Hierarchical Traffic Management of Multi-AGV Systems with Deadlock Prevention Applied to Industrial Environments

Federico Pratissoli[1], Riccardo Brugioni[1], Nicola Battilani[2], Lorenzo Sabattini[1]

***Abstract*—This paper concerns the coordination and the traffic management of a group of Automated Guided Vehicles (AGVs) moving in a real industrial scenario, such as an automated factory or warehouse. The proposed methodology is based on a three-layer control architecture, which is described as follows: 1) the Top Layer (or Topological Layer) allows to model the traffic of vehicles among the different areas of the environment; 2) the Middle Layer allows the path planner to compute a traffic sensitive path for each vehicle; 3) the Bottom Layer (or Roadmap Layer) defines the final routes to be followed by each vehicle and coordinates the AGVs over time. In the paper we describe the coordination strategy we propose, which is executed once the routes are computed and has the aim to prevent congestions, collisions and deadlocks. The coordination algorithm exploits a novel deadlock prevention approach based on time-expanded graphs. Moreover, the presented control architecture aims at grounding theoretical methods to an industrial application by facing the typical practical issues such as graphs difficulties (load/unload locations, weak connections, ...), a predefined roadmap (constrained by the plant layout), vehicles errors, dynamical obstacles, etc. In this paper we propose a flexible and robust methodology for multi-AGVs traffic-aware management. Moreover, we propose a coordination algorithm, which does not rely on ad hoc assumptions or rules, to prevent collisions and deadlocks and to deal with delays or vehicle motion errors.**

***Note to Practitioners*—This paper concerns the coordination and the traffic management of a group of Automated Guided Vehicles (AGVs) moving in a real industrial scenario, such as an automated factory or warehouse. The proposed methodology is based on a three-layer control architecture, which is described as follows: 1) the Top Layer (or Topological Layer) allows to model the traffic of vehicles among the different areas of the environment; 2) the Middle Layer allows the path planner to compute a traffic sensitive path for each vehicle; 3) the Bottom Layer (or Roadmap Layer) defines the final routes to be followed by each vehicle and coordinates the AGVs over time. In the paper we describe the coordination strategy we propose, which is executed once the routes are computed and has the aim to prevent congestions, collisions and deadlocks. The coordination algorithm exploits a novel deadlock prevention approach based on time-expanded graphs. Moreover, the presented control architecture aims at grounding theoretical methods to an industrial application by facing the typical practical issues such as graphs difficulties (load/unload locations, weak connections, ...), a predefined roadmap (constrained by the plant layout), vehicles errors, dynamical obstacles, etc. In this paper we propose a flexible and robust methodology for multi-AGVs traffic-aware management. Moreover, we propose a coordination algorithm, which does not rely on ad hoc assumptions or rules, to prevent collisions and deadlocks and to deal with delays or vehicle motion errors.**

***Index Terms*—Multi-robot system, Automated factory, Mobile robotics, Software, Traffic management, Deadlock**

[1] Department of Sciences and Methods for Engineering (DISMI), University of Modena and Reggio Emilia, Italy {federico.pratissoli, lorenzo.sabattini}@unimore.it, brugioniriccardo@gmail.com,

[2] Industria Tecnologica Italiana S.r.l (IT-I), Reggio Emilia, Italy nicola.battilani@it-i.it

This paper describes the results found during the research project in collaboration with Proxaut s.r.l.

This work was supported by the COLLABORATION Project through the Italian Ministry of Foreign Affairs and International Cooperation.

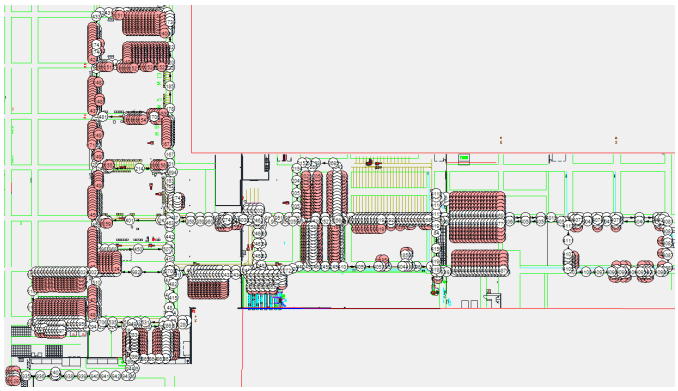

Fig. 1: The figure shows the factory structure (red, green, and black drawings) and the plant graph $\mathcal{G}$ of our use case, which is elicited by a real application. In particular, red nodes indicate load or unload locations, while red nodes indicate transit areas.

I. Introduction

Automated warehouses and automated factories are spreading as the solution to the increasing demand and the Automated Guided Vehicle (AGV) systems are consequently gaining popularity and relevance. AGVs increase efficiency, flexibility, and reduce costs by helping to automate a manufacturing facility or warehouse [1]. As such, several strategies have been developed to deal with major issues in automated warehouses, such as traffic management or safety and performance guarantees [2]. Along these lines, motion coordination of a high number of vehicles has become a widely studied research topic in the field of multi-robot systems [3].

Different strategies have been deepened to increase global performance indices such as efficiency, safety, scalability, or robustness to failures [4]. Two approaches can be mainly found in literature: centralized and decentralized approaches. The former strategy is able to find the optimal solution for the

multi-robot system planning and control problem [5]. Thus, centralized approaches show generally better performances when compared to decentralized ones [6]. However, the problem complexity — and hence computational costs — quickly becomes impractical while increasing the number of robots over a few tens [7]. Moreover, centralized control methods are usually classified as coupled or decoupled. Coupled methods [1] control the whole system exploiting classical single-vehicle motion planning control methods. The task is to find a path for each agent in the group, from a start to a goal position, which avoids collisions between agents. Two robots are in a collision if they are located in the same position at the same time. This Multi-Agent Path Finding problem (MAPF) [8] can be solved for example exploiting the well-known Conflict-Based Search (CBS) optimal algorithm [9], a typical coupled approach. These approaches generally find optimal solutions, at the cost of a high computational effort and limited scalability. Decoupled methods [10], [11], [12] simplify the vehicles coordination and control problem by dividing it into two main phases: path planning and motion coordination. Decoupled control algorithms are generally much faster than coupled ones, however, they may find sub-optimal solutions and they are prone to generating deadlocks.

In decentralized approaches [11], [13], [14], each robot communicates with its neighboring ones to estimate the global performance of the system and to make local control decisions [15]. Decentralized methodologies are computationally less demanding than centralized ones. The computation for the multi-robot system coordination is shared among the agents, making the system easily scalable to large scenarios [16]. However, these methodologies present some disadvantages: they generally find a sub-optimal solution to the coordination problem and, especially, they may fail in finding a solution and a feasible path for every agent in the system due to deadlocks [17], [18]. Moreover, the complexity may still be high in terms of exchanged messages or message size.

For these reasons, although relevant studies have been proposed to manage the coordination of automated vehicles exploiting decentralized approaches, the traffic and coordination of AGVs in industrial scenarios is generally managed by a centralized supervisor. In other words, there exists a central computational unit that manages all the information coming from the environment and the vehicles to optimally coordinate the vehicles. Moreover, decoupled approaches are preferable to coupled ones, since these generally require a significant amount of computational effort, are generally more vulnerable to failures and are hardly scalable to large teams of robots [19]. The management of a fleet of AGVs includes dealing with a multi-robot path planning problem: in decoupled control strategies this is generally decomposed into modules to reduce the complexity [20]. A hierarchical architecture control strategy can be exploited to solve planning and coordination problems for large-scale infrastructures [17]. Every robot is coordinated along its planned path to its goal in order to avoid conflicts and dissolve deadlocks.

Typically, in modern automatic warehouses and factories the movements and trajectories of the automatic vehicles are defined by the roadmap, which consists of a set of predefined virtual paths, as illustrated in [21]. The roadmap depends on the plant structure and has a relevant role in the traffic management of the AGVs affecting the performance of the control system. Thus, typically, in industrial applications, once a path has been computed, it is assigned to every AGV and their coordination is managed over the plant following a set of traffic rules manually defined during the installation of the system [22]. This approach requires a lot of personnel working during the AGV system deployment in the plant and when variations are required in the system, since all the exceptions have to be manually managed. The work illustrated in [23], [24] describes the coordination diagrams, which are tools used for representing the possible collisions among the vehicles, useful to handle the coordination of the AGVs limiting the use of manual traffic rules. Several methodologies have been proposed for the definition of a roadmap, typically based on random sampling techniques [25], [26] or using probabilistic methods [27]. Another strategy is proposed by [28] which is more focused on the roadmap definition for industrial scenarios. These works aim to find a solution for the automatic generation of the roadmap, building the framework for traffic management of the AGVs upon such solution. Thus, the roadmap is defined to help the coordination of vehicles in the warehouse or factory. However, this approach excludes the majority of the industrial scenarios where the roadmap is provided a priori and represents then a constraint for the traffic manager: e.g. the coordination strategy proposed in [28] can not be applied in these industrial scenarios. Moreover, the mentioned methods are designed for managing robots in static, dedicated environments and cease to work if key assumptions on the infrastructure or on the fleet are dropped, as typically happens in industrial applications [29].

In this paper, we present a centralized and decoupled control strategy to efficiently coordinate the movements of a fleet of automated vehicles in a real industrial scenario. The proposed approach allows us to build a control system scalable to large fleets of vehicles and able to efficiently deal with the issues and performance requirements typical of an industrial application. This study is built upon the work presented in [30], where preliminary results were introduced on a hierarchical approach and methodology to design a flexible traffic manager able to coordinate a fleet of real AGVs in an industrial scenario. The control strategy exploits a multiple layer approach, which allows to simplify path planning computation, model the traffic and coordinate the vehicles on the planned routes. The proposed control architecture is built upon a predefined and fixed roadmap of the plant. In fact the structure and organization of an automated warehouse or factory is usually designed to optimize and simplify the coordination of multiple AGVs in the plant. The presented work considers the common case where the AGV system has to be deployed in an existing and structured plant, where the roadmap of the plant is given and cannot be modified. The proposed methodology aims to introduce a flexible and an efficient system able to deal with non-idealities and issues introduced typically by a real and industrial implementation of the roadmap, such as the presence of locations where AGVs are required to perform loading/unloading operations (red nodes in Fig. 1 and Fig. 3),

lack of redundancy or bidirectional single vehicle corridors. The proposed strategy, as stated in [30], aims to be robust to the issues that usually affect industrial scenarios, such as dynamic obstacles and communication errors [31]. While the overall architecture was only presented in [30] in a preliminary form, in this paper we propose the full coordination strategy, including a detailed description of the coordination method implemented on the lowest layer of the hierarchical architecture, which aims to avoid the collisions among the AGVs, which are moving on the precomputed routes. The coordination algorithm, described in Section VI, exploits the prediction of future locations of the AGVs to solve the future conflicts. Furthermore, deadlock issues were not considered in [30]: in this paper we present a deadlock prevention algorithm, which exploits the time expanded graphs [32].

Deadlock situations arise when a group of tasks becomes interlocked in such a way that they cannot be completed. The authors in [33] theorize the general conditions sufficient to generate a deadlock situation. The most relevant one for a multi AGV system defines that a deadlock state is generated by a circular chain of tasks, such that each task holds the resources needed by the next task in the chain. Different approaches were studied to prevent deadlock occurrence in a multitasking system for various scenarios [34], [33]. In a multi vehicle system scenario, the authors expressed the traffic deadlock in terms of graphs and show how a circuit (directed loop) in the generated graph is a necessary and sufficient condition for a deadlock.

In this paper, we propose a methodology based on time expanded graphs to deal with traffic deadlocks for every time step in the trajectory followed by the AGVs. The proposed methodology has been integrated into the control software of real industrial plants, in collaboration with Proxaut s.r.l.: this allowed us to validate the proposed methodology in a real use case. In this work, we describe the experiments we conducted in one automated factory and we compare the performance of the developed software with the company's one usually implemented on its plants.

The rest of the paper is organized as follows. Section II aims at providing the reader with the main background notions that will be used in this paper. Section III gives an overview and a brief description of the control architecture proposed in this paper. Section IV describes in detail the proposed multi-layer architecture, the role of the layers in the coordination and planning and their interconnections. Section V focuses on the proposed path planning strategies adopted to consider the traffic status in the path computation. In Section VI we describe the whole traffic manager algorithm, the AGVs coordination methodology and the proposed deadlock prevention policy. Finally, Section VII reports the simulations and the experiments conducted on the real industrial plant and Section VIII deals with the conclusion and future works.

II. Algebraic Graph Theory

In this section, we recall some notions on graph theory that will be used in the paper to model interconnections in our multi-layered architecture. The reader is referred to [35] for additional details.

Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a directed graph characterized by a set $\mathcal{N}(\mathcal{G})$ of vertices or nodes and a set $\mathcal{E}(\mathcal{G}) \subseteq \mathcal{N}(\mathcal{G}) \times \mathcal{N}(\mathcal{G})$ of edges. Given an edge $(i, j) \in \mathcal{E}$, then the node j is a neighbor of i. Let $\mathcal{K}$ be a subgraph of $\mathcal{G}$, then $\mathcal{K}$ is a graph where the node set $\mathcal{N}(\mathcal{K})$ is a subset of $\mathcal{N}(\mathcal{G})$ and the edge set $\mathcal{E}(\mathcal{K})$ is a subset of $\mathcal{E}(\mathcal{G})$. A path is a finite sequence of edges that joins a sequence of nodes, which are all distinct. In a path of length L, the nodes can be listed as $\{n_1, n_2, \ldots, n_L\}$, such that the edges are (n_i, n_{i+1}), where $i = 1, 2, \ldots, L-1$.

Let $\mathcal{G}$ be a directed and *connected* graph, then at least one path exists between each pair of nodes in $\mathcal{N}(\mathcal{G})$. Moreover, $\mathcal{G}$ is a *strongly connected* graph if, for any pair of nodes u, v in $\mathcal{N}(\mathcal{G})$, a directed path can be defined from u to v and from v to u. Contrariwise, $\mathcal{G}$ is a *weakly connected* graph if, for any pair of nodes u, v in $\mathcal{N}(\mathcal{G})$, a directed path exists from u to v, but not necessarily from v to u.

Given a directed graph $\mathcal{G}$, we assume each edge in $\mathcal{E}(\mathcal{G})$ is associated with a positive weight. Thus, given two nodes $u, v \in \mathcal{N}(\mathcal{G})$, the minimum cost path from u to v can be computed by using standard graph search algorithms (e.g., Dijkstra, A*, DFS — see [36], [37] for details). In this paper, we exploit the graph to represent the predefined feasible routes to be followed by vehicles in the plant. In this sense, the graph nodes $\mathcal{N}(\mathcal{G})$ represent the reachable locations in the plant and the edges $\mathcal{E}(\mathcal{G})$ model the available roads between two nodes. Accordingly, the edge weight can be arbitrarily defined to quantify relevant information: in this paper, we will use the weights to quantify the expected travel time. Standard graph search algorithms are used to compute the path from a start to a goal location in the plant minimizing the travel time.

Moreover, let us introduce the time-expanded approach [38] which constructs the time-expanded digraph in which every node corresponds to a specific time event and edges between nodes represent the connections between the two events. Every node of the time-expanded graph consists of a static graph, hence a time-expanded graph can be considered as a sequence of static graphs (see Fig. 2). Let $\mathcal{G}_t(t)$ be a directed time-expanded graph. We have that $\mathcal{G}_t(t) = \{\mathcal{G}(1), \mathcal{G}(2), \ldots, \mathcal{G}(i)\}$, where $\mathcal{G}(i)$ is a snapshot of $\mathcal{G}_t$ at time $t = i$ and is denoted by $\mathcal{G}(i) = (\mathcal{E}(i), \mathcal{N}(i))$. In the proposed control strategy, we exploit the time expanded graph to represent the *precedences* and the result of the negotiation between vehicles following the execution of the coordination algorithm. Each node of the graph represents an automated vehicle in a particular instant of time and the directed edges between the nodes represent the result of the negotiation determined by the coordination diagram (see Sec. VI-A). The utility of the time expanded graph is to represent the precedences and the negotiation over time. Hence, for example, in Figure 2, at the instant of time $t = 4$ the AGV represented by node A has to wait for the AGV C, and similarly, the AGV D has to wait for the AGV B.

III. System Overview

The system proposed in this paper is composed of *(i)* a multi-layer architecture to model the environment, and *(ii)* a traffic-aware coordination and path planning strategy. The

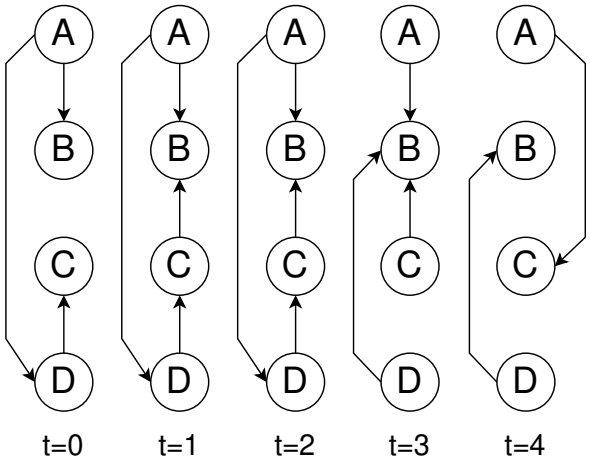

Fig. 2: An example of directed time expanded graph used in the proposed coordination strategy. The graph considered in figure is composed by four nodes: A, B, C, D. The graph is considered over time and the edge set $\mathcal{E}(\mathcal{G}(i))$ at time $t = i$ can be different from the edge set $\mathcal{E}(\mathcal{G}(i+1))$ at time $t = i+1$.

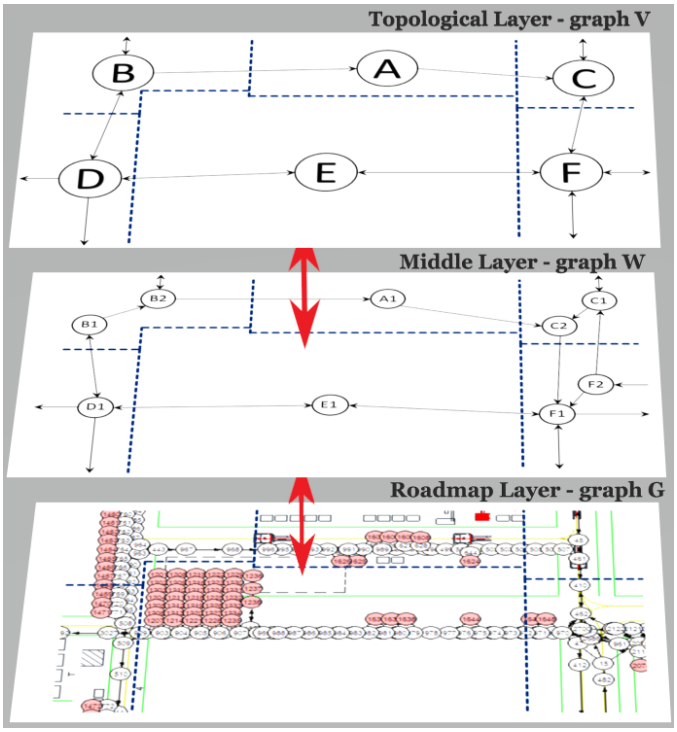

Fig. 3: Overview of the proposed multi-layer architecture. The figure shows a portion of the three graphs, one for each layer, of the studied use case, from the bottom: the roadmap layer, the middle layer, the topological layer. On the roadmap layer the red nodes indicate the load or unload locations in the factory, while the white nodes indicate the transit locations. The blue dotted lines show the different plant areas, divided by topology, each one assigned to a sector S.

main objective is to coordinate a large number of AGVs in an industrial environment, preventing traffic congestion. The control approach is centralized to meet the performances and the efficiency required by an industrial application, hence the trajectories are optimally computed for every vehicle with a global knowledge over the industrial plant. Furthermore, the control architecture is decoupled in the path planning and in the coordination strategy to reduce the computational cost and make the system easily scalable. The multi-layer architecture, pictorially represented in Fig. 3, will be detailed in Section IV.

To model the traffic evolution quantitatively, we divide the environment into three layers: *(i)* the Top Layer (or Topological Layer), responsible for traffic management; *(ii)* the Middle Layer, responsible for the path planning of the AGVs; *(iii)* the Bottom Layer (or Roadmap Layer), containing all possible vehicle routes and handling the fleet coordination. The roadmap models all the possible trajectories the vehicles can follow and all the locations the vehicles can occupy in the plant. Hence, the roadmap model consists in a graph where the nodes represent the locations sampled in the plant. The accurate tracking of the AGV over the roadmap is guaranteed by the high number of points (the graph nodes) that sample the feasible trajectories. A trajectory is, indeed, modeled as a sequence of connected nodes.

Hereafter, the term "path" refers to a generic sequence of nodes computed by the planner over a graph, while "route" refers to the specific sequence of nodes computed over the roadmap, which is necessary to determine the trajectories to be followed by the vehicles. Upon the computation of the route and trajectory for each vehicle, it is possible to determine the location of the vehicles over time, since they move with a known velocity profile. As a result, it is possible to predict the future location of the vehicles and take action to prevent congestion and deadlocks.

The path planning and coordination strategy, detailed in Section V, exploits information from the three layers, to define an optimal and feasible route for each AGV. In detail, the Top Layer is based on a discretization of the environment in topological areas (intersections, load/unload areas, congestion risk areas, etc.) and provides quantitative information about the current AGVs traffic in each of them. Each AGV's path planning consists of two main steps. A path is computed over the Middle Layer, which models the actual connections among the areas of the Topological Layer and knows the traffic information of these areas. Subsequently, the computed path is exploited to plan the actual feasible route of the AGVs over the Bottom Layer, where the trajectories are built over time and conflicts among the AGVs are avoided by means of coordination.

Given a route, the trajectory following problem is the problem of computing a kynodynamically feasible temporal profile of the vehicle control inputs to move the robot along the given route. This problem is beyond the scope of this paper and will not be addressed. Thus, we assume each AGV is equipped with an on-board control algorithm capable of ensuring safety (i.e. [39], [40]) and accurate trajectory tracking (i.e. [41], [42]).

Once the route is computed from the path planner in the Bottom Layer, the AGVs have to be coordinated on the respective paths to prevent collisions and deadlocks. A col-

lision between two AGVs occurs when they occupy the same location at the same time. The purpose of the coordination is to modify the time the vehicle reaches the conflicting locations. In this paper we propose a coordination strategy based on a future prediction of the AGVs trajectories able to solve future conflicting situation and prevent congestions or deadlocks. Deadlock situations arise when a group of vehicles becomes interlocked in such a way that they cannot complete their tasks. The coordination between AGVs is represented exploiting *precedence* graphs, where the nodes are the AGVs and the edges represent the precedences between them. This information is extended over time through the time expanded graph where every layer coincides with a precedence graph at definite time step. Avoiding cyclic loop inside the graph we prevent deadlocks [33].

We assume that the coordination strategy runs on a central elaboration unit, communicating with the AGVs. In particular, the central control process periodically sends a command to the AGV, which contains the future steps of the route the vehicle has to follow and the time the vehicle has to wait in a node (location) that the traffic manager has set to manage congestions and deadlocks. Moreover, we assume each on-board low-level control of the vehicle can deal with communication non-idealities, such as packet loss or transmission delays. Hence, communication issues are assumed non-critical for the low-level control. However, it may occur that the AGV is not able to execute a given command because of communication issues, robot hardware issues or safety issues. Since all these issues bring to a stop/slowdown of the vehicle, they can be equally treated by the central control traffic manager as unexpected execution errors on the interested vehicle. Therefore, the proposed control strategy needs to be robust to the delays introduced on the AGV motion by these errors (in the video attached we show some of the experiments conducted).

IV. Multi-Layer Architecture

We here detail the proposed multi-layer architecture by describing all layers, their interconnections and how they are used for coordination and traffic management.

A. Roadmap Layer (Bottom Layer)

The roadmap of the plant represents all the possible routes the AGV can follow and the locations the AGV can cross in the environment. The Bottom Layer provides a modeling of the roadmap through a graph.

This layer is responsible for the planning of the route to be followed by the AGV. The *plant graph* $\mathcal{G}$ is then introduced to abstract the description of the roadmap: the set of nodes $\mathcal{N}(\mathcal{G})$ represents the locations in the plant the vehicles can reach, and the set of edges $\mathcal{E}(\mathcal{G})$ represents the feasible trajectories connecting those locations. Thus, let p_i, p_j be two locations in the plant, associated with nodes $i, j \in \mathcal{N}(\mathcal{G})$. Then, the edge (i, j) exists in $\mathcal{E}(\mathcal{G})$ if there exists a kinematically feasible route from p_i to p_j. Each edge in $\mathcal{E}(\mathcal{G})$ is weighted by the distance between the two corresponding nodes and it also contains the average time required by the vehicle to travel this distance. Therefore, the planned route traveled by the vehicle and modeled as a sequence of nodes in $\mathcal{N}(\mathcal{G})$ is able to describe the vehicle's motion over time. Therefore, the planned route traveled by the vehicles, which is represented as a sequence of nodes in $\mathcal{N}(\mathcal{G})$, serves as an effective means of describing the vehicle's motion over time. The traffic manager planner computes the route over the graph $\mathcal{G}$ that not only minimizes the time required by the vehicle to reach its goal, but also minimizes the traffic congestion in certain areas of the plant. This traffic sensitive path planning is performed thanks to the topological layer.

In this paper we consider a common industrial scenario in which the AGV system has to be deployed in an already existing plant. The roadmap, hence, is fixed and can not be changed to optimize the traffic management of vehicles as in [28]. The plant graph, built upon the roadmap, is consequently treated as a constraint for the modeling, planning and the design of the control architecture.

A representative example of a roadmap of a middle-size plant is depicted in Fig. 1: we will hereafter refer to this example as a use case for the proposed system.

B. Topological Layer (Top Layer)

The Top Layer is the most abstract layer, and considers the topological representation of the factory or warehouse as a set of *sectors*. Sectors correspond to plant areas, such as intersections, corridors, load or unload areas, relevant for traffic management. In particular, each sector $\mathcal{S}$ can be distinguished from the others according to either topological, geometrical or logistical characteristics or other particular constraints.

The set of interconnected sectors defines a directed graph $\mathcal{V}$, referred to as the *Sectors Graph*. Each node $S \in \mathcal{N}(\mathcal{V})$ represents a sector and is defined as follows. Let $\mathcal{N}(\mathcal{G}_S)$ be the subset of nodes in $\mathcal{N}(\mathcal{G})$ located inside the sector S, where $\mathcal{G}$ is the plant graph and $\mathcal{N}(\mathcal{G})$ the respective set of nodes (see Section IV-A). Thus, we have that $\mathcal{G}_S$ is a subgraph of $\mathcal{G}$ and $\mathcal{E}(G_S) \subseteq \mathcal{E}(\mathcal{G})$. We also assume that all nodes in $\mathcal{N}(\mathcal{G})$ cannot lay on the boundary of two sectors in $\mathcal{V}$, i.e., for each $u \in N(\mathcal{G}), u \in N(S_i) \wedge u \notin N(S_j) \iff S_i = S_j$. It is worth noting that the subgraph $\mathcal{G}_S$ is not necessarily connected.

Let every edge in $\mathcal{E}(\mathcal{V})$ be the connection between two sectors. Let $S_1, S_2 \in \mathcal{N}(\mathcal{V})$ be two sectors, then an edge in the plant graph $\mathcal{G}$ that connects a node in G_{S_1} with a node in G_{S_2} implies that an edge in $\mathcal{E}(\mathcal{V})$ exists to connect S_1 with S_2.

We remark that the main role of the *Sectors Graph* is to model the traffic of vehicles in the plant and, through the subsector graph (which will be defined in See section IV-C), to define a traffic-sensitive path planner (i.e., to avoid congestions.)

Each plant area defined by the respective sector can manage a maximum number of AGVs without occurring in traffic issues and congestions. This number is the capacity of the sector. In the proposed approach the sector's capacity is set by the user considering the plant area dimension, its topology and the presence of escape ways for vehicles in case of high

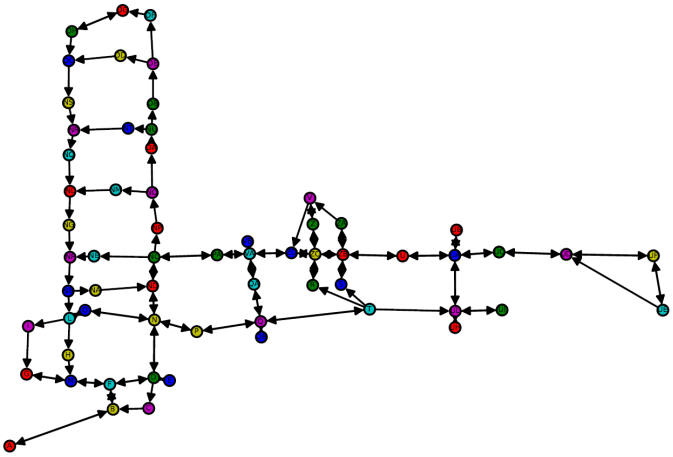

Fig. 4: A full representation of the sector graph $\mathcal{V}$ used to model the traffic in our use case, which is elicited by a real industrial application.

traffic situations. A sector is labeled as congested if it contains more vehicles than its capacity.

The *Sectors Graph* of the considered use case is shown in Fig. 4. Every node of the graph $\mathcal{V}$ keeps the information about the respective sector capacity. The weight of the edge that connects the sector i with the sector j is computed at time t as:

$$w_{i,j}(t) = T_j(t) + D_{i,j} \tag{1}$$

where T_j is the traffic weight for the specific sector j defined as follow:

$$T_j(t) = \begin{cases} K \cdot \frac{N_j(t)}{C_j - N_j(t)} & \text{if } C_j > N_j \\ K \cdot C_j & \text{otherwise} \end{cases} \tag{2}$$

where $N_j(t)$ is the number of AGVs within the sector j at time t, C_j is the capacity of the sector j, $D_{i,j}$ is the Euclidean distance computed between the centers of the two sectors i and j, which is proportional to the travel time given the constant velocity of the AGV, and K is a static gain. Therefore, the path planner has the possibility to compute the optimal path that not only minimizes the distance to the destination but also takes into account the traffic situation and the potential for congestion (see Sec. IV-C).

Partitioning the environment in sectors is a critical operation, that requires human intervention. In particular, human operators are required to provide the boundaries between different sectors and each sector capacity since both are application dependent. Conversely, the set $\mathcal{N}(\mathcal{V})$ can be automatically computed, e.g., using the Voronoi partitioning proposed in [12]. The sector is individuated to isolate a particular area in the plant, such as intersections, corridors or load/unload locations or parking locations. Once the sectors have been defined, it is then necessary to tune each sector capacity based on the specific characteristics of the respective plant area.

C. Middle Layer

The Middle Layer has a similar definition to that of the Topological Layer and it allows to have a traffic sensitive path planner by exploiting the information stored in the Sectors Graph $\mathcal{V}$ thorough the parameter in (2). Over this layer, the controller integrates the traffic information collected on the Top Layer with the aim to obtain the routes, defined on the Roadmap Layer, to be followed by the AGVs. In particular, as shown in Fig. 3, each sector in the sectors graph $\mathcal{V}$ corresponds to one or more subsectors on the Middle Layer. Thus, similarly to $\mathcal{V}$ on the Top Layer, we define the *subsector graph* $\mathcal{W}$, in which each node is a subsector (that will be defined hereafter) and each edge is a connection between two subsectors.

The subsector graph is exploited by the controller to define a feasible route on the roadmap that minimizes the traffic of vehicles modeled by the sector graph. Finally, the planner generates from the path on the subsector graph $\mathcal{W}$ the optimal route on the plant graph $\mathcal{G}$.

As described in [30] a subsector consists in a subset of the roadmap nodes belonging to a sector. In particular, Let $S \in \mathcal{N}(\mathcal{V})$ be a sector, and $\mathcal{G}_S$ the respective subgraph, then $\mathcal{N}(\mathcal{G}_S)$ is the set of nodes on the plant graph that belongs to the sector S, such that $\mathcal{N}(\mathcal{G}_S) \subseteq \mathcal{N}(\mathcal{G})$. Thus, let $U_S \in \mathcal{N}(\mathcal{W})$ be a subsector and $\mathcal{G}_{U_S}$ the respective subgraph: then, $\mathcal{N}(\mathcal{G}_{U_S})$ is the set of nodes in the plant graph that belong to subsector U_S, such that $\mathcal{N}(\mathcal{G}_{U_S}) \subseteq \mathcal{N}(\mathcal{G})$. Finally, we have that U_S is a subsector of S if $\mathcal{N}(\mathcal{G}_{U_S}) \subseteq \mathcal{N}(G_S)$.

The subsectors are determined from the sectors graph. We consider two neighboring sectors S_h and S_k, and $\mathcal{G}_{S_h}$ and $\mathcal{G}_{S_k}$ the respective subgraphs. Let the node $n_1 \in \mathcal{N}(G_{S_h})$, if a path exists that connects n_1 to one node in $\mathcal{N}(\mathcal{G}_{S_k})$, then n_1 is *connected* to sector S_k. Thus, a subsector U_S of S_h can be defined as the set nodes in $\mathcal{N}(\mathcal{G}_{S_h})$ that are connected to the same neighboring sectors. Moreover, let U_{S_1}, U_{S_2} be two subsectors over $\mathcal{W}$. Let U_{S_i} belonging to S_i and $\mathcal{G}_{U_{S_i}}$ be the subgraph of $\mathcal{G}$ associated with S_i, $i \in \{1, 2\}$. Then, an edge from U_{S_1} to U_{S_2} exists in $\mathcal{E}(\mathcal{W})$ if there exists a path over $\mathcal{G}$ connecting two nodes $(u, v) \in \mathcal{N}(\mathcal{G}_{U_{S_1}}) \times \mathcal{N}(\mathcal{G}_{U_{S_2}})$. The edge from U_{S_1} to U_{S_2} is weighted according to (1), as for the edge from S_1 to S_2. The weight on the edge that connects two subsectors belonging to the same sector is zero. Depending on the subgraph connectivity defined within the sector, in the extreme cases, a subsector can coincide with a single roadmap node or with the whole sectors it belongs to. Each sector contains at least one subsector. Moreover, let $\mathcal{G}_{U_S}$ be a subgraph of the plant graph $\mathcal{G}$, defined by a subsector $U_S \in \mathcal{W}$, whose node set is given by $\mathcal{N}(\mathcal{G}_{U_S})$, then $\mathcal{G}_{U_S}$ is at least weakly connected by definition.

For further information about the subsector division and why this is necessary the reader is referred to [30].

V. Path Planning

In this section, we describe the proposed strategy based on the hierarchical architecture for the multi-AGV path planning. The method builds upon the architecture presented in Section IV and consists of exploiting all the information provided by the layers to obtain a traffic sensible route planning

and replanning strategy robust to errors and uncertainties introduced by a real scenario.

A. Path Planning

We assume that each AGV is assigned to a mission, which involves moving from a start location $v_s \in \mathcal{N}(\mathcal{G})$ to a goal location $v_g \in \mathcal{N}(\mathcal{G})$[1].

Let $S_s, S_g \in \mathcal{N}(\mathcal{V})$ and $U_{S_s}, U_{S_g} \in \mathcal{N}(\mathcal{W})$ be the subsectors and sectors containing the start and goal, respectively. Thus, a path is computed on $\mathcal{W}$ that connects subsectors U_{S_s} and U_{S_g}. The traffic status, which characterizes every sector $S \in \mathcal{N}(\mathcal{V})$, is monitored and modeled on the Top Layer over the sectors graph and, at each time t is shared with the respective subsectors U_S in $\mathcal{W}$. Thus, the traffic weight is the same for all the subsectors belonging to the same sector, and is the one computed on the Top Layer for the respective sector according to (2). Finally, the edges in $\mathcal{E}(\mathcal{W})$, similarly to the edges in $\mathcal{E}(\mathcal{V})$, are weighted according to (1). Therefore, the path planner aims to optimize the balance between the distance, and the time, needed for the AGV to reach its destination and the level of traffic and the likelihood of congestion in certain areas of the plant.

Finally, the path, represented as a sequence of subsectors, is transformed by the planner into a corresponding route on the roadmap, represented as a sequence of nodes. The roadmap layer contains the information required to model the vehicles' trajectories over time. The route, composed of a sequence of sampled locations within the plant, describes the amount of time necessary for the vehicle to traverse each section of the road as defined by an edge on the plant graph. Therefore, as the average velocity of the AGVs is assumed to be known and constant, given a planned path, it is possible to predict the vehicles' locations over time, as well as the evolution of the traffic within each sector and, ultimately, throughout the entire plant. At every time step, the path for each AGV is replanned and the traffic status is updated consequently. Thus, the optimal solution found by the planner is the best compromise between the traffic conditions and minimal path to the goal.

The planner in the Middle Layer computes the optimal path $\pi_W = \{U_1, \ldots, U_n\}$ on graph $\mathcal{W}$, which aims for each AGV to avoid congested areas minimizing the time required to reach the goal. The path planning algorithm is based on the Dijkstra's algorithm [36] on $\mathcal{W}$, with edge weights defined as in (1). In other words, to exploit the traffic modeling, the properties of each sector are inherited by all its subsectors.

Finally, the path π_W computed over W becomes a constraint for the route π_G computed over the roadmap and plant graph $\mathcal{G}$. Hence, the route π_G is planned over a portion of the plant graph $\mathcal{G}$, which consists in a subgraph of $\mathcal{G}$ constituted by only the nodes in $\mathcal{N}(\pi_W)$, where $\mathcal{N}(\pi_W) \subseteq \mathcal{N}(\mathcal{G})$ is the union of the sets of nodes belonging to every subsector in π_W. The route planning consists in searching a feasible path from v_s to v_g on the defined portion of plant graph exploiting the Dijkstra's algorithm. It is worth noting that, since the subgraph of $\mathcal{G}$ computed on $\mathcal{N}(U_i)$ is guaranteed to be at least weakly connected, a route can always be found. Each AGV's path is dynamically replanned in order to face variations in the traffic status and occurrence of unpredictable events (e.g., the AGV slows down or stops due to the presence of human operators). Each time a path is recomputed, the traffic status is updated to include the last information about the current (and future) locations of the AGVs. The more frequent is the path replanning, the more robust is the traffic coordination, since a frequent path replanning allows a robust traffic coordination with respect to dynamic obstacles, traffic congestion, or tracking/communication errors.

The proposed planning and replanning strategy is summarized in Algorithm 1 and consists in the following steps:

i) The optimal path π_W from the start subsector to the goal subsector is computed/updated over $\mathcal{W}$ (Algorithm 1, line 2). The path planner minimizes a combination of the traveled distance, and hence the expected travel time, and the currently known traffic cost.

ii) The path expressed as a sequence of subsectors π_W corresponds to a sequence of sectors π_V, which are the sectors that will be crossed by the AGV following the planned path. Thus, the number of vehicles inside each sector can be monitored and a model of the traffic status over time can be easily generated. The sector graph $\mathcal{V}$ weights are updated according to (1) following the traffic status changes (Algorithm 1, lines 3 and 4).

iii) The traffic information held by the sector graph V is inherited by the subsector graph W. Hence, the weights of the set $\mathcal{E}(W)$ are updated according to the new traffic information (Algorithm 1, line 5).

Algorithm 1: Traffic modeling and path planning

```
1 if state != goal then
2     π_W ← dijkstra(W);
3     T ← update_traffic_status(π_W);
4     V ← update_weights (V, T);
5     W ← update_weights (W, V);
6 end
```

VI. The Traffic Manager

The centralized and decoupled controller requires an efficient coordination strategy in order to prevent collisions inside the paths computed previously by the planner. Thus, a proper coordination algorithm together with the described path planning algorithm complete the traffic manager proposed in this paper. Inter-agent collisions are then avoided by exploiting the coordination strategy described in this section, which is inspired by the coordination diagram approach described by [23]. The proposed strategy aims to an efficient negotiation between colliding vehicles to manage the traffic and prevent deadlocks. The main traffic manager loop is resumed in Algorithm 2, where the coordination process is executed in line 18. A mission is assigned to an AGV only when idle. Each mission involves achieving a loading location, loading a cargo, moving to the unloading location and unloading the

[1]Mission assignment is interesting yet beyond the scope of this paper. The interested reader is referred, e.g., to [43] for previous studies on the topic.

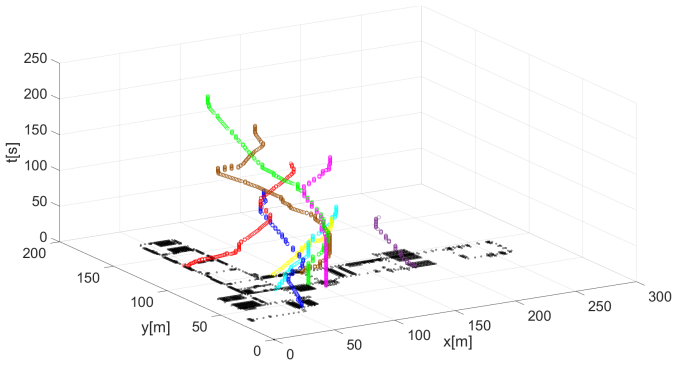

Fig. 5: Visualization of coordination over time of 8 AGVs in the use case plant. The figure provides a visualization of the coordination of 8 AGVs evolved over time in the use case plant considered. On the plane XY the figure shows the node coordinates (the location) in the plant, while, on the Z axis the figure shows the time at which a node is crossed. Different colors indicates different vehicles, i.e. red is for AGV with number 1, blue is for AGV with number 6, etc. It is worth noting that the lines, as sequences of colored nodes, never cross, since the simultaneous presence of two (or more) AGVs on the same node at the same time never takes place. This means that the coordination has been carried on properly and there are no collisions in the AGV trajectories.

cargo. If the AGV has no more missions to be assigned, it is automatically sent to the battery charge location. At every time step, the AGVs positions are updated since the AGV location is frequently monitored in order to deal with delays in the trajectory following. Every time period T the path of each vehicle is computed again and the weights in the sector and subsector graphs are updated according to the new traffic status. The smaller is the time period T, the more frequent is the path computation and the traffic update in the execution of the traffic manager algorithm. The path planning is followed by the coordination algorithm (Algorithm 2, line 18) to deal with collisions and deadlocks. Every time the coordination algorithm is executed the vehicles motion over time is changed in order to face conflicts and, hence, the traffic status has to be updated accordingly (Algorithm 2, line 19).

The higher is the value of T, the lower are both the computational effort and the robustness to dynamical obstacles or vehicle errors. Low values of T imply frequent optimal paths computation and traffic status updates, which makes the control more responsive to unpredictable events and traffic congestions.

A. AGVs Coordination on the Roadmap Layer

The path planner computes the route that optimizes the balance between the vehicles traveled distance, and hence the traveled time, and the level of traffic over the sectors and the plant. The path planner provides the necessary information to consider the evolution of the trajectory over time. Therefore, once the paths are computed for each AGV, we need to verify that no collision occurs during the vehicle's trajectory

Algorithm 2: Traffic management

```
1  Compute_initial_traffic_and_paths();
2  t ← current_time;
3  T ← time step update;
4  while true do
5      update_AGVs_positions();
6      if current_time ≥ t + T then
7          foreach AGV do
8              startnode ← current_position;
9              if goalnode then
10                 update_path(Algorithm 1);
11                 π_G ← π_W;
12             else
13                 goalnode ← assign_new_mission();
14                 update_path(Algorithm 1);
15                 π_G ← π_W;
16             end
17         end
18         coordination_AGVs(Algorithm 3);
19         update_traffic();
20         t ← current time;
21     end
22 end
```

Algorithm 3: AGVs coordination

```
1  H ← time horizon window;
2  t ← current time;
3  collision_free ← false;
4  while not collision_free do
5      foreach t in H do
6          foreach AGV do
7              collision_free, conflicting_AGVs ←
                 get_conflicts(t,AGV);
8              if not collision_free then
9                  AGV_winner, AGV_losers ←
                     negotiation(t, conflicting_AGVs);
10                 AGV_winner, AGV_losers ← dead-
                     lock_policy(t,AGV_winner,AGV_losers);
11                 solve_conflict(t,
                     AGV_winner,AGV_losers);
12             end
13         end
14     end
15 end
```

following. We say that there is a conflict (or a collision) whenever two or more AGVs are required to occupy the same node/edge at the same time. Conflicts are detected while assuming the current paths are traversed with the robot's nominal velocity profile (e.g., minimum time) and accurate tracking.

More in details, neglecting for the moment delays or execution errors, and assuming a nominal trajectory profile, we are able to precisely predict the location of each AGV over a predefined time horizon H. Therefore, conflicts and congestions can be predicted and prevented by sequencing the robot access through the shared resource according to priorities and negotiation. It is worth noting that the vehicles stops or slowdowns are decided by the coordination strategy and, hence, the trajectory model over time and the prediction of the vehicles locations is updated accordingly. This computation over time on the future locations traveled by the vehicles implies longer execution time but ensures conflict prevention.

It should be noted that each AGV possesses the capability to accurately determine its own location within the environment, and is known to frequently transmit updates concerning its location, status, and any potential errors to the central computational unit. To ensure accurate and efficient operation, the location of each vehicle is constantly measured and monitored, in order to promptly identify any execution errors that may cause delays in the trajectory following and requiring an update in the prediction of the AGVs' locations over H. The potential delay of the vehicle on the followed trajectory is subsequently incorporated into the time model of the vehicle and into the coordination strategy to account for any execution errors, thus enabling a reliable prediction of the robot's location. This is crucial for managing future conflicts between the AGVs and preventing deadlocks and congestion.

The selection of the parameter H plays a crucial role to determine how large the prediction time window is, the computational cost and the effectiveness of the coordination algorithm. It follows that as the value of H increases, the system exhibits greater robustness against deadlock and congestion, as there is a larger temporal margin for preventing potential conflicts. However, the higher is the value of H, the higher will be the computational burden of the coordination algorithm. The selection of the parameter H is contingent with the structure of the factory, and, as such, the layout of the plant. In particular, the time horizon has to be chosen sufficiently large to allow the coordination strategy to anticipate and manage the vehicles movements in the longest dead-end present in the plant. Inadequate choice of H will result in suboptimal performance and may lead to vehicles being stuck in a dead-end. Furthermore, it should be noted that the value of H cannot be selected arbitrarily large due to the computational cost associated with the coordination algorithm and thus, a trade-off must be found between the desired performance of the algorithm and the computational resources that are available.

The result of a correct traffic management and, hence, of a correct coordination of AGVs over time can be visualized in Fig. 5, where the locations (or nodes on $\mathcal{G}$) at every time step of each AGVs are plotted. The figure shows how the routes of the AGVs, distinguished with different colors, do not intersect each other when extended in a prediction over time. This means the vehicles are able to reach the final destination without occurring into conflicts and congestions.

Algorithm 4: Negotiation

```
1  AGV_winner ← None;
2  AGV_losers ← None;
3  AGV_winner,AGV_losers =
     motion_error(conflicting_AGVs);
4  if (AGV_winner is None) OR (AGV_losers is None)
     then
5  |  AGV_winner,AGV_losers =
   |    path_obstruction(conflicting_AGVs, T);
6  end
7  if (AGV_winner is None) OR (AGV_losers is None)
     then
8  |  AGV_winner,AGV_losers =
   |    time_to_conflict(conflicting_AGVs);
9  end
10 return AGV_winner,AGV_losers
```

The coordination strategy of the AGVs is resumed in Algorithm 3 and can be divided in 4 main phases:

i) *conflict identification*: the localization in time and space of a conflict between a group of AGVs, i.e. AGVs occupying the same area at the same time. The time horizon H defines how far in time a conflict can be detected and possibly prevented. H is lower-bounded by the topology of the plant, since it has to be chosen wide enough to avoid two AGVs stuck in a bidirectional single vehicle corridor. Every future time steps, hence, has to be checked for possible collisions (Algorithm 3, line 4).

ii) *negotiation*: once we know which are the AGVs in conflict and at which time step the collision will happen, we have to define a policy to assign the precedence on the motion to the vehicles. The negotiation to define which AGV has to wait to solve the conflict is based on simple rules. These rules consider the mission priority, if the AGV has to free an intersection, if the vehicle is stopped due to an internal error and the time required to reach the collision spot. The proposed algorithm incorporates a mechanism to assess with simple rules the status of vehicles regarding their execution and movement. If a vehicle is determined to be stopped due to an execution error, it is deemed as the most probable loser and would have to wait for other vehicles to complete their motion (line 3, algorithm 4). Additionally, the algorithm assesses the potential path obstruction between vehicles in the portion of the route considered within the time window T. The vehicle that obstructs the path of another vehicle is considered as the most probable winner and is selected to clear the passage in order to prevent further obstructions (line 5, algorithm 4). Finally a simple FIFO policy is chosen to end the negotiation, which means that the first vehicle that reaches the collision spot is the negotiation winner, and other AGVs involved in the

collision wait for their motion (line 8, algorithm 4). Hence, at the end of the negotiation there will be a winning AGV and (possibly multiple) loser AGVs (Algorithm 3, line 9). Those rules can be chosen to minimize or maximize a performance index by solving an optimization problem online but, for this specific case, we consider some very basic simple rules, since the efficiency of the solution of the negotiation algorithm is checked by the deadlock prevention policy which guarantees the coordination of the vehicles.

iii) *deadlock prevention*: the use of simple rules for the negotiation policy could lead to the generation of deadlocks. Hence, the proposed coordination strategy is embedded with an effective deadlock prevention policy (DPP), which has the aim to identify and prevent deadlocks before they occur. The DPP we propose, described in details in the following section, extends the concept of precedence graph used in multitasking systems by introducing a temporal variable that is necessary and efficient for identifying and preventing deadlocks in time.

iv) *solve conflict*: the conflict is ready to be solved once the conflicting time step is found and the loser(s) and winning AGVs are defined. The list of commands to be send to the loser(s) is modified to make the AGV(s) waiting for a time length sufficiently large to solve the collision. The process is repeated to check for other collisions and solve them until no more collisions are found in the time window H.

B. Deadlock prevention policy

The implementation of a deadlock prevention policy is an essential strategy for maximizing the possibilities of finding an optimal solution in the coordination of AGVs. The negotiation process for resolving conflicts is commonly based on a set of straightforward rules and can potentially result in deadlock situations that cannot be resolved.

We can identify two types of deadlock scenarios: cyclic deadlocks and acyclic deadlocks, that differ in the nature of the resource locked between the processes involved [44]. Cyclic deadlocks represent a category of deadlock in which the resources that are locked in place exhibit a circular dependency. This is characterized by the existence of a chain of processes, each of which possesses a resource that is being sought by the subsequent process within the chain. Acyclic deadlocks, on the other hand, are a type of deadlock in which the resource dependency forms a tree or a directed acyclic graph (DAG) structure. Hence there are no circular dependencies and each process holds a resource that is not required by any other process in the chain. It is worth noting that in our scenario, acyclic deadlocks are generally generated by a vehicle that is halted as a result of an error, which subsequently locks the resource in question, which in this case is the moving space. Therefore, the coordination strategy lacks the capability to address this issue, and the traffic manager must take steps to update the trajectories of the vehicles in order to limit the congestions until the error is resolved. Finally, second level deadlocks or nested deadlocks are acyclic phenomena typically generated by a cyclic deadlock. In fact, the vehicle that locks resources and generates a tree dependency structure is halted in a cyclic chain of locked resources. Therefore, by preventing the emergence of cyclic deadlocks, it is also possible to prevent the occurrence of nested deadlocks. In the presented control architecture we propose a deadlock prevention policy based on time expanded graphs. The result of the negotiation process, for every future conflict found in the time window H, sets which are the AGVs that have to stop (the losers) and the ones that have to move (the winners) in order to solve the future collision. The set of precedences between the conflicting AGVs in a particular time step t can be represented by a graph. In the proposed approach we exploit the time expanded graph in order to represent the set of precedences between conflicting AGVs in the whole time window H. The analysis of this graph of precedences becomes useful when dealing with deadlocks, since these are caused by cyclical precedences in the time window H [33]. Hence, having an acyclic time expanded graph representing the precedences ensures that the computed trajectories in H are deadlock free.

The time step t at which the collision is predicted to happen is denoted as $t_{conflict}$. In order to solve the conflict, the AGV that has lost the negotiation has to be stopped or slowed down a time window before the collision time in order to allow the winning AGV to cross. This time window is denoted as t_{delta}, and the time step t at which the AGV is stopped is denoted as t_{wait}, so that $t_{wait} + t_{delta} = t_{conflict}$. Every time the future trajectory to be followed by the AGV is changed to solve a collision, the prediction of the AGV motion has to be updated accordingly introducing the delay caused by the stop or slowdown of the vehicle.

The negotiation process brings to a set of possible solutions to solve the collisions in the time horizon H, and the deadlock prevention process is responsible in finding the deadlock free solutions. The time expanded graph is generated over the set of precedences in the time window H obtained from the negotiation process. Every graph node indicates the AGV ID and every edge represents a precedence, with a winner AGV and a loser AGV: for each incoming (outgoing) edge the AGV is a winner (loser) for the respective precedence.

For every collision detected in the predicted trajectory, the time expanded graph is built iteratively, every precedence edge is added step by step from the $t_{conflict}$ until the t_{wait}, when the conflict is solved. For example, Fig. 6 shows a coordination between 3 AGVs: AGV 1 and AGV 2 have a conflict predicted at time $t = t_4$ and, to avoid the collision with sufficient space, AGV 1 has to wait 3 time steps before at time $t = t_1$. Similarly, AGV 2 and AGV 3 have a conflict at time $t = t_5$, where the AGV 2 is stopped 2 seconds before at time $t = t_3$. Hence, at time $t = t_4$ and $t = t_3$ we have two conflict situations which have been solved with AGV 1 stopped for AGV 2 and AGV 2 stopped for AGV 3.

After every negotiation at every time step the time expanded graph of precedences is updated and analyzed in order to ensure that the new edges introduced do not bring to a deadlock scenario. This is the case shown in Fig. 7, where the conflict between AGV 1 and AGV 3 at time $t = t_8$ could

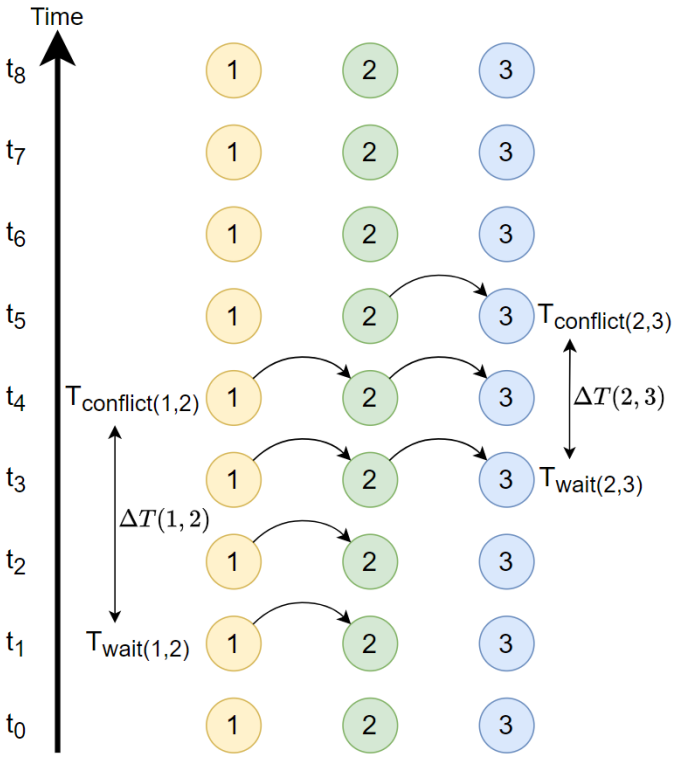

Fig. 6: A representation through the time expanded graphs of the precedences defined to coordinate three AGVs. Every layer corresponds to the graph representing the precedences, as result of the vehicles coordination, at a particular time step. For example, at time t_4 the coordination is managed by making the AGV 1 waiting for the AGV 2 and the AGV 2 waiting for the AGV 3.

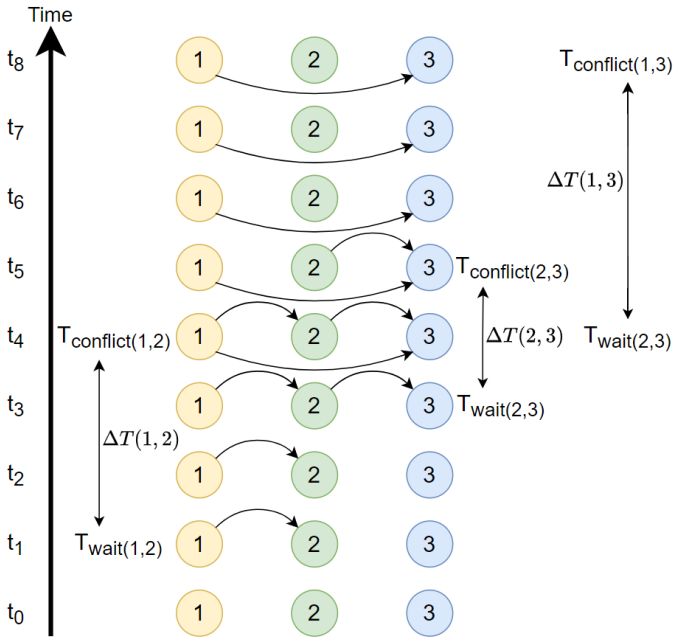

Fig. 7: A representation through the time expanded graphs of the precedences defined to coordinate three AGVs. Every layer corresponds to the graph representing the precedences, as result of the vehicles coordination, at a particular time step. The time expanded graph helped to preventing a cyclic set of precedences and hence a deadlock at time t_4. The coordination is managed by making the AGV 1 waiting for the AGV 3 and not vice versa to prevent a deadlock scenario between the three vehicles.

generate a loop cycle in the graph and, hence, a deadlock. The negotiation process gives the solution in which the AGV 3 is either a loser or a winner. In the first scenario, a loop cycle would occur at time $t = t_4$ generating a deadlock between the AGVs 1, 2 and 3. Exploiting the information stored in the time expanded graph, we can exclude the first scenario from the possible solutions and make the AGV 3 a winner adding an edge from node 1 to node 3.

Every time a new conflict is found in the predicted trajectories of the AGVs, the time expanded graph is updated accordingly and every layer is checked to be acyclic to prevent the occurrence of deadlocks. Hence, if a cycle is detected, the found solution from the negotiation is not admissible and another one is evaluated.

VII. Implementation

The proposed architecture was implemented, for validation and evaluation, on an industrial use case, exploiting the NetworkX library [45] in Python language. This library provides the tools to easily manage the graphs. The graphs are strongly related to the layout of the infrastructure (warehouse or factory) and are computed offline once the feasible routes in the plant are defined. Thus, the planner in the traffic manager does not see any change in the structure and connectivity of these graphs. However, the path computation is sensible to the changes in terms of attributes and weights linked with nodes and edges, which follow the changes in the traffic status. We remark that the route of each AGV is processed as a sequence of connected nodes. Every node occupied by an AGV means a location in the plant occupied by the vehicle. All the directions the AGV can follow from its location (node) are represented by the edges outgoing from the respective node.

The proposed control architecture makes the multi-AGV system able to deal with unexpected events, such as dynamic obstacles or vehicles errors, that change the expected traffic scenario. A frequent path re-planning and traffic status updating is the key to allow the vehicles to avoid unforeseen congested areas due to dynamic objects. Thus, the coordination algorithm is frequently executed to keep reactive the coordination of vehicles over the planned routes.

A software library has been developed to model the AGV motion over the time steps. Since the average of the AGV velocity is constant and known, the time required to travel every edge in the roadmap is known. Hence, the planned route, which consists in a list of adjacent nodes over the plant graph, can be projected over time. The coordination algorithm manages the vehicle possible conflicts stopping or slowing it down in free collision nodes, so that the time the AGV takes to reach the conflicting node is modified. Moreover, every time an unexpected event occurs, the schedule of the AGVs

changes and an execution of the control algorithm is necessary to update the model of the AGVs motion over time. A frequent execution of the control algorithm is computationally demanding, but makes the systems able to compensate the delays introduced by vehicles alarms, obstacles or communication errors.

A. Computational Complexity

Coupled approaches typically have a time complexity that grows with the size of the configuration space, which grows exponentially with the number of robots and plant dimension [46]. The computational effort of the proposed software is mainly defined by two components: the path planning and the coordination of vehicles. The complexity of the algorithm to coordinate the AGVs preventing conflicts and deadlocks is analyzed below. The path computation is lightened by the hierarchical architecture, specially in large environments and complex and high redundancy roadmaps: the Dijkstra's algorithm is modified to compute a path π_G over the plant graph $\mathcal{G}$ constrained to be contained into the subgraph of $\mathcal{G}$ defined by π_W, see Section V. The path π_W is computed exploiting the Dijkstra's algorithm over the graph of subsectors $\mathcal{W}$.

The computational efficiency of the proposed multi-layer path planning architecture is compared with the conventional approach, in which the Dijkstra search algorithm is applied directly on the roadmap graph(see Figure 8). In the conducted test, executed on a average laptop with Intel core i7-10510U and NVIDIA Geforce MX230, a range of graphs with varying dimensions and complexities were considered. The time needed to complete the path planning computation for 8 AGVs between two of the most distant nodes in the graph was analyzed using both the standard approach and the proposed multi-layer architecture. The results, depicted in Figure 8, demonstrate that as the dimension and complexity of the graph increase, the difference in computation time between the standard and proposed approach becomes increasingly pronounced. It is noteworthy that a low computational time for path planning is crucial for the effectiveness of the replanning feature in the proposed control strategy.

The computational effort incurred by the coordination algorithm, which includes the deadlock prevention policy, must also be taken into consideration. Analyzing Algorithm 3, similarly to [47], it consists in two main inherited loops over the number of AGVs N and over the time steps t in the time window H. The processes inside these loops are mainly independent from H and N and require a constant amount of resources to be executed. The deadlock prevention algorithm requires a relevant computational effort. In order to determine the computational effort we consider the worst case scenario for the iterative generation of the time expanded graph: this is constituted by $H+1$ layers, each one containing N nodes. In the worst case, we suppose all the AGVs are in collision for the whole time window H. Thus, once an AGV winner is chosen, every layer of the time expanded graph has to be updated with all the edges representing the precedences between the AGVs. The computational complexity of the deadlock prevention algorithm becomes $O(NH)$. Since this process is executed inside Algorithm 3, the computational complexity of the whole coordination strategy becomes $O(N^2H^2)$

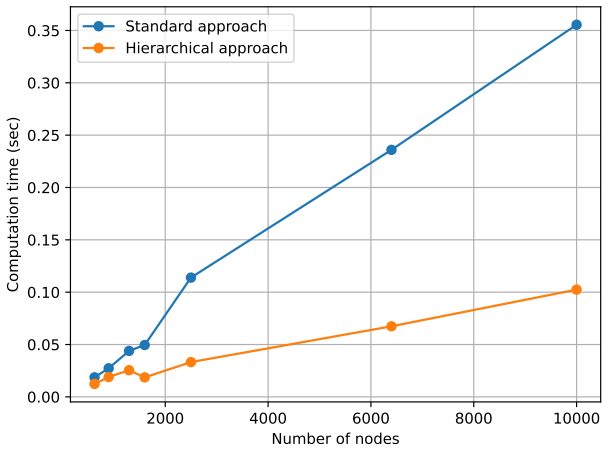

Fig. 8: The figure illustrates the computational benefits of a decoupled approach as compared to a standard coupled approach in terms of the path planning computation. The graph displays the computation time in seconds required to calculate the path for 8 AGVs in a roadmap with varying size between two of the most distant nodes in the graph. The discrepancy in computation time becomes more pronounced as the dimension and complexity of the graph increases.

The deadlock prevention algorithm computational cost has been tested in some simulations since its high expected computational demanding. Starting from a set of precomputed paths from a set of critical mission previously chosen for the test, we evaluated the number of iterations required to end properly the coordination. The tests were executed varying the number of AGVs from 2 to 8. The results are shown in Fig. 9 and indicates that the average and the maximum number or iterations are well below the maximum theoretical values estimated previously. In fact, as a result of the efficient planning and traffic modeling described in Section V, only a part of the AGVs operating in the system needs to be coordinated in order to prevent deadlocks, and therefore the worst-case scenario previously theorized is efficiently prevented.

We notify that the time required to complete the traffic manager computation is, typically, in the range of few seconds. In particular, the computation time has to be inferior to the time period T, where T determines how frequently the control algorithm is executed and, hence, how frequently the path planning algorithm and the coordination algorithm are executed (see Algorithm 2).

Finally, we conducted an evaluation of the efficiency in the traffic management and coordination of a variable fleet of AGVs in the considered industrial use case. We studied a comparison between the time the AGV is ideally required to reach its goal (i.e. without stops due to traffic, obstacles, ecc.) and the actual time, tested in simulations, with delays introduced by the vehicle coordination. In particular, we compute the management efficiency as the ratio between the time the vehicle is effectively moving and the overall

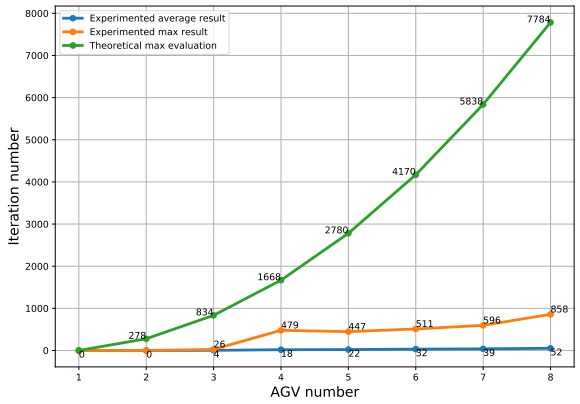

Fig. 9: The figure shows an evaluation of the deadlock prevention policy algorithm computational cost. We compared the theoretical exponential computational cost with the one tested with several simulations. The orange and blue line show respectively the highest and the average number of iterations tested from simulations. The green line shows the theoretical expected number of iteration in the worst computational scenario.

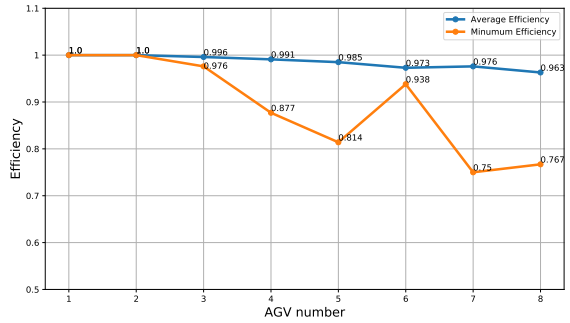

Fig. 10: The figure shows an evaluation of the efficiency in the traffic management and vehicles coordination. The results were obtained by a series of simulations conducted on the industrial use case scenario. The efficiency parameter aims to determine how much time is wasted due to the coordination. The blue line shows the average efficiency seen in several tests. The orange line shows the worsts values obtained from some tests. It is worth noting how the efficiency only slightly decreases, increasing the number of AGVs.

time required to complete the task, i.e. the combination of the moving time and the waiting time of the vehicle: $\eta = T_{moving}/(T_{moving} + T_{waiting})$. Figure 10 shows the results about the efficiency η from the tests conducted with up to 8 AGVs. The efficiency decreases with a large fleet of AGVs, since the traffic management and the vehicles coordination increase in complexity and congestions are more likely to happen.

Fig. 11: A photo taken during the experiments conducted in the automated factory. The 4 AGVs are coordinating in a load and unload cargo area.

B. Experimental validation

The software developed on the proposed control architecture has been initially tested on a simulation environment provided by Proxaut s.r.l., which emulates the nominal activity of a working day in the automated factory. Moreover, the simulator allows to test unexpected events that normally occur during the factory operation, such as alarms, vehicle errors, communication errors and delays introduced by operators. Finally, the software has been implemented and validated on a real industrial environment in collaboration with the company, see Fig. 11. The critical parameters that require careful selection to optimize the performance of the software for a specific application are the time horizon H and update frequency T of the traffic manager. These parameters are chosen based on the dimensions and layout of the plant under consideration. For the considered use case, a time horizon of 100 seconds was selected, as the longest dead-end in the plant takes approximately 80 seconds to be traversed by the AGV. The computational complexity of the traffic manager algorithm is within the range of 7 seconds of execution, and thus, a time frequency of 10 seconds was selected, which is confirmed through multiple simulations.

In the nominal operation, the software manages 8 AGVs over the provided plant. The video attached in the supplementary material shows some of the tests conducted in the automated factory. The performance of the control architecture proposed in this paper has been compared to the commercial software developed by the company. The company's software implements a traffic coordination of the AGVs based on manually synthesized traffic rules, which are compared among all the AGVs at every iteration. Given the path for each AGV, that is a sequence of nodes, each iteration consists in assigning the following node to each AGV. Moreover, this software does not consider the traffic status, and does not allow replanning the path: hence, the shortest path is assigned to each AGV. At each iteration, future conflicts are detected as intersections among the planned paths. Such conflicts are solved imposing stops to one or more AGVs. The AGVs to be stopped are defined by means of a greedy algorithm, computing all the possible combinations. Since such computation is performed at each iteration, the computational cost is very large, which makes it difficult to manage the traffic in large factories, as the considered use case.

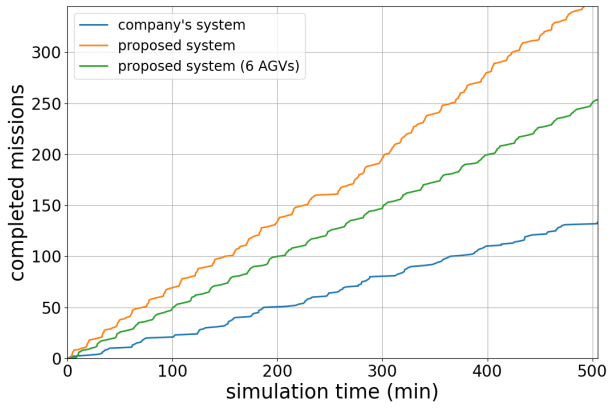

Fig. 12: The figure shows the number of missions completed over time by the fleet of AGVs. The blue line indicates the company's software performance at full potential, while the orange one indicates the proposed software performance at full potential. The proposed software managed to coordinate 8 vehicles simultaneously. The company's software managed to coordinate 6 vehicles simultaneously. The green line shows the performance of the proposed software when constrained to move only 6 AGVs simultaneously.

The developed software, based on the architecture proposed in this paper, has been installed at the automated factory managing the traffic of the fleet of 8 AGVs.

In order to compare the two systems, a set of critical missions has been chosen and repeated over time. A mission consists in a load and unload task the AGV has to accomplish. Every mission chosen for the experiment has very distant load and unload positions, in order to generate long paths. However, all the load positions are close to each other, in order to generate high traffic scenarios (the same holds for the unload positions). Figure 12 shows the number of mission completed over time. A mission is considered completed when the AGV has accomplished both loading and unloading tasks. The performance difference, which was evaluated in terms of total number of mission completed per hour, is evident: with a set of critical missions, the proposed system was able to complete more than 340 missions in 8 hours of simulation, while the company's system was able to complete approximately 130 missions in the same amount of time. In fact, the company's software, under a critical set of missions, is able to coordinate no more than 6 AGVs, leaving 2 vehicles parked. In order to perform a fair comparison, the proposed system was tested in the same conditions, leaving 2 AGVs parked, and approximately 250 missions were accomplished in the same amount of time.

The attached video shows some of the experiments conducted to test and validate the software based on the proposed control strategy. Firstly, several simulations were executed to test the reliability and the robustness of the control software. Subsequently, the developed architecture has been implemented in the industrial use case environment, to manage the traffic and coordinate the movements of a fleet of AGVs. Various traffic scenarios and congestions were faced during the experiments. Moreover, the control software was tested to deal with unexpected events that cause the vehicle to stop affecting the traffic and the coordination. These events are typically static or dynamic obstacles, vehicles alarms or errors.

VIII. Conclusions and Future Works

In this paper we present a novel methodology for the coordination and the traffic management of a fleet of AGVs deployed in a real industrial environment. The proposed multi-layer control architecture has the aim to improve the efficiency, the robustness and the flexibility of the global system. The control strategy consists in two main parts: a hierarchical path planning approach and a coordination strategy able to deal with collisions, deadlocks and vehicles errors.

The methodology proposed aims at the realization of a control software implementable on the majority of the industrial layouts and able to deal with the non idealities of a real implementation. Unlike what is usually found in literature, no assumptions are made on the roadmap and the presence of dynamical obstacles or execution errors has been considered. A novel hierarchical approach based on a sector and subsector division of the plant has been introduced to better model the traffic evolution and to design a traffic sensitive path planner. We propose a coordination strategy which is not based on predefined ad hoc rules able to manage the vehicles over time avoiding collisions and congestions. Moreover, the coordination methodology exploits a novel approach to recognize and deal with deadlocks based on time expanded graphs. Finally, the whole control architecture is robust to unpredictable events (typical of a real industrial implementation), such as dynamical obstacles, vehicles delays, communications errors.

The validity of the developed software is supported by different simulations and, especially, by tests executed in a real automated factory. The results show the hierarchical planner and traffic manager performs better compared to the company's control software based on common planning algorithms and predefined traffic rules. Moreover, the proposed strategy was able to cope with high traffic situations and unpredictable events, such as dynamic obstacles or alarms, during the experiments conducted in the real environment.

No assumption was made on homogeneity of the AGVs: while evaluation has been performed using homogeneous AGVs only, the proposed coordination strategy can be easily adapted to deal with heterogeneous AGVs. This can be achieved, for instance, opportunely tuning the weights of the graph.

The sector definition proposed in this paper requires a human operator assistance to differentiate the plant areas, such as intersections, corridors, load/unload positions, etc. Current work aims at developing a reliable method to automatically define the different plant areas and automatically generate the sectors clustering the roadmap nodes.

References

[1] H. Andreasson, A. Bouguerra, M. Cirillo, D. N. Dimitrov, D. Driankov, L. Karlsson, A. J. Lilienthal, F. Pecora, J. P. Saarinen, A. Sherikov *et al.*, "Autonomous transport vehicles: Where we are and what is missing," *IEEE Robotics & Automation Magazine*, vol. 22, no. 1, pp. 64–75, 2015.

[2] F. Oleari, M. Magnani, D. Ronzoni, and L. Sabattini, "Industrial agvs: Toward a pervasive diffusion in modern factory warehouses," in *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP)*, 2014, pp. 233–238.

[3] L. E. Parker, "Path planning and motion coordination in multiple mobile robot teams," *Encyclopedia of complexity and system science*, pp. 5783–5800, 2009.

[4] Y. Zhang and H. Mehrjerdi, "A survey on multiple unmanned vehicles formation control and coordination: Normal and fault situations," in *2013 International conference on unmanned aircraft systems (ICUAS)*. IEEE, 2013, pp. 1087–1096.

[5] S. M. LaValle and S. A. Hutchinson, "Optimal motion planning for multiple robots having independent goals," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 912–925, 1998.

[6] A. Cámara, D. Silva, P. Henriques Abreu, and E. Oliveira, "Comparing a centralized and decentralized multi-agent approaches to air traffic control," 10 2014.

[7] T. Siméon, S. Leroy, and J.-P. Lauumond, "Path coordination for multiple mobile robots: A resolution-complete algorithm," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 42–49, 2002.

[8] R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. S. Kumar *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Twelfth Annual Symposium on Combinatorial Search*, 2019.

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

[10] R. Olmi, "Traffic management of automated guided vehicles in flexible manufacturing systems," 2011.

[11] I. Draganjac, D. Miklić, Z. Kovačić, G. Vasiljević, and S. Bogdan, "Decentralized control of multi-agv systems in autonomous warehousing applications," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 4, pp. 1433–1447, 2016.

[12] L. Sabattini, V. Digani, C. Secchi, and C. Fantuzzi, "Hierarchical coordination strategy for multi-agv systems based on dynamic geodesic environment partitioning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, oct. 2016.

[13] C. Wei, K. V. Hindriks, and C. M. Jonker, "Multi-robot cooperative pathfinding: A decentralized approach," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 2014, pp. 21–31.

[14] D. Herrero-Perez and H. Martinez-Barbera, "Decentralized coordination of automated guided vehicles," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, 2008, pp. 1195–1198.

[15] P. Yang, R. A. Freeman, and K. M. Lynch, "Multi-agent coordination by decentralized estimation and control," *IEEE Transactions on Automatic Control*, vol. 53, no. 11, pp. 2480–2496, 2008.

[16] L. Pallottino, V. G. Scordio, A. Bicchi, and E. Frazzoli, "Decentralized cooperative policy for conflict resolution in multivehicle systems," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1170–1183, 2007.

[17] W. Zhang, M. Kamgarpour, D. Sun, and C. J. Tomlin, "A hierarchical flight planning framework for air traffic management," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 179–194, 2011.

[18] M. Jager and B. Nebel, "Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, vol. 3. IEEE, 2001, pp. 1213–1219.

[19] R. Luna and K. E. Bekris, "Efficient and complete centralized multi-robot path planning," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 3268–3275.

[20] Y. Guo and L. E. Parker, "A distributed and optimal motion planning approach for multiple mobile robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 3. IEEE, 2002, pp. 2612–2619.

[21] I. F. Vis, "Survey of research in the design and control of automated guided vehicle systems," *European Journal of Operational Research*, vol. 170, no. 3, pp. 677–709, 2006.

[22] V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi, "Towards decentralized coordination of multi robot systems in industrial environments: A hierarchical traffic control strategy," in *2013 IEEE 9th International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE, 2013, pp. 209–215.

[23] R. Olmi, C. Secchi, and C. Fantuzzi, "Coordination of industrial agvs," *International Journal of Vehicle Autonomous Systems*, vol. 9, no. 1-2, pp. 5–25, 2011.

[24] ——, "Coordination of multiple agvs in an industrial application," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1916–1921.

[25] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[26] J. P. Van Den Berg, D. Nieuwenhuisen, L. Jaillet, and M. H. Overmars, "Creating robust roadmaps for motion planning in changing environments," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 1053–1059.

[27] R. Geraerts and M. H. Overmars, "A comparative study of probabilistic roadmap planners," in *Algorithmic Foundations of Robotics V*. Springer, 2004, pp. 43–57.

[28] V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi, "Ensemble coordination approach in multi-agv systems applied to industrial warehouses," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 922–934, 2015.

[29] A. MANNUCCI, "Intra-logistics with integrated automatic deployment: from one to multi-mobile robot systems," 2020.

[30] F. Pratissoli, N. Battilani, C. Fantuzzi, and L. Sabattini, "Hierarchical and flexible traffic management of multi-agv systems applied to industrial environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 10 009–10 015.

[31] A. Mannucci, L. Pallottino, and F. Pecora, "Provably safe multi-robot coordination with unreliable communication," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3232–3239, 2019.

[32] Y. Wang, Y. Yuan, Y. Ma, and G. Wang, "Time-dependent graphs: Definitions, applications, and algorithms," *Data Science and Engineering*, vol. 4, no. 4, pp. 352–366, 2019.

[33] E. G. Coffman, M. Elphick, and A. Shoshani, "System deadlocks," *ACM Computing Surveys (CSUR)*, vol. 3, no. 2, pp. 67–78, 1971.

[34] J. W. Havender, "Avoiding deadlock in multitasking systems," *IBM Systems Journal*, vol. 7, no. 2, pp. 74–84, 1968.

[35] C. Godsil and G. Royle, "Algebraic graph theory," *Graduate text in mathematics, Springer, New York*, 2001.

[36] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[37] S. Bandi and D. Thalmann, "Path finding for human motion in virtual environments," *Computational Geometry*, vol. 15, no. 1-3, pp. 103–127, 2000.

[38] F. Schulz, D. Wagner, and K. Weihe, "Dijkstra's algorithm on-line: An empirical case study from public railroad transport," *Journal of Experimental Algorithmics (JEA)*, vol. 5, pp. 12–es, 2000.

[39] R. V. Bostelman, T. H. Hong, and R. Madhavan, "Towards agv safety and navigation advancement obstacle detection using a tof range camera," in *ICAR'05. Proceedings., 12th International Conference on Advanced Robotics, 2005.* IEEE, 2005, pp. 460–467.

[40] M. Boehning, "Improving safety and efficiency of agvs at warehouse black spots," in *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE, 2014, pp. 245–249.

[41] D. Chen, Z. Shi, P. Yuan, T. Wang, Y. Liu, M. Lin, and Z. Li, "Trajectory tracking control method and experiment of agv," in *2016 IEEE 14th International Workshop on Advanced Motion Control (AMC)*. IEEE, 2016, pp. 24–29.

[42] P. S. Pratama, A. V. Gulakari, Y. D. Setiawan, D. H. Kim, H. K. Kim, and S. B. Kim, "Trajectory tracking and fault detection algorithm for automatic guided vehicle based on multiple positioning modules," *International Journal of Control, Automation and Systems*, vol. 14, no. 2, pp. 400–410, 2016.

[43] L. Sabattini, V. Digani, M. Lucchi, C. Secchi, and C. Fantuzzi, "Mission assignment for multi-vehicle systems in industrial environments," in *Proceedings of the IFAC Symposium on Robot Control (SYROCO)*, Salvador, Brazil, aug. 2015.

[44] T. Shimomura and K. Ikeda, "Two types of deadlock detection: cyclic and acyclic," in *Intelligent Systems for Science and Information*. Springer, 2014, pp. 233–259.

[45] A. Hagberg, D. Schult, and P. Swart, "Networkx: Python software for the analysis of networks," *Mathematical Modeling and Analysis, Los Alamos National Laboratory*, 2005.

[46] M. Peasgood, C. M. Clark, and J. McPhee, "A complete and scalable strategy for coordinating multiple robots within roadmaps," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 283–292, 2008.

[47] S. MohaimenianPour, M. Behbooei, and S. S. Ghidary, "Adaptive multi-agent path planning with dynamic heuristic," in *Intelligent Autonomous Systems 13*. Springer, 2016, pp. 591–603.

Federico Pratissoli received the B.S. and M.S. degree in mechatronic engineering and the Ph.D. degree in Industrial Innovation Engineering from the University of Modena and Reggio Emilia, Italy, in 2016, 2018 and 2023 respectively. In 2018 was Visiting Student at Sheffield University, Sheffield, UK. He has been a Visiting Researcher with the University of Cambridge, Cambridge, UK. His main research interests include multi-robot systems, UAV systems, coordination and path planning, multi-AGV industrial systems, distributed control and multi-agent learning.

Riccardo Brugioni received the B.Sc. and M.Sc. degrees in mechatronic engineering from the University of Modena and Reggio Emilia, Italy, in 2018 and 2021, respectively. He has been working, as Chassis Controls Engineer, in Silk Sports Car Company since 2021. His main research interests include multi-AGV industrial systems, integrated active chassis controls and sensorless controls for IPM motors.

Nicola Battilani received B.Sc., M.Sc. in mechatronic engineer and the Ph.D. degree in Industrial Innovation Engineering from the University of Modena and Reggio Emilia, Italy, in 2011, 2014 and 2017 respectively. He was a research scientist with Lagadic team at Inria Rennes Bretagne Atlantique, Rennes, France, in 2016. His research topics include mobile autonomous robots, machine vision, human robot collaboration, human machine interface and industrial systems.

Lorenzo Sabattini received the B.Sc. and M.Sc. degrees in mechatronic engineering from the University of Modena and Reggio Emilia, Italy, in 2005 and 2007, respectively, and the Ph.D. degree in control systems and operational research from the University of Bologna, Italy, in 2012. In 2010, he has been a Visiting Researcher with the University of Maryland at College Park, College Park, MD, USA. He has been an Associate Professor with the Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, since 2018. His main research interests include multirobot systems, decentralized estimation and control, and mobile robotics. He has been a Founding Co-Chair of the IEEE RAS Technical Committee on Multi-Robot Systems, and has served as the corresponding co-chair from 2014 to 2021. He has been serving as Associate Editor for IEEE Robotics and Automation Letters from 2015 to 2018, and for IEEE Robotics and Automation Magazine from 2017 to 2019. He is currently serving as Editor for the IEEE ICRA and IEEE/RSJ IROS conferences, and as Associate Editor for the International Journal of Robotics Research (IJRR)