IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 4 January 2024, accepted 30 January 2024, date of publication 5 February 2024, date of current version 9 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3362293

RESEARCH ARTICLE

Traffic Management of Multi-AGV Systems by Improved Dynamic Resource Reservation

PARIKSHIT VERMA[1], JOSEP M. OLM[2,3], AND RAÚL SUÁREZ[3], (Senior Member, IEEE)

[1]Tavil Ind, S.A.U., 17854 Sant Jaume de Llierca, Spain
[2]Department of Mathematics, Universitat Politècnica de Catalunya, 08028 Barcelona, Spain
[3]Institute of Industrial and Control Engineering, Universitat Politècnica de Catalunya, 08028 Barcelona, Spain

Corresponding author: Josep M. Olm (josep.olm@upc.edu)

The work of Parikshit Verma was supported by Generalitat de Catalunya through the Industrial Doctorate Project under Grant 2021 DI 16. The work of Josep M. Olm was supported in part by the Government of Spain through the Agencia Estatal de Investigación Project under Grant PID2021-122821NB-I00, and in part by the Generalitat de Catalunya through the AGAUR Project under Grant 2021 SGR 00376 and the Industrial Doctorate Project under Grant 2021 DI 16. The work of Raúl Suárez was supported in part by the Spanish Government under Project PID2020-114819GB-I00, and in part by Generalitat de Catalunya through the Industrial Doctorate Project under Grant 2021 DI 16.

ABSTRACT Automated guided vehicles (AGVs) are widely used for material handling in warehouses and automated production lines due to their high efficiency and low failure rate with respect to human operated load carriers. However, AGVs usually interact with each other because of the restricted capacity of the layout, and conflicts arise. Although many traffic scheduling algorithms have been proposed to address the AGV fleet control problem, most of them are inefficient for collision and deadlock avoidance in dynamic environments. This paper proposes an improved dynamic resource reservation (IDRR) based method which renders time-efficient task completion and deadlock-free movements of multiple AGVs in a manufacturing system. Unlike traditional approaches, most of which adopt a dynamic single agent reservation of the shared resource points and/or force path deviations, IDRR exploits dynamic multiple reservations of shared resource points. This is combined with a conflict detection and resolution method that accommodates the AGV motions when they meet at a resource point. Extensive, realistic simulation results demonstrate the feasibility and efficiency of the proposed collision and deadlock prevention method in productivity, travelled distance, and time completion of the assigned tasks. The proposal can be implemented on both central and local controllers.

INDEX TERMS Multi-AGV systems, collision and deadlock avoidance, dynamic resource reservation, traffic control.

I. INTRODUCTION

The use of Automated Guided Vehicles (AGVs) to transport materials, which dates back to the 1950's [1], is rapidly increasing and has a high potential growth in the next years [2] within the context of Industry 4.0 and flexible manufacturing systems (FMS). However, the deployment of fleets of AGVs, the so-called Multi-AGV Systems (MAGVS), entails a number of challenging problems that run from task allocation to route execution [2].

Among these problems stands the aim of preserving the system liveness [3], [4], i.e. of coordinating the motion of a set of AGVs so that all the transport tasks can be completed timely and in a deadlock-free way, namely without collisions and/or deadlocks. Indeed, such an issue has been attracting research interest since decades [5], and it is seen as a major concern in risk analysis of AGV operation [6].

Conflicts may be faced not only through an efficient control strategy but also taking into account other features and parameters such as the specific industrial layout, the size of the fleet, or the physical dimensions [7] and kinematic properties [8], [9] of the AGVs. In any case, regarding the overall performance of the system, a combination of avoidance plus recognition and resolution seems to be the best alternative [10]. Hence, conflict avoidance and resolution methods may be classified along three different

The associate editor coordinating the review of this manuscript and approving it for publication was Emanuele Crisostomi.

axes: control architecture, route planning and execution, and control algorithm [11].

The control architecture is centralized when a single computer manages the whole MAGVS. Alternatively, it is decentralized when intelligence is split among the system components, which communicate between them, and decisions are taken locally [12], [13]. Although most of the literature deals with centralized schemes, the seek of increased robustness, flexibility, and scalability are making decentralization a current trend in the area [2], [14].

The path that an AGV has to follow from an initial to a final position may be computed offline, or online during execution. This features static and dynamic routing algorithms, respectively, while the combination is termed as semi-dynamic [15] or hybrid [11], [16]. Static routers, although able to reduce the possibility of conflict by an efficient planning [17], [18], and/or use Internet of Things (IoT) resources [19], answer poorly to environmental changes because their response capability relies on waiting times. In turn, fully dynamic routers are at an early stage and still have room for improvement in terms of optimality [15] or layout genericity [20], which makes hybrid solutions a current interesting option. The reader is referred to [15] for a thorough and up-to-date classification of the existing literature in static, semi-dynamic, and fully dynamic collision avoidance algorithms. Similar to routing strategies, multi-agent path finding (MAPF) algorithms are also an interesting topic [21], [22], [23]; the main difference with motion planning strategies as the one presented in this paper is that, in case of potential conflicts, the latter works on the basis of shortest paths with conflict avoidance or resolution based on temporal synchronization of movements, while the former looks for longer, deadlock-free paths. From the optimization side, these strategies target different metrics such as completion time, operation costs, waiting times of AGVs, travelling distance, and delivery delay time, among others [15].

Regarding the control algorithm, strategies are classified as time window, Petri net, and zone control-based approaches [11], [24]. Solving shortest path problems within time windows is at the core of the routing algorithms of the first type [25]. Petri nets, which are a discrete-event system tool that allows to represent systems with concurrency, have been often used to model layouts and paths to be followed by the AGVs, and to develop collision avoidance algorithms [26]. Instead, zone-control methods divide the layout into different areas, and traffic is controlled in such a way that just a limited number of AGVs at a time is allowed in each area [2]. Another excellent classification of AGV traffic control literature, now from the control algorithm side, is available in [11].

Time window-based strategies are mainly used in unidirectional layouts [11], while Petri net-based controllers have a high computational burden. Hence, zone-control has become the most popular traffic management technique [27], and several interesting zone-control algorithms recently published are discussed below. Common to all of them is the division of the layout into zones, known as resource points, with each AGV occupying a single zone.

The chain of reservations (COR) method [28] considers a reservation of the planned path for an AGV in such a way that no other AGV can access a common resource point until the first one frees it. In this way, collisions and deadlocks are definitely avoided. However, reservations are static as they are computed before sending the route plan to the AGV, and this may lead to unavoidable waiting times for AGVs with shared paths at their respective initial positions before they can start moving.

The dynamic resource reservation (DRR) technique [29] proposes a reservation of resource points based on a continued update of the travelling information of each AGV, in particular of the resource points left to visit during task execution (known as the residual route). Resource points are made free once the reserving AGV has visited them, but the common (or shared) residual route is not accessible to any other AGV. Significant improvements in average waiting times and total travel time are reported, even when the number of AGVs and tasks scale up. The acquiring parking spots on the fly (APF) strategy [30] refines DRR by allowing AGVs in a waiting state to look for a nearby parking spot out of the main path while the requested resource point is not freed, thus alleviating potential traffic congestions due to road blocking.

The structural on-line control policy (SOCP) [31] divides the layout into deadlock free and deadlock risk zones, usually corresponding to unidirectional and bidirectional paths, respectively. Therefore, the path plan is formed by a collection of consecutive segments of both types. Resource points in deadlock free segments are sequentially reserved, occupied, and made free once the AGV has visited them. Instead, deadlock risk segments are fully reserved for a single AGV and no one else may access the shared residual route. The SOCP algorithm outperforms COR and DRR, but when no deadlock free segments can be identified in shared routes it boils down to DRR. An improvement of the SOCP for a square topology layout is reported in [32].

The so far discussed COR, DRR and SCOP policies fall within the static routing category: a route is planned for every task using shortest path-based algorithms, such as Dijkstra [33], and it is never altered during execution. Differently, the spare zone-based hierarchical (SZH) coordination algorithm [11] proposes a hybrid approach in which spare resource points are identified next to shared resource points. Then, the lower priority AGV is diverted to the spare zone while the higher priority AGV goes through, and resumes its original path afterwards. SZH shows a much better performance than COR, but compromises additional layout resource points to those of the initially assigned path. Besides, spare zone allocations are made ahead of time, before the task starts, which results in longer waiting times when the number of tasks scales up due to eventual failures in locating spare

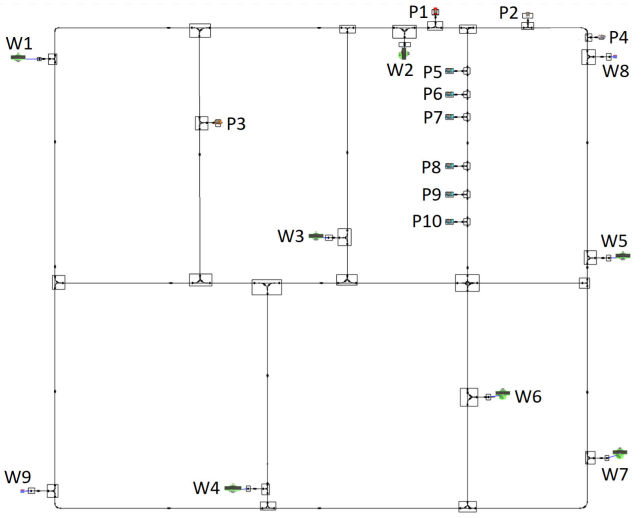

FIGURE 1. Example of a FMS layout where W1 to W7 are pick-up stations, W8 and W9 are drop-off stations, and P1 to P10 are parking stations.

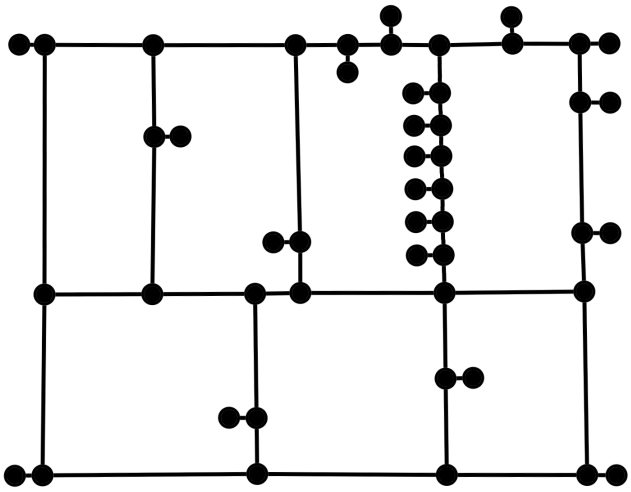

FIGURE 2. Graph model of the layout portrayed in Figure 1.

points. This limits the effectiveness of SZH to workspaces with a high enough number of spare zones. A dynamic variation of the strategy, known as dynamic spare point application (DSPA), has exhibited better performances in terms of timespan, waiting time, and travelled distance [24].

Two key drawbacks common to these zone-control based architectures are: (i) AGVs are heading to a parking station after completing a task and before undertaking a new assignment: this entails extra travelled distances per task, which results in longer execution times and higher energy consumption; and (ii) only one AGV is allowed to travel through a shared zone; consequently, AGVs requesting access to an already busy shared zone have to either wait for the AGV occupying it to clear the area [28], [29], [31], and/or to divert their routes through nearby spare resource points when available [11], [24]. Hence, waiting times can scale up undesirably in layouts with fewer nodes and longer aisles.

This paper presents an improved DRR (IDRR) collision and deadlock free MAGVS control policy where both issues are overcome. Namely:

1) in IDRR, new tasks can be assigned without requiring the AGV to first head to a parking place; AGVs are directed to parking places after completing a task just in case the pending tasks list is empty;

2) IDRR lets different AGVs to travel along shared zones in a deadlock-free way. This is achieved, on the one hand, allowing multiple reservations of shared resource points, which are defined so that they can accommodate more than one AGV, and, on the other hand, using a set of traffic rules that detects, classifies, and solves conflicts when they eventually arise. The conflict solving actions may run from waiting states to forcing the AGVs to reach the same resource point and maneuvering therein to sort out the situation.

3) Differently from spare zone-based approaches, IDRR is a static routing strategy: AGVs do not divert from their originally devised paths, which are obtained using a shortest distance criterion.

4) IDRR relies on a two layer architecture that combines the symbolic information associated to the layout representation and the geometric information linked to the specific position of the AGV within the workspace.

5) IDRR may be implemented in a centralized or a semi-decentralized way; however, for the sake of brevity, here the presentation is made as a centralized approach.

The paper is organized as follows. The problem to be solved and the layout modeling are introduced in Section II. Then, the traffic management strategy is described in Section III, and a set of illustrative conflict resolution examples is gathered in Section IV. Realistic numerical simulations are presented in Section V. Finally, conclusions and suggestions for further research are drawn in Section VI.

II. PROBLEM DESCRIPTION AND LAYOUT MODELING

An industrial process of pallet transportation in an FMS is considered. This encompasses, for example, to carry a pallet from a palletizing station to a storing area, and to move pallets between processing stations, among other actions. In any case, we refer to the origin and destination of the AGVs as pick-up and drop-off stations, respectively.

A completely known, Manhattan grid-like industrial layout is considered, where we distinguish: (a) pick-up and drop-off stations, both termed as working (W) stations, (b) parking (P) stations, where AGVs may also charge their batteries, and (c) aisles connecting these elements (see Figure 1). To refer to a generic station, either W or P, we will use WP. As for the aisles, we consider three and four aisle intersections, also known as three-way (T) and four-way (X) crossings. To refer to a generic intersection, either T or X, we will use TX. In a typical manufacturing system WP stations are located next to the aisles, hence the small path to reach them from the aisle implicitly creates a TX crossings. Besides, WP stations are

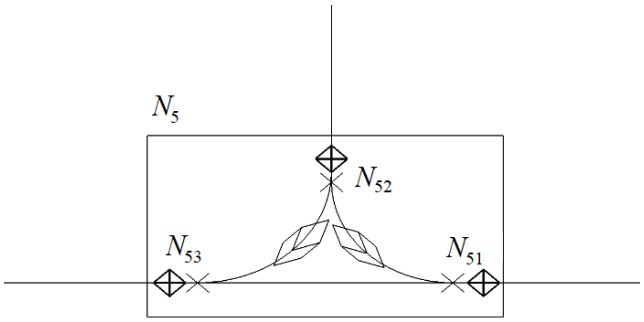

FIGURE 3. Example of a node (N_5) containing three CPs (N_{51}, N_{52}, N_{53}).

assumed to allow single occupancy. Moreover, it is assumed that the number of P stations in the layout is greater or equal that the number of deployed AGVs. Finally, the aisles are assumed bi-directional, but with at most one AGV travelling in an aisle at a time.

The layout is modelled as a weighted, undirected graph $G = (N, E)$, where $N = \{N_i, 1 \leq i \leq n_N\}$ is the set of n_N nodes with each node N_i representing either a TX crossing or a WP station, and $E = \{E_{ij};\ \forall N_i, N_j \text{ adjacent}\}$ is the set of edges $E_{ij} = \{N_i, N_j\}$ linking any two adjacent nodes, N_i and N_j. This constitutes the symbolic layer of the map model. Figure 2 portrays the graph model of the layout depicted in Figure 1.

Notice from the above description that a WP-node is always adjacent to a TX-node, which links it to the layout. It is assumed that WP-nodes have degree 1, while TX-nodes may be adjacent to, at most, one WP-node. The subsets of TX- and WP-nodes are denoted by $\mathcal{TX}$ and $\mathcal{WP}$, respectively.

Node-to-node displacements take place between specific geometric positions, termed as Control Points (CPs), identified in the vicinity of each node. The set of CPs associated to node N_i is denoted as $CP_i = \{N_{ij}, 1 \leq j \leq n_{N_i}\}$, with the number of CPs depending on the type of node, i.e. $n_{N_i} = 3$ for T-nodes, $n_{N_i} = 4$ for X-nodes, and $n_{N_i} = 1$ for WP-nodes. This constitutes the geometric layer of the map model. As an example, the CP distribution of a T-node is illustrated in Figure 3. Each CP can be occupied by a single AGV. However, their location within a node guarantee enough physical space (i) to have AGVs at different CPs of the same node, and also (ii) to allow them to evolve within the CPs of the node without colliding. Hence, nodes can also be used to solve conflicts between AGVs.

The set of AGVs is $R = \{R_i,\ i = 1 \ldots n_R\}$, with R_i denoting an individual AGV, and n_R standing for the total number of AGVs. These AGVs carry out transportation assignments that can be allocated according to different criteria.

Transportation assignments are always split as node-to-node displacements within the layout. Hence, when an AGV at node N_k has an assignment that consists of picking up a pallet at node N_l and dropping it off at node N_m, this is actually considered as two tasks, one from N_k to N_l and another one from N_l to N_m, with the AGV becoming idle after the eventual completion of the second task. Notice that this also allows to consider as tasks single node-to-node displacements such as moving to a parking/charging station. Therefore, tasks always occur between WP-nodes.

In any case, after a task is allocated, a shortest path is planned which outputs a set of nodes to be visited in a sequential manner. It is assumed that the same shortest path is always selected between any two nodes. These nodes act as symbolic locations for the motion planning function, while the exact geometric positions, or CPs, to be visited during route execution is decided in real time according to some traffic rules, except for the one corresponding to the destination, which is fixed.

This is formalized as follows:

Definition 1: A task for R_i is an ordered sequence of n_{R_i} adjacent nodes that has to be followed. Namely, $T_i = (T_{ij})_{1 \leq j \leq n_{R_i}}$, with each T_{ij} standing for a specific layout node.

Remark 1: Notice that T_{ij} is used to index the nodes of a task in a sequential way. For example, if task T_4 requires visiting nodes N_3, N_{14}, and N_8, then $T_4 = (T_{41}, T_{42}, T_{43}) = (N_3, N_{14}, N_8)$.

The symbolic allocation of an AGV at a certain time step during a trajectory is gathered in the following definitions:

Definition 2: Let R_i be at node N_j; then, its position is $P_i = N_j$. In case that R_i, when performing task T_i, is in the edge (T_{ij}, T_{ij+1}), then its position is considered to be the destination node: $P_i = T_{ij+1}$.

Definition 3: The traveling information, TI_i, that R_i following T_i communicates to the central controller consists of the current node position, $P_i = T_{ij}$, and the next node to visit according to T_i, i.e. T_{ij+1}. Namely, $TI_i = (T_{ij}, T_{ij+1})$.

The basic motion procedure considers an allocation mechanism that can be described as follows: an AGV R_i performing task T_i and currently located in a CP at node T_{ij} requests the central controller to proceed to T_{ij+1} by sending it its TI_i; when permission is granted R_i leaves the corresponding CP in T_{ij}, which becomes immediately freed, and proceeds to the nearest CP in T_{ij+1} that, at the same time, is reserved.

III. TRAFFIC CONTROL

The traffic management strategy proposed here works as a conflict detection and resolution method that aims at improving the global time of completion of the tasks. Essentially, it uses a resource allocation mechanism based on specific traffic rules where resources are TX-nodes, which may act as conflict resolution zones, and CPs. The approach works by detecting conflicts between pairs of AGVs in the system, classifying them, and allowing AGV motions according to the traffic rules, which depend on the type of conflict. Possible actions encompass either waiting at the current CP, performing a maneuver between the CPs of the current node (where the system ensures enough space for a

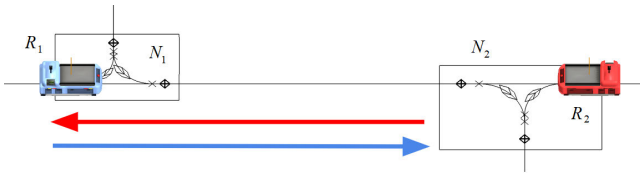

FIGURE 4. Example of a head-on conflict.

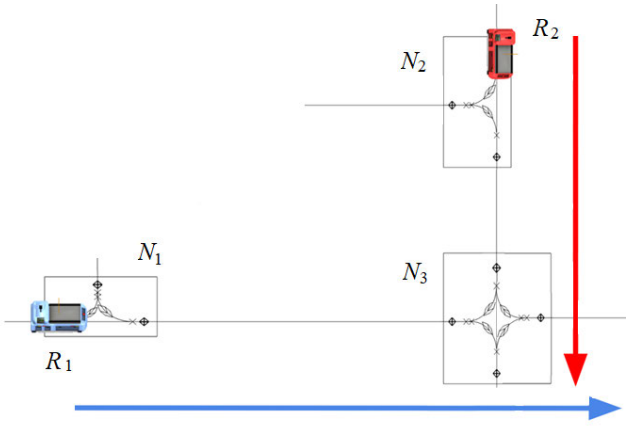

FIGURE 5. Example of an intersection conflict.

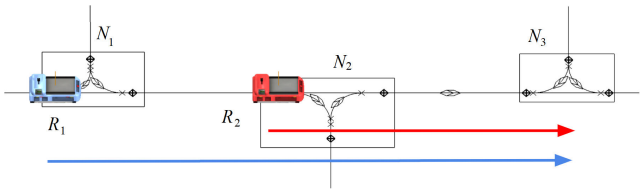

FIGURE 6. Example of a pursuit conflict.

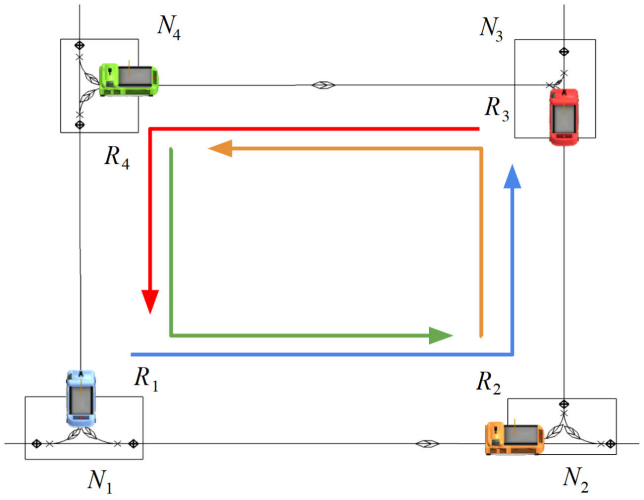

FIGURE 7. Example of a loop conflict.

collision free resolution), or moving forward to the next node of their trajectories while other AGVs wait. After resolving the conflict, the AGVs resume their planned paths.

Hence, the main stages of the MAGVS traffic control proposal are: (a) conflict detection and classification, (b) allocation of resources according to the traffic rules, and (c) conflict resolution. Each of these stages is discussed in the following subsections.

A. CONFLICT DETECTION AND CLASSIFICATION

Let us first define the concepts that will be used to detect and classify conflicts, namely residual routes and shared routes.

Definition 4: Let AGV R_i performing task T_i be at position $P_i = T_{ij}$. Its residual route, L_i, is the sequence of remaining nodes of T_i to be visited including P_i, i.e. $L_i = \{T_{ik}\}_{j \le k \le n_r}$.

Definition 5: The shared route of AGVs R_i and R_j, Σ_{ij}, is the intersection of its residual routes. Namely, $\Sigma_{ij} = L_i \cap L_j$.

Definition 6: The shared route of R_i, Σ_i, is the subset of nodes of L_i that also belong to the residual route of any other AGV. Namely,

$$\Sigma_i = \bigcup_{\substack{j=1 \\ j \neq i}}^{n_R} \Sigma_{ij}.$$

Remark 2: Notice from Definition 5 that $\Sigma_{ij} = \Sigma_{ji}$, but, in general, $\Sigma_i \neq \Sigma_j$.

Hence, conflicts may arise for R_i when $\Sigma_i \neq \emptyset$. Moreover, the analysis of the sequence of nodes in residual routes allows to classify the type of conflict according to the following definition.

Definition 7: Let the residual routes of R_i, R_j be $L_i = \{T_{ik}\}_{k_i \le k \le n_{R_i}}$, $L_j = \{T_{jk}\}_{k_j \le k \le n_{R_j}}$, respectively.

(i) *If $T_{ik_i} = T_{jk_j+1}$ and $T_{ik_i+1} = T_{jk_j}$, then R_i, R_j have a head-on conflict (see Figure 4).*

(ii) *If $T_{ik_i+1} = T_{jk_j}$ and $T_{ik_i+2} \neq T_{jk_j+1}$, then R_i, R_j have an intersection conflict at T_{ik_i+1} (see Figure 5).*

(iii) *If $T_{ik_i+1} = T_{jk_j}$ and $T_{ik_i+2} = T_{jk_j+1}$, then R_i, R_j have a pursuit conflict (see Figure 6).*

(iv) *Let $R_i, R_j, \ldots, R_k, R_l$ be such that their positions $P_i, P_j, \ldots, P_k, P_l$ are adjacent and form a cycle in the corresponding graph, G; if every pair $(R_i, R_j), \ldots, (R_k, R_l), (R_l, R_i)$ has a pursuit conflict as described in item (iii), then $R_i, R_j, \ldots, R_k, R_l$ have a loop conflict (see Figure 7).*

Remark 3: The intersection conflict presented in Definition 7.ii takes place when the residual routes of two AGVs have one single common node. This generalizes the so-called cross collision [11], as it does not demand the AGV trajectories to be orthogonal. Consequently, it allows to classify, and resolve, as an intersection conflict also the case of two AGVs travelling in the same direction that are at the final stage of a pursuit situation, i.e. when there is just one node left in their shared route. This is why, according to Definition 7.iii, for a pursuit conflict to exist the shared route is required to contain at least two adjacent nodes. In turn, this affects the loop conflict of Definition 7.iv, which is introduced as a closed chain of pursuit conflicts.

The MAGVS fleet control strategy proposed here carries out a dynamic allocation of AGVs to nodes following traffic rules that may let multiple AGVs to access shared routes, taking appropriate resolving actions should a conflict arise.

B. RESOURCE ALLOCATION: PRELIMINARIES

The traffic rules for the allocation of R_i to the requested node from the traveling information, $TI_i = (N_x, N_y)$, with $N_x = T_{ij}$, $N_y = T_{ij+1}$, communicated by AGV R_i, take into account not only shared routes, but also the states of R_i, of the CPs associated to N_y, of N_y itself, and, in case of conflict, the type of conflict.

The central controller, on receiving the allocation request from R_i, checks the type of conflict between R_i and the AGVs currently allocated in N_y, if any, and decides on the permission assigned to R_i towards its allocation to N_y. On successful allocation, R_i leaves N_x by its exit CP and arrives in N_y by its entry CP, where it first asks for a resolution within N_y should any other AGV be therein. Once this is sorted out and the exit path from N_y is cleared, R_i is ready to be allocated to the next node of its trajectory.

These concepts are defined below.

Definition 8: Assume that R_i, allocated to N_x, sends its travelling information, $TI_i = (N_x, N_y)$. The exit CP of R_i in N_x is the nearest CP of N_x to N_y. In turn, the entry CP of R_i in N_y is the nearest CP of N_y to N_x (see Figure 8).

Definition 9: The possible states, S_i, of an AGV R_i are: (a) idle: R_i is stopped at a WP-node and ready to accept a new task; (b) resuming: R_i has been allocated to the requested node in its planned path and is moving towards it; (c) waiting: R_i is requesting for allocation to the next node in its planned path and waiting for an answer; (d) resolving: R_i is undergoing a conflict resolution. These states are denoted as:

$$S_i = \begin{cases} 0 & \text{if } R_i \text{ idle} \\ 1 & \text{if } R_i \text{ resuming} \\ 2 & \text{if } R_i \text{ waiting} \\ 3 & \text{if } R_i \text{ resolving.} \end{cases}$$

Definition 10: The possible states, SN_{ij}, of a control point N_{ij} are:

$$SN_{ij} = \begin{cases} 1 & \text{if } N_{ij} \text{ is occupied/reserved by an AGV} \\ 0 & \text{if } N_{ij} \text{ is free.} \end{cases}$$

Differently from [29], a key point of the traffic control scheme is the exploitation of shared routes by allowing multiple AGVs to access them, as well as the use of nodes to solve conflicts without forcing the AGVs to divert from their originally planned paths should a situation arise. Hence, the viability of a node to act as a conflict resolution zone is given by the number of AGVs that can accommodate, and the number of AGVs that it is currently accommodating. This is based on the two definitions below.

Definition 11: The capacity, C_i, of a node N_i is the number of its associated CPs. Namely, $C_i = |CP_i|$.

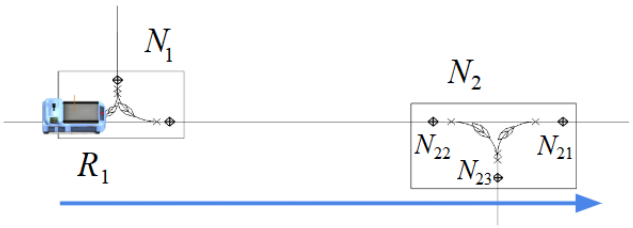

FIGURE 8. Example of AGV R_1 travelling through node N_2 where the entry CP is N_{22} and exit CP is N_{21}.

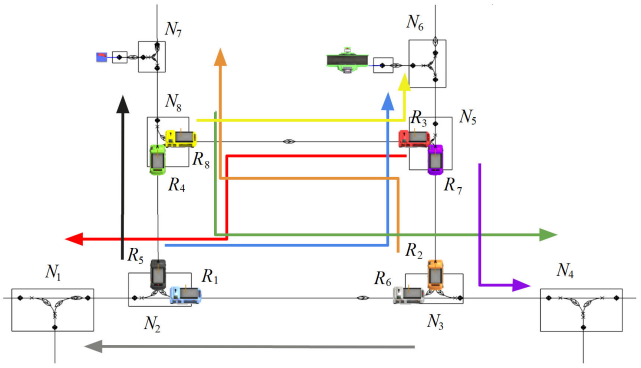

FIGURE 9. Example of a loop deadlock due to full occupancy of nodes. The trajectories of the AGVS are in the following colors: R_1, blue; R_2, orange; R_3, red; R_4, green; R_5, black; R_6, grey; R_7, purple; R_8, yellow.

Definition 12: The state, SN_i, of a node N_i is given by the number of its occupied or reserved CPs. Namely, $SN_i = \sum_{j=1}^{n_{N_i}} SN_{ij}$.

Then, nodes that can accommodate a minimum of two AGVs, i.e. such that $C(N_i) \geq 2$, and at a certain time step have at least an extra free CP to provide the degree of freedom required by the other AGVs in the node to maneuver therein, i.e. $S(N_i) \leq C(N_i) - 1$, may act as a conflict resolution zone. Therefore, only TX-nodes fulfill these requirements, as WP-nodes have a capacity of 1. Moreover, an upper bound of 2 is set for the state of TX-nodes to guarantee that both T and X crossings have room enough for conflict resolutions.

Notice that having adjacent nodes with full occupancy (i.e. with state equal to 2) may induce deadlocks different from the loop conflict of Definition 7.iii (see for instance the example in Figure 9). This situation is handled guaranteeing that adjacent TX-nodes in residual routes are never saturated.

On the other hand, loop conflicts are prevented with the below defined condition, which is checked every time a request for allocation is processed. The idea is not to allow the allocation of AGV R_k to node N_i in case this creates a loop conflict in at least one of the cycles where N_i belongs to. Hence, we first state when a cycle including N_i is prone to undergo a loop conflict, and then we extend it to all the cycles containing N_i.

Definition 13: Let $\mathcal{CY}$ denote the set of all the cycles in the layout graph G, let $CY = (N_i, N_j, \ldots, N_k, N_i) \in \mathcal{CY}$ denote one of its elements, and assume that $SN_i < 2$. The possible

pursuit states of CY with respect to N_i are: (i) troubled, if the allocation of an additional AGV in N_i results in a loop conflict in CY, and (ii) not troubled, otherwise. These states are denoted as:

$$S_P(CY) = \begin{cases} 1 & \text{if } CY \text{ is troubled} \\ 0 & \text{otherwise.} \end{cases}$$

Definition 14: Let $N_i \in N$ be a node of the layout graph G, and let $CY_{N_i} = \{CY_{ij},\ 1 \le j \le n_{CY_i}\} \subseteq \mathcal{CY}$ be the set of all the cycles in G containing N_i, with CY_{ij} denoting each of the cycles. N_i is said to verify the cycle pursuit-free condition iff none of the cycles of CY_{N_i} is troubled with respect to N_i, i.e.

$$SF_P\left(CY_{N_i}\right) = \sum_{j=1}^{n_{CY_i}} S_P\left(CY_{ij}\right) = 0.$$

Remark 4: (i) In graph cycles, only the first and last nodes are equal. Hence, just TX-nodes may belong to cycles.

(ii) The obtaining of CY_{N_i} for every node, N_i, of the graph can be performed offline, as it does not change in fixed layouts. This is a relatively easy task for moderate size associated graphs, and depth-first search techniques may be used in the general case [34].

(iii) The computation of CY_{N_i} can be skipped in specific cases. Namely, let n_S be the number of nodes of the smallest cycle that can be formed in the graph, and recall that n_R stands for the number of AGVs in the layout; then, a sufficient condition for the cycle pursuit-free condition to be verified is that $n_R < n_S$. This can be exploited in layouts with a very low number of AGVs and/or when design conditions allow to increase the number of nodes in cycles by, for example, introducing additional parking spots.

C. CONFLICT RESOLUTION FUNCTIONS

The conflict resolution strategy proposed in IDRR includes accommodating two AGVs at the same node and maneuvering therein to resolve the situation and avoid deadlocks. Two functions are defined to manage this issue: the first one, $Res_1(\cdot)$, is responsible of placing an AGV at an appropriate CP of the current node so as to free the entry CP of the AGV requesting allocation therein; the second one, $Res_2(\cdot)$, deals with two AGVs already at the same node, and places them at their respective exit CPs. Specific details for each one are given below.

- $Res_1\left(R_i, R_j\right)$: this maneuver action is triggered by an AGV, R_i, currently at node N_x, when: (a) it has asked for allocation to an adjacent node, N_y; (b) another AGV, R_j, which has a head-on or intersection conflict with R_i, is occupying or reserving the entry CP of R_i in N_y; (c) there is no other AGV than R_j in N_y.
  When $Res_1\left(R_i, R_j\right)$ is invoked the following actions are performed: 1) The states of R_i, R_j are set to resolving, i.e. $S_i = S_j = 3$; in case that R_j has reserved the CP in N_y but it is still resuming towards it, its state is set to resolving once it arrives in N_y. 2) A resolving planner is run that generates a sequence of movements of R_j within the CPs of N_y so as to finally place it at a free CP in N_y that it is neither the entry nor the exit CP of R_i in N_y, and this destination CP for R_j is reserved, this yielding $SN_y = 2$. In case that N_y is an X-node, the free CP is arbitrarily selected in an ordered way according to its indexation in CP_j. 3) The planned sequence of movements is run, so way is left for R_i to access N_y; in case that R_j is not yet in N_y, this action is kept as a request and it is conducted once R_j arrives in it. 4) The states of R_i, R_j are set to waiting, i.e. $S_i = S_j = 2$.
- $Res_2\left(R_i, R_j\right)$: this maneuver action is triggered by an AGV, R_i, when: (a) it has arrived in its entry CP to a node, N_x; (b) another AGV, R_j, which has a head-on or intersection conflict with R_i, is occupying or reserving a different CP in N_x which is not its exit CP; (c) R_j is not in resolving state, i.e. $S_j \neq 3$; (d) there are no AGVs other than R_i and R_j in N_x.
  When $Res_2\left(R_i, R_j\right)$ is invoked the following actions are performed: 1) The states of R_i, R_j are set to resolving, i.e. $S_i = S_j = 3$; in case that R_j has reserved the CP in N_x but it is still resuming towards it, its state is set to resolving once it arrives in N_x. 2) A resolving planner is run that generates a sequence of movements of R_i and R_j within the CPs of N_x so as to finally place them at their respective exit CPs in N_x. 3) The planned sequence of movements is run, so both R_i and R_j are potentially clear to leave N_x; in case that R_j is not yet in N_x, this action is kept as a request and it is conducted once R_j effectively arrives in it. 4) The states of R_i, R_j are set to waiting, i.e. $S_i = S_j = 2$.

The resolving planners in item (2) of the actions of both functions are implemented using a fast-forward (FF) planner equivalent to the one presented in [35] and [36]. Essentially, the predicates and actions that can be executed by the involved AGVs, as well as the preconditions and effects of the actions, are initially defined, and, using them, the planner calculates a sequence of synchronized and collision free AGV actions according to the initial and goal state of the conflict, i.e. the original and final CPs of the AGVs within the node.

D. THE IDRR TRAFFIC CONTROL ALGORITHM

Assume that AGV R_i is initially at a WP-node N_p in an idle state, $S_i = 0$, and receives an assignment from the central controller consisting in travelling to another WP-node, N_q, where R_i is due to perform either a pick-up, drop-off or parking action. A shortest path is then planned and a task $T_i = \left(T_{ij}\right)_{1 \le j \le n_{R_i}}$, with $T_{i1} = N_p$, $T_{in_{R_i}} = N_q$, is delivered to R_i, and the route execution begins.

Hence, let R_i arrive in its entry CP at a node of the trajectory, say $T_{ik} = N_x$, with $N_x \neq N_q$; then it gets into waiting state, i.e. $S_i = 2$, and sends both the state and the travelling information, $TI_i = \left(N_x, N_y\right)$, with $N_y = T_{ik+1}$, to the central controller. This decides on the allocation of R_i according to the set of traffic rules displayed in Table 1. The set of rules is iteratively called and applied, starting from

Rule 1, till the AGV R_i reaches the WP-node $T_{in_{R_i}}$, which completes the task T_i. After performing the corresponding action therein, if any, R_i becomes idle, i.e. $S_i = 0$, hence available to take a new assignment.

When an AGV becomes idle at a working station, and the assignment list is empty, a task is created for R_i that sends it to the nearest available parking station, where it waits in idle state for a new assignment.

Theorem 1: The IDRR traffic control algorithm is deadlock-free.

Proof: The set of traffic rules does not allow deadlocks by construction, as shown below:

- Rules 1-4 arrange the location of R_i at an appropriate CP in N_x according to an eventual pending request from another AGV that wants to access N_x and needs its entry CP to be free, and/or to the eventual presence of a second AGV, say R_j, at N_x. In the second case the goal is to locate both AGVs at their respective exit CPs in N_x. Once this is sorted out and R_i has a potentially clear exit path from N_x, the procedure continues with Rule 5 for $Ny \in \mathcal{TX}$, while it jumps to Rule 7 for $Ny \in \mathcal{WP}$. Thus, no deadlocks can be generated. Note that Rules 1-4 do not produce any action when $N_x \in \mathcal{WP}$, as WP nodes can accommodate just one AGV.
- Rules 5-6 do not allow R_i to resume to $Ny \in \mathcal{TX}$ neither when N_y already has 2 AGVs, nor when the move may result in a loop conflict. Thus, in both cases deadlocks are avoided. This is not checked when $Ny \in \mathcal{WP}$, as WP nodes can neither accommodate more than one AGV, nor belong to a cycle (recall Remark 4.i).
- Rule 7 allows R_i to resume only if N_y is empty, so a deadlock is not possible. Instead, when N_y already hosts one AGV, say R_k, if $N_y \in \mathcal{WP}$ then R_i has to wait for R_k to leave N_y, as there is no room at WP nodes for conflict resolution; however, if $N_y \in \mathcal{TX}$ further checks are done from Rule 8. Thus, again, no deadlocks are possible.
- Rule 8 classifies the conflict between R_i, located at N_x, and R_k, located at N_y. If the conflict is intersection or head-on, Rules 9 or 10 are called. In case of pursuit, R_i has to wait so as to keep a one node safety distance between; hence, no deadlock is generated.
- Rule 9 checks the path saturation condition. If the move results in two adjacent nodes becoming saturated in either of the paths of R_i and R_k, R_i has to wait. Not permitting full occupancy in two adjacent TX-nodes of residual routes ensures the existence of room for intersections and head-on conflict resolution, thus no deadlock is possible.

  Let us now clarify why this rule is not applied when some of the involved nodes belong to $\mathcal{WP}$:

  (i) When R_i is at $N_x \in \mathcal{WP}$ and has a head-on conflict with an AGV R_k allocated to N_y, it is because R_k aims at reaching N_x. As $N_x \in \mathcal{WP}$, it turns out that R_k will always be able to move to N_x once R_i leaves N_x and arrives in N_y; therefore, the state of N_y will eventually be 1 again, and path saturation, if any, will be just momentary.

TABLE 1. The IDRR traffic rules.

Rule	Action
1	Check whether there exists any pending maneuver request involving R_i to be solved by $Res_1(\cdot)$ or $Res_2(\cdot)$. If YES, turn R_i into resolving state (i.e. $S_i = 3$) and perform the corresponding actions; once back into waiting state (i.e. $S_i = 2$), apply Rule 2. Else, apply Rule 2.
2	Check the state of node N_x. If $SN_x = 2$, then apply Rule 3; else (i.e. $SN_x = 1$), apply Rule 5.
3	Check if the other AGV in N_x, say R_j, is at R_i's exit CP. If YES, then apply Rule 4; else, apply Rule 5.
4	Check the state of R_j. If $S_j = 3$, keep R_i waiting (i.e. $S_i = 2$) and EXIT; else, request a maneuver between R_i and R_j via $Res_2(\cdot)$. Once the maneuver is completed and R_i, R_j are in waiting state (i.e. $S_i = S_j = 2$) at their respective exit CPs in N_x, if $N_y \in \mathcal{TX}$, apply Rule 5; else (i.e. $N_y \in \mathcal{WP}$), apply Rule 7.
5	Check whether the state of node N_y is $SN_y = 2$. If YES, keep R_i waiting at N_x (i.e. $S_i = 2$) and EXIT; else (i.e. $SN_y \leq 1$), apply Rule 6.
6	Check the cycle pursuit-free condition for N_y. If $SF_P(CY_{N_y}) \neq 0$, keep R_i waiting at N_x (i.e. $S_i = 2$) and EXIT; else, apply Rule 7.
7	Check the state of node N_y. If $SN_y = 0$, set R_i to resume (i.e. $S_i = 1$) and EXIT; elseif ($SN_y = 1$ AND $N_y \in \mathcal{WP}$), keep R_i waiting at N_x (i.e. $S_i = 2$) and EXIT; else (i.e. $SN_y = 1$ AND $N_y \in \mathcal{TX}$), apply Rule 8.
8	Check the type of conflict between R_i and the AGV already allocated at node N_y, say R_k. If the conflict is intersection OR (head-on AND $N_x \in \mathcal{TX}$), apply Rule 9; elseif it is (head-on AND $N_x \in \mathcal{WP}$), apply Rule 10; else, R_i keeps waiting at N_x (i.e. $S_i = 2$) and EXIT.
9	Check the state of the nodes following N_y in the residual routes of R_i, R_k, say, T_{ip+2}, T_{kl+1}. If $ST_{ip+2} = 2$ OR $ST_{kl+1} = 2$, keep R_i waiting at N_x (i.e. $S_i = 2$) and EXIT; else, apply Rule 10).
10	Check the position of R_k at N_y. If R_k is not at the entry CP of R_i in N_y, set R_i to resume (i.e. $S_i = 1$) and EXIT; else, set R_i to resolving state (i.e. $S_i = 2$), and request a maneuver for R_k via $Res_1(\cdot)$. Once the maneuver is completed and R_k is in waiting state at a free CP in N_y that is neither the entry nor the exit CP of R_i in N_y, set R_i to resume (i.e. $S_i = 1$) and EXIT.

(ii) Neither $ST_{ip+2} = 2$ nor $ST_{kl+1} = 2$ is verified when either T_{ip+2} or T_{kl+1} is a WP-node, i.e. in this case it is irrelevant whether the WP-node is busy, or not. Otherwise, taking into account WP-nodes in this path saturation condition would prevent the adjacent TX-node to be used as a conflict resolution zone when the WP-node is busy and an AGV waiting to access the WP-node is allocated to this adjacent TX-node. This, in turn, could induce deadlocks.

- Rule 10 checks if R_k is blocking the entrance of R_i into N_y. If so, R_k is relocated to an appropriate CP within N_y using $Res_1(R_i, R_k)$, and once this move is completed, R_i resumes. Recall that, as $SN_y = 2$ is induced as soon as $Res_1(R_i, R_k)$ is invoked and till R_i effectively reaches N_y, the path saturation condition prevents other AGVs to be allocated neither to N_x nor to N_y during the full accommodation process. Otherwise, a loop of requests via $Res_i(\cdot)$ might take place at N_x and/or N_y, which may result in a deadlock.

□

E. IMPLEMENTATION

The AGV traffic control algorithm presented in Section III-D is currently implemented in a centralized way. Specifically, the central controller is responsible for task assignment and route planning, and also commands route execution. In turn, the AGV controllers perform actions according to the instructions of the central controller, and update it on its position by communicating the travelling information when necessary. Algorithms 1 and 2 detail the control policy implemented at the central controller. The former refers to the task assignment and route planning functions of the MAGV system, and the latter refers to the traffic control system.

Algorithm 1 Central Controller: Task and Route Planning

```
1:  while system is operative do
2:      Update assignment list A
3:      if ∃A_j ∈ A then
4:          if (∃R_i ∈ R)&(S_i = 0) then
5:              Find shortest route T_i to accomplish A_j
6:              Send task T_i to R_i
7:              Remove A_j from A
8:              Set S_i = 2
9:          end if
10:     end if
11:     if A = ∅ then
12:         if (∃R_i ∈ R)&(S_i = 0)&(P_i =W station) then
13:             Find shortest route T_i to nearest free P station
14:             Send task T_i to R_i
15:             set S_i = 2
16:         end if
17:     end if
18: end while
```

As indicated in Algorithm 1, during system operation any new assignment received by the central controller is added to a list of pending assignments, A (lines 1-2). While A in nonempty, assignments are allocated to idle AGVs by searching the list of ordered AGVs for the first idle one (line 4). Upon assigning to the first idle AGV, R_i, the first pending assignment, A_j, a shortest route is created from the AGV's initial position, P_i, to the final WP destination node, and the corresponding task, T_i, is created and sent to R_i; then, A_j is removed from the pending assignments A, and R_i is set into waiting state, i.e. $S_i = 2$, (lines 3-9). When there are no pending assignments, the idle AGVs are sent to a parking station (P node), where it remains in idle state until receiving a new assignment (lines 11-17).

Whenever R_i has been assigned a task, T_i, and is in waiting state, the AGV requests the traffic control system for allocation to the next node by sending the travelling information, $TI_i = (N_x, N_y)$. Algorithm 2 shows the procedure followed by the traffic controller while there are AGVs active in the system. First, it updates the residual route L_i, the shared routes between R_i and all other AGVs, Σ_{ij}, and uses this information to also update the shared route of R_i, Σ_i (lines 2-8). As long as the AGV is not at its WP destination node, the traffic rules described in Section III-D are applied (lines 9-10). Otherwise, when the AGV arrives in this final node, it is set to idle state (lines 11-12).

Algorithm 2 Central Controller: Traffic Control

```
1:  while system is operative do
2:      if (∃R_i ∈ R)&(S_i = 2) then
3:          Read the travelling information TI_i = (N_x, N_y)
4:          Update residual route L_i from TI_i = (N_x, N_y)
5:          for all R_j in R, j ≠ i do
6:              Create shared route Σ_ij
7:          end for
8:          Update whole shared route Σ_i
9:          if N_y ≠ ∅ then
10:             Execute Traffic Rules
11:         else (i.e. N_y = ∅)
12:             Set S_i = 0
13:         end if
14:     end if
15: end while
```

Recall that when N_x is already a WP-node the AGV performs the assigned pick-up, drop-off, or charging task therein before becoming idle. However, for the sake of simplicity, in the current approach we assume that this action is instantaneously performed, so the AGV becomes idle immediately after completing a task.

IV. EXAMPLES OF CONFLICT RESOLUTION

The operation of the traffic control algorithm and, in particular, of the conflict resolution functions $Res_1(\cdot)$ and

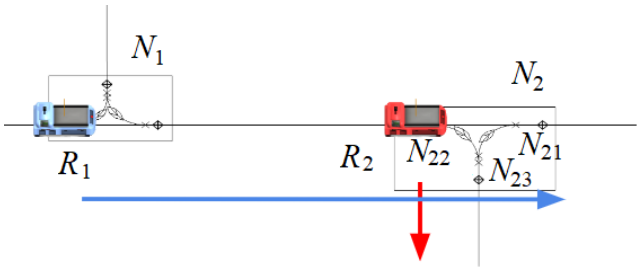

FIGURE 10. Example of resolution of an intersection conflict using $Res_1(\cdot)$.

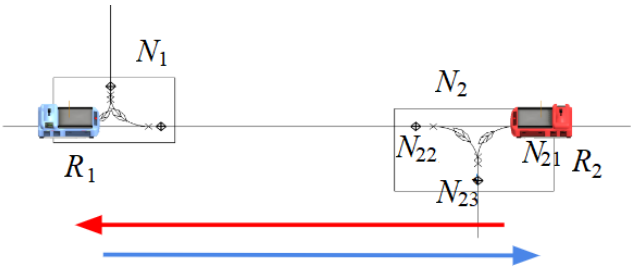

FIGURE 11. Example of resolution of a head-on conflict using $Res_2(\cdot)$.

$Res_2(\cdot)$ introduced in Subsection III-C, are exemplified in this section.

A. EXAMPLE: USE OF $RES_1(\cdot)$

Assume a layout with only two AGVs, R_1 and R_2, that at a certain time step are allocated as in Figure 10, namely, $P_1 = N_1$, $P_2 = N_2$, with their residual routes being $L_1 = \{N_1, N_2, \ldots\}$, $L_2 = \{N_2, N_3, \ldots\}$, where N_3 is assumed adjacent to N_2 and in the travel direction of R_2. Assume also that R_1 communicates first its travelling information, $TI_1 = (N_1, N_2)$, to the central controller, which is subsequently processed.

Since $SN_1 = SN_2 = 1$, the algorithm jumps from Rule 1 to Rule 5, and then to Rule 6. Besides, as there are no other AGVs in the layout, the cycle pursuit-free condition at N_2 is verified, i.e. $SF_P\left(CY_{N_2}\right) = 0$, so it advances to Rule 7, and then, recalling that $SN_2 = 1$, to Rule 8. According to Figure 10, the type of conflict between R_1 and R_2 is intersection (see Definition 7 in Section III-A), hence Rule 9 is applied. Again, as there are no more AGVs in the layout, the state of the nodes following N_2 in the residual routes of R_1, R_2 is 0, and the algorithm arrives in Rule 10. Then, as R_2 is at N_{22}, which is the entry CP of R_1 in N_2, $Res_1(R_1, R_2)$ is triggered. This sets R_1, R_2 in resolving state, i.e. $S_1 = S_2 = 3$, and the resolving planner is called, generating the following plan. The initial location of R_2 is at N_{22}, which has a busy state, i.e. $SN_{22} = 1$, and the final location is R_2 at a free CP in N_2 which is neither the entry nor the exit CP of R_1 in N_2. As N_2 is a T crossing, the goal location has to be N_{23}, this being the exit CP of R_2 in N_2. Once at this point, the state of N_{23} is set to reserved, i.e. $SN_{23} = 1$, and R_2 moves from N_{22} to N_{23}. When R_2 reaches N_{23}, R_1 is allocated to N_{22}, the state of R_2 is set to waiting, i.e. $S_2 = 2$, and that of R_2 is set to resuming, i.e. $S_1 = 1$, thus allowing it permission to proceed.

B. EXAMPLE: USE OF $RES_2(\cdot)$

Assume again a layout with only two AGVs, R_1 and R_2, allocated as in Figure 11, namely, $P_1 = N_1$, $P_2 = N_2$, with their residual routes being $L_1 = \{N_1, N_2, \ldots\}$, $L_2 = \{N_2, N_1, \ldots\}$. Let R_1 communicate its travelling information, $TI_1 = (N_1, N_2)$, to the central controller before R_2.

Considering that $SN_1 = SN_2 = 1$, $SF_P\left(CY_{N_2}\right) = 0$, that the conflict between R_1 and R_2 is head-on (see Definition 7 in Section III-A), and that there are no more AGVs in the layout, the algorithm advances up to Rule 10 as it did in the previous example, but following a different sequence of actions. In this case R_2 is not blocking the entry CP of R_1 in N_2, so R_1 is granted permission to resume, its CP in N_1 is deallocated, and it is allocated at N_{22}, i.e. $SN_{22} = 1$. Therefore, the head-on conflict is to be solved at node N_2, which has now state $SN_2 = 2$.

Assume that, while R_1 is on its way to N_{22}, R_2 requests permission to proceed to N_1 by sending its travelling information, $TI_2 = (N_2, N_1)$. As $SN_2 = 2$ in Rule 2, Rule 3 is triggered, and it turns out that R_2 has its exit CP, N_{22}, reserved by R_1. Hence, $Res_2(R_2, R_1)$ is called in Rule 4. The states of R_1 and R_2 changes to resolving, i.e. $S_1 = S_2 = 3$ (waiting for R_1 to reach N_{22}, if necessary), and the resolving planner is called, generating the following plan. The initial location is R_1 at N_{22} and R_2 at N_{21}, and the goal location is the opposite, i.e. R_1 at N_{21} and R_2 at N_{22}. The maneuver action synchronizes the motion, first sending R_2 from N_{21} to N_{23}, which is reserved, i.e. $SN_{23} = 1$. Then, when R_2 arrives in N_{23}, R_1 is allocated to N_{21} and moves towards it. Finally, when R_1 arrives in N_{21}, R_2 is allocated to N_{22} and proceeds to it. Upon arrival, N_{23} is freed, i.e. $SN_{23} = 0$. Finally, both AGVs are set to waiting state, i.e. $S_1 = S_2 = 2$.

Now both R_1 and R_2 have their exit paths from N_2 potentially cleared, and Rule 7 is triggered to proceed.

C. EXAMPLE: MULTIPLE CONFLICT RESOLUTION

Assume a layout with nine AGVs initially located as depicted in Figure 12. The non TX-nodes are all working stations, in particular, N_1, N_2, N_8 and N_9 are pick-up stations, while N_5, N_{12}, N_{15} and N_{16} are drop-off stations; the parking stations are elsewhere in the layout, accessible by the dashed edges stemming from N_3, N_4, N_{13} and N_{14}. The travel informations, TI_i, are assumed to be processed instantly and in a sequential order, from TI_1 to TI_9, at every time step. Besides, the AGVs take one time step to travel between adjacent nodes, and also to complete $Res_i(\cdot)$ movements within the CPs of a node when necessary.

Within this framework, let the residual routes at $t = 0$ be:

$$L_1 = \{N_1, N_3, N_6, N_{10}, N_{13}, N_{15}\}$$
$$L_2 = \{N_5, N_6, N_7, N_8\}$$

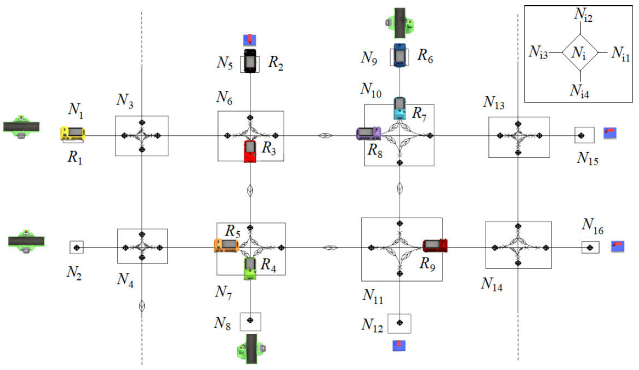

FIGURE 12. Example of a multiple conflict resolution: positions at $t = 0$.

$$L_3 = \{N_6, N_5\}$$
$$L_4 = \{N_7, N_6, N_5\}$$
$$L_5 = \{N_7, N_{11}, N_{12}\}$$
$$L_6 = \{N_9, N_{10}, N_{11}, N_{12}\}$$
$$L_7 = \{N_{10}, N_9\}$$
$$L_8 = \{N_{10}, N_6, N_3, N_1\}$$
$$L_9 = \{N_{11}, N_7, N_8\}.$$

The CP location of the AGVs at every time step is gathered in Table 2. The main motion details are as follows:

- $t = 0$: R_1, R_2 and R_5 are given permission to resume; R_3 triggers $Res_2(R_3, R_2)$ to resolve a head-on conflict at N_6, and R_9 triggers $Res_2(R_9, R_5)$ to resolve an intersection conflict at N_{11};
- $t = 1$: upon arrival of R_2 in N_6, and of R_5 in N_{11}, $Res_2(R_3, R_2)$ and $Res_2(R_9, R_5)$ are initiated;
- $t = 2$: upon completion of $Res_2(R_3, R_2)$ and $Res_2(R_9, R_5)$, R_2, R_3, and R_5 are allowed to resume; R_4 triggers $Res_2(R_4, R_2)$ to resolve a head-on conflict at N_7; R_8 is also allowed to resume;
- $t = 3$: R_1 is given permission to resume, as R_8 is not blocking its entrance to N_6, and R_8 triggers $Res_2(R_8, R_1)$ to resolve a head-on conflict therein; upon arrival of R_2 in N_7, $Res_2(R_4, R_2)$ is initiated; R_3 and R_5 arrive in N_5 and N_{12}, respectively, which completes their tasks;
- $t = 4$: upon arrival of R_1 in N_6, $Res_2(R_8, R_1)$ is initiated; upon completion of $Res_2(R_4, R_2)$, both AGVs are allowed to resume; as R_8 has left N_{10}, R_6 triggers $Res_1(R_6, R_7)$ as it has the entrance to N_{10} blocked by R_7, and the maneuver is initiated; R_9 is allowed to resume to N_7;
- $t = 5$: R_2 arrives in N_8, which completes its task; R_4 and R_9 arrive in N_6 and N_7, respectively, and they have to remain therein till R_3 and R_2 complete their assignments at N_5 and N_8 and de-allocate these nodes, which are the targets of R_4 and R_9, respectively; upon completion of $Res_1(R_6, R_7)$, R_6 is allowed to resume to N_{10}, and R_7 triggers $Res_2(R_7, R_6)$ to resolve the corresponding intersection conflict at N_{10}; upon completion of $Res_2(R_8, R_1)$, R_8 is allowed to resume;

TABLE 2. Example of a multiple conflict resolution: CP location of the nine AGVs R_i at every time step, from $t = 0$ to $t = 10$.

	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9
0	N_{11}	N_{51}	N_{64}	N_{74}	N_{73}	N_{91}	N_{102}	N_{103}	N_{111}
1	N_{33}	N_{62}	N_{64}	N_{74}	N_{113}	N_{91}	N_{102}	N_{103}	N_{111}
2	N_{33}	N_{64}	N_{62}	N_{74}	N_{114}	N_{91}	N_{102}	N_{103}	N_{113}
3	N_{33}	N_{72}	N_{51}	N_{74}	N_{121}	N_{91}	N_{102}	N_{61}	N_{113}
4	N_{63}	N_{74}	N_{51}	N_{72}	N_{121}	N_{91}	N_{102}	N_{61}	N_{113}
5	N_{61}	N_{81}	N_{51}	N_{64}	N_{121}	N_{91}	N_{101}	N_{63}	N_{71}
6	N_{61}	N_{81}	N_{51}	N_{64}	N_{121}	N_{102}	N_{101}	N_{31}	N_{71}
7	N_{61}	N_{81}	N_{51}	N_{64}	N_{121}	N_{104}	N_{102}	N_{11}	N_{71}
8	N_{103}	N_{81}	N_{51}	N_{64}	N_{121}	N_{112}	N_{91}	N_{11}	N_{71}
9	N_{133}	N_{81}	N_{51}	N_{64}	N_{121}	N_{112}	N_{91}	N_{11}	N_{71}
10	N_{151}	N_{81}	N_{51}	N_{64}	N_{121}	N_{112}	N_{91}	N_{11}	N_{71}

- $t = 6$: upon arrival of R_6 in N_{10}, $Res_2(R_7, R_6)$ is initiated; R_8 is allowed to resume;
- $t = 7$: R_1 is allowed to resume; upon completion of $Res_2(R_7, R_6)$, R_6 and R_7 are allowed to resume; R_8 arrives in N_1, which completes its task;
- $t = 8$: R_1 is allowed to resume; R_6 arrives in N_{11} and has to remain therein till R_5 completes its assignment at N_{12} and de-allocates this node, which is the target of R_6; R_7 arrives in N_9, which completes its task;
- $t = 9$: R_1 is allowed to resume;
- $t = 10$: R_1 arrives in N_{15}, which completes its task.

V. NUMERICAL RESULTS

The proposed IDRR traffic control approach has been validated using the simulation software Flexsim, which allows modelling the layout, adding AGVs, pick-up, drop-off and parking stations, and tracking of the key system parameters, including battery usage.

Two different layouts with distinctive task rationales have been selected to conduct two sets of experiments, labelled as Test 1 and Test 2; details are provided in the next subsections.

A. TEST 1: REAL LAYOUT

In Test 1, in-silico experiments were run to simulate operations in a real FMS of dimensions 154 m × 141 m with the layout shown in Figure 1. Transportation assignments were randomly generated in real time at pick-up stations with time intervals according to a uniform distribution, which were allocated to idle AGVs by searching within the list of AGVs for the first idle one, as mentioned in Subsection III-E. The simulation time was set to nine hours, corresponding to an operational shift within the FMS.

All the AGVs in the system operated at a speed of 1 m/s, and with the same acceleration and deceleration. For the sake of simplicity, as indicated in Subsection III-E and in order to avoid jeopardizing the time-based quantitative performance analysis of the IDRR algorithm, loading, unloading, and charging actions were assumed to last 0 s.

The quantitative metrics considered are:

1) Productivity per hour: number of transportation assignments completed per hour, which includes picking the

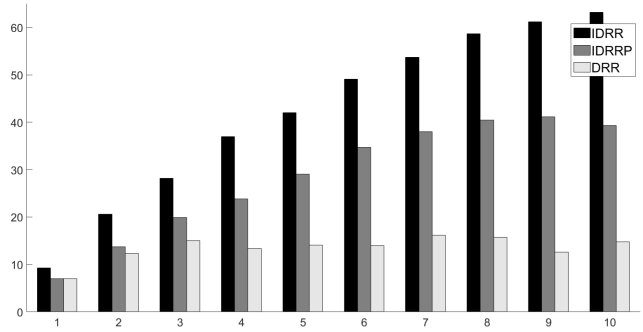

FIGURE 13. Test 1.1: productivity per hour for 1 to 10 AGVs. IDRR: black; IDRRP: dark grey; DRR: light grey.

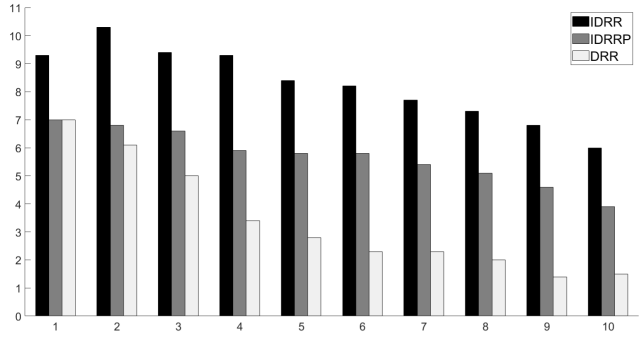

FIGURE 14. Test 1.1: average productivity per AGV-hour for 1 to 10 AGVs. IDRR: black; IDRRP: dark grey; DRR: light grey.

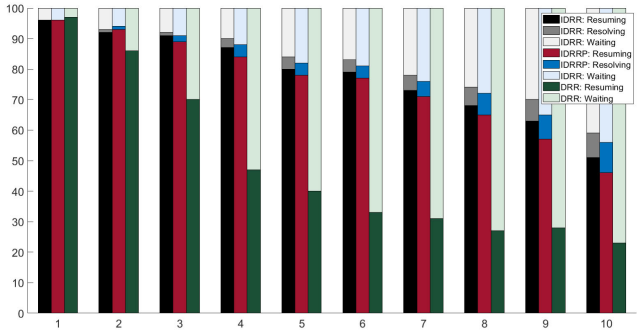

FIGURE 15. Test 1.1: state chart for 1 to 10 AGVs. IDRR resuming: black; IDRR resolving: dark grey; IDRR waiting: light grey; IDRRP resuming: red; IDRRP resolving: dark blue; IDRR waiting: light blue; DRR resuming: green; DRR waiting: light green.

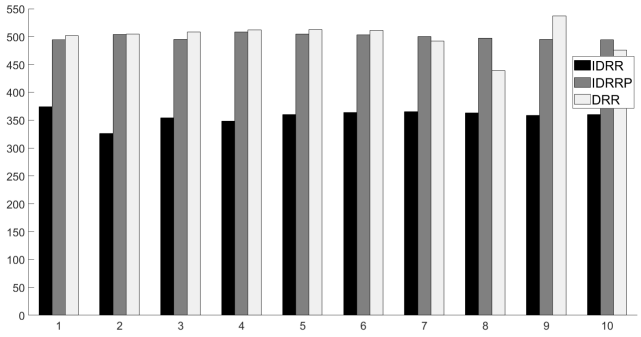

FIGURE 16. Test 1.1: average travel distance per assignment for 1 to 10 AGVs. IDRR: black; IDRRP: dark grey; DRR: light grey.

pallet at a pick-up station and delivering it at a drop-off station.

2) State times: time spent in idle, resuming, waiting, and resolving states during the operation.
3) Average travel distance per assignment: total travelled distance when completing transportation assignments, starting from the position where the AGV receives the assignment till the position where it becomes idle and ready to be allocated a new one, divided by the number of assignments.

According to the specific analyzed aspects, these parameters may be given in global numbers, i.e. for all the AGVs in the system, in average, i.e. divided by the number of AGV in the system, or in individual values, i.e. for each AGV in the system.

Two sets of numerical experiments were conducted in this layout. In Test 1.1, simulations with the number of AGVs scaling from $n_R = 1$ to $n_R = 10$ were run within a continuous flow of tasks that kept the AGVs fully busy, i.e. idle states were not allowed to settle; in this case the analysis is based on global performance values, and IDRR is compared with DRR [29] and IDRRP, a version of IDRR that forces AGVs to head to a parking station after completing a task and before undertaking a new one, as required by DRR. Then, in order to gain insight on individual AGV dynamics under IDRR, results were obtained in Test 1.2 for a system with $n_R = 5$ where the tasks flow was slowed down to allow intervals of idle states in AGVs, which also allowed to test the IDRR idle vehicle management policy.

As of Test 1.1, Figure 13 shows the IDRR productivity per hour for 1 to 10 AGVs in the layout. Notice that for 1 to 4 AGVS it scales almost proportionally; however, from 5 AGV onwards the effect of traffic lowers the increase rate, and the productivity tends to stabilize. In fact, an additional AGV in a system with 9 of them just results in 2 more assignments completed per hour. This is consistent with the average productivity per AGV-hour plotted in Figure 14: it stays constant around 9 for 1 to 4 AGVs, and then it decreases steadily. Further confirmation stems from the state chart of Figure 15: the time fraction of the waiting state increases slowly for low traffic, i.e. $n_R \leq 4$, and then it raises continuously up to 40% for $n_R = 10$.

However, it is most remarkable from Figures 13-15 that IDRR clearly outperforms DRR for all $n_R \in \{1, \ldots, 10\}$, with improvements up to a 386% for 9 AGVs, when traffic management is more relevant. As mentioned in Section I, DRR requires the AGVs to return to a parking station after completing an assignment in order to be allocated a new one; instead, in IDRR, assignments are undertaken from the drop-off station if there is a queue of them. Hence, even for $n_R = 1$, the productivity of IDRR is higher than that of DRR, and as in this particular case there are no traffic-induced delays, the effect of this assignment criterion is neat.

TABLE 3. Test 1.1: metrics for 1 to 10 AGVs. Productivity: productivity per hour; AProductivity: average productivity per AGV-hour; Resuming, Resolving, and Waiting times are in % of the total simulation time; ATravel: average travel distance per assignment, in meters.

AGVs	1	2	3	4	5	6	7	8	9	10
Productivity										
IDRR	9.3	20.6	28.2	37.0	42.0	49.1	53.7	58.7	61.2	63.2
IDRRP	7	13.7	19.9	23.8	29.1	34.7	38.0	40.5	41.2	39.3
DRR	7	12.3	15.0	13.4	14.1	14.0	16.2	15.7	12.6	14.8
AProductivity										
IDRR	9.3	10.3	9.4	9.3	8.4	8.2	7.7	7.3	6.8	6.0
IDRRP	7	6.8	6.6	5.9	5.8	5.8	5.4	5.1	4.6	3.9
DRR	7	6.1	5.0	3.4	2.8	2.3	2.3	2.0	1.4	1.5
Resuming										
IDRR	96%	92%	91%	87%	80%	79%	73%	68%	63%	51%
IDRRP	96%	93%	89%	84%	78%	77%	71%	65%	57%	46%
DRR	97%	86%	70%	47%	40%	33%	31%	27%	28%	23%
Resolving										
IDRR	0%	1%	1%	3%	4%	4%	5%	6%	7%	8%
IDRRP	0%	1%	2%	4%	4%	4%	5%	7%	8%	10%
DRR	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Waiting										
IDRR	4%	7%	8%	10%	16%	17%	22%	26%	30%	41%
IDRRP	4%	6%	9%	12%	18%	19%	24%	28%	35%	44%
DRR	3%	14%	30%	53%	60%	67%	69%	73%	72%	77%
ATravel										
IDRR	374	326	354	348	360	364	365	363	359	360
IDRRP	494	504	495	508	505	503	500	497	495	494
DRR	502	505	508	512	513	511	492	439	537	476

This is also seen in Figure 16, which portrays the average distance travelled by an AGV per completed assignment: while in IDRR it stays around 360 m, in DRR it scales up to 510 m.

Another source of differences between IDRR and DRR lays on the fact that the former allows traffic in shared routes: when the number of working stations entails short trips and/or infrequent route intersections this might not be that important; instead, in a layout like that in Figure 1, with few working stations and long distances to run, shared routes are ubiquitous, which impairs dramatically the performance of DRR. The quantitative benefits of IDRR's traffic management strategy with respect to DRR can be isolated from the parking policy by looking at the IDRRP results. Firstly, notice from Figures 13 and 14 that both IDRRP and DRR have the same productivity for one AGV; besides, the average travelled distance in both cases is analogous for 1 to 10 AGVs (not exactly equal, as tasks are not the same in each simulation but just randomly generated using the same uniform distribution function). This indicates that AGVs are effectively parking between tasks under IDRRP. Once at this point, it is straightforward that the traffic management of IDDR, independently of the parking effect, is significantly better than that DRR, raising to a 230% for 9 AGVs.

The numerical data corresponding to Test 1.1 are gathered in Table 3.

Test 1.2 was run for 5 AGVs and with the assignment flow slowed down so as to allow idle states to settle, which required the idle management policy to be applied: AGVs that become idle at a working station and do not have a task assignment are set to the nearest available parking station. Figure 17 plots the productivity per hour for each AGV. Notice that the average completed assignments, around 6, is lower that than in Test 1.1 for 5 AGVs, which amounted above 8.

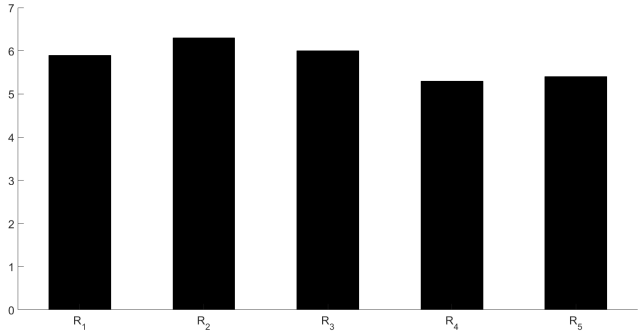

FIGURE 17. Test 1.2: productivity per hour for 5 AGVs under IDRR.

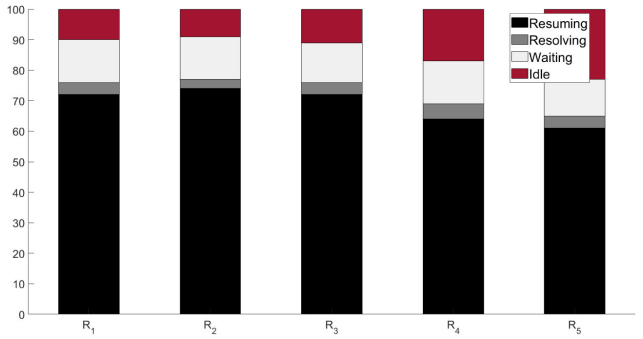

FIGURE 18. Test 1.2: state chart for 5 AGVs under IDRR. Resuming: black; resolving: dark grey; waiting: light grey; idle: red.

TABLE 4. Test 1.2: metrics for 5 AGVs under IDRR. Productivity: productivity per hour; Resuming, Resolving, Waiting, and Idle times are in % of the total simulation time.

	R_1	R_2	R_3	R_4	R_5
Productivity	5.9	6.3	6.0	5.3	5.4
Resuming	72%	74%	72%	64%	61%
Resolving	4%	3%	4%	5%	4%
Waiting	14%	14%	13%	14%	12%
Idle	10%	9%	11%	17%	23%

This is consistent with the current existence of idle states. In any case, the productivity is very similar among the AGVs, indicating that the proposed algorithm is balanced with respect to task distribution. The corresponding state chart is portrayed in Figure 18: as expected, now idle states appear in a non negligible way, and AGVs are less time in resuming state than in Test 1.1. Note also that the idle state increases with the order of the AGVs in the list, this is due to the procedure used for the assignment of the AGVs to new tasks (presented in Subsection III-E); for instance, the idle time of R_5 is larger than the idle time of R_1 because, when both are in idle state, R_1 will always receive a new assignment before R_5. The metrics for Test 1.2 are displayed in Table 4.

TABLE 5. Test 2: metrics for 1 to 10 AGVs. End time in [s].

Number of AGVs	DRR	IDRRP	IDRR
1	30020	30019	20139
2	15863	15697	10853
3	12892	10856	8108
4	13027	8447	6198
5	14952	6989	5249
6	15118	6025	4732
7	14224	5403	4404
8	14990	4985	4068
9	16080	4759	3979
10	15306	4677	3957

B. TEST 2: A BENCHMARK LAYOUT

A benchmark layout borrowed from [31], see Figure 19, has been used in this test to further validate the IDRR traffic management policy. It consists of a 50 m × 40 m square topology where we distinguish: (i) three sources of work pieces, I1-I3; (ii) three sinks for work pieces, O1-O3; (iii) six machines, M1-M6; (iv) eighteen working (W) stations, S1-S18, one half of pick-up kind, and the other half of drop-off kind, and (v) ten parking (P) stations, C1-C10.

At the beginning of the simulation, each source point, Ii, has twenty working pieces, labelled as Part-i type. Pieces have to be picked up one at a time from the corresponding W station next to Ii, taken to two machines, one after the other one, where each piece is dropped, processed for 20 s, then picked, and finally dropped at the W station of the corresponding sink point, Oj. The route to be followed by each type of piece is detailed below, with parentheses indicating sources, sinks, and machines:

- Part 1: (I1)-S7-S14-(M4)-S13-S5-(M3)-S6-S12-(O3);
- Part 2: (I2)-S10-S3-(M2)-S4-S16-(M5)-S15-S9-(O2);
- Part 3: (I3)-S11-S1-(M1)-S2-S18-(M6)-S17-S8-(O1).

Hence, three independent transportation assignments are associated to each route: from the source W station to the W drop-of station of the first machine, from the W pick-up station of the first machine to the W drop-of station of the second machine, and from the W pick-up station of the second machine to the W drop-of station of the sink. This represents 180 tasks to be assigned to the AGVs available at the layout (3 types of pieces, 20 pieces of each type, and 3 moves per piece). The AGVs are initially parked at the P stations C1-C10, travel at a speed of 1 m/s, and the simulation finishes when the last piece has been dropped and the transporting AGV reaches its parking position.

Ten experiments were run, with an increasing number of AGVs, using IDRR, IDRRP, and DRR. With IDRR, when an AGV drops a piece at a W station, it is immediately assigned a new task as long as there are pieces left to transport. Instead, with IDRRP and DRR, a new assignment takes place only after the AGV has reached its parking position. The total times taken by all the simulations are collected in Table 5.

Notice that IDRR is outperforming by far DRR also in this Test, reaching a maximum improvement of 304% again

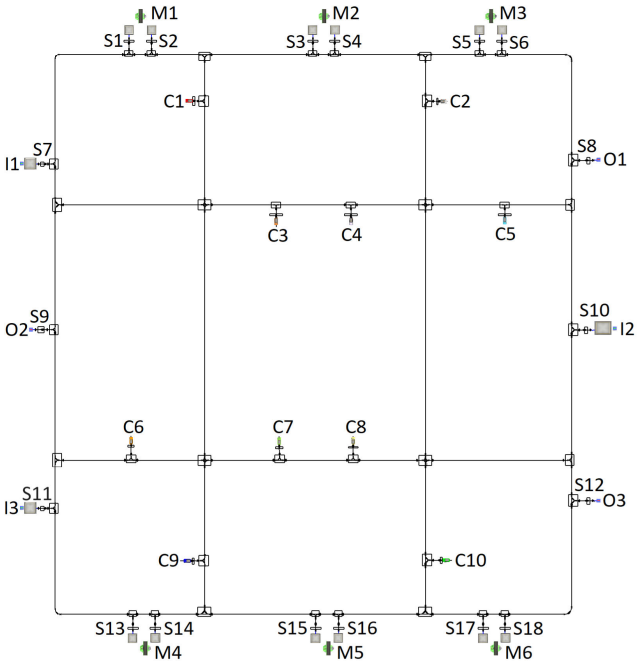

FIGURE 19. A benchmark layout. C1-C10: parking stations; I3-I3: sources of work pieces; O1-O3: sinks to work pieces; S1, S3, S5, S8, S9, S12, S14, S16, S18: drop-off stations; S2, S4, S6, S7, S10, S11, S13, S15, S17: pick-up points; M1-M6: machines.

for 9 AGVS. Interestingly, in this case the specific task rationale hides the effect of the parking policy followed by DRR for a low number of AGVs, the improvement falling to 60% for 3 AGVs. From that point onwards an increasing traffic density escalates the number of conflicts, which plays against DRR efficiency with respect to IDRR. This masking of the parking effect is further confirmed by the IDRRP numbers, although in this case the traffic management strategy, which is that of IDRR, makes its results converge to those of IDRR for a growing number of AGVs: in fact, for 10 AGVs the improvement of IDRR with respect to IDRRP is just of 18%. On the other hand, it is again confirmed from DRR and IDRRP performances that not only the parking policy but also the traffic management by itself makes an important difference between IDRR and DRR.

Finally, it is worth mentioning that: (i) the 1 s mismatch between DRR and IDRRP for 1 AGV represents a $3 \cdot 10^{-5}$ relative error difference, and it is due to numerical rounding; (ii) the fact that DRR results oscillate from a certain number of AGVs in the layout is also observed in [31], and it is a consequence of the highly nonlinear effect of traffic conflicts on the system throughput.

VI. CONCLUSION

A traffic management strategy for MAGVS that guarantees collision and deadlock avoidance was presented in this paper. The policy, termed as IDRR, falls within the static routing category, uses a zone control approach and can be

implemented in a centralized or semi-decentralized way. Its key element is that shared resource points allow multiple reservations/occupancy, and can be used as conflict resolution areas between AGVs. Extensive numerical simulations illustrated the effectiveness of the proposal and showed, using standard metrics, that it clearly outperforms the so-called DRR controllers, thus enabling an FMS to transport material faster and hence enhancing the productivity of the whole logistic operation.

Currently ongoing work is dealing with the implementation of the IDRR architecture in a real industrial MAGVS using a semi-centralized approach. In turn, further research will be oriented towards; (i) the obtaining of fully decentralized algorithms and implementations; (ii) considering some optimization criteria in the assignments of idle AGVs to new required tasks; (iii) assessing the performance of IDRR in a specific layout against changes in the number and location of parking and working stations; (iv) assessing the cost of communications and of that of energy consumption, and designing a recharging policy for AGVs.

ACKNOWLEDGMENT

The authors wish to thank Tavil Ind. S.A.U., for committing facilities and resources to the project and in particular to its Research and Development Director, Pol Toldrà, for his guidance and support at the company.

REFERENCES

[1] H. Fazlollahtabar and M. Saidi-Mehrabad, "Methodologies to optimize automated guided vehicle scheduling and routing problems: A review study," *J. Intell. Robotic Syst.*, vol. 77, nos. 3–4, pp. 525–545, Mar. 2015.

[2] M. De Ryck, M. Versteyhe, and F. Debrouwere, "Automated guided vehicle systems, state-of-the-art control algorithms and techniques," *J. Manuf. Syst.*, vol. 54, pp. 152–173, Jan. 2020.

[3] E. Roszkowska and S. A. Reveliotis, "On the liveness of guidepath-based, zone-controlled dynamically routed, closed traffic systems," *IEEE Trans. Autom. Control*, vol. 53, no. 7, pp. 1689–1695, Aug. 2008.

[4] S. Reveliotis and T. Masopust, "Efficient liveness assessment for traffic states in open, irreversible, dynamically routed, zone-controlled guidepath-based transport systems," *IEEE Trans. Autom. Control*, vol. 65, no. 7, pp. 2883–2898, Jul. 2020.

[5] E. G. Coffman, M. J. Elphick, and A. Shoshani, "System deadlocks," *ACM Comput. Surv.*, vol. 3, no. 2, pp. 67–78, Jun. 1971.

[6] A. A. Tubis and H. Poturaj, "Risk related to AGV systems—Open-access literature review," *Energies*, vol. 15, no. 23, p. 8910, Nov. 2022.

[7] X. Chen, W. Wu, and R. Hu, "A novel multi-AGV coordination strategy based on the combination of nodes and grids," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 6218–6225, Jul. 2022.

[8] J. Yang, F. Yu, E. Wu, and Y. Yang, "Energy saving of automated terminals considering AGV speed variation," in *Proc. Int. Symp. Sens. Instrum. 5G IoT era (ISSI)*, Nov. 2022, pp. 68–73.

[9] C. Liang, Y. Zhang, and L. Dong, "A three stage optimal scheduling algorithm for AGV route planning considering collision avoidance under speed control strategy," *Mathematics*, vol. 11, no. 1, p. 138, Dec. 2022.

[10] M. Müller, J. H. Ulrich, T. Reggelin, S. Lang, and L. S. Reyes-Rubiano, "Comparison of deadlock handling strategies for different warehouse layouts with an AGVS," in *Proc. Winter Simul. Conf. (WSC)*, K. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, Eds., Dec. 2020, pp. 1300–1311.

[11] Y. Zhao, X. Liu, S. Wu, and G. Wang, "Spare zone based hierarchical motion coordination for multi-AGV systems," *Simul. Model. Pract. Theory*, vol. 109, May 2021, Art. no. 102294.

[12] M. P. Fanti, A. M. Mangini, G. Pedroncelli, and W. Ukovich, "Decentralized deadlock-free control for AGV systems," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2015, pp. 2414–2419.

[13] M. P. Fanti, A. M. Mangini, G. Pedroncelli, and W. Ukovich, "A decentralized control strategy for the coordination of AGV systems," *Control Eng. Pract.*, vol. 70, pp. 86–97, Jan. 2018.

[14] T. Schmidt, K. Reith, N. Klein, and M. Däumler, "Research on decentralized control strategies for automated vehicle-based in-house transport systems—A survey," *Logistics Res.*, vol. 13, no. 1, Nov. 2020, Art. no. 10.

[15] J. Chen, X. Zhang, X. Peng, D. Xu, and J. Peng, "Efficient routing for multi-AGV based on optimized ant-agent," *Comput. Ind. Eng.*, vol. 167, May 2022, Art. no. 108042.

[16] S. Maza, "Hybrid supervisory-based architecture for robust control of bi-directional AGVs," *Comput. Ind.*, vol. 144, Jan. 2023, Art. no. 103797.

[17] L. Feng, X. Chen, T. Zhang, and W. Wu, "A conflict-reducing path planning algorithm in automated warehouses," in *Proc. IEEE Int. Conf. Netw., Sens. Control (ICNSC)*, vol. 1, Dec. 2021, pp. 1–5.

[18] X. Chen, T. Zhang, W. Wu, and R. Hu, "A novel searching method of fringe blocks for AGV deadlock avoidance," in *Proc. IEEE Int. Conf. Netw., Sens. Control (ICNSC)*, vol. 1, Dec. 2021, pp. 1–6.

[19] H. Xiao, X. Wu, D. Qin, and J. Zhai, "A collision and deadlock prevention method with traffic sequence optimization strategy for UGN-based AGVS," *IEEE Access*, vol. 8, pp. 209452–209470, 2020.

[20] K. M. Kim, C. H. Chung, and Y. J. Jang, "Deadlock avoidance dynamic routing algorithm for a massive bidirectional automated guided vehicle system," in *Proc. Winter Simul. Conf. (WSC)*, Dec. 2022, pp. 1–12.

[21] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 219, pp. 40–66, Feb. 2015.

[22] M. Cap, P. Novak, A. Kleiner, and M. Selecky, "Prioritized planning algorithms for trajectory coordination of multiple mobile robots," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 3, pp. 835–849, Jul. 2015.

[23] H. Ma, T. K. S. Kumar, J. Li, and S. Koenig, "Lifelong multi-agent path finding for online pickup and delivery tasks," in *Proc. Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, vol. 2, Jan. 2017, pp. 837–845.

[24] R. Xu, H. Feng, J. Liu, and W. Hong, "Dynamic spare point application based coordination strategy for multi-AGV systems in a WIP warehouse environment," *IEEE Access*, vol. 10, pp. 80249–80263, 2022.

[25] N. Smolic-Rocak, S. Bogdan, Z. Kovacic, and T. Petrovic, "Time windows based dynamic routing in multi-AGV systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 1, pp. 151–155, Jan. 2010.

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

[27] Q. Li, J. T. Udding, and A. Pogromsky, "Zone-control-based traffic control of automated guided vehicles," in *Coordination Control of Distributed Systems* (Lecture Notes in Control and Information Sciences), vol. 456. Cham, Switzerland: Springer, 2015, pp. 53–60.

[28] W. Małopolski, "A sustainable and conflict-free operation of AGVs in a square topology," *Comput. Ind. Eng.*, vol. 126, pp. 472–481, Dec. 2018.

[29] Y. Zhao, X. Liu, G. Wang, S. Wu, and S. Han, "Dynamic resource reservation based collision and deadlock prevention for multi-AGVs," *IEEE Access*, vol. 8, pp. 82120–82130, 2020.

[30] M. Sauer, A. Dachsberger, L. Giglhuber, and L. Zalewski, "Decentralized deadlock prevention for self-organizing industrial mobile robot fleets," in *Proc. IEEE Int. Conf. Omni-Layer Intell. Syst. (COINS)*, Aug. 2022, pp. 1–6.

[31] J. Zając and W. Małopolski, "Structural on-line control policy for collision and deadlock resolution in multi-AGV systems," *J. Manuf. Syst.*, vol. 60, pp. 80–92, Jul. 2021.

[32] W. Małopolski and J. Zając, "AGVs collision and deadlock handling based on structural online control policy: A case study in a square topology," *Appl. Sci.*, vol. 11, no. 14, p. 6494, Jul. 2021.

[33] E. W. Djikstra, "A note on two problems in connection with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, 1959.

[34] R. Sedgewick, *Algorithms*. Reading, MA, USA: Addison-Wesley, 1983.

[35] P. Verma, M. Diab, and J. Rosell, "Automatic generation of behavior trees for the execution of robotic manipulation tasks," in *Proc. 26th IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2021, pp. 1–4.

[36] O. Ruiz-Celada, P. Verma, M. Diab, and J. Rosell, "Automating adaptive execution behaviors for robot manipulation," *IEEE Access*, vol. 10, pp. 123489–123497, 2022.

PARIKSHIT VERMA received the degree in mechatronics engineering from Thapar University, India, in 2016, and the master's degree in automatic control and robotics from UPC, Barcelona. He is currently pursuing the Industrial Ph.D. degree in mobile robotics from UPC and Tavil Ind. S.A.U., Spain. He has working experience in the industry. He is passionate about applied robotics and the integration of robots in real and complex industrial scenarios.

JOSEP M. OLM received the M.Sc. degree in physics from the University of Barcelona, Barcelona, Spain, in 1989, and the Ph.D. degree in physics from Universitat Politècnica de Catalunya, Barcelona, in 2004.

Since 2003, he has been with Universitat Politècnica de Catalunya, where he is currently an Associate Professor with the Department of Mathematics and a Researcher with the Institute of Industrial and Control Engineering. He is the author or coauthor of 90 journal articles and conference papers and one book. His research interests include control theory, including sliding, adaptive, repetitive, and complex network control.

Dr. Olm served as an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, from 2014 to 2015.

RAÚL SUÁREZ (Senior Member, IEEE) received the degree in electronic engineering from the National University of San Juan, San Juan, Argentina, in 1984, and the Ph.D. degree in electronic engineering from Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, in 1993.

He was responsible for the research line on process control at the Institute of Industrial and Control Engineering (IOC), from 1998 to 2003, where he was the Deputy Director, from 2003 to 2009, and the Director, from 2009 to 2016. Since 2010, he has been responsible for the Division of Robotics, IOC. He was the Coordinator of the Ph.D. Program on Robotics, UPC, from 1995 to 2023. He is currently a Research Supervisor with IOC, UPC. He is the coauthor of about 100 scientific papers published in international conference proceedings and journals. His research interests include intelligent robotics, particularly in the areas of motion planning, human-like movements, robotized grasping, and dexterous manipulation. He was the President of the IEEE RAS Spanish Chapter, from 2019 to 2023.