Toward Optimal Multi-Vehicle Path Planning With Hybrid Roadmaps

Marianna Turrà, *Graduate Student Member, IEEE*, Silvia Proia, *Member, IEEE*, Alessandro Bonetti, *Graduate Student Member, IEEE*, and Lorenzo Sabattini, *Senior Member, IEEE*

***Abstract*—Effective path planning for multi-vehicle autonomous systems is essential to guarantee safe and efficient operation in industrial environments composed of both structured and unstructured settings. In the proposed framework, the overall environment is explicitly modeled as the integration of static and dynamic areas, each exhibiting distinct spatial and temporal properties. Static areas correspond to structured zones, such as corridors or production lines, and are represented by roadmaps with fixed topology. Dynamic areas, instead, represent unstructured portions of the environment and are modeled as gridmaps to enable adaptive navigation under changing occupancy conditions. Building upon this representation, a hybrid roadmap unifying static and dynamic areas is proposed, with a hierarchical control architecture for coordinated multi-vehicle navigation. The high-layer planner determines the optimal sequence of areas to traverse by solving a multi-objective optimization (MOO) problem that balances efficiency, safety, and traffic-related factors. On the other hand, the low-layer planner generates feasible paths within each area according to its structural constraints. The MOO formulation is embedded in a bi-level optimization framework, where the outer-level employs a quasi-adaptive self-tuning process for objective weights, addressing the limitations of subjective or suboptimal weight selection. Experimental validation across two realistic industrial layouts demonstrates improved mission efficiency, reduced travel-time variability, and scalability with fleet size, confirming the effectiveness of the proposed method for Industry 4.0 applications.**

***Index Terms*—Multi-vehicles autonomous system, roadmap, path planning, multi-objective optimization.**

I. Introduction

IN RECENT years, Industry 4.0 paradigm promotes the digitalization and interconnection of manufacturing systems through the deployment of cyber-physical architectures, internet of things technologies, and data-centric decision-making frameworks [1], with emerging opportunities from deep learning to human-in-the-loop approaches and digital twins [2]. Among the most critical challenges, path planning (PP) represents a core functional requirement for ensuring safe, efficient, and adaptive operation of autonomous vehicle fleets in complex environments [3]. In response, significant research efforts have been directed toward the design and analysis of algorithms capable of addressing PP problems under a wide range of operational conditions and across diverse application domains [4]. On the basis of environmental layout and dynamic factors, two broad application domains can be identified: structured and unstructured settings, each characterized by distinct constraints, advantages, and challenges that fundamentally influence the formulation of effective planning strategies. Structured environments are defined by regularity and predictability in both spatial configuration and dynamic behavior. Variability over time is minimal, with obstacle positions and operational conditions remaining largely constant. Common examples include manufacturing facilities, warehouses, and other controlled industrial settings, where spatial layouts and process dynamics conform to predefined standards. A high degree of environmental regularity often allows for the adoption of static PP approaches, in which dynamic elements are either negligible or sufficiently constrained to be disregarded. Accordingly, PP tends to focus on minimizing travel time [5], avoiding obstacles [6], and ensuring seamless integration with other system components [3].

Conversely, the complexity of PP increases significantly in unstructured or partially structured environments, characterized by high variability and limited predictability. Frequent changes in spatial layout, presence of dynamic agents, and occurrence of unexpected obstacles introduce uncertainty that PP must address [7]. In this context, recent research has focused on model-free and artificial intelligence-based methods to enhance robustness in uncertain and dynamically changing environments [8], [9], [10]. Outdoor zones, temporary construction sites, and mobile workspaces represent common examples, often influenced by human activity, machinery movement, or environmental factors. Real-time perception, continuous environmental interpretation, and adaptive planning strategies become essential to maintain safety and achieve navigation goals in such contexts [11]. In parallel, optimization-based approaches have been extended to multi-phase trajectory planning to improve convergence and control robustness [12], although they remain limited to single-vehicle scenarios and lack scalability to cooperative or large-scale navigation.

Operational scenarios in real-world applications increasingly exhibit mixed characteristics, combining structured and unstructured elements. Planning approaches must therefore balance prior knowledge with responsiveness to environmental

Received 20 June 2025; revised 5 November 2025 and 3 February 2026; accepted 16 February 2026. Date of publication 18 March 2026; date of current version 1 July 2026. The Associate Editor for this article was S. F. Ghoreishi. *(Corresponding author: Marianna Turrà.)*

The authors are with the Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, 42121 Reggio Emilia, Italy (e-mail: marianna.turra@unimore.it; silvia.proia@unimore.it; alessandro.bonetti@unimore.it; lorenzo.sabattini@unimore.it).

Digital Object Identifier 10.1109/TITS.2026.3666800

© 2026 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

changes. Hybrid planning strategies, integrating predefined paths with real-time adaptation, offer a promising framework for managing uncertainty in partially known or evolving environments. Building upon this paradigm, the work introduces a hierarchical multi-vehicle PP strategy, combining the strengths of static and dynamic planning within a multi-objective optimization (MOO), while a quasi-adaptive self-tuning process autonomously adjusts objective weights.

The remainder of this paper is structured as follows. Section II summarizes the related works and main contributions of this work. Section III describes the hybrid environment, while Section IV outlines the hierarchical planner architecture. Section V introduces the quasi-adaptive self-tuning process for automated weight tuning. The experimental setup and results are discussed in Section VI. Finally, concluding remarks and potential future directions are discussed in Section VII.

II. Related Works and Contributions

Path planning refers to the process of determining an optimal continuous path that guides a system from an initial state to a desired goal state [13]. The planned path is typically required to ensure safe navigation while optimizing a given cost function, which may account for factors such as distance, time, energy consumption, and system dynamics [14]. In detail, effective PP requires an appropriate environmental model. A common approach involves discretizing the space into a graph, enabling the use of efficient graph search algorithms [15]. Based on the regularity and predictability of the environment, in this work, a formal distinction is made between structured and unstructured settings.

In particular, structured environments exhibit fixed spatial configurations and deterministic system dynamics. Motion constraints are typically encoded through predefined roadmaps, defined as structured sets of virtual or physical paths representing admissible transitions between navigable locations [16]. The use of predefined paths enables efficient planning and low computational cost [3], particularly in space-constrained industrial domains, including automated warehouses and production lines, where task repeatability and tight integration with scheduling systems are critical [17], [18]. Nonetheless, despite their computational efficiency, structured settings are restricted by limited scalability and flexibility. Growing industrial complexity and scale demand higher flexibility, including fault tolerance, modular integration of additional units, and support for a broader range of tasks – requirements that static planning frameworks struggle to meet due to rigid modeling assumptions [19].

Unstructured environments have gained increasing relevance as a response to the limitations of structured settings. The absence of constraints imposed by fixed navigation graphs enables greater adaptability in dynamic, partially known, and continuously evolving industrial scenarios [20]. Continuous generation of feasible paths is supported, allowing autonomous vehicles to react to real-time changes, avoid unexpected obstacles, and explore novel solutions without the need for full global map updates [21]. This paradigm shift facilitates scalable and fault-tolerant solutions capable of meeting the demands of smart factories, logistics platforms, and other complex operational ecosystems [20]. However, the increased flexibility of unstructured settings introduces notable complexity. Unstructured environments typically exhibit high variability in layout, obstacle distribution, and agent behavior. In such contexts, predefined roadmaps often fail due to limited spatial coverage and lack of adaptability. Even flexible variants, such as probabilistic roadmaps [22], may suffer from poor completeness and limited reactivity in the presence of dynamic or complex geometries. To overcome the above limitations, a shift from fixed-path navigation to free-space planning becomes necessary. A prominent approach consists in discretizing the entire workspace using gridmaps [23], which provide a systematic and dense representation of free and occupied regions. Unlike predefined roadmaps, grid-based models generate homogeneous and updatable graph structures, enabling frequent replanning and enhanced responsiveness in highly variable environments.

Building upon the strengths and addressing the limitations of both structured and unstructured settings, the system proposed in this work introduces a unified framework that brings together static and dynamic settings within a hybrid representation. The approach combines the advantages of predefined roadmaps with the flexibility of gridmaps, leading to the concept of a hybrid roadmap, that, to the best of our knowledge, has not yet been formally addressed in the literature. The hybrid representation is employed to plan across distinct operational zones, ensuring deterministic behavior in static areas (e.g., loading/unloading points or machine interfaces) and adaptive, coordinate-based strategies in open or less structured spaces (e.g., storage or transit areas). This dual approach combines the flexibility and real-time responsiveness required in dynamic settings with the robustness and predictability of structured environments. Transitions between the two layouts are managed by a central unit (CU), ensuring global path consistency while enabling local adaptability to environmental changes. The resulting system offers a scalable and resilient solution and represents a significant step forward in the development of fully integrated autonomous vehicle ecosystems within the Industry 4.0 paradigm.

Unlocking the full potential of hybrid planning architectures requires not only structural flexibility, including seamless connectivity between different graph representations, but also the integration of autonomous and intelligent decision-making mechanisms. Autonomy, defined as the capacity to independently make and execute decisions without external control [24], highlights the necessity for intelligent agents capable of operating effectively in dynamic and unpredictable environments. Reliance on single-objective optimization proves insufficient to capture the complexity and variability inherent to real-world operations.

In the specific context of PP, approaches based on single-objective formulations have been shown to exhibit limited adaptability and to produce solutions that are often inadequate when evaluated against the multifaceted requirements of practical applications [25]. Despite the increasing need for coordination and the growing operational complexity in industrial multi-vehicle systems, the adoption of MOO strategies within the industrial domain remains relatively limited.

A substantial portion of existing research tends to focus on optimizing single metrics, such as travel distance, completion time, or throughput [26], [27], while equally important aspects, such as safety, energy consumption, and path smoothness, are frequently overlooked or treated as secondary objectives, limiting the robustness and applicability of the solutions.

A holistic approach to PP should instead incorporate multiple, potentially conflicting objectives that more accurately reflect the complexity of real operational environments. As argued in [28], neglecting factors, such as safety or path smoothness, can substantially degrade system performance, particularly in dynamic or partially known environments. Similarly, [29] demonstrates that solutions based on single-objective optimization may yield paths that are either unsafe or impractical for deployment, reinforcing the need for MOO methods that produce context-aware, adaptive, and operationally viable solutions.

To address the challenges in dynamic environments, this work proposes a hierarchical architecture consisting of two main layers, each designed for real-time operation and continuous replanning in multi-vehicle scenarios. At the high-layer, a MOO problem is solved to determine the optimal sequence of area crossings for each vehicle, evaluating multiple, often conflicting criteria that reflect dynamic environmental conditions. Trade-offs among objectives are balanced to generate area sequences that are both computationally efficient and suitable for deployment, ensuring robustness and responsiveness. The resulting sequences are then transferred to the low-layer, where they are transformed into feasible physical paths suitable for real-time multi-vehicle navigation.

Although the advantages of MOO approaches have been demonstrated across various autonomous mobility domains beyond the industrial context [30], the integration of autonomous decision-making mechanisms remains considerably underdeveloped. Within the limited body of research that addresses MOO in autonomous systems, multi-objective decision making (MODM) –which focuses on generating a set of optimal trade-off solutions– and multi-attribute decision making (MADM)– which involves selecting the most suitable solution based on predefined criteria– are frequently treated as distinct stages [31]. This separation limits the development of fully autonomous and context-aware planning pipelines. Even in domains where MOO techniques are more established, such as uncrewed aerial vehicle mission planning, existing approaches frequently rely on the generation of Pareto-optimal solution sets capturing trade-offs among competing objectives, including mission duration, connectivity, and energy consumption [32], [33]. However, the selection of a final solution is typically delegated to human operators or facilitated through basic interactive tools, limiting system autonomy and reducing responsiveness under dynamic or uncertain conditions. Similarly, [34] proposes a two-phase fuzzy multi-objective optimization framework for multi-uncrewed ground vehicles trajectory planning, in which the priority sequence among objectives is manually predefined through a β-fuzzy goal programming scheme.

In contrast, the system proposed in this work adopts a quasi-adaptive self-tuning process that integrates MODM and MADM to address the limitations of subjective and potentially suboptimal weight selection. The system is integrated within a bi-level optimization framework, where the outer-level optimizes the set of designated criteria with the quasi-adaptive self-tuning process, influenced by the responses of the inner-level entity, which solves the MOO problem formulated in the hierarchical architecture. Since the trade-offs among conflicting objectives are highly context-dependent, the ability to adjust weights dynamically enhances the relevance and quality of the resulting solutions [35]. The term *quasi-adaptive* reflects the system's ability to adjust weight configurations as needed, without requiring continuous or fully online optimization. Weight adjustment can be triggered in response to significant changes in the operational environment or to insights derived from analysis of historical data. For instance, in highly congested environments, objectives related to collision avoidance and traffic flow management tend to take precedence [36], whereas in low-demand scenarios, minimizing travel distances becomes more relevant for optimizing resource efficiency [37].

Summarizing, the main contributions of this paper are:

- Introduction of an innovative hierarchical multi-vehicle PP strategy within a hybrid roadmap, enabling deterministic behavior in static environments and adaptive, real-time decision-making for dynamic conditions. The system supports continuous replanning and real-time operation in industrial settings.
- Formulation of a MOO problem integrated into a hierarchical architecture, where the high-layer determines the optimal sequence of area crossings for each vehicle, and the low-layer generates corresponding feasible physical paths. The evaluation of multiple and potentially conflicting criteria enables more robust and responsive PP, particularly within dynamic environments.
- Integration of a quasi-adaptive self-tuning process within a bi-level optimization framework, enabling dynamic adjustment of objective weights in response to environmental changes or historical data. The approach enhances performance without requiring manual intervention.

III. Hybrid Roadmap

In the considered industrial environment, a fleet of Γ autonomous vehicles, indexed by $\gamma \in \mathcal{F} := \{1, \ldots, \Gamma\} \subseteq \mathbb{N}$, operates within a *hybrid roadmap*, under the following modeling assumption (see Fig. 1).

Assumption 1 (Vehicle Modeling): Each vehicle is modeled as a point mass moving within a two-dimensional environment, neglecting kinematic and dynamic effects.

It partitions the environment into A areas $a_i \in \mathcal{A} := \{a_1, \ldots, a_A\}$, comprising a set of S static areas $a_i \in \mathcal{S} := \{a_1, \ldots, a_S\}$, and a set of D dynamic areas $a_i \in \mathcal{D} := \{a_{S+1}, \ldots, a_A\}$, with $A = S + D$. The entire system is formally modeled as a directed unweighted graph $\mathcal{G}^A = (\mathcal{V}^A, \mathcal{E}^A)$, with the vertex set $\mathcal{V}^A(\mathcal{G}^A)$ representing the vertices within each area and the edge set $\mathcal{E}^A(\mathcal{G}^A)$ defining their connections.

The *hybrid roadmap* is composed of three distinct types of vertices, each fulfilling a specific role in maintaining the graph's structure and connectivity. The *start/goal vertices* $\zeta_z \in \mathcal{V}^\zeta := \{\zeta_1, \ldots, \zeta_Z\}$ form the subset $\mathcal{V}^\zeta \subseteq \mathcal{V}^A$, representing

locations that may serve either as the origin or destination of a *task*. In the remainder of the manuscript, for the sake of notation clarity, start and goal vertices are explicitly denoted as v_γ^s or v_γ^g, indicating respectively the start and goal vertex associated with the γ-th vehicle. A given vertex $\zeta_z \in \mathcal{V}^\zeta$ may assume either role depending on the assigned *task*. Thoroughly, a *task* is defined as a pair $(v_\gamma^s, v_\gamma^g) \in \mathcal{V}^\zeta \times \mathcal{V}^\zeta$, assigned to the γ-th vehicle, with $\gamma \in \mathcal{F}$. The *intermediate vertices* $\iota_i \in \mathcal{V}^\iota := \{\iota_1, \ldots, \iota_I\} \subseteq \mathbb{N}$, denoted by the subset $\mathcal{V}^\iota \subseteq \mathcal{V}^A$, preserve the graph's connectivity, enabling uninterrupted transitions across the areas. Lastly, the *attachment/detachment vertices* $\mu_m \in \mathcal{V}^\mu := \{\mu_1, \ldots, \mu_M\}$, represented by the subset $\mathcal{V}^\mu \subseteq \mathcal{V}^A$, act as interface points between static and dynamic areas, allowing the γ-th vehicle to move between different area types and navigate within the heterogeneous environment. The presence of interface points is particularly important in dynamic areas, where the use of a roadmap supports the execution of precise, robust, and safe maneuvers required in more complex or less structured spaces.

On the one hand, *static areas* are modeled as fixed graphs, to facilitate efficient PP, given their predefined and consistent layout across tasks, which simplifies the management of vehicle movements over time.

Definition 1 (Static Area): A static area in the hybrid roadmap is modeled as a predefined roadmap, where dynamic changes are not expected and the admissible paths are explicitly defined. The i-th static area is modeled as a directed unweighted subgraph $\mathcal{K}_i^S = (\mathcal{V}_i^S, \mathcal{E}_i^S)$ with $i \in \{1, \ldots, S\}$, i.e., $\mathcal{K}_i^S \subseteq \mathcal{G}^A$, with the vertex set $\mathcal{V}_i^S$, defined as follows:

$$\mathcal{V}_i^S = \bigcup_{\omega \in \{\zeta, \iota, \mu\}} \mathcal{V}^\omega(\mathcal{K}_i^S). \tag{1}$$

Instead, the edge set $\mathcal{E}_i^S \subseteq \mathcal{V}_i^S \times \mathcal{V}_i^S$ defines the possible connections between vertices, enabling vehicle movement within the structured environment of the static area.

On the other hand, *dynamic areas* are modeled as dense gridmaps $\Omega^D := \bigcup_{i=S+1}^{A} \mathcal{K}_i^D$, offering flexible navigation in unstructured environments while avoiding obstacles.

Definition 2 (Dynamic Area): A dynamic area in the hybrid roadmap is defined as a portion of the environment modeled as a gridmap, that supports adaptive navigation under changing conditions, with traversable paths determined by the current occupancy. The i-th dynamic area is modeled as an undirected unweighted subgraph $\mathcal{K}_i^D = (\mathcal{V}_i^D, \mathcal{E}_i^D)$, with $i \in \{S+1, \ldots, A\}$, i.e., $\mathcal{K}_i^D \subseteq \mathcal{G}^A$. The vertex set $\mathcal{V}_i^D(\mathcal{K}_i^D)$ forms a discrete grid of integer coordinates, offering flexibility in path selection:

$$\mathcal{V}_i^D = \{(x, y) \in \Omega^D \mid (x, y) \notin \Omega^D \setminus \mathcal{K}_i^D, x, y \in \mathbb{Z}\}. \tag{2}$$

Instead, the edge set $\mathcal{E}_i^D(\mathcal{K}_i^D)$ is defined as:

$$\mathcal{E}_i^D = \{((x, y), (x', y')) \in \Omega^D \mid \max(|x - x'|, |y - y'|) = 1\}. \tag{3}$$

Specific dynamic areas containing a work station feature two distinguished types of vertices, i.e., a start/goal vertex $\zeta_z \in \mathcal{V}^\zeta(\mathcal{K}_i^D)$ and an attachment/detachment vertex $\mu_m \in \mathcal{V}^\mu(\mathcal{K}_i^D)$, with $i \in \{S+1, \ldots, A\}$, which ensure proper vehicle access to the work station. They are formally incorporated into the vertex and edge sets as follows:

$$\begin{aligned} \mathcal{V}_i^D &= \{\{(x, y) \in \Omega^D \mid (x, y) \notin \Omega^D \setminus \mathcal{K}_i^D, x, y \in \mathbb{Z}\} \cup \{\zeta_z, \mu_m\}\}, \\ \mathcal{E}_i^D &= \{((x, y), (x', y')) \mid \max(|x - x'|, |y - y'|) = 1\} \cup \{(\zeta_z, \mu_m)\}. \end{aligned} \tag{4}$$

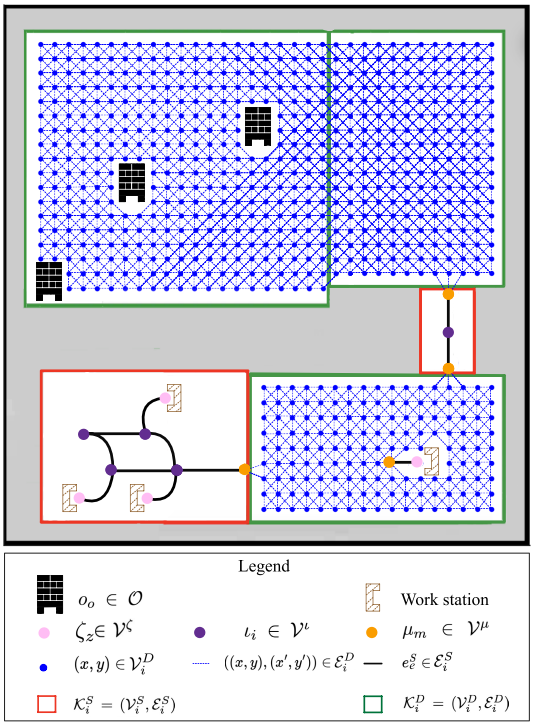


Fig. 1. Representation of the hybrid roadmap employed for vehicles navigation, consisting of static and dynamic areas. Static areas, delineated by red borders, are modeled as graphs with predefined vertices and edges, while dynamic areas, enclosed by green borders, are represented as dense gridmaps, enabling flexible and adaptive PP. The legend outlines key elements of the environment, including obstacles, work stations, start/goal vertices, intermediate vertices, and attachment/detachment vertices, as well as the vertices and edges that define the roadmap structure.

Note that the edge sets $\mathcal{E}_i^S$ with $i \in \{1, \ldots, S\}$ and $\mathcal{E}_i^D$ with $i \in \{S+1, \ldots, A\}$ must possess adequate smoothness and maintain C^1 continuity by using cubic Bézier curves [38].

To effectively coordinate the planning process and represent dynamic areas, the system operates in discrete time, divided into time steps indexed by $k \in \{0, 1, 2, \ldots\}$, each with fixed duration ΔT. The discrete time instants are denoted by $t(k) = k\Delta T$ and at each k the spatial configuration of the environment is updated to reflect possible changes in obstacle occupancy.

Assumption 2 (Nominal Operational Conditions): The present study assumes nominal operational conditions, which refer to ideal or reference operating scenarios where all system parameters and environmental factors are assumed to behave according to their expected values, without the influence of disturbances, noise, or variability.

In dynamic areas, *obstacles* are defined as spatial configurations that are currently occupied and cannot be traversed by the vehicles. The set of obstacles at each time step k is denoted by $\mathcal{O}(k)$. Each obstacle $o_o(k) \in \mathcal{O}(k) := \{o_1(k), \ldots, o_{O(k)}(k)\}$

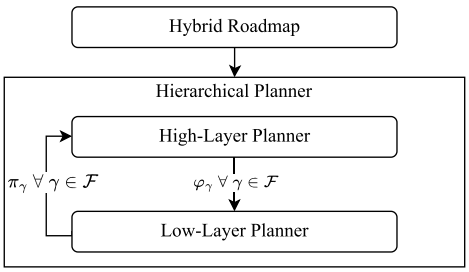


Fig. 2. Overview diagram of the proposed methodology, illustrating the hierarchical path-planning architecture consisting of two planner layers, i.e., the high-layer planner which outputs the area path φ_γ, and the low-layer planner which generates the physical path π_γ for each $\gamma \in \mathcal{F}$.

consists of a set of discrete coordinates, formulated as:

$$\mathcal{O}(k) = \{(x(k), y(k)) \in \Omega^D \mid g(x(k), y(k)) = 1\} \tag{5}$$

where $g(\cdot) : \mathbb{Z}^2 \to \{0, 1\}$ denotes an indicator function that determines whether a given point $(x(k), y(k))$ is occupied by an obstacle $o_o(k) \in \mathcal{O}(k)$ at the k-th time step. Specifically, $g(\cdot)$ is defined as:

$$g(x(k), y(k)) = \begin{cases} 1 \text{ if } (x(k), y(k)) \text{ is occupied by an obstacle,} \\ 0 \text{ otherwise.} \end{cases} \tag{6}$$

By combining structured and adaptive components, the system optimizes paths dynamically in response to real-time conditions rather than relying solely on a predefined graph. The *connectivity map* extends the *hybrid roadmap* by defining valid transitions between areas. It is represented as a directed and weighted graph $\mathcal{G}^Y = (\mathcal{V}^Y, \mathcal{E}^Y)$, where $\mathcal{V}^Y$, with $|\mathcal{V}^Y| = A$, is the set of vertices, with the i-th vertex corresponding to the area $a_i \in \mathcal{A}$. The edges $e_e \in \mathcal{E}^Y := \{e_1, \ldots, e_E\} \subseteq \mathcal{V}^Y \times \mathcal{V}^Y$ encode valid transitions between adjacent areas, ensuring connectivity in the roadmap. Instead, E represents the total number of edges in the graph, and each edge $e_e \in \mathcal{E}^Y$ is a transition between two vertices $i, j \in \mathcal{V}^Y$.

IV. Hierarchical Planner

The proposed method presents a centralized architecture to address PP, where the CU manages fleet operations by collecting information on the hybrid roadmap and the mission list and by monitoring detailed data for each vehicle, including position and speed.

The main objective is to optimize PP by leveraging the hybrid roadmap and incorporating areas not commonly employed as alternative paths. By proactively adapting to dynamic conditions, e.g., changes in obstacle layout, traffic density, user preferences, or environmental constraints, the proposed approach enhances productivity, maximizes the use of available space, and enables smoother navigation through complex environments.

Figure 2 illustrates the hierarchical PP architecture, consisting of two planner layers, i.e., the high-layer planner, which outputs the area path φ_γ, and the low-layer planner, which generates the physical path π_γ, for each $\gamma \in \mathcal{F}$. The two layers are designed to operate sequentially, enabling the continuous computation of the optimal paths π_γ for each $\gamma \in \mathcal{F}$, ensuring the successful execution of the tasks. The system is built upon the hybrid roadmap (as detailed in Section III), which offers a structured representation of both static and dynamic areas.

In the course of task execution, the fleet operates under a hierarchical strategy, where each layer is responsible for a specific aspect of the PP process.

To efficiently organize and manage operations across the fleet, a mission is defined as a set of individual tasks, where each task is assigned to a single vehicle, and each vehicle is responsible for only one task. All tasks that constitute a mission are collected in a mission list, which specifies their assignment and priority order.

Assumption 3 (Mission List): The sequenced list of tasks, referred to as the mission list, is given a priori. As a consequence, each vehicle is assigned a predefined task from the list, each characterized by a distinct and ordered priority level.

Note that task assignment falls beyond the scope of this paper and will not be addressed further.

In detail, at the upper level, the high-layer planner defines the overall path strategy by considering a designated set of criteria, and associated weights, which reflect their relative importance in the decision-making process (see Section IV-A). Specifically, for each $\gamma \in \mathcal{F}$, the planner determines the optimal sequence of edges ξ_γ, referred to as the *edge path*, which defines the connectivity between the start area a^s_γ – containing the start vertex v^s_γ–and the goal area a^g_γ, associated with the goal vertex v^g_γ.

Definition 3 (Edge Path): For each $\gamma \in \mathcal{F}$, the edge path $\xi_\gamma := \langle e_{\gamma,1}, \ldots, e_{\gamma,L_\gamma} \rangle$ is an ordered sequence of L_γ edges $e_l = (i, j) \in \mathcal{E}^Y$, for each $l \in \{1, \ldots, L_\gamma\}$, where $i, j \in \mathcal{V}^Y$ represent the i-th and j-th areas that form the l-th edge in the sequence. Moreover, for each $l \in \{1, \ldots, L_\gamma\}$, $e \in \{1, \ldots, E\}$ exists such that $e_{\gamma,l} = e_e \in \mathcal{E}^Y$.

The *edge path* ξ_γ, for the γ-th vehicle, is determined by optimizing the objective function $f_H(\cdot)$, that will be described in Section IV-A, with the aim of minimizing the traversal cost. Based on ξ_γ, the corresponding sequence of areas φ_γ, for the γ-th vehicle and referred to as the *area path*, is derived to specify the areas the vehicle must traverse to complete its task and serves as input for the low-layer planner.

Definition 4 (Area Path): For each $\gamma \in \mathcal{F}$, an a^s_γ-a^g_γ path, denoted as the area path, is defined as an ordered sequence of N_γ areas: $\varphi_\gamma = \langle a^s_\gamma = a_{\gamma,1}, \ldots, a_{\gamma,N_\gamma} = a^g_\gamma \rangle$ where, for each $n \in \mathcal{N}_\gamma := \{1, \ldots, N_\gamma\}$, $i \in \{1, \ldots, A\}$ exists such that $a_{\gamma,n} = a_i \in \mathcal{A}$ and $a_{\gamma,n} \neq a_{\gamma,n+1}$, $\forall n \in \{1, \ldots, N_\gamma - 1\}$.

Instead, the low-layer planner refines the *area path* φ_γ by computing the physical path π_γ, which optimizes the objective function $f_L(\cdot)$ while ensuring a seamless connection from v^s_γ to v^g_γ, for each $\gamma \in \mathcal{F}$ (see Section IV-B).

During the execution of tasks, governed by the mission list (see Assumption 3), both layers continuously update their respective outputs, i.e., φ_γ and π_γ, for each $\gamma \in \mathcal{F}$ vehicle, to dynamically adapt to environmental changes, based on the criteria evaluated by the high-layer. A detailed description of the interaction between the two layers, as well as the role of the hierarchical PP strategy in ensuring coherent and adaptive behavior across the fleet, is thoroughly described in Section IV-C.

A. High-Layer Planner

The high-layer planner is responsible for assigning, for each $\gamma \in \mathcal{F}$, an *area path* φ_γ, defining the sequence of areas the

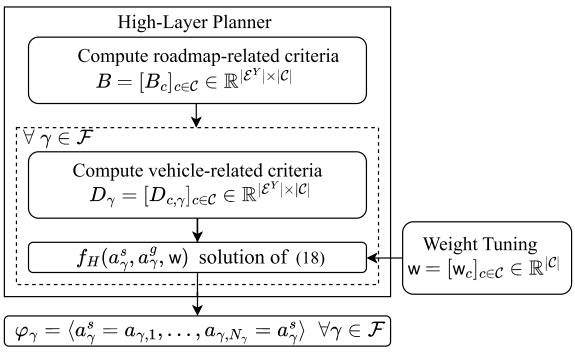


Fig. 3. Logical structure of the high-layer planner, which combines roadmap-related and vehicle-related criteria to assign optimal area paths φ_γ to each γ-th vehicle, using the weight vector w computed by the quasi-adaptive self-tuning process or manually assigned.

vehicle must traverse to complete its task. Formulated as an integer linear programming (ILP) problem, the criteria-based area PP (CBAPP) features a scalarized MOO that integrates the designated criteria, weighted according to the quasi-adaptive self-tuning process (see Section V) or manually set based on operational needs, with the flexibility for arbitrary adjustments to suit different decision-making strategies. The resulting solution ensures task completion and minimizes traversal costs while adhering to the feasibility constraints of the shortest path problem [39].

In the CBAPP problem, the areas are classified into two distinct subsets. On the one hand, normal areas, denoted as $\mathcal{N} \subseteq \mathcal{A}$, consist of standard areas for PP. On the other hand, storage areas, represented as $\mathcal{W} \subseteq \mathcal{D}$, consist of dynamic areas primarily designated for material storage. However, when not fully occupied, they may also be utilized for PP, similar to normal dynamic areas.

Remark 1: The set of normal areas is given by:

$$\mathcal{N} = (\mathcal{D} \setminus \mathcal{W}) \cup \mathcal{S}.$$

Remark 2: The total set of areas is defined as $\mathcal{A} = \mathcal{N} \cup \mathcal{W}$ or equivalently $\mathcal{A} = \mathcal{S} \cup \mathcal{D}$, with cardinality $A = |\mathcal{A}|$.

As illustrated in Fig. 3, the high-layer planner follows a structured computational flow to generate feasible and cost-efficient paths. The process begins by defining the set of C criteria, indexed by $c \in \mathcal{C} := \{1, \ldots, C\} \subseteq \mathbb{N}$. Each criterion is a real-valued vector that assigns a quantitative value to each edge $e_e \in \mathcal{E}^Y$.

The C criteria are partitioned into two distinct matrices, capturing either *roadmap-related* or *vehicle-related* criteria. Specifically, the *roadmap-related* criteria are collected into the matrix $B \in \mathbb{R}^{|\mathcal{E}^Y|\times|\mathcal{C}|}$, defined as:

$$B = [B_c]_{c\in\mathcal{C}}, \quad B_c \in \mathbb{R}^{|\mathcal{E}^Y|} \tag{7}$$

where each $B_c \in \mathbb{R}^{|\mathcal{E}^Y|}$ represents the values associated with the c-th roadmap-related criterion for each edge $e_e \in \mathcal{E}^Y$.

Similarly, for each $\gamma \in \mathcal{F}$, the *vehicle-related* criteria are stored into the matrix $D_\gamma \in \mathbb{R}^{|\mathcal{E}^Y|\times|\mathcal{C}|}$, defined as:

$$D_\gamma = \left[D_{c,\gamma}\right]_{c\in\mathcal{C}}, \quad D_{c,\gamma} \in \mathbb{R}^{|\mathcal{E}^Y|} \tag{8}$$

where each $D_{c,\gamma} \in \mathbb{R}^{|\mathcal{E}^Y|}$ represents the values associated with the c-th vehicle-related criterion for each edge $e_e \in \mathcal{E}^Y$ for the γ-th vehicle. By construction, the matrices B and D_γ, $\forall\gamma \in \mathcal{F}$, exhibit a structural exclusivity that ensures the separation of *roadmap-related* and *vehicle-related criteria*, guaranteeing that their respective values do not overlap, as formalized in the following property.

Property 1 (Mutual Exclusivity of Roadmap and Vehicle Matrices): Let $B \in \mathbb{R}^{|\mathcal{E}^Y|\times|\mathcal{C}|}$ and $D_\gamma \in \mathbb{R}^{|\mathcal{E}^Y|\times|\mathcal{C}|}$, with $\gamma \in \mathcal{F}$, be matrices that, by construction, satisfy the following condition:

$$B_{e,c}D_{e,c,\gamma} = 0, \quad \forall e \in \{1, \ldots, |\mathcal{E}^Y|\}, \quad \forall c \in \mathcal{C},$$

where $B_{e,c}D_{e,c,\gamma}$ represents the criterion value associated with the e-th edge in the respective tables B and D_γ, with $\gamma \in \mathcal{F}$. This condition implies that, for each edge $e_e \in \mathcal{E}^Y$, the nonzero entries of B and D_γ, for each $\gamma \in \mathcal{F}$, form a disjoint partition, i.e., if $B_{e,c} \neq 0$, then $D_{e,c,\gamma} = 0$, and, conversely, if $D_{e,c,\gamma} \neq 0$, then $B_{e,c} = 0$.

Once all criteria are defined and normalized to ensure consistency, eliminating scale disparities that could affect the optimization process, they are weighted using the vector

$$w = [w_c]_{c\in\mathcal{C}} \in \mathbb{R}^{|\mathcal{C}|}. \tag{9}$$

For its computation, we propose the quasi-adaptive self-tuning process, which is presented in Section V. The proposed mechanism is particularly useful in scenarios where criteria preferences are unavailable. Alternatively, the weights can be manually defined when prior knowledge of the industrial context or specific performance priorities are known in advance.

Let us introduce the constraints that define the CBAPP problem. It is worth noting that the CBAPP is solved only when the start area a^s_γ and the goal area a^g_γ of the γ-th vehicle are distinct. Solving the problem becomes redundant when the two coincide –i.e., when both the start vertex v^s_γ and the goal vertex v^g_γ belong to the same area $a_i \in \mathcal{A}$. For the sake of notational clarity, we denote the start area a^s_γ as the vertex $s \in \mathcal{V}^Y$ and the goal area a^g_γ as the vertex $g \in \mathcal{V}^Y$, for the γ-th vehicle. In addition, each edge $e_e \in \mathcal{E}^Y$ is represented as an ordered pair (i, j), where $i, j \in \mathcal{V}^Y$. Without loss of generality, the model is presented for the γ-th vehicle, and therefore, for clarity and simplicity, the explicit dependency on γ is omitted in the following formulation.

The first contribution to the CBAPP problem is represented by the binary decision variable $e_{(i,j)}$, where $e_{(i,j)} = 1$ denotes the selection of edge (i, j), and $e_{(i,j)} = 0$ indicates its exclusion, for each $\gamma \in \mathcal{F}$:

$$e_{(i,j)} \in \{0, 1\}, \quad \forall\ (i, j) \in \mathcal{E}^Y \tag{10}$$

where $e \in \mathbb{R}^{|\mathcal{E}^Y|}$.

Furthermore, several flow conservation constraints are incorporated into the CBAPP problem to ensure path consistency. The first constraint enforces the selection of exactly one edge in the cut-set of the start area s, which initiates the path:

$$\sum_{(s,j)\in\delta(s)} e_{(s,j)} = 1 \tag{11}$$

where $\delta(s) = \{(s, j) \in \mathcal{E}^Y \mid j \in \mathcal{V}^Y, j \neq s\}$ represents the set of edges incident to s.

Similarly, the second flow conservation constraint ensures that exactly one edge is selected in the cut-set of the destination area g, enforcing path completion:

$$\sum_{(j,g)\in\delta(g)} e_{(j,g)} = 1 \tag{12}$$

where $\delta(g) = \{(j,g) \in \mathcal{E}^Y \mid j \in \mathcal{V}^Y, j \neq g\}$ denotes the set of edges incident to g.

Finally, the last flow constraint ensures that each intermediate i-th area in the path has an even number of selected incident edges, preserving flow conservation:

$$\sum_{(j,i)\in\delta(i)} e_{(j,i)} - \sum_{(i,j)\in\delta(i)} e_{(i,j)} = 0, \quad \forall i \in \mathcal{V}^Y \setminus \{s,g\} \tag{13}$$

where $\delta(i)$ denotes the set of edges incident to i, defined as:

$$\delta(i) = \{(j,i) \in \mathcal{E}^Y \mid j \in \mathcal{V}^Y, j \neq i\}, \quad \forall i \in \mathcal{V}^Y \setminus \{s,g\}. \tag{14}$$

Having defined the constraints, we now proceed to describe the objective function, which assigns the γ-th vehicle, $\forall \gamma \in \mathcal{F}$, an edge path ξ_γ, that minimizes the total cost, given by:

$$\sum_{(i,j)\in\mathcal{E}^Y} \left(\sum_{c\in\mathcal{C}} (D_{(i,j),c} + B_{(i,j),c}) w_c \right) e_{(i,j)}. \tag{15}$$

The proposed formulation addresses a multi-vehicle CBAPP problem composed of Γ subproblems. Specifically, the CU is responsible for solving an individual MOO subproblem for each vehicle $\gamma \in \mathcal{F}$. The decomposition allows the planner to generate vehicle-specific paths that are consistent with the tasks objectives. Summing up, let us present CBAPP as a separate binary MOO problem for each $\gamma \in \mathcal{F}$, expressed as follows:

$$\begin{aligned}
\underset{e_{(i,j)}}{\text{minimize}} \quad & \sum_{(i,j)\in\mathcal{E}^Y} \left(\sum_{c\in\mathcal{C}} \left(D_{(i,j),c} + B_{(i,j),c}\right) w_c \right) e_{(i,j)} \\
\text{subject to} \quad & \sum_{(s,j)\in\delta(s)} e_{(s,j)} = 1, \\
& \sum_{(j,g)\in\delta(g)} e_{(j,g)} = 1, \\
& \sum_{(j,i)\in\delta(i)} e_{(j,i)} - \sum_{(i,j)\in\delta(i)} e_{(i,j)} = 0, \quad \forall i \in \mathcal{V}^Y \setminus \{s,g\}, \\
& e_{(i,j)} \in \{0,1\}, \quad \forall\, (i,j) \in \mathcal{E}^Y.
\end{aligned} \tag{16}$$

The objective function in (16) represents the cumulative weighted cost of a feasible path for $\gamma \in \mathcal{F}$, computed over the connectivity map $\mathcal{G}^Y$. The optimization problem in (16) is formulated as an ILP, whose optimal solution is given by:

$$\xi_\gamma = \arg\min_{e_{(i,j)}} (16). \tag{17}$$

Specifically, (16) is solved on the connectivity map $\mathcal{G}^Y$, where the edges are weighted based on the evaluated criteria and their respective weights.

According to Definition 4, the edge path ξ_γ must be mapped to the area path φ_γ, for the γ-th vehicle, achieved employing Dijkstra's algorithm [40], which reduces (16) to a shortest path problem.

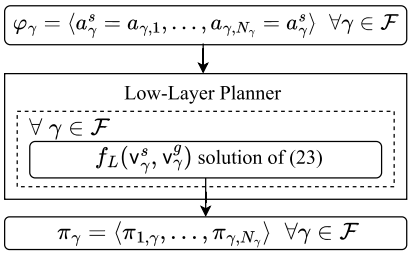


Fig. 4. Logical structure of the low-layer planner, which computes the physical path π_γ for each γ-th vehicle starting from the area path φ_γ, in each replanning step.

Hence, let us rewrite (16) in a compact form, defining $f_H(\cdot)$ as the total cost function in terms of the decision variable $e_{(i,j)}$, incorporating the transformation into the *area path* φ_γ for the γ-th vehicle:

$$\begin{aligned}
\varphi_\gamma = \ & \arg\min_{e_{(i,j)}} f_H(a^s_\gamma, a^g_\gamma, w) \\
& \text{subject to} \ \ (10),\ (11),\ (12),\ (13)\ .
\end{aligned} \tag{18}$$

B. Low-Layer Planner

The low-layer planner is responsible for assigning the γ-th vehicle, for each $\gamma \in \mathcal{F}$, a physical path π_γ, as depicted in Fig. 4, connecting the start vertex v^s_γ to the goal vertex v^g_γ, avoiding obstacles $\mathcal{O}(k)$ at each time step k and guaranteeing the mutual exclusivity of paths. To this end, let $\hat{\Pi}(k) = \{\hat{\pi}_1, \ldots, \hat{\pi}_\Gamma\}$ be the set of paths for all vehicles, ordered by length in descending order. For the γ-th vehicle, the mutual exclusivity is guaranteed by considering the subset $\{\hat{\pi}_i(k)\}_{i\in\{1,\ldots,\gamma-1\}}$ consisting of the paths already optimized for vehicles with longer paths.

Once the area path φ_γ is determined by the high-layer planner defined in Section IV-A, the specific path $\pi_{\gamma,n}$ within each area $a_{\gamma,n}$ with $n \in \mathcal{N}_\gamma$ is computed separately for each $\gamma \in \mathcal{F}$ (see Definition 4).

In the n-th area of the sequence φ_γ, the physical path $\pi_{\gamma,n}$ is optimized to minimize travel cost while accounting for potential obstacles and the mutual exclusivity of paths. Each area $a_{\gamma,n}$ with $n \in \mathcal{N}_\gamma$ is classified as either static, if $a_{\gamma,n} \in \mathcal{S}$, or dynamic, if $a_{\gamma,n} \in \mathcal{D}$, for each $\gamma \in \mathcal{F}$. In a static area $a_{\gamma,n} \in \mathcal{S}$, where no obstacles are present, the obstacle avoidance constraint is not required. The problem, for the k-th time step, is formally defined as follows:

$$\left.\begin{aligned}
& \pi_{\gamma,n} = \arg\min_{\pi_u \in \mathcal{P}_{\gamma,n}} f^*_L(\cdot) \\
& \text{subject to} \quad \mathcal{P}_{\gamma,n} \cap \mathcal{O}(k) \cap \{\hat{\pi}_i(k)\}_{i=1}^{\gamma-1} = \emptyset
\end{aligned}\right\} \quad \forall n \in \mathcal{N}_\gamma \tag{19}$$

where:

$$\mathcal{O}(k) = \begin{cases} \emptyset, & \text{if } a_{\gamma,n} \in \mathcal{S}, \\ \mathcal{O}(k), & \text{if } a_{\gamma,n} \in \mathcal{D}, \end{cases} \quad \forall n \in \mathcal{N}_\gamma, \tag{20}$$

and $\mathcal{P}_{\gamma,n} := \{\pi_1, \ldots, \pi_{P_{\gamma,n}}\}$, with $P_{\gamma,n} = |\mathcal{P}_{\gamma,n}|$, denote the set of feasible paths within the n-th area for the γ-th vehicle, where $\pi_u \in \mathcal{P}_{\gamma,n}$. Instead, the input to $f^*_L(\cdot)$ changes depending

on the position of the area $\{a_{\gamma,n}\}_{n=1}^{N_\gamma}$ in the sequence, as follows:

$$\pi_{\gamma,n} = \arg\min_{\pi_u \in \mathcal{P}_{\gamma,n}} \begin{cases} f_L^*(v_\gamma^s, \mu_m^s \in \mathcal{V}^\mu), & \text{if } a_{\gamma,n} = a_\gamma^s, \\ f_L^*(\mu_m^{a_{\gamma,n-1}} \in \mathcal{V}^\mu, \mu_j^{a_{\gamma,n}} \in \mathcal{V}^\mu), & \text{if } a_{\gamma,n} \neq a_\gamma^s \\ & \wedge a_{\gamma,n} \neq a_\gamma^g \\ f_L^*(\mu_m^g \in \mathcal{V}^\mu, v_\gamma^g), & \text{if } a_{\gamma,n} = a_\gamma^g, \\ f_L^*(v_\gamma^s, v_\gamma^g), & \text{if } a_\gamma^s = a_\gamma^g. \end{cases} \tag{21}$$

Notably, if an area $a_{\gamma,n}$ with $n \in \mathcal{N}_\gamma$ is the initial area a_γ^s in φ_γ for the γ-th vehicle, the proposed method determines the shortest path from the start vertex v_γ^s to the first detachment point $\mu_m^s \in \mathcal{V}^\mu$. For intermediate areas, where $a_{\gamma,n} \neq a_\gamma^s$ and $a_{\gamma,n} \neq a_\gamma^g$, with $n \in \{2, \ldots, N_\gamma - 1\}$, the path is computed from the attachment point $\mu_m^{a_{\gamma,n-1}} \in \mathcal{V}^\mu$ of the preceding area $a_{\gamma,n-1}$ to the detachment point $\mu_j^{a_{\gamma,n}} \in \mathcal{V}^\mu$ of the current area $a_{\gamma,n}$ of the sequence φ_γ. In the final area a_γ^g, the path extends from the last attachment point $\mu_m^g \in \mathcal{V}^\mu$ to the goal vertex v_γ^g. If φ_γ consists of a single area, the path is computed directly as the shortest path from v_γ^s to v_γ^g.

To solve (19), the A* algorithm [41] is applied to compute the optimal sequence of adjacent vertices required to traverse the area. In the case of a static area $a_{\gamma,n} \in \mathcal{S}$ with $n \in \mathcal{N}_\gamma$, (19) is performed using a graph-based approach on $\mathcal{K}_i^S$, with $i \in \{1, \ldots, S\}$ (see Definition 1). Conversely, for a dynamic area $a_{\gamma,n} \in \mathcal{D}$, with $n \in \mathcal{N}_\gamma$, the A* algorithm is employed to determine the optimal sequence of coordinates $\pi_{\gamma,n}$, for the γ-th vehicle, with $n \in \mathcal{N}_\gamma$, on the grid $\mathcal{K}_i^D \in \Omega^D$ (see Definition 2).

The final physical path π_γ, for the γ-th vehicle, is then obtained by concatenating the optimal subpaths $\pi_{\gamma,n}$, with $n \in \mathcal{N}_\gamma$, traversed by the vehicle, within each visited area $\{a_{\gamma,n}\}_{n \in \mathcal{N}_\gamma}$, in accordance with the ordered sequence of areas φ_γ, as:

$$\pi_\gamma = \langle \pi_{\gamma,1}, \ldots, \pi_{\gamma,N_\gamma} \rangle \quad \forall \gamma \in \mathcal{F} \tag{22}$$

where π_γ forms the complete physical path that the γ-th vehicle must follow to complete its task, as shown in Fig. 4.

Let us rewrite (21) and its concatenation (22) in a compact form, by defining $f_L(\cdot)$ as the function expressing the total cost of the path for the γ-th vehicle. This leads to the formulation of the following problem, referred to as the single-vehicle PP problem:

$$\pi_\gamma = \arg\min_{\langle \pi_{\gamma,1}, \ldots, \pi_{\gamma,N_\gamma} \rangle} f_L(v_\gamma^s, v_\gamma^g) \quad \text{subject to (18).} \tag{23}$$

C. High–Low Layer Interaction

This section provides a detailed description of the sequential coordination between the high-layer and low-layer planners, illustrating how the hierarchical PP strategy ensures coherent computation and dynamic adaptation of the physical paths π_γ for the γ-th vehicle, $\forall \gamma \in \mathcal{F}$.

At each discrete time instant $t(k)$, the position $p_\gamma(k)$ of each γ-th vehicle is estimated through a real-time localization system –whose design and implementation lie beyond the scope of this work– and subsequently processed by the CU. Acting as the core decision-making entity, the CU is responsible for managing mission execution by computing the physical paths π_γ, $\forall \gamma \in \mathcal{F}$ and guaranteeing their mutual exclusivity.

In accordance with Assumption 3, which defines the priority structure governing PP among vehicles, the CU considers previously assigned physical paths $\{\pi_\gamma\}_{\gamma \in \mathcal{F}}$ and iteratively adjusts them by evaluating, in real-time, both roadmap-related criteria B and vehicle-related criteria D_γ, $\forall \gamma \in \mathcal{F}$. Through this mechanism, the CU ensures that each physical path π_γ, for each $\gamma \in \mathcal{F}$, is dynamically adapted to maintain optimal vehicle performance throughout the mission. The iterative process continues until all vehicles have successfully reached their respective goals and the overall mission is completed in an integrated and efficient manner.

Algorithm 1 takes as input $\mathcal{F}$, the start vertices $\{v_\gamma^s\}_{\gamma \in \mathcal{F}}$, the goal vertices $\{v_\gamma^g\}_{\gamma \in \mathcal{F}}$, the start areas $\{a_\gamma^s\}_{\gamma \in \mathcal{F}}$, the goal areas $\{a_\gamma^g\}_{\gamma \in \mathcal{F}}$, and the weight vector w (Algorithm 1, lines 1-2). The process concludes once all vehicles have reached their designated destinations $\{v_\gamma^g\}_{\gamma \in \mathcal{F}}$ (Algorithm 1, line 30).

At time step $k = 0$, Algorithm 1 initializes the planning process for the γ-th vehicle, $\forall \gamma \in \mathcal{F}$, which is derived through the integration of both high-layer and low-layer planning strategies (Algorithm 1, lines 25-28). Note that all vehicles depart simultaneously, following the initial planned paths $\{\pi_\gamma(k)\}_{\gamma \in \mathcal{F}}$, which are subject to adjustment through replanning in response to evolving conditions.

Algorithm 1 Multi-Vehicle Path Planning

1 **Input:** $\mathcal{F}$, $\{v_\gamma^s, v_\gamma^g, a_\gamma^s, a_\gamma^g\}_{\gamma \in \mathcal{F}}$, w;
2 **Output:** $\{\pi_\gamma\}_{\gamma \in \mathcal{F}}$;
3 **Function** `HighLayer` $(a_\gamma^s, a_\gamma^g, w)$ **:**
4 $\quad \varphi_\gamma \leftarrow \arg\min f_H(a_\gamma^s, a_\gamma^g, w)$ s.t. (10), (11), (12), (13);
5 $\quad$ **return** φ_γ;
6 **Function** `LowLayer` $(\varphi_\gamma, v_\gamma^s, v_\gamma^g, \mathcal{O}(k), \{\hat{\pi}_i(k)\}_{i=1}^{\gamma-1})$ **:**
7 $\quad \pi_\gamma \leftarrow \arg\min f_L(v_\gamma^s, v_\gamma^g)$ s.t. (18);
8 $\quad$ **return** π_γ;
9 **Function** `RePlanning` $(\mathcal{F}, \{v_\gamma^g, a_\gamma^g, \pi_\gamma(k)\}_{\gamma \in \mathcal{F}}, w, k, \hat{\Pi}(k))$ **:**
10 $\quad k = k + 1$;
11 $\quad B(k) \leftarrow$ *EvaluateRoadmapCriteria*(k);
12 $\quad$ **foreach** $\gamma \in \mathcal{F}$ **do**
13 $\quad\quad D_\gamma(k) \leftarrow$ *EvaluateVehicleCriteria*(γ, k);
14 $\quad\quad p_\gamma(k) \leftarrow$ *CurrentPosition*(γ);
15 $\quad\quad$ **if** $B(k) \neq B(k-1) \vee D_\gamma(k) \neq D_\gamma(k-1)$ **then**
16 $\quad\quad\quad a_\gamma(k) \leftarrow$ *FindArea*$(p_\gamma(k))$;
17 $\quad\quad\quad \varphi_\gamma(k) \leftarrow$ `HighLayer` $(a_\gamma(k), a_\gamma^g, w)$;
18 $\quad\quad\quad \pi_\gamma(k) \leftarrow$ `LowLayer` $(\varphi_\gamma(k), p_\gamma(k), v_\gamma^g, \mathcal{O}(k), \{\hat{\pi}_i(k)\}_{i=1}^{\gamma-1})$;
19 $\quad\quad$ **if** $p_\gamma(k) = v_\gamma^g$ **then**
20 $\quad\quad\quad \mathcal{F} = \mathcal{F} \setminus \{\gamma\}$;
21 $\quad$ **if** $\mathcal{F} = \emptyset$ **then**
22 $\quad\quad$ **return** $\{\pi_\gamma(k)\}_{\gamma \in \mathcal{F}}$;
23 $\quad \hat{\Pi}(k) = \{\pi_\gamma(k)\}_{\gamma \in \mathcal{F}}$;
24 $\quad$ **return** `RePlanning` $(\mathcal{F}, \{v_\gamma^g, a_\gamma^g, \pi_\gamma(k)\}_{\gamma \in \mathcal{F}}, w, k, \hat{\Pi}(k))$;
25 $k = 0$;
26 **foreach** $\gamma \in \mathcal{F}$ **do**
27 $\quad \varphi_\gamma(k) \leftarrow$ `HighLayer`$(a_\gamma^s, a_\gamma^g, w)$;
28 $\quad \pi_\gamma(k) \leftarrow$ `LowLayer`$(\varphi_\gamma, v_\gamma^s, v_\gamma^g, \mathcal{O}(k), \emptyset)$;
29 $\hat{\Pi}(k) \leftarrow$ *Prioritize*$(\{\pi_\gamma(0)\}_{\gamma \in \mathcal{F}})$;
30 $\{\pi_\gamma(k)\}_{\gamma \in \mathcal{F}} \leftarrow$ `RePlanning` $(\mathcal{F}, \{v_\gamma^g, a_\gamma^g, \pi_\gamma(k)\}_{\gamma \in \mathcal{F}}, w, k, \hat{\Pi}(k))$

Once the initial paths $\{\pi_\gamma(k)\}_{\gamma\in\mathcal{F}}$ have been established, the *Prioritize*$(\cdot)$ function (Algorithm 1, line 29) ranks the vehicles in descending order based on their path lengths, establishing a priority hierarchy where vehicles with longer path are planned first, forming the ordered set $\hat{\Pi}(k)$ used to enforce mutual exclusivity.

On the one hand, the `HighLayer(·)` function (see Section IV-A) determines the optimal sequence of areas φ_γ by minimizing the objective function $f_H(a^s_\gamma, a^g_\gamma, w)$, for each $\gamma \in \mathcal{F}$ (Algorithm 1, lines 3–5). On the other hand, the `LowLayer(·)` function (see Section IV-B) generates the physical path π_γ based on φ_γ, by optimizing $f_L(v^s_\gamma, v^g_\gamma)$ with respect to the obstacle configuration $\mathcal{O}(k)$ and the set of paths $\{\hat{\pi}_i(k)\}_{i=1}^{\gamma-1}$ already assigned to higher-priority vehicles, at each time step k, for each $\gamma \in \mathcal{F}$ (Algorithm 1, lines 6–8).

The algorithm proceeds to the `RePlanning(·)` routine (Algorithm 1, line 30), where it dynamically adjusts the paths of all vehicles at each time step k based on real-time environmental conditions.

During the replanning phase (Algorithm 1, lines 9–23), the CU assesses the roadmap-related criteria B, using the *EvaluateRoadmapCriteria*$(\cdot)$ function (Algorithm 1, line 11) and vehicle-related criteria, via the *EvaluateVehicleCriteria*$(\cdot)$ function D_γ, $\forall\gamma \in \mathcal{F}$ (Algorithm 1, lines 12-13), with the aim of performing dynamic adjustments to the vehicles paths $\{\pi_\gamma(k)\}_{\gamma\in\mathcal{F}}$ based on the current conditions of the environment and the fleet. If the evaluated criteria for the γ-th vehicle have changed with respect to the previous planning, a new path $\pi_\gamma(k)$ is computed by determining the start area $a_\gamma(k)$ through the *FindArea(·)* function (Algorithm 1, line 16), employing the vehicle's current position $p_\gamma(k)$ (Algorithm 1, line 14) as input (Algorithm 1, lines 15–18). Specifically, $p_\gamma(k)$ of each $\gamma \in \mathcal{F}$ is estimated by the CU using the *CurrentPosition(·)* function.

Once the γ-th vehicle reaches its goal vertex v^g_γ, it is removed from the set, i.e., $\mathcal{F} = \mathcal{F} \setminus \{\gamma\}$ (Algorithm 1, lines 19–20), ensuring no further updates for vehicles that have completed their task. The process continues iteratively and ends when all vehicles reach their destinations $\{v^g_\gamma\}_{\gamma\in\mathcal{F}}$, with the final paths $\{\pi_\gamma(k)\}_{\gamma\in\mathcal{F}}$ returned as output (Algorithm 1, lines 21–22). At the end of each time step, the set $\hat{\Pi}(k)$ is updated with the newly computed paths (Algorithm 1, line 23-24), maintaining the priority order for the subsequent recursive call to the `RePlanning(·)` routine.

V. Quasi-Adaptive Self-Tuning for Automated Weighting

The hierarchical planner is integrated within a bi-level optimization framework [42], where the outer-level authority aims to optimize the set of designated criteria being influenced by the responses of an inner-level entity, which pursues its own operational goals, i.e., the PP process in the context of this manuscript. To overcome the limitations of subjective and potentially suboptimal weight selection in MOO, a quasi-adaptive self-tuning process is introduced at the outer-level. This approach is tightly coupled with the decision-making process at the inner-level, structured according to the hierarchical architecture outlined in Section IV, which integrates both the CBAPP problem (16) (see Section IV-A) and the single-vehicle PP problem (23) (see Section IV-B). Notably, both problems (16) and (23) are solved for the entire fleet, jointly defining the multi-vehicle PP problem. Operating in real-time, the inner-level continuously provides feedback that supports adaptive weight adjustments at the outer-level.

Weight configurations determined through the quasi-adaptive self-tuning process are designed to remain robust over medium to long-term. Despite being computed offline, they generalize effectively across diverse missions, capturing different priority trade-offs without introducing subjective biases and mitigating the need for frequent manual intervention or tuning. In cases where the relative importance of the objectives evolves, the optimization process can be re-executed to identify a more appropriate configuration aligned with the updated priorities.

Let us introduce the constraints that define the outer-level problem employed to determine the optimal weight values. The first component consists of the weight variables, defined as follows:

$$w_c \geq 0, \quad \forall c \in \mathcal{C}. \tag{24}$$

Subsequently, a normalization constraint is imposed to ensure that the sum of all weights equals one, guaranteeing that each criterion contributes proportionally to the overall evaluation:

$$\sum_{c\in\mathcal{C}} w_c = 1. \tag{25}$$

Additionally, a descending order constraint is introduced to enforce distinct weight values:

$$w_c - w_{c+1} > 0, \quad \forall c \in \{1, \ldots, C-1\}. \tag{26}$$

Having established the constraints, we now proceed to describe the objective function associated with the outer-level optimization. The aim is to minimize the total mission time of the fleet, with the specific formulation of the performance metric influenced by system-level priorities and the operational context defined by the user. The adoption of time as a metric to evaluate path optimality is justified by its central role in optimizing productivity [43].

In this regard, we define a general objective function $R(w)$, which is to be maximized and is defined as inversely proportional to a time-based performance metric $T(w)$ evaluated over the fleet. Formally, the reward function takes the form:

$$R(w) = T(w)^{-1} \tag{27}$$

where $T(w)$ represents the output of the inner-level optimization process given a weight vector w, as described in Section IV. In particular, $T(w)$ is designed to reflect user preferences with respect to mission time and can be expressed differently depending on the considered operational scenario.

A first possible formulation consists in minimizing the fleet completion time across the fleet:

$$T(w) = \sum_{\gamma\in\mathcal{F}} (t_\gamma - t^*_\gamma) \tag{28}$$

where t_γ represents the actual time required for the γ-th vehicle to complete its assigned task, and t^*_γ denotes the minimum theoretical time corresponding to the shortest path without constraints, with $\gamma \in \mathcal{F}$.

Algorithm 2 Bi-Level Optimization Problem

1 **Output:** w^*;
2 **Function** `InnerProblem(w)`:
3 $\{t_\gamma\}_{\gamma\in\mathcal{F}} \leftarrow$ solution of (23);
4 $T(w) \leftarrow$ *ComputeMetric*$(\{t_\gamma\}_{\gamma\in\mathcal{F}})$;
5 $R(w) = T(w)^{-1}$;
6 **return** $R(w)$;
7 $w \leftarrow$ Computed s.t. (24), (25), (26);
8 $w^* \leftarrow$ *OuterProblem*(`InnerProblem(w)`);

Alternatively, the objective may focus on minimizing the overall mission duration, defined by the completion time of the vehicle with the longest assigned task, to reduce disparities in workload distribution across the fleet:

$$T(w) = \max_{\gamma\in\mathcal{F}} (t_\gamma - t^*_\gamma). \tag{29}$$

Instead, a third formulation targets the average mission time per vehicle, aiming to minimize the mean deviation from the corresponding ideal path durations:

$$T(w) = |\mathcal{F}|^{-1} \sum_{\gamma\in\mathcal{F}} (t_\gamma - t^*_\gamma). \tag{30}$$

The selection among the aforementioned temporal objectives is determined by the specific priorities of the industrial context. While the first objective emphasizes overall time performance, the second focuses on fairness by minimizing excessive delays for any individual vehicle, and the third offers a balanced compromise between global performance and individual performance. Note that these objectives are presented as representative examples; alternative formulations may also be considered, depending on the operational goals and constraints of the specific application. Finally, the bi-level optimization problem can be formulated as follows:

$$\begin{aligned} \underset{w_c}{\text{maximize}} \quad & R(w) \\ \text{subject to} \quad & \sum_{c\in\mathcal{C}} w_c = 1, \\ & w_c - w_{c+1} > 0, \quad \forall\, c \in \{1,\dots,C-1\}, \\ & w_c \geq 0, \quad \forall\, c \in \mathcal{C}, \\ & \{t_\gamma\}_{\gamma\in\mathcal{F}} \text{ solution of (23).} \end{aligned} \tag{31}$$

Let us rewrite (31) in a compact form by defining $f_W(\cdot)$ as the function that represents a feasible optimized vector w^*:

$$\begin{aligned} w^* = \ & \arg\max_{w_c} f_W(w) \\ & \text{subject to } \{t_\gamma\}_{\gamma\in\mathcal{F}} \text{ solution of (23).} \end{aligned} \tag{32}$$

From an operational perspective, we present Algorithm 2 to compute the optimized weight vector w^* (Algorithm 2, line 1), for the inner-level problem (see Section IV). Specifically, the candidate weight vector w is initialized to satisfy the feasibility constraints in (24), (25), and (26) (Algorithm 2, line 7), and sent to the outer-level function *OuterProblem*$(\cdot)$, which iteratively updates it based on feedback from the inner-level problem (Algorithm 2, line 8). In particular, the outer optimization can be solved using several methods from the literature, such as evolutionary algorithms [44] (e.g., genetic algorithms (GA) [45]), Bayesian optimization [46], neural network [47], deep neural network [48], or other metaheuristics approaches.

In the `InnerProblem(·)`, the hierarchical optimization problem (23) is addressed by incorporating the weight vector w within the ILP formulation of the CBAPP problem (16). The resulting mission completion times, denoted as $\{t_\gamma\}_{\gamma\in\mathcal{F}}$, are derived from the multi-vehicle PP strategy, which involves minimizing the single-vehicle PP problem (23) while satisfying the CBAPP problem (16), both solved for the entire fleet of vehicles under a given weight configuration w. Based on $\{t_\gamma\}_{\gamma\in\mathcal{F}}$, the time-based performance metric $T(w)$ is computed using the function *ComputeMetric*$(\cdot)$, in accordance with one of the possible optimization criteria (e.g., 28, 29, and 30). Finally, the reward function $R(w)$ is calculated and subsequently returned to the outer-level optimization process (Algorithm 2, lines 2–6).

The outer-level problem continues to explore the weight space until a stopping condition is satisfied or convergence is achieved, ultimately returning the optimized weight vector w^* (Algorithm 2, line 8).

Given the potential variability in preference specification, the quasi-adaptive self-tuning process provides an automated approach to determine weights. However, manual assignment is still an option when specific preferences or predefined configurations are available, allowing for adaptation to different operational settings.

VI. Case Study

A. System Setup

Our numerical simulations are designed to assess the effectiveness of the proposed hierarchical MOO PP strategy across hybrid roadmaps. The setup process involves defining the operating environment –including the layout and the associated roadmap– along with the criteria adopted in the CBAPP (see Section IV-A), the input parameters, and the methodology used to assign weights within the CBAPP.

The simulations, implemented in Python 3.9.6, are intended to replicate realistic industrial conditions, enabling accurate and reliable performance evaluations. All experiments are conducted on a laptop equipped with an Apple M1 (8-core CPU with 4 performance cores and 4 efficiency cores, 8-core GPU and 16-core Neural Engine) and 8 GB of RAM.

In detail, the effectiveness and adaptability of the proposed hybrid architecture are evaluated across two distinct configurations representative of an industrial plant environment. The two layouts primarily differ in the presence of a central corridor and in their overall connectivity, which is quantified by the algebraic connectivity (AC) of $\mathcal{G}^Y$, defined as the real part of the second smallest eigenvalue in modulus of its Laplacian matrix [49]. Notably, the layout including the corridor provides alternative paths that increase the AC by 20.9%. In both cases, the layout spans 100×110 m^2 and is designed to accommodate up to 30 vehicles. For each configuration, tests are conducted both with and without the replanning strategy, and under three fleet size scenarios, defined by deployments of 10, 20, and 30 vehicles. To ensure statistical robustness and capture consistent performance trends, each scenario is repeated

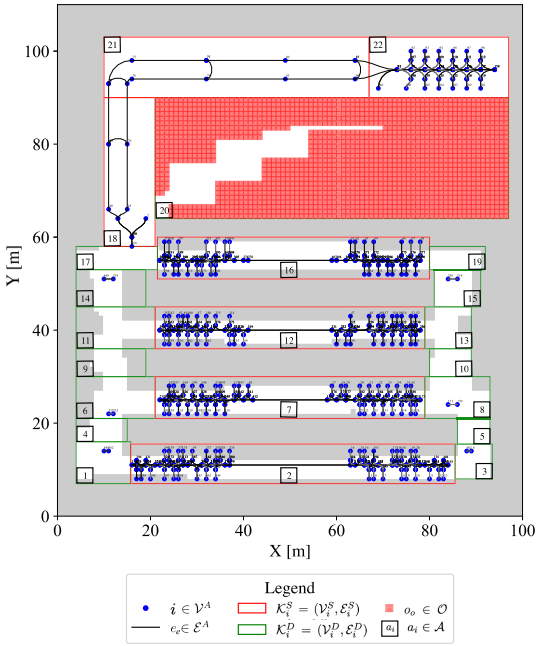


Fig. 5. Hybrid roadmap for autonomous vehicle navigation within a constrained layout. The storage area, labeled as Area 20, exhibits high occupancy density, which restricts vehicle movement.

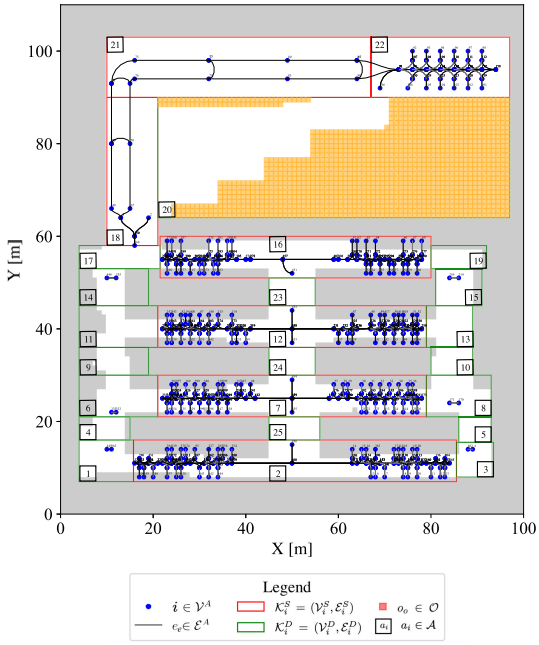


Fig. 6. Hybrid roadmap for autonomous vehicle navigation with a central corridor. The storage area, labeled as Area 20, has medium obstacles occupancy density –as indicated by the orange color– allowing for feasible navigation.

over 50 independent simulation runs, resulting in a total of 600 simulations.

The first configuration, illustrated in Fig. 5, features a layout without a central corridor, resulting in a constrained environment with restricted vehicle path options and diminished operational flexibility. The plant comprises multiple functional areas –either static or dynamic–interconnected via a hybrid roadmap. A dedicated storage area is included, where pallets are systematically stored and retrieved. As a key component of the simulations, the storage area has a significant impact

TABLE I

AREA SPECIFICATIONS OF ROADMAPS SHOWN IN FIG. 5 AND FIG. 6

Area ID	Reference Frame	Width [m]	Height [m]	Type
1	(4, 7)	11.75	9	Dynamic
2	(15.75, 7)	69.75	8.50	Static
3	(85.50, 8)	8	7.50	Dynamic
4	(4, 16)	11	5	Dynamic
5	(86, 15.50)	7	5.50	Dynamic
6	(4, 21)	17	9	Dynamic
7	(21, 21)	58	9	Static
8	(79, 21)	14	9	Dynamic
9	(4, 30)	15	6	Dynamic
10	(80, 30)	9	6	Dynamic
11	(4, 36)	17	9	Dynamic
12	(21, 36)	58	9	Static
13	(79, 36)	10	9	Dynamic
14	(4, 45)	15	8	Dynamic
15	(81, 45)	10	8	Dynamic
16	(21.50, 51)	58.50	9	Static
17	(4, 53)	17.50	5	Dynamic
18	(10, 58)	11	32	Static
19	(80, 53)	12	5	Dynamic
20	(21, 64)	76	26	Dynamic
21	(10, 90)	57	13	Static
22	(67, 90)	30	13	Static
23	(45, 45)	10	6	Dynamic
24	(45, 30)	10	6	Dynamic
25	(45, 16)	11	5	Dynamic

on multi-vehicle PP. Pallets are placed at predefined locations within the area and measure 1×1 m^2. The detailed specifications of each area (IDs 1 to 22) are provided in Table I, which lists the Area ID (1st column, Table I) –matching the numerical indicators shown as black squares in Fig. 5– along with the reference frame (2nd column, Table I), defined as the coordinates of the lower-left corner, which serves as the origin for positioning each area within the layout. Table I also reports the width (3rd column, Table I), height (4th column, Table I), and typology of each area (5th column, Table I), classified as either static or dynamic.

In contrast, the second configuration (see Fig. 6) features a central corridor running longitudinally through the plant layout. As a result, efficiency improves through a more even distribution of vehicle movement across the facility. Detailed specifications for all 25 areas considered in this configuration are provided in Table I.

The second key component in the simulations is the distribution of pallet occupation within the storage area. The number of pallets varies throughout the operational cycle, directly influencing the accessibility of alternative paths and the performance of navigation strategies. High levels of occupation –as illustrated in Fig. 5– restrict the use of the storage area for vehicle transit, resulting in adjustments to the planned paths. Note that the level of pallet occupation is independent of the specific layout configuration –whether featuring a central corridor (see Fig. 6) or not (see Fig. 5)– and is instead solely determined by the characteristics and demands of the operational cycle. By decoupling layout configuration from pallet occupation, the specific contribution of each factor to system behavior can be more precisely examined.

Table II summarizes occupation thresholds and storage capacities for both scenarios. Accurate modeling of storage area dynamics within the simulations relies on distinguishing between maximum capacity with open passage (1st row, Table II) –the highest number of pallets that can be stored

TABLE II
STORAGE CAPACITY AND DENSITY THRESHOLDS

Storage Capacity		
Description	**N° Pallets**	**Occupation**
Max. capacity with open passage	1595	80.70%
Max. area capacity	1976	100%
Density Threshold		
Description	**N° Pallets**	**Occupation**
Low density	0–493	0%–25%
Medium-low density	494–987	25%–50%
Medium density	988–1580	50%–80%
High density	1581–1976	80%–100%

*Pallet size fixed at $1 \times 1\,\mathrm{m}^2$

while still allowing vehicle transit– and maximum area capacity –the total number of pallets the storage area can hold regardless of transit constraints (2nd row, Table II). This distinction offers a realistic depiction of how storage utilization influences vehicle movement and traffic flow. By incorporating reference values, the simulations model various operational conditions, ranging from lightly occupied storage areas that permit free vehicle access to highly congested states that obstruct transit and force replanning. Four occupancy thresholds characterize different levels of storage use –less than 25%, 25% to 50%, 50% to 80%, and above 80% of total capacity– supporting evaluation of system performance (3rd to 6th rows, Table II).

Defining the CBAPP problem (16) requires a careful selection of evaluation criteria. As outlined in Section IV-A, the PP strategy is assessed by the high-layer planner using a structured set of well-established roadmap-related B and vehicle-related criteria D_γ, with $\gamma \in \mathcal{F}$. The selected criteria are listed in Table III, where each criterion is associated with its corresponding category (1st column, Table III), its label –i.e., preferability, occupancy, attachment/detachment vertex, traversed areas, farthest vertex, and traffic– (2nd column, Table III), and a detailed explanation of the computation and evaluation procedure (3rd column, Table III). The fourth column of Table III defines the subset of areas to which each criterion applies. When a criterion is not relevant to all areas in $\mathcal{A}$, its evaluation is restricted to the specified subset, and a zero value is assigned elsewhere. To enhance usability, all criteria are defined with respect to areas, enabling users to work directly with area-based values. However, since the CBAPP problem (see Section IV-A) requires criteria to be associated with edges, the conversion from area-based to edge-based values is automatically handled by our system. Specifically, each edge $e_e = (i, j) \in \mathcal{E}^Y$ is assigned with the value corresponding to its destination area $a_j = j$, with $a_j \in \mathcal{A}$ and $j \in \mathcal{V}^Y$. The weight w_c, with $c \in \mathcal{C}$, assigned to each criterion appears in the fifth column of Table III. Each weight results from the outer-level of the bi-level optimization process (31) (see Section V). Among the available evaluation metrics $T(w)$, fleet completion time is selected as the optimization objective, determining the final weight configuration.

Computation of w is performed using a GA, chosen for its ability to balance exploration, exploitation, and computational efficiency. Standard GA operators –mutation, crossover, and elitism [51]– are used. To preserve genetic diversity and avoid premature convergence to local optima, a mutation rate is set to 0.80. A crossover rate of 0.25 allows recombination while preserving high-quality individuals, and an elitism rate of 0.50 ensures the best half of the population is retained, promoting a convergence of 0.947 of the unit simplex.

To further improve solution quality and robustness, the GA is executed using a multi-start strategy [52], consisting of 32 independent runs. This approach mitigates the risk of convergence to suboptimal solutions by promoting broader exploration of the search space. Each run spans 50 generations with a population of 20 individuals, resulting in a total of 32,000 candidate solutions evaluated. To enhance population diversity and reduce redundancy, a control mechanism is applied to eliminate duplicate solutions across generations [53].

Regarding the input parameters of Algorithm 1, several key values are required (see Table IV). The fleet is assumed to move at constant speed, with each vehicle traveling at 3 m/s under normal traffic conditions (1st row, Table IV). A speed reduction to 1 m/s is applied in the presence of congestion, defined as the occurrence of more than 20% of the fleet occupying the same area.

Additional inputs include the mission list (2nd row, Table IV), the preferability values associated with different areas (3rd row, Table IV), the occupancy level of the storage area and the pallet allocation (4th row, Table IV). To simulate realistic conditions, both the mission list and roadmap-related criteria are randomly generated during the simulations. In contrast, vehicle-related criteria are not defined as input parameters; instead, they are dynamically computed based on the system's current state and its evolution over time. Temporal parameters are also incorporated to regulate the system's dynamic behavior, including a replanning interval (5th row, Table IV) set to 5 s and a path computation timeout (6th row, Table IV) fixed at 30 s. The 5 s replanning interval is defined as a compromise between responsiveness and computational load, acting as an upper bound that guarantees synchronized updates across all vehicles. Conversely, the 30 s path computation timeout enhances robustness by preventing the planner from getting stuck indefinitely in complex or infeasible scenarios.

B. Results

The experimental results are presented by configuration, where each simulation run corresponds to the completion of a single mission. First, we report the outcomes obtained with the first configuration (see Fig. 5), which is characterized by the absence of a central corridor. Results are presented for three scenarios involving 10, 20, and 30 vehicles, respectively, considering both strategies with and without replanning. Then, we analyze the alternative configuration (see Fig. 6), which includes a dedicated corridor. The same structure is followed to ensure consistency. Finally, a comparative analysis highlights the differences observed between the two configurations based on the following performance indicators:

TABLE III
CRITERIA AND WEIGHT VALUES

Category	Criterion (c)	Description	Target Set	Weight (w_c)
Roadmap-related B	Preferability	User-defined penalization value for transits in the i-th area	$a_i \in \mathcal{A}$	0.15
	Occupancy	Total storage area occupied by each obstacle $o_o(k) \in \mathcal{O}(k)$	$a_i \in \mathcal{W}$	0.17
Vehicle-related $D_\gamma,\ \forall \gamma \in \mathcal{F}$	Attachment/Detachment Vertex	Euclidean distance from the start v_γ^s and goal v_γ^g vertices to possible attachment/detachment vertices $\mu_m \in \mathcal{V}^\mu$	$a_i \in \hat{\mathcal{K}}^*$	0.23
	Traversed Areas	Minimum number of areas that must be traversed to reach the goal area a_γ^g, starting from the i-th area. Evaluated through breadth-first search [50]	$a_i \in \mathcal{A}$	0.02
	Farthest Vertex	Geometric efficiency metric computed as the Euclidean distance between the farthest vertex of the i-th area (with respect to the goal vertex v_γ^g) and the midpoint of each edge connecting the i-th area to adjacent areas	$a_i \in \mathcal{N}$	0.37
	Traffic	Estimated number of vehicles in the i-th area [16]	$a_i \in \mathcal{A}$	0.06

*Where $\hat{\mathcal{K}} \subseteq \mathcal{N}$ denotes the subset of normal areas adjacent to the start a_γ^s or goal areas a_γ^g, for the γ-th vehicle

TABLE IV
USER-DEFINED INPUT PARAMETERS

Parameter	Value / Description
Vehicle's speed	3 m/s
Mission list	User-defined*
Preferability criterion	User-defined*
Occupancy criterion and pallet allocation	User-defined*
Replanning interval	5 s
Path computation timeout	30 s

*Randomly generated

- *Total mission time (TMT)*: The total duration of mission execution, defined as the time taken by the vehicle performing the longest task in the mission list;
- *Cumulative mission time (CMT)*: The cumulative mission time across all simulation runs, obtained by summing the maximum task durations (i.e., the TMT) from each run;
- *Travel time spread (TTS)*: The average imbalance in tasks distribution, computed for each simulation run as the difference between:
  - *Total travel time*: The sum of all vehicles tasks durations in a mission;
  - *Mean travel time*: The average of vehicle travel times.

  This difference is then averaged across all runs.

Based on the simulation results for the first configuration, the impact of replanning is evaluated across the three scenarios. The results, illustrated in Fig. 7-a, Fig. 7-c, and Fig. 7-e, correspond to fleets of 10, 20, and 30 vehicles, respectively. The horizontal axis represents the simulation index, while the vertical axis indicates the TMT (in seconds) required to complete each task of the mission list. The comparison between the two strategies shows that the absence of replanning (blue line in Fig. 7) leads to consistently longer TMT across all scenarios. The corresponding mean and variance values – reported both in Fig. 7 and Table V– provide quantitative insight into performance consistency. Notably, the higher variance observed without replanning suggests greater variability due to unresolved congestion and inefficient path selection.

A closer analysis of each scenario highlights how replanning improves time savings. In the case of 10 vehicles (see Fig. 7-a), the replanning strategy reduces CMT by 7.58% and lowers the TTS by 11.99%. With 20 vehicles (see Fig. 7-c), the improvements reach 9.25% in CMT and 14.32% in the TTS, indicating more effective congestion management and shorter CMT. A slight reduction in variance is also observed under the replanning condition (orange line Fig. 7-c), suggesting enhanced robustness to variability. In the 30-vehicle scenario (see Fig. 7-e), the benefits of replanning are further amplified, with a 9.89% reduction in CMT and a 14.70% decrease in the TTS. The variance associated with the absence of replanning (blue line Fig. 7-e) increases significantly, indicating substantial unpredictability in system performance when dynamic traffic handling is not implemented. An overall trend emerges, indicating that larger fleet sizes are associated with greater effectiveness of the replanning strategy. The results are summarized in Table V, highlighting the percentage reductions in CMT. Statistical validation through p-value confirms the significance of the observed differences across all scenarios.

As with the first configuration, the second one is analyzed through simulations to evaluate the effect of replanning versus the non-replanning strategy. The results, shown in Fig. 7-b, Fig. 7-d, and Fig. 7-f, correspond to scenarios with 10, 20, and 30 vehicles, respectively. Performance differences between the two strategies are even more pronounced. In the 10-vehicle scenario (see Fig. 7-b), enabling replanning reduces CMT by 29.05%, with a statistically significant p-value of 1.40×10^{-7}. Variance also decreases substantially –from 2,068.36 s^2 to 362.77 s^2 (an 82.46% reduction)– indicating more robust and predictable outcomes. Additionally, the TTS drops by 42.72%, reflecting improved consistency in mission durations. The observed trend continues and intensifies with larger fleet sizes. In the 20-vehicle scenario (see Fig. 7-d), CMT is reduced by 33.13% ($p = 7.66 \times 10^{-12}$), with a 73.07% decrease in variance and a 43.57% reduction in the TTS. The results highlight the proposed methodology's effectiveness in managing higher

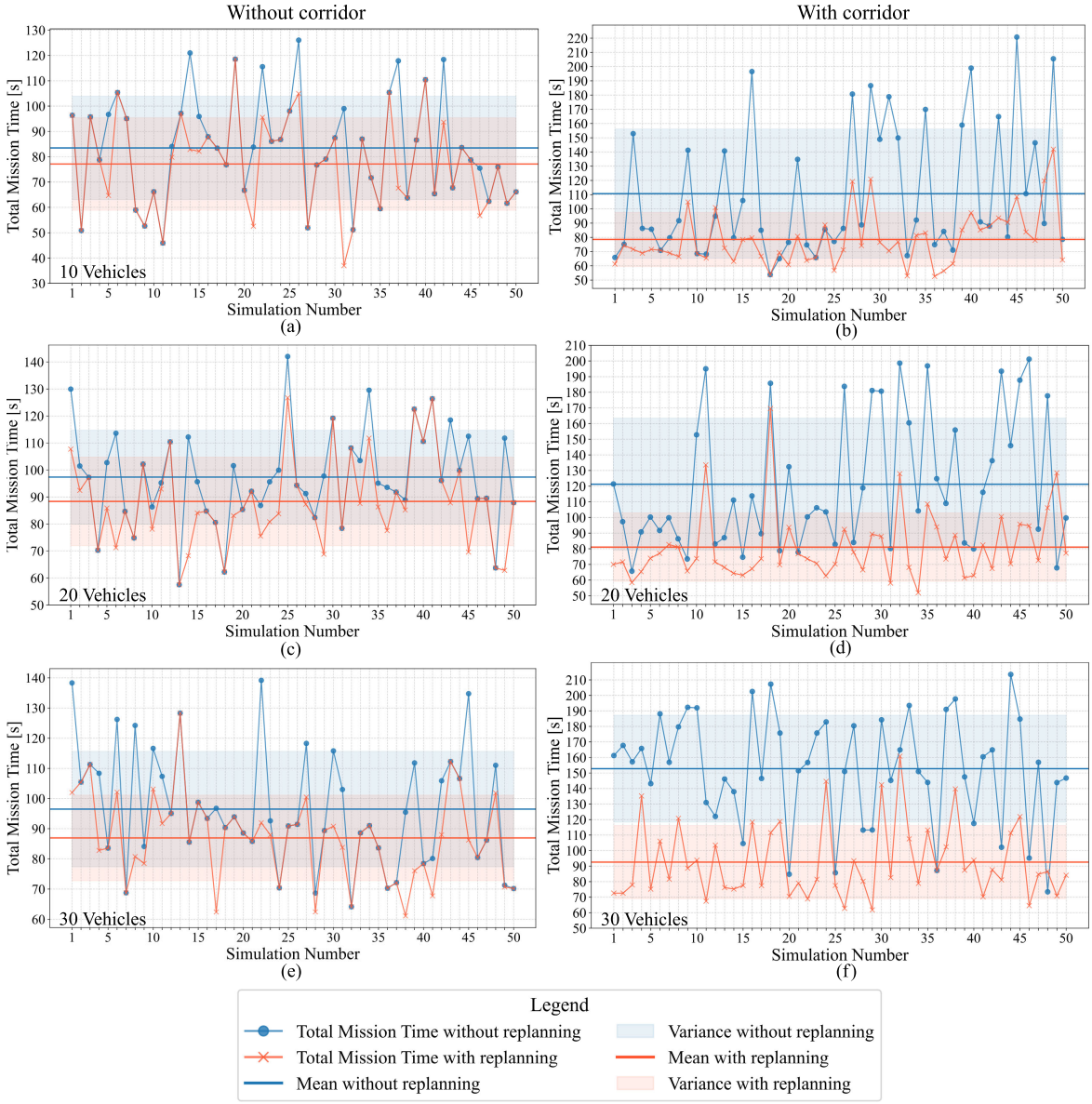


Fig. 7. Comparison of TMT for the two configurations with and without corridor. Each column includes three plots representing fleet scenarios with 10, 20, and 30 vehicles, respectively, under the same configuration. In each plot, the TMT's trends over 50 simulations are shown for both strategies with replanning (red) and without (blue). Mean and variance values are reported to enhance performance differences and assess robustness to variability.

complexity and dynamic variability through replanning. In the 30-vehicle scenario (see Fig. 7-f), performance improvements are most significant: a 39.39% reduction in CMT ($p = 1.78 \times 10^{-14}$), a 53.25% drop in variance, and a 49.25% narrowing of the TTS. The corresponding numerical results are reported in Table VI.

Finally, the comparison between the two configurations (see Fig. 7, Table V, and Table VI) highlights the critical role of layout connectivity in determining system performance. Introducing a central corridor substantially increases AC by providing multiple alternative paths, which enables a more efficient redistribution of vehicle flows and reduces congestion in high-traffic areas. This structural advantage translates into a marked improvement in mission metrics: for a fleet of 30 vehicles, the CMT reduction rises from 9.89% in the layout without the corridor to 39.39% with the corridor. Variance analysis further confirms enhanced reliability, with the corridor-based configuration exhibiting significantly lower variability in mission completion times. In the absence of the corridor, performance improvements are sporadic and inconsistent (see Fig. 7-a, Fig. 7-c, and Fig. 7-e), with TMT often remaining unchanged. Conversely, the corridor-based layout consistently achieves better outcomes, with reduced TMT and TTS across all simulation runs, demonstrating the effectiveness and robustness of the proposed hybrid navigation methodology under diverse traffic conditions.

The computational performance of the replanning process is evaluated for both corridor-free and corridor-based configurations. Results show that the corridor-based setup consistently achieves lower computation times across all scenarios. As

TABLE V
COMPARISON OF PERFORMANCE WITH AND WITHOUT REPLANNING - CONFIGURATION WITHOUT CORRIDOR

Fleet Size	**10 Vehicles**		**20 Vehicles**		**30 Vehicles**	
Description	**Without Replanning**	**With Replanning**	**Without Replanning**	**With Replanning**	**Without Replanning**	**With Replanning**
Travel time spread	42.63 s	37.52 s	54.27 s	46.50 s	57.08 s	48.69 s
Mean (illustrated in Fig. 7)	83.47 s	77.14 s	97.36 s	88.35 s	96.49 s	86.95 s
Variance (illustrated in Fig. 7)	417.53 s^2	334.92 s^2	303.84 s^2	269.84 s^2	366.66 s^2	200.75 s^2
Reduction (%) of variance	19.78%		11.19%		45.25%	
Change (%) in the travel time spread	11.99%		14.32%		14.70%	
Cumulative mission time reduction	7.58% (p-value $= 0.27 \times 10^{-2} < 0.05$)		9.25% (p-value $= 1.93 \times 10^{-5} < 0.05$)		9.89% (p-value $= 2.87 \times 10^{-5} < 0.05$)	

TABLE VI
COMPARISON OF PERFORMANCE WITH AND WITHOUT REPLANNING - CONFIGURATION WITH CORRIDOR

Fleet Size	**10 Vehicles**		**20 Vehicles**		**30 Vehicles**	
Description	**Without Replanning**	**With Replanning**	**Without Replanning**	**With Replanning**	**Without Replanning**	**With Replanning**
Travel time spread	62.28 s	35.67 s	72.33 s	40.82 s	101.24 s	51.37 s
Mean (illustrated in Fig. 7)	110.71 s	78.55 s	121.25 s	81.08 s	152.78 s	92.59 s
Variance (illustrated in Fig. 7)	2,068.36 s^2	362.77 s^2	1,779.74 s^2	479.18 s^2	1,187.66 s^2	555.18 s^2
Reduction (%) of variance	82.46%		73.07%		53.25%	
Change (%) in the travel time spread	42.72%		43.57%		49.25%	
Cumulative mission time reduction	29.05% (p-value $= 1.40 \times 10^{-7} < 0.05$)		33.13% (p-value $= 7.66 \times 10^{-12} < 0.05$)		39.39% (p-value $= 1.78 \times 10^{-14} < 0.05$)	

TABLE VII
COMPUTATIONALTIMES REQUIRED TO DETERMINE THE PATHS FOR THE ENTIRE FLEET DURING A SINGLEREPLANNING STEP

Configuration	**Time**	**10 Vehicles**	**20 Vehicles**	**30 Vehicles**
Corridor-free	Average	0.45	0.84	1.70
	Min-Max	0.01-1.32	0.03-2.49	0.06-3.69
Corridor-based	Average	0.41	0.75	1.27
	Min-Max	0.01-1.23	0.03-1.94	0.07-2.97

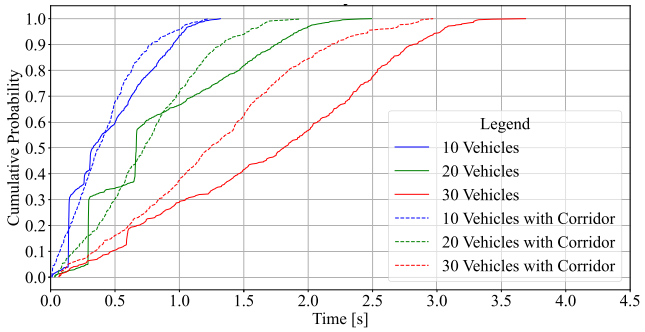


Fig. 8. Cumulative probability of completing the path computation for the entire fleet during a single replanning step, reported for the three scenarios in both the configurations.

shown in Table VII, for fleets of 10, 20, and 30 vehicles, the computational average time is always lower when the corridor is present, and peak values are also notably reduced. Specifically, the computational average time ranges from 0.41 s to 1.27 s in the corridor-based configuration, while reaching up to 1.70 s in the corridor-free case. Moreover, variability –expressed as the min–max range– is significantly smaller in the corridor-based layout, indicating more robust and predictable computational behavior.

The cumulative probability distribution in Fig. 8 further confirms the observed trend. Aggregated across all scenarios, the corridor-based configuration consistently achieves higher probabilities of completing replanning within shorter time intervals compared to the corridor-free setup. Notably, both configurations reach a 100% probability of completing within the 5 s upper bound across all fleet sizes. The steeper slope of the cumulative curves further highlights its higher computational efficiency and reliability as the fleet size increases, confirming that the nominal 5 s threshold can be reduced if required by real-time constraints.

VII. CONCLUSION AND OUTLOOKS

This paper presents a hierarchical PP framework for autonomous vehicle fleets operating in hybrid roadmaps that combine static (graph-based) and dynamic (grid-based) areas for robust and flexible navigation. A bi-level optimization structure integrates multi-objective planning with a quasi-adaptive self-tuning mechanism, which dynamically adjusts

objective weights based on environmental conditions, removing the need for manual tuning. Simulations on two industrial layouts demonstrate significant improvements in total and cumulative mission time, and consistency, particularly for larger fleets, with results statistically validated across all scenarios. Future work will integrate PP with task scheduling and validate the system in real-world scenarios, relaxing the current assumptions on nominal conditions and simplified models to account for vehicle dynamics and uncertainties.

REFERENCES

[1] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues," *J. Ind. Inf. Integr.*, vol. 6, pp. 1–10, Jun. 2017.

[2] R. Chai, D. Liu, T. Liu, A. Tsourdos, Y. Xia, and S. Chai, "Deep learning-based trajectory planning and control for autonomous ground vehicle parking maneuver," *IEEE Trans. Autom. Sci. Eng.*, vol. 20, no. 3, pp. 1633–1647, Jul. 2023.

[3] I. P. Vlachos, R. M. Pascazzi, G. Zobolas, P. Repoussis, and M. Giannakis, "Lean manufacturing systems in the area of industry 4.0: A lean automation plan of AGVs/IoT integration," *Prod. Planning Control*, vol. 34, no. 4, pp. 345–358, Mar. 2023.

[4] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, "Path planning techniques for mobile robots: Review and prospect," *Expert Syst. Appl.*, vol. 227, Oct. 2023, Art. no. 120254.

[5] Y. Zhang and A. Khani, "An algorithm for reliable shortest path problem with travel time correlations," *Transp. Res. B, Methodol.*, vol. 121, pp. 92–113, Mar. 2019.

[6] A. J. Moshayedi, J. Li, and L. Liao, "AGV (automated guided vehicle) robot: Mission and obstacles in design and performance," *J. Simul. Anal. Novel Technol. Mech. Eng.*, vol. 12, no. 4, pp. 5–18, 2019.

[7] A. Faisal, M. Kamruzzaman, T. Yigitcanlar, and G. Currie, "Understanding autonomous vehicles," *J. Transp. Land Use*, vol. 12, no. 1, pp. 45–72, Jan. 2019.

[8] O. Tutsoy, D. Asadi, K. Ahmadi, S. Y. Nabavi-Chashmi, and J. Iqbal, "Minimum distance and minimum time optimal path planning with bioinspired machine learning algorithms for faulty unmanned air vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 8, pp. 9069–9077, Aug. 2024.

[9] Ö. Tutsoy and M. Brown, "Reinforcement learning analysis for a minimum time balance problem," *Trans. Inst. Meas. Control*, vol. 38, no. 10, pp. 1186–1200, 2015.

[10] R. Chai, H. Niu, J. Carrasco, F. Arvin, H. Yin, and B. Lennox, "Design and experimental validation of deep reinforcement learning-based fast trajectory planning and control for mobile robot in unknown environment," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 4, pp. 5778–5792, Apr. 2024.

[11] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 1, no. 1, pp. 187–210, 2018.

[12] R. Chai, A. Tsourdos, S. Chai, Y. Xia, A. Savvaris, and C. L. P. Chen, "Multiphase overtaking maneuver planning for autonomous ground vehicles via a desensitized trajectory optimization approach," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 74–87, Jan. 2023.

[13] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "A survey of path planning algorithms for mobile robots," *Vehicles*, vol. 3, no. 3, pp. 448–468, Aug. 2021.

[14] Á. Madridano, A. Al-Kaff, D. Martín, and A. de la Escalera, "Trajectory planning for multi-robot systems: Methods and applications," *Expert Syst. Appl.*, vol. 173, Jul. 2021, Art. no. 114660.

[15] F. Menebröker, D. Lünsch, M. Hantzsch, and U. Pazarci, "Generalization of a stateful graph search algorithm applied to heterogeneous mobile robot path planning," in *Proc. IEEE 20th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2024, pp. 2201–2208.

[16] V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi, "Ensemble coordination approach in multi-AGV systems applied to industrial warehouses," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 3, pp. 922–934, Jul. 2015.

[17] P. Z. H. Sun et al., "AGV-based vehicle transportation in automated container terminals: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 1, pp. 341–356, Jan. 2023.

[18] D. M. Matos, P. Costa, H. Sobreira, A. Valente, and J. Lima, "Efficient multi-robot path planning in real environments: A centralized coordination system," *Int. J. Intell. Robot. Appl.*, vol. 9, no. 1, pp. 217–244, Mar. 2025.

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

[20] Y. Guo, Z. Guo, Y. Wang, D. Yao, B. Li, and L. Li, "A survey of trajectory planning methods for autonomous driving—Part I: Unstructured scenarios," *IEEE Trans. Intell. Vehicles*, vol. 9, no. 9, pp. 5407–5434, Sep. 2024.

[21] T. Lackner, J. Hermann, C. Kuhn, and D. Palm, "Review of autonomous mobile robots in intralogistics: State-of-the-art, limitations and research gaps," *Proc. CIRP*, vol. 130, pp. 930–935, Jan. 2024.

[22] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Trans. Robot. Autom.*, vol. 14, no. 1, pp. 166–171, Feb. 1998.

[23] R. Stern et al., "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proc. Int. Symp. Combinat. Search*, vol. 10, 2021, pp. 151–158.

[24] M. Fisher, V. Mascardi, K. Y. Rozier, B.-H. Schlingloff, M. Winikoff, and N. Yorke-Smith, "Towards a framework for certification of reliable autonomous systems," *Auto. Agents Multi-Agent Syst.*, vol. 35, no. 1, pp. 1–14, Apr. 2021.

[25] C. F. Hayes et al., "A practical guide to multi-objective reinforcement learning and planning," *Auto. Agents Multi-Agent Syst.*, vol. 36, no. 1, p. 26, Apr. 2022.

[26] F. Matoui, B. Boussaid, B. Metoui, and M. N. Abdelkrim, "Contribution to the path planning of a multi-robot system: Centralized architecture," *Intell. Service Robot.*, vol. 13, no. 1, pp. 147–158, Jan. 2020.

[27] B. Tang, K. Xiang, M. Pang, and Z. Zhanxia, "Multi-robot path planning using an improved self-adaptive particle swarm optimization," *Int. J. Adv. Robotic Syst.*, vol. 17, no. 5, pp. 1–19, Sep. 2020.

[28] F. Gul, W. Rahiman, S. S. N. Alhady, A. Ali, I. Mir, and A. Jalil, "Meta-heuristic approach for solving multi-objective path planning for autonomous guided robot using PSO–GWO optimization algorithm with evolutionary programming," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 7, pp. 7873–7890, Jul. 2021.

[29] A. Hidalgo-Paniagua, M. A. Vega-Rodríguez, J. Ferruz, and N. Pavón, "Solving the multi-objective path planning problem in mobile robotics with a firefly-based approach," *Soft Comput.*, vol. 21, no. 4, pp. 949–964, Feb. 2017.

[30] M.-Y. Zhang, S.-C. Yang, X.-J. Feng, Y.-Y. Chen, J.-Y. Lu, and Y.-G. Cao, "Route planning for autonomous driving based on traffic information via multi-objective optimization," *Appl. Sci.*, vol. 12, no. 22, p. 11817, Nov. 2022.

[31] N. A. Azhar, N. A. M. Radzi, and W. S. H. M. W. Ahmad, "Multi-criteria decision making: A systematic review," *Recent Adv. Elect. Electron. Eng.*, vol. 14, no. 8, pp. 779–801, 2021.

[32] Y. Liao and V. Friderikos, "Energy and age Pareto optimal trajectories in UAV-assisted wireless data collection," *IEEE Trans. Veh. Technol.*, vol. 71, no. 8, pp. 9101–9106, Aug. 2022.

[33] C. Ramirez-Atencia, G. Bello-Orgaz, M. D. R-Moreno, and D. Camacho, "Solving complex multi-UAV mission planning problems using multi-objective genetic algorithms," *Soft Comput.*, vol. 21, no. 17, pp. 4883–4900, Sep. 2017.

[34] R. Chai et al., "A two phases multiobjective trajectory optimization scheme for multi-UGVs in the sight of the first aid scenario," *IEEE Trans. Cybern.*, vol. 54, no. 9, pp. 5078–5091, Sep. 2024.

[35] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: New insights," *Structural Multidisciplinary Optim.*, vol. 41, no. 6, pp. 853–862, Jun. 2010.

[36] F. Pratissoli, R. Brugioni, N. Battilani, and L. Sabattini, "Hierarchical traffic management of multi-AGV systems with deadlock prevention applied to industrial environments," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 3, pp. 3155–3169, Jul. 2024.

[37] Z. Han, D. Wang, F. Liu, and Z. Zhao, "Multi-AGV path planning with double-path constraints by using an improved genetic algorithm," *PLoS ONE*, vol. 12, no. 7, Jul. 2017, Art. no. e0181747.

[38] J. Choi, R. E. Curry, and G. H. Elkaim, "Continuous curvature path generation based on Bézier curves for autonomous vehicles," *IAENG Int. J. Appl. Math.*, vol. 40, no. 2, pp. 91–101, May 2010.

[39] G. Yu and J. Yang, "On the robust shortest path problem," *Comput. Operations Res.*, vol. 25, no. 6, pp. 457–468, Jun. 1998.

[40] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[41] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Feb. 1968.

[42] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: From classical to evolutionary approaches and applications," *IEEE Trans. Evol. Comput.*, vol. 22, no. 2, pp. 276–295, Apr. 2018.
[43] T. Qiuyun, S. Hongyan, G. Hengwei, and W. Ping, "Improved particle swarm optimization algorithm for AGV path planning," *IEEE Access*, vol. 9, pp. 33522–33531, 2021.
[44] S. Liu, Q. Lin, J. Li, and K. C. Tan, "A survey on learnable evolutionary algorithms for scalable multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 27, no. 6, pp. 1941–1961, Dec. 2023.
[45] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: Past, present, and future," *Multimedia Tools Appl.*, vol. 80, no. 5, pp. 8091–8126, Feb. 2021.
[46] X. Wang, Y. Jin, S. Schmitt, and M. Olhofer, "Recent advances in Bayesian optimization," *ACM Comput. Surveys*, vol. 55, no. 13s, pp. 1–36, Dec. 2023.
[47] S. Kang, K. Li, and R. Wang, "A survey on Pareto front learning for multi-objective optimization," *J. Membrane Comput.*, vol. 7, no. 2, pp. 128–134, Jun. 2025.
[48] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia, and C. L. P. Chen, "Design and implementation of deep neural network-based control for automatic parking maneuver process," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 4, pp. 1400–1413, Apr. 2022.
[49] L. Sabattini, C. Secchi, and N. Chopra, "Decentralized estimation and control for preserving the strong connectivity of directed graphs," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2273–2286, Oct. 2015.
[50] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2022.
[51] B. Alhijawi and A. Awajan, "Genetic algorithms: Theory, genetic operators, solutions, and applications," *Evol. Intell.*, vol. 17, no. 3, pp. 1245–1256, Jun. 2024.
[52] R. Martí, "Multi-start methods," in *Handbook of Metaheuristics*. Cham, Switzerland: Springer, Oct. 2003, pp. 211–230.
[53] T. Park and K. R. Ryu, "A dual-population genetic algorithm for adaptive diversity control," *IEEE Trans. Evol. Comput.*, vol. 14, no. 6, pp. 865–884, Dec. 2010.



Marianna Turrà (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. (cum laude) degrees in management engineering from the University of Modena and Reggio Emilia, Italy, in 2022 and 2024, respectively, where she is currently pursuing the Ph.D. degree in industrial innovation engineering. Her research activity focuses on the development of optimization models for the management of heterogeneous vehicle fleets operating in complex industrial environments.



Silvia Proia (Member, IEEE) received the double M.Sc. degree (Hons.) in mechanical engineering with major in mechatronics and robotics from the Polytechnic University of Bari, Italy, and NYU Tandon School of Engineering, USA, in 2020, and the Ph.D. degree in Industry 4.0 from the Polytechnic University of Bari, in 2024. She is currently a Post-Doctoral Researcher with the Department of Science and Methods for Engineering, University of Modena and Reggio Emilia, Italy. Her research activity focuses on decision, modeling, and control of complex systems, with a particular emphasis on collaborative and cooperative robotic systems, human–robot and human–drone interaction, and multi-agent systems. She is/was an Associate Editor of major IEEE conferences, including IEEE ICRA, IEEE IROS, and IEEE SMC (2024–2026).



Alessandro Bonetti (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in mechatronics engineering from the University of Modena and Reggio Emilia, Italy, in 2019 and 2022, respectively, where he is currently pursuing the Ph.D. degree in industrial innovation engineering.

He has been a Visiting Scholar with the National University of Singapore (NUS), Singapore. He is currently working as a Research and Development Software Developer with Gruppo Tecnoferrari S.p.A. His research interests include multi-agent pathfinding applied to mobile robotics and traffic management of AGV fleets.



Lorenzo Sabattini (Senior Member, IEEE) received the Ph.D. degree in control systems and operational research from the University of Bologna, Italy. He is currently an Associate Professor with the Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Italy. His research interests include multi-robot systems, decentralized estimation and control, and human–robot interaction. He leads the Multi-Robot Systems Research Group, ARSControl Laboratory, and has coordinated several national and European research projects, including Horizon Europe and PRIN initiatives. He has served in editorial roles for leading international journals and conferences in robotics and automation.

Open Access funding provided by 'Università degli Studi di Modena e Reggio Emilia' within the CRUI CARE Agreement