossing $k'$, i.e., set $c_{j}'=c_{j-1}'=\cdots=c_{1}'$, with $m_{j'}=m_j$ (thus lane $j$ does not exist anymore); and choose any out-lanes of crossing $k'$ that are not in $L_{\text{blocked}} \setminus [i]$ as the neighbouring lanes of lane $j'$. If there is a vehicle in $c_{j'}$, it will pick its next zone freely in $Y_{c_{j'}}$.

b. if lane $j$ is an exit-lane of a depot, then let $c_{j}'=c_{j-1}'=\cdots=c_{1}'$ be a new entry-lane (with the index $j'$) of the depot. If there is a vehicle in $c_{j'}$, it will pick its next zone freely in $Y_{c_{j'}}$.

If $j=i_2$, remove it from the set of in-lanes of crossing $k$, then do (a) or (b) with $m_1$ substituted by $m_j$. Besides, let $c_{j}'=c_{j-1}'=\cdots=c_{1}'$ be a new in-lane (with the index $j'$) of crossing $k$, i.e., set $c_{j'}=c_{j-1}'$. (Note that lane $j'$ is composed of only one zone, which is occupied by the broken vehicle.)

**Step 4**: Modify the next zone of each vehicle in $V_{\text{change}}$ such that it will not try to enter any lane in $L_{\text{blocked}} \setminus [i]$. We have the following two possibilities:

a. Let the vehicle be in an at-crossing zone $c_2$ of a crossing $k$. Then, if $k \neq k'$, the vehicle resets its next zone to be the SZ of an out-lane of the crossing $k'$, which is not in $L_{\text{blocked}} \sim [i]$; if $k = k'$, the vehicle resets its next zone to be the SZ, say zone $c_2$, of an out-lane of crossing $k$, such that $(c_2,c_2) \notin L_1(c_2,c_2)$. In either of the cases, if this change would introduce black cycles into the operation graph, then remove the black cycles using the same zone splitting trick as described in Step 4 in Section 5.2.1.

b. Let the vehicle be in a depot. Then the vehicle resets its next zone to be the SZ of an exit-lane of the depot that is not in $L_{\text{blocked}} \sim [i]$.

**Step 5**: Remove lane $i_2$ from the set of out-lanes of crossing $k$. For each in-lane (say lane $j$) of crossing $k$ but lane $j'$, reset the set of its neighbouring lanes to include any out-lane (say $j'$) of the crossing that satisfies $(c_{j'}c_{j'}) \notin L_2(c_{j'},c_{j'})$.

**Step 6**: Resume the token assignment procedure.

Figure 11. An illustration of an at-crossing breakdown: the broken vehicle is coloured black; $L_{\text{blocked}}=[i]$; $V_{\text{blocked}}=[c_1c_2]$ and $V_{\text{change}}=[v_3v_4]$

As for the on-lane vehicle breakdown case, by the end of the emergency traffic control steps for both the at-depot and at-crossing breakdowns, the routes of the vehicles in $V_{\text{blocked}}$ and $V_{\text{change}}$ are in general not complete; the routes of the other healthy vehicles may need to be adjusted as well. Again, full freedom in the route re-assignment/adjustment is guaranteed and the system will be free of collisions and deadlocks under the normal traffic control after it resumes. The proofs are similar to those for the on-lane breakdown cases and are thus omitted.

6. Concluding Remarks

In this paper, we have extended the discrete-event zone-control model and traffic control for AGV systems presented in our previous work [1, 2], such that the requirement on the zone size is relaxed. This improves substantially the utilization of the workspace, especially for systems with a
large number of zones. We have proved that the traffic control guarantees the avoidance of collision and deadlocks. Moreover, the traffic control inherits the appealing properties of its predecessor in [1, 2]: it is time-efficient and imposes no constraints for the route assignment on the vehicles. A simulation-based case study of overall system performance is presented, in which an AGV system is applied to quayside container transportation at an automated container terminal; our results are compared with those published in two existing papers.

Using the extended zone-control model, we have also proposed some emergency traffic control schemes that reconfigure the guide-path and regulate the vehicles’ behaviour when a vehicle breakdown occurs. Various cases are addressed depending on whether to remove the faulty vehicle and the state of the faulty vehicle. The AGV system is guaranteed free of collisions and deadlocks within the duration of the emergency control and after the normal traffic control takes over. The freedom of routing for the healthy vehicles that is ensured in the fault-free case is retained.

7. References

[1] Q. Li, A. C. Adriaansen, J. T. Udding, and A. Y. Pogromsky. Design and control of automated guided vehicle systems: a case study. In Proc. of the 18th IFAC World Congress, pages 13852–13857, Milano, Italy, 2011.
[2] Q. Li, J. T. Udding, and A. Pogromsky. Zone-control-based traffic control of automated guided vehicles. In J. H. van Schuppen and T. Villa, editors, Coordination Control of Distributed Systems, pages 53–60. Springer, 2015.
[3] I. F. A. Vis. Survey of research in the design and control of automated guided vehicle systems. European Journal of Operational Research, 170:677–709, 2006.
[4] T. Le-Anh and M.B.M. De Koster. A review of design and control of automated guided vehicle systems. European Journal of Operational Research, 171:1–23, 2006.
[5] S. Berman, E. Schechtman, and Y. Edan. Evaluation of automatic guided systems. Robotics and Computer-Integrated Manufacturing, 25:522–528, 2009.
[6] M. B. Duinkerken, J. A. Ottes, and G. Lodewijks. Comparison of routing strategies for agv systems using simulation. In Proc. of the 2006 Winter Simulation Conference, pages 1523–1530, Monterey, Canada, 2006.
[7] E. A. Ross, F. Mahmoudi, and C. T. Mosier. Tandem configuration automated guided vehicle systems: A comparative study. Decision Sciences, 27:81–102, 1996.
[8] D. Sinriech and J. M. A. Tanchoco. Design procedures and implementation of the segmented flow topology (sft) for discrete material flow systems. IIE Transactions, 29:323–335, 1997.
[9] SavantAutomation. Agv basics. http://www.agvsystems.com/agvs-basics/basics-agvs/, 2015. Accessed on 1 Sep 2015.
[10] E. G. Coffman, M. J. Elphick, and A. Shoshani. System deadlocks. Computing Surveys, 3:67–78, 1971.
[11] A. N. Habermann. Prevention of system deadlocks. Communications of the ACM, 12:373–378, 1969.
[12] M. Singhal. Deadlock detection in distributed systems. IEEE Computer, 22:37–48, 1989.
[13] Z. A. Banaszak and B. H. Krogh. Deadlock avoidance in flexible manufacturing systems with concurrently competing process flow. IEEE Trans. on Robotics and Automation, 6:724–734, 1990.
[14] N. Viswanadham, Y. Narahari, and T. L. Johnson. Deadlock prediction and deadlock avoidance in flexible manufacturing systems using petri net models. IEEE Trans. on Robotics and Automation, 6:713–723, 1990.
[15] L. Zeng, H.P. Wang, and S. Jin. Conflict detection of automated guided vehicles: A petri net approach. International Journal of Production Research, 29:865–879, 1991.
[16] N. Q. Wu and M. C. Zhou. Modeling and deadlock avoidance of automated manufacturing systems with multiple automated guided vehicles. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 35:1193–1202, 2005.
[17] C. C. Lee and J. T. Lin. Deadlock prediction and avoidance based on petri nets for zone-control automated guided vehicle systems. International Journal of Production Research, 33:3249–3265, 1995.
[18] G. P. Huang, M. D. Jeng, L. Xie, and D. H. Chung. Siphon-based deadlock prevention policy for flexible manufacturing systems. IEEE Transactions on Systems, Man, and Cybernetics, Part A, 36:1248–1256, 2006.
[19] Z.W. Li, M.C. Zhou, and N.Q. Wu. A survey and comparison of petri net-based deadlock prevention policies for flexible manufacturing systems. IEEE Transactions on Systems, Man, and Cybernetics - Part C, 38:173–188, 2008.
[20] M. P. Fanti, B. Maione, S. Mascolo, and B. Turchiano. Event-based feedback control for deadlock avoidance in flexible production systems. IEEE Trans. on Robotics and Automation, 13:347–363, 1997.
[21] M. P. Fanti, B. Maione, S. Mascolo, and B. Turchiano. Event-based controller to avoid deadlock and collisions in zone-control agvs. International Journal of Production Research, 40:1453–1478, 2002.
[22] S. A. Reveliotis. Conflict resolution in agv systems. IIE Transactions, 32:647–659, 2002.
[23] L. Kalinovcic, T. Petrovic, S. Bogdan, and V. Bobanac. Modified banker’s algorithm for scheduling in multi-agv systems. In IEEE International Conference on Automation Science and Engineering, Trieste, Italy, 2011.

[24] L.T.M.E. Penners. Investigating the effect of layout and routing strategy on the performance of the adapto system. Master’s thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, April 2014.

[25] S. A. Reveliotis, E. Roszkowska, and J. Y. Choi. Generalized algebraic deadlock avoidance policies for sequential resource allocation systems. IEEE Trans. on Automatic Control, 52:2345–2350, 2007.

[26] C. O. Kim and S. S. Kim. An efficient real-time deadlock-free control algorithm for automated manufacturing systems. International Journal of Production Research, 35:1545–1560, 1997.

[27] G. Bocewicz, R. Wójcik, and Z. Banaszak. Design of admissible schedules for agv systems with constraints: a logic-algebraic approach. Lecture Notes in Computer Science, 4496:578–587, 2007.

[28] C. W. Kim and J. M. Tanchoco. Conflict-free shortest-time bidirectional agv routing. International Journal of Production Research, 29:2377–2391, 1991.

[29] C. W. Kim and J. M. Tanchoco. Operational control of a bidirectional automated guided vehicle system. International Journal of Production Research, 31:2123–2138, 1993.

[30] L. Qiu and W. J. Hsu. A bi-directional path layout for conflict-free routing of agvs. International Journal of Production Research, 39:2177–2195, 2001.

[31] R. H. Mühling, E. Köhler, E. Gawrilow, and B. Stenzel. Conflict-free real-time agv routing. Preprint 026-2004, Technische Universität Berlin, Institute of Mathematics, 2004.

[32] M. Lehmann, M. Grunow, and H.-O Günther. Deadlock handling for real-time control of agvs at automated container terminals. OR Spectrum, 28:631–657, 2006.

[33] K. H. Kim, S. M. Jeon, and K. R. Ryu. Deadlock prevention for automated guided vehicles in automated container terminals. OR Spectrum, 28:659–679, 2006.

[34] J. J. M. Evers and S. A. J. Koppers. Automated guided vehicle traffic control at a container terminal. Transportation Research Part A: Policy and Practice, 30:21–34, 1996.

[35] M. S. Yeh and W. C. Yeh. Deadlock prediction and avoidance for zone-control agvs. International Journal of Production Research, 36:2879–2889, 1998.

[36] L. M. Rajeeva, H. G. Wee, W. C. Ng, C. P. Teo, and N. S. Yang. Cyclic deadlock prediction and avoid-

[37] A. Nazeem, S. Reveliotis, Y. Wang, and S. Lafortune. Designing compact and maximally permissive deadlock avoidance policies for complex resource allocation systems through classification theory: The linear case. IEEE Transactions on Automatic Control, 56:1818–1833, 2011.

[38] Y.F. Chen, Z.W. Li, and M.C. Zhou. Optimal supervisory control of flexible manufacturing systems by petri nets: a set classification approach. IEEE Transactions on Automation Science and Engineering, 11:549–656, 2014.

[39] P.J. Egbelu and J.M.A. Tanchoco. Potentials for bi-directional guide-path for automated guided vehicle based systems. International Journal of Production Research, 24:1075–1097, 1986.

[40] F. S. Hsieh. Fault-tolerant deadlock avoidance algorithm for assembly processes. IEEE Trans. on Systems, Man, and Cybernetics - Part A: Systems and Humans, 34:65–79, 2004.

[41] G. Y. Liu, Z. W. Li, K. Barkaoui, and A. M. Al-Ahmari. Robustness of deadlock control for a class of petri nets with unreliable resources. Information Sciences, 235:259–279, 2013.

[42] D. Xiang, Y. L. Zhang, and Y. Pan. Practical deadlock-free fault-tolerant routing in meshes based on the planar network fault model. IEEE Trans. on Computers, 58:620–633, 2009.

[43] A. B. Ahmed, A. B. Ahmed, and A. B. Abdallah. Deadlock-recovery support for fault-tolerant routing algorithms in 3d-noc architectures. In Proc. of the 7th IEEE International Symposium on Embedded Multicore/Manycore System-on-Chip, pages 67–72, 2013.

[44] I.F.A. Vis and I. Harika. Comparison of vehicle types at an automated container terminal. OR Spectrum, 26:117–143, 2004.

[45] H. Y. Bae, R. Choe, T. Park, and K. R. Ryu. Comparison of operations of agvs and alvs in an automated container terminal. Journal of Intelligent Manufacturing, 22:413–426, 2011.

[46] A. C. Adriaansen. An automated guided vehicle system in a container terminal. Master’s thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, February 2011.

[47] T. Park, Y. H. Bae, R. Choe, and K. R. Ryu. Travel time estimation and deadlock-free routing of an agy system. In H. Haasis, H. Kreowski, and B. Scholz-Reiter, editors, Dynamics in Logistics, pages 77–84, Springer, 2008.

[48] D. Briskorn, A. Drexl, and S. Hartmann. Inventory-based dispatching of automated guided vehicles on container terminals. OR Spectrum, 28:611–630, 2006.
A. Proofs of the Lemmas and Theorems

A.1. Proofs of some preliminary lemmas

We consider an AGV system with N vehicles denoted by \(v_1, \ldots, v_N\).

For convenience of the presentation, throughout this section, we use the following notations:

- \(l_i\): lane \(i\);
- \(c_i^k\): the \(k\)th zone of lane \(i\);
- \(Z_i\): the set of the zones in \(l_i\);
- \(N_i\): the set of the neighbouring lanes of \(l_i\);
- \(d\): depot;
- \(r\): crossing \(i\);
- \(v\): vehicle \(i\);
- \(f_i\): the set of the in-lanes of crossing \(i\);
- \(o_i\): the set of the out-lanes of crossing \(i\);
- \(A\): the set of the at-crossing zones of crossing \(i\);
- \(B\): the set of the SZs of all the lanes;
- \(D\): the set of all depots;
- \(A\): the set of all at-crossing zones;
- \(W\): the set of the non-EZs of all the lanes;
- \(F\): the set of the EZs of the entry-lanes of all the depots;
- \(Y_{c}^0\): the set of the neighbouring zones of zone \(c\);
- \(\tilde{Y}_c^1\): \(\{c: c \in Y_{c}^0\}\);
- \(|Z|\): the cardinality of the set \(Z\).

**Lemma 1** Any zone must belong to either of the four disjoint sets \(D, A, W\) and \(F\).

**Proof.** Let \(c\) be any zone in \(c\). Clearly, if \(c\) is a depot, \(c \in D\); otherwise \(c\) must be a zone on a lane. Now, if \(c\) is a non-EZ zone, then obviously \(c \in W\); otherwise by the guide-path layout assumption (A2), \(c\) is an EZ of either an entry-lane of a unique depot or an in-lane of a unique crossing. In the former case, \(c \in F\); while in the latter, \(c \in A\).

The lemma below shows that each zone in \(c\) has at least one neighbouring zone and cannot have itself as a neighbouring zone.

**Lemma 2** For any \(c \in c\), \(Y_{c}^0 = \emptyset\); and \(|Y_{c}^1| = 1\) for any \(c \in W \cup F\). In addition, \(c \in Y_{c}^0\) for any \(c \in c\).

**Proof.** From the definition of neighbouring zone, it is straightforward that, for each \(c \in W \cup F\), the set \(Y_{c}^0\) is non-empty. Now, if \(c \in A\), \(c\) is the EZ of an in-lane of a crossing that must have at least one neighbouring lane whose SZ is a neighbouring zone of \(c\). The conclusion that \(|Y_{c}^1| = 1\) for any \(c \in W \cup F\) is obvious.

The second part of the lemma is obvious for \(c \in W \cup F\). By Lemma 1, we still need to justify the claim for the case \(c \in A\). Let \(c\) be the EZ of \(l_i \in f_i\), i.e., \(c \in A\), and suppose, by contradiction, that \(c \in Y_{c}^0\). Then, by the definition of neighbouring zone, \(c\) is also an SZ of an out-lane \(l_{ij}\) of the crossing \(i\). Due to the guide-path layout assumption (A1), this implies that \(j = k\) and \(c\) is the only zone of \(l_{ij}\). Since \(l_{ij}\) is both the in-lane and out-lane of crossing \(i\), by the guide-path layout assumptions (A2) and (A3), it can be neither an entry-lane nor exit-lane of a depot. However, because \(l_{ij}\) has only one zone, from the guide-path layout assumption (A4) and the arguments above, we know that there can be at most one in-lane of crossing \(i\), i.e., \(l_{ij}\) itself, which has \(l_{ij}\) as a neighbouring lane. This, however, contradicts with the guide-path layout assumption (A4). □

**Lemma 3**

1. Let \(c\) be the SZ of a lane \(l_{ij}\). If \(l_{ij}\) is an exit-lane of a depot \(d_i\), then \(\tilde{Y}_c^1 = \{d_i\}\); if \(l_{ij}\) is an out-lane of a crossing \(k\), then \(\tilde{Y}_c^1 \subseteq A\); if \(l_{ij}\) is neither an exit-lane of a depot nor an out-lane of a crossing, then \(\tilde{Y}_c^1 = \emptyset\).

2. If \(c \in c\setminus (B \cup D)\), then \(|\tilde{Y}_c^1| = 1\) and \(\tilde{Y}_c^1 \cap D = \emptyset\).

3. If \(c \in A\), then \(|\tilde{Y}_c^1| = 0\) or \(1\).

**Proof.** We first note that if \(c\) is the SZ of a lane, it is clear that any zone in \(W \cup F\) cannot have \(c\) as a neighbouring zone, i.e., \(\tilde{Y}_c \cap (W \cup F) = \emptyset\), which, by Lemma 1, implies \(\tilde{Y}_c \subseteq A \cup D\). Now, suppose that \(c\) is the SZ of \(l_{ij}\), which is an exit-lane of the depot \(d_i\). By definition, \(d_i \in \tilde{Y}_c\). Furthermore, by the guide-path layout assumption (A3), \(l_{ij}\) cannot be an exit-lane of any other depot nor an out-lane of any crossing. Therefore, any zone in \((A \cup D) \setminus \{d_i\}\) cannot have \(c\) as a neighbouring zone. This shows that \(\tilde{Y}_c = \{d_i\}\). If \(l_{ij}\) is an out-lane of the crossing \(k\), i.e., \(c \in B\), then, again because of the guide-path layout assumption (A3), \(l_{ij}\) cannot be an out-lane of any other crossing nor an exit-lane of any depot. Hence, any zone in \((A \cup D) \setminus A\) cannot have \(c\) as a neighbouring zone, which implies \(\tilde{Y}_c \subseteq A\). If \(l_{ij}\) is neither an exit-lane of a depot nor an out-lane of a crossing, then none of the zones in \((A \cup D)\), can have \(c\) as a neighbouring zone and thus \(\tilde{Y}_c = \emptyset\).

If \(c \in c\setminus (B \cup D)\), then \(c\) must be the \(k\)th, \(k \geq 2\), zone on a lane. It is easy to show that the only zone that has \(c\) as the neighbouring zone is the \(k-1\)th zone on the same lane (the...
proof is lengthy but straightforward). Hence, \(|\tilde{Y}_c|=1\) and \(\tilde{Y}_c \cap \mathcal{D}_k = \emptyset\).

Now, let \(c \in \mathcal{A}\), then \(c\) is an EZ of an in-lane \(l\) of a crossing. If \(l\) contains more than one zone, then \(c \in \mathcal{C}_l \setminus \mathcal{D}_k\) and thus, by the second claim of the lemma, \(|\tilde{Y}_c|=1\). If \(l\) contains only one zone, which is \(c\), then by the guide-path layout assumption (A3), there can be only three possibilities: (i) \(l\) is an exit-lane of a depot \(d\); (ii) \(l\) is an out-lane of a crossing; (iii) \(l\) is neither an exit-lane of a depot nor an out-lane of a crossing. In case (i), from the first part of the lemma, we know that \(\tilde{Y}_c = |d|\) and \(|\tilde{Y}_c|=1\). In case (ii), by the first claim of the lemma, \(\tilde{Y}_c \subseteq \mathcal{A}\); but, according to the guide-path layout assumption (A4), there can be only one in-lane of crossing \(k\) that has \(l\) as its neighbouring lane; hence, \(\tilde{Y}_c\) is either a singleton or empty. In case (iii), \(\tilde{Y}_c\) is clearly empty. □

Lemma 4 If a vehicle leaves or goes ahead to pass a zone \(c_1 \in \mathcal{C}_l\), then it starts occupying a zone \(c_2 \in \mathcal{C}_l\). If a vehicle starts occupying a zone \(c_2 \in \mathcal{C}_l\), then it leaves or goes ahead to pass a zone \(c_3 \in \mathcal{C}_l\).

Proof. Suppose that a vehicle leaves \(c_1\) by an LEA. The LEA then changes the state of the vehicle from \((c_1,c_2) \in \mathcal{S}_k\) to \((c_1,c_2) \in \mathcal{S}_k\), where \(c_2 \in \mathcal{Y}_{c_1}\) (by Lemma 2). This implies that the vehicle occupies only \(c_1\) before the event and occupies both \(c_1\) and \(c_2\) afterwards. Suppose that a vehicle goes ahead to pass \(c_1\) by a GOA. The GOA then changes the state of the vehicle from \((c_1,c_2) \in \mathcal{S}_k\) to \((c_1,c_2) \in \mathcal{S}_k\), where \(c_2 \in \mathcal{Y}_{c_1}\). This means that the vehicle occupies only \(c_1\) and \(c_2\) before the event and occupies \(c_1\) or \(c_2\) afterwards.

For the second part of the lemma, we first note that, if a vehicle starts occupying a zone, it must be a consequence of the occurrence of a vehicle event. Furthermore, we show that this event cannot be an ARR. By contradiction, assume that the event was an ARR; we then know that, before the event, the state of the vehicle must have the form as either \(s=(c_1,c_2) \in \mathcal{S}_k\) or \(s=(c_1,c_2) \in \mathcal{S}_k\). If \(s=(c_1,c_2) \in \mathcal{S}_k\) then the zones \(c_1\) and \(c_2\) were both occupied by the vehicle before the ARR, but the event would change the state of the vehicle to \((c_1,c_2) \in \mathcal{S}_k\), where \(c_2 \in \mathcal{Y}_{c_1}\), which means that \(c_2\) was the only zone occupied by the vehicle after the event (i.e., no new zone is occupied by this event). The proof is very similar for the case \(s=(c_1,c_2) \in \mathcal{S}_k\) hence omitted. Thus, we know that a vehicle can start occupying a zone only by either an LEA or GOA. Lastly, from the proof of the first part of the lemma we see that if the vehicle starts occupying \(c_2\), then it must leave or go ahead to pass a zone in \(c_1 \in \tilde{Y}_{c_2}\). □

A.2. Proof of the collision and deadlock avoidance

A.2.1. Proof of the collision avoidance

First, we present a technical lemma.

Lemma 5 By Rule 1 and Rule 2, for each \(c \in \mathcal{B}\), at most one vehicle can start occupying \(c\) at any point in time.

Proof. First, from Lemma 4, we know that, to start occupying a zone \(c\), a vehicle has to trigger an LEA or GOA to leave or go ahead to pass a zone in \(\tilde{Y}_c\). Let \(c \in \mathcal{B}\), namely \(c\) is an SZ of a lane \(l\). Then, by the guide-path layout assumption (A3), \(l\) can be in one of the three cases described in the first claim of Lemma 3. If \(l\) is an exit-lane of a depot \(d\), then by Lemma 3, \(\tilde{Y}_c = |d|\). Thus, to start occupying \(c\), a vehicle needs to leave or go ahead to pass \(d\). By Rule 1, at most one vehicle can do this at any point in time. If \(l\) is an out-lane of a crossing, then, from Lemma 3, we see that \(\tilde{Y}_c \subseteq \mathcal{A}\). Thus, to start occupying \(c\), a vehicle needs to leave or go ahead to pass a zone in \(\mathcal{A}\). Because of Rule 2, at most one vehicle can do this at any point in time. If \(l\) is neither an exit-lane of a depot nor an out-lane of a crossing, then the set \(\tilde{Y}_c\) is empty and thereby no vehicle can start occupying \(c\) at all times. □

Proof of collision avoidance:

We show, by contradiction, that the conditions (C1) and (C2) are both satisfied all the time. Firstly, if (C1) does not hold all the time, then there must be some event(s) occurring at a time instant \(t\), such that each non-depot zone is occupied by at most one vehicle before \(t\) and either of the following situations appears at \(t\): (1) at least two vehicles start occupying the same zone \(c\), which was available before the event(s); (2) at least one vehicle starts occupying a zone \(c\), which was occupied by one vehicle before the event(s). By Lemma 4, in both cases, the vehicles must leave or go ahead to pass the zones in \(\tilde{Y}_c\). Further, note that, in case (1), the zone \(c \in \mathcal{C}_l \setminus \mathcal{D}_k\). This is because any depot is always available and, by Lemma 5, at most one vehicle can start occupying any zone in \(\mathcal{B}\) at any time. Thus, from the second claim of Lemma 3, we know that, in case (1), each of the vehicles leaves or goes ahead to pass the same non-depot zone by an LEA or GOA event, which means that the vehicles occupy the same non-depot zone before the events (a contradiction). Case (2) also cannot be true as Rule 3 and Rule 4 require that a vehicle can start occupying a zone only if the zone is available.

Next, we show that (C2) always holds, i.e., there cannot be two vehicles having conflicting states at any crossing. If this was not the case, then there exists some moment \(t\) and two vehicles \(v_i\) and \(v_j\) having, respectively, state \(s\) and \(s'\) at \(t\), such that one of the three conflicting cases (X1)-(X3) happens at a crossing \(k\); and there was no conflicting state at any crossing before that (for it is assumed that there is no initial collision). Here we only show that the case in which \(s,s' \in \mathcal{S}_k\) and \((s_2,s_3) \in \mathcal{B}((s,s'),s')\) is not possible (the proofs for the rest can be lengthy but share a similar line of reasoning and are thereby omitted). First, note that the two vehicles cannot change their states to be \(s\) and \(s'\) simultaneously at \(t\), as otherwise Rule 2 is violated (for two crossing-passing GOAs at crossing \(k\) are triggered at the same time).
Consequently, at time $t$ either $v_i$ triggers a GOA, which changes the state to $s$, while $v_j$ is in state $s'$; or $v_i$ triggers a GOA, which changes the state to be $s'$, while $v_j$ is with $s$. However, neither situation is allowed by Rule 4.

A.2.2. Proof of the deadlock avoidance

We first note that the the result of collision avoidance still holds here, since Rules 2' and 4' are stronger than Rules 2 and 4. In other words, the AGV system always satisfies the conditions (C1) and (C2) in Section 3.1.

We also point out some important properties of the operation graph:

(P1) For each off-crossing zone $c$, there is a fixed non-empty set of arcs incident on $c$ at any time; for each at-crossing zone $c$, the set of arcs incident on $c$ can be empty and time varying;

(P2) If $(c_1, c_2) \in E$, then $c_2 \in Y_{t_i}$;

(P3) If $c_1$ and $c_2$ are, respectively, the current and next zone of a vehicle, then $(c_1, c_2) \in E$.

(P4) There can be at most one arc incident on any non-depot zone.

The properties (P1)-(P3) are direct from the definitions of the vehicle state and the operation graph. The property (P4) can be proved as follows: let $c \in C \setminus D$. Then, by Lemma 1, $c \in W \cup F \cup \mathcal{A}$. If $c \in W \cup F$, then Lemma 2 shows that $|Y_{t_i}| = 1$. From the property (P2), we see that there can be at most one arc incident on $c$. If $c \in \mathcal{A}$ the conclusion can be obtained from the definition of the arcs and the condition (C1).

We will sometimes interchange the words “zone” and “vertex” where no confusions arise.

As a key step in our proof of the deadlock avoidance, we show that there is no black cycle in the operation graph at any time. First, we need the following lemma:

**Lemma 6** An ARR cannot change the colour of a zone from white or grey to black.

**Proof.** By definition, an ARR is responsible for two types of vehicle state transitions. If an ARR changes a vehicle state from $(c_1, c_2, c_3) \in S_2$ to $(c_2, c_3, c_1) \in S_1$, then $c_2$ is the only resulting black zone, which was already black before the state transition; if an ARR changes a vehicle state from $(c_1, c_2, c_3) \in S_3$ to $(c_2, c_3, c_1) \in S_2$, then $c_2$ is the only resulting black zone, which was already black before the state transition. Therefore, the claim is justified. $\square$

**Lemma 7** An LEA or GOA cannot add arcs in the operation graph. An ARR adds an arc in the operation graph if and only if it causes a vehicle to arrive at an at-crossing zone and the added arc is incident on the at-crossing zone.

**Proof.** From the definition of the operation graph, we know that an arc can be added only when an event is triggered to turn an at-crossing zone to be the current zone of a vehicle from not being the current zone of any vehicle. In addition, the added arc is incident on the at-crossing zone. Consequently, it is straightforward to see that an arc cannot be added by either an LEA or a GOA.

Now, suppose that a vehicle, say $v$, arrives at a zone $c_2$ by triggering an ARR, which means that the ARR changes the state of the vehicle from $(c_1, c_2, c_3) \in S_2$ to $(c_2, c_3, c_1) \in S_1$ or from $(c_1, c_2, c_3) \in S_3$ to $(c_2, c_3, c_1) \in S_2$. In both cases, $c_2$ becomes the current zone of $v$ after the state transition. In addition, $c_2$ could not be the current zone of any vehicle before the transition because it was not the current zone of $v$ and could not be occupied by any other vehicle (due to condition (C1)). Therefore, the ARR adds an arc in the operation graph if and only if $c_2$ is an at-crossing zone; and the added arc is incident on $c_2$. $\square$

**Lemma 8** Non-crossing-passing events (i.e., ARRs, off-crossing LEAs and off-crossing GOAs) cannot introduce black cycles in the operation graph.

**Proof.** First, from Lemma 6 we know that an ARR does not turn any zone from white or grey to black. Furthermore, note that an ARR may add an arc in the operation graph, but the arc cannot be in a black cycle. In fact, by Lemma 7, an ARR adds an arc $(c_2, c_3)$ in the operation graph only if it makes a vehicle arrive at $c_2 \in \mathcal{A}$. By Lemma 3, we have either $\tilde{Y}_{t_i} = \{c_1\}$, where $c_1$ is a zone in $C_i$ or $\tilde{Y}_{t_i} = \emptyset$. Thus, from the property (P2) of the operation graph, it follows that $c_i$ is the only possible zone on which an arc pointing to $c_2$ can be incident. However, in this case, it is easy to see that $c_2$ must be coloured white by the ARR. Therefore, the added arc $(c_2, c_3)$ cannot be an arc of any black cycle.

On the other hand, we know from Lemma 7 that an off-crossing LEA or off-crossing GOA does not add any arcs in the operation graph. While the event may colour a zone black, the black zone cannot be involved in any black cycle. We only justify this point for an off-crossing LEA (the proof for an off-crossing GOA is similar and hence omitted). Suppose that an off-crossing LEA changes the state of a vehicle from $(c_1, c_2, c_3)$ to $(c_2, c_3, c_1)$, where $c_1 \in C_i \setminus \mathcal{A}$. Due to Rule 3, the zone $c_2$ is coloured from white to black if $c_2 \in C_i \setminus D$. It follows that, in this case, $\tilde{Y}_{t_i} = \{c_1\}$. In fact, obviously $c_1 \in C_i \setminus F$ (otherwise $c_2 \in D$); and hence from Lemma 1, we know that $c_2 \in W \cup D$. If $c_1 \in W$, then clearly $c_2 \in C_i \setminus B$ and, by Lemma 3, $\tilde{Y}_{t_i} = \{c_1\}$; if $c_1 \in D$, then $c_2$ is the SZ of an exit-lane of $d_1$ and we have again $\tilde{Y}_{t_i} = \{c_1\}$ by Lemma 3. By the property (P2) of the operation graph, this means that $c_1$ is the only possible zone on which an arc pointing to $c_2$ can be incident. However, it is easy to check
that $c_1$ is coloured from black to grey by the off-crossing LEA. Therefore, $c_2$ cannot be in any black cycle. □

**Lemma 9** There is no black cycle in the operation graph at any time.

**Proof.** First, note that it is assumed that there is no initial black cycles. Thus, by Lemma 8 and Rule 2', if there was a black cycle, it must have resulted from a crossing-passing event. By contradiction, suppose that there was a crossing-passing event that does introduce black cycles in the operation graph; and let this happen at time instant $t$ for the first time. By Rule 2', there is only one crossing-passing event that takes place at $t$. Assume that the event is either an at-crossing LEA, which changes the state of a vehicle from $(c_1, c_2, c_3) \in S_1^c$ to $(c_1, c_2, c_3) \in S_2^c$ or an at-crossing GOA, which changes the state of a vehicle from $(c_1, c_2, c_3) \in S_2^c$ to $(c_1, c_2, c_3) \in S_3^c$. Now, taking any operational vehicle that is in a zone and intends to trigger an LEA, it can be easily seen that each blocked operational vehicle must be in a zone and intends to trigger an LEA, which is not permitted by the traffic control.

We know that at least one of the operational vehicles is in a black zone, as otherwise all vehicles would be in depots (note that the non-operational vehicles are all in depots as assumed) and by Rule 3 at least one operational vehicle is not blocked. Now, taking any operational vehicle that is in a black zone $c$, if its next zone $c'$ is occupied, then there must be exactly one other operational vehicle in the zone, which is black. In addition, due to the property (P3) of the operation graph, $(c, c')$ is an arc in $G_c$. Since there is no black cycle in $G_c$, it follows that there must be an operational vehicle $v$, with the state $s=(c, c, c)$, whose next zone $c_2$ is available and hence white. Apparently, $c_1$ (the current zone of $v$) must be an at-crossing zone and belongs to $A_k$ for a $k$, as otherwise an LEA is allowed to occur for $s$ by Rule 3. In addition, we see from Rule 4' that the only reason $v$ is blocked is that triggering an LEA would introduce a black cycle containing $c_2$ in the operation graph. Note that the LEA, if that happens, would colour $c_2 \in B_k$ black, yet not add any arcs in the graph (Lemma 7). Thus, there must be a cycle $\Gamma$ in $G_c$ that contains $c_2$, in which all the other zones are black. By the property (P4) of the operation graph and noting that any depot is always white, we further see that such a cycle must be unique. In addition, it is clear that $c_1$ cannot be in $\Gamma$, for it would be coloured grey by the occurrence of the LEA, which would make $\Gamma$ a black cycle. Hence, by the property (P2) of the operation graph and Lemma 3, there exists an operational vehicle $v_i (v_i \neq v)$, which is in a $c_2 \in \overset{\Rightarrow}{A} (c_2 \neq c_1)$ contained in $\Gamma$, and is with $c_3$ as its next zone (i.e., $v_i$ is in the state $(c_3, c_2, c_2)$). This gives rise to a contradiction, for then $v_i$ can be allowed to trigger an LEA by Rule 4', which would colour $c_2$ black and $c_3$ grey; thus $v_i$ is not blocked. In fact, this LEA would not introduce a black cycle; because if it did, then, as it does not add arcs in the operation graph, there would be a cycle other than $\Gamma$ in $G_c$, containing $c_2$ but not $c_3$, with all the zones except $c_2$ black.

B. Supporting Arguments (Proofs) for Some Statements in Section 5.2

A – From an implementation point of view, this is fine, as the vehicle is not physically occupying the zone $c_3$ when its state is $(c_1, c_2, c_3)$ (see also Section 2.2).

B – Clearly $c_2$ is the SZ of a lane; otherwise, from the second claim of Lemma 3, we know that $c_1 = [c_3]_1$ and $c_1$ and $c_2$ would be on the same lane. According to the first claim of Lemma 3, $c_2$ must be the SZ of an out-lane of a crossing and $c_1$ is an at-crossing zone of the crossing.

C – This is a strengthened version of the guide-path layout assumption (A3) in Section 2.1.

D – $\nu_{\text{change}_m, c} \cap \nu_{\text{blocked}} = \emptyset$, because each vehicle in $\nu_{\text{change}_m, c}$ is in the EZ of a lane but any vehicle in $\nu_{\text{blocked}}$ cannot be.

E – Let $s$ be the state of a vehicle in $\nu_{\text{blocked}}$. The conclusion is obvious if $s \in S^c$, as the vehicle is already in a zone in $\Omega$. For the case $s=(c_1, c_2, c_3) \in S^c$, if $c_2 \in \Omega$, then the vehicle will be in $c_2$ by triggering just one ARR; if $c_1 \in \Omega$, it follows that we also have $c_2 \in \Omega$, as otherwise $c_2 = c_1'$ and thus the zone $c_2'$ would be occupied by both the vehicle and the broken vehicle when the breakdown happens (which contradicts with the collision avoidance result in the previous section). Similar reasoning applies to the case where $s \in S_b$.

Now we show that, after Step 2 and before the emergency traffic control scheme finishes, any vehicle occupying a zone $c \in \Omega$ must be a vehicle in $\nu_{\text{blocked}}$ in the zone $c$. In fact, if it was not the case then there must be a vehicle $v \in \nu_{\text{blocked}}$ that is not occupying any zone in $\Omega$ when the breakdown occurs, but starts occupying a zone in $\Omega$ during the emergency traffic control. However, by Lemma 4, this would require the triggering of a crossing-passing LEA or GOA (at
crossing \( k \), which is not possible due to the suspension of the crossing token (as per Step 1).

F \ – Such a lane exists due to the guide-path layout assumption (A5).

G \ – This vehicle is obviously not the broken one since the broken vehicle now only occupies zones on lane \( i’ \), which is not an out-lane of any crossing.

H \ – For the vehicles \( V_{\text{blocked}} \) occupying non-EZs on the lane \( i_2 \), the claim can be proved by the same arguments in (E) with \( j \) substituted by \( n_i \). If a vehicle \( v \) in \( V_{\text{blocked}} \) is occupying a zone on a lane \( j \in L_{\text{blocked}} \setminus \{i_1\} \), the state of the vehicle, denoted by \( s \), can have the following possibilities: (1) \( s = (c_1^1, c_1^2) \in S_1^i \) and \( c_1^i \) is a zone on lane \( j \); (2) \( s = (c_1^1, c_1^2, c_1^3) \in S_2^i \) with \( c_1^i \) and/or \( c_2^i \) being a zone on lane \( j \); (3) \( s = (c_1^1, c_2^i) \in S_3^i \) with at least one of the zones \( c_1^1, c_2^i, c_3^i \) is on the lane \( j \). Note that case (1) is trivial. For case (2), one only needs to note that, if \( c_1^1 \) is on lane \( j \), then \( c_2^i \) must also be on the same lane, for otherwise it is easy to see from the definition of \( L_{\text{blocked}} \) that the vehicle \( v \) would have conflicted with the broken vehicle at crossing \( k \) when the breakdown occurs. For case (3), one may use the same reasoning to show that \( c_2^i \) must be on lane \( j \) if either \( c_1^1 \) or \( c_3^i \) is on that lane.

I \ – Since \( j \) is an in-lane of crossing \( k \), \( k \neq k' \) because of the guide-path layout assumption (A6).

J \ – By the guide-path layout assumption (A7) and the fact that each lane in \( L_{\text{blocked}} \) is an in-lane crossing \( k \), we have the situation where there is an out-lane of crossing \( k' \) that is not in \( L_{\text{blocked}} \). In addition, this lane must be different from lane \( i_2 \) due to the guide-path layout assumption (A3).

K \ – By Lemma 3, the vehicle is either in an at-crossing zone of a crossing or in a depot at the moment it is informed of the breakdown and maintains its state after Step 1 due to the suspension of the crossing and depot tokens by the step.

L \ – Clearly, in this case, the vehicle has the SZ of a lane in \( L_{\text{blocked}} \) as the next zone, as otherwise it would have the SZ of lane \( i_2 \) as the next zone and hence \( k' = k \).

M \ – The existence of such a lane can be justified by a similar reasoning as given in (J).

N \ – In this case, the vehicle has the SZ of lane \( i_2 \), i.e., the zone \( c_2^i \) as the next zone. If this was not true, then the vehicle would have the SZ of a lane in \( L_{\text{blocked}} \) as the next zone. This, however, implies that there was a lane which was both an in-lane and out-lane of the crossing \( k \) (before Step 3). This contradicts the assumption (A6).

O \ – Such a lane exists, otherwise the vehicle would be in the set \( V_{\text{blocked}} \) because it is occupying \( c_2^i \) the EZ of an in-lane of crossing \( k \) and \( (c_2^i, c) \in R_{L}(c_1, c_2) \) for all \( (c_2^i, c) \in R_k \).

P \ – The existence of such a lane can be justified by a similar reasoning as given in (J).

Q \ – Such a lane \( j' \) exists, because otherwise lane \( j \) would have been contained in the set \( L_{\text{blocked}} \) and removed from the set of in-lanes of crossing \( k \) in Step 3.

C. Proof of the Collision and Deadlock Avoidance with Emergency Traffic Control

C.1. Proof of the collision and deadlock avoidance with vehicle removal

First, we see that Step 1 simply makes each healthy vehicle trigger feasible events to be in a zone and temporarily holds its intention to trigger a new event. The removal of the faulty vehicle in Step 2 cannot cause collisions and deadlocks as it simply colours some zones to be white and possibly removes an arc in the operation graph. Therefore, no collisions and deadlocks can appear due to Step 1 and Step 2; as is also the case after the system resumes, as guaranteed by the normal traffic control rules presented in Section 3.

C.2. Proof of the collision and deadlock avoidance without vehicle removal

We only show the proof for the on-lane breakdown case described in Section 5.2.1. The proofs for the other two cases are similar and are hence omitted.

Proof of the collision avoidance:

The same proof for the collision avoidance in Section A.2.1 applies here as all arguments still hold with the emergency control scheme.

Proof of the deadlock avoidance:

As a key step, we first show that the emergency control scheme cannot induce any black cycles in the operation graph. First, in Step 1, the temporary suspension of all the crossing-passing events can be simply seen as (finite) delays of the intention to trigger crossing-passing events from the vehicles. In Step 2, the ARRs are the events that are always allowed to be triggered by the normal traffic control. Hence, Step 1 and Step 2 simply comply with the normal traffic control and therefore cannot cause black cycles. In Step 3, lane \( i \) is removed and two new lanes (i.e., lane \( i' \) and lane \( i'' \)) are added; and each of the blocked vehicles in \( V_{\text{blocked}} \), which is in a zone on lane \( i' \), chooses its new next zone. Note that this leads to both removal and addition of arcs in the operation graph that are not caused by the occurrence of vehicle events. We now justify that this step cannot generate black cycles either. By contradiction, suppose that the step did cause black cycles. Then, note first that the black cycles must be caused by the addition of arcs just mentioned for the following two reasons: (1) no
crossing-passing events can occur due to Step 1 and no non-crossing-passing events can cause black cycles according to Lemma 8, (2) no black cycles can be induced by arc removal. Furthermore, the black cycles must be caused by the adding of lane \( i' \) and the resetting of the next zones for the vehicles in \( V_{\text{black}} \). as lane \( i' \) is simply part of the previously existing lane \( i \). This would imply that the black cycles must contain a zone in \( c_1^{i'}, c_2^{i'}, ..., c_{j-1}^{i'}, \) which is impossible. In fact, by the property \((P2)\) of the operation graph (see Section A.2.2) and the second claim of Lemma 3, the only arc pointing to \( c_k^{i'}, k = 2, 3, ..., j-1 \) must be incident on \( c_{k-1}^{i'} \); but there can be no arc incident on \( c_1^{i'} \) due to the property \((P2)\) and the first claim of Lemma 3, as lane \( i' \) is neither an exit-lane of a depot nor an out-lane of a crossing. Finally the changing of the next zones of the vehicles in \( V_{\text{change,ns}} \) in Step 4 cannot generate black cycles due to the black cycle prevention measure by using the SZ splitting concept.

Now, note that by the emergency control scheme, we have in fact isolated the faulty vehicle from the rest of the AGV team: for the running of the healthy vehicles, we could simply “remove” the faulty vehicle and “cut” the zones occupied by it (i.e., either \( c_j^{i'} \) or \( c_j^{i'} \) and \( c_{j+1}^{i'} \), depending on the state of the faulty vehicle) from the guide-path. This is because lane \( i' \) is neither an out-lane of any crossing nor an exit lane of a depot. We further see that, when excluding the zones occupied by the faulty vehicle (such that the lane \( i' \) becomes shorter) the left guide-path would still satisfy the assumptions \((A1)-(A4)\). Therefore, the healthy vehicles are guaranteed to be free of collisions and deadlocks under the normal traffic control after the emergency traffic control ends, regardless of their routes.