Control Engineering Practice 175 (2026) 107135

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

Structure-aware constraint configuration for distributed online trajectory generators: An application to mobile robot navigation

Koen de Vos [a,*], Tom Minten [a], Herman Bruyninckx [a,b,c], René van de Molengraft [a], Dinesh Krishnamoorthy [a,d], Elena Torta [a]

[a] Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands
[b] Department of Mechanical Engineering, KU Leuven, Leuven, Belgium
[c] Flanders Make, Leuven, Belgium
[d] Department of Engineering Cybernetics, NTNU, Trondheim, Norway

ARTICLE INFO

Keywords:
Multi-robot navigation
Knowledge representation
Motion planning

ABSTRACT

We present a framework for the automatic configuration of constraints in a disturbed model predictive control (MPC) formulation for multi-agent trajectory generation, leveraging structural knowledge encoded in a robot world model. Agent routes are planned as sequences of areas to be traversed, enabling the systematic selection of relevant environmental and inter-agent constraints through queries of the world model. The introduction of sub-areas, such as lanes, further reduces computational complexity by limiting the activation of coupling constraints. In addition, the framework supports reasoning over discrete actions, such as yielding to other agents, allowing agents to satisfy high-level task and environmental constraints within the continuous MPC formulation. The proposed framework is validated in both simulation and hardware-in-the-loop experiments using a distributed computing setup. The results demonstrate improved task execution performance while significantly reducing computation time when compared with baseline distributed formulations.

1. Introduction

Multi-robot navigation in structured environments, such as corridors, and intersections found in warehouses, farms, and factories, requires agents to generate collision-free trajectories in real time while coordinating with one another. A widely used approach for this is Model Predictive Control (MPC), which formulates trajectory generation as a constrained optimization problem over a finite horizon, balancing goal-directed progress with reactive adaptation to disturbances (Ferranti et al., 2018; Van Parys & Pipeleers, 2016; de Vos et al., 2025b). In multi-robot settings, MPC serves as the local planner within the standard navigation hierarchy (Siegwart et al., 2011): a global planner assigns routes, a local planner reactively generates dynamically feasible trajectories, and a low-level motion controller executes the resulting setpoints.

When all agents are planned for jointly, a coupled approach commonly used in multi-agent path finding (MAPF), planners can reason explicitly about inter-agent interactions. However, this comes at the cost of scalability: computational requirements grow rapidly with the number of agents (Stern et al., 2019), and long-horizon coupled plans are sensitive to small disturbances that can trigger costly replanning. The framework proposed in this paper therefore falls within the decoupled class, where each agent plans independently and coordination is handled at runtime. In this setting individual trajectory problems are more tractable, however, distributed solvers such as the Alternating Direction Method of Multipliers (ADMM) must iteratively exchange information across agents to reach consensus, requiring high communication bandwidth and low-latency channels to remain feasible in real time. To reduce the requirements on the communication channel, the proposed framework uses structured knowledge of the environment, represented as a world model of areas, lanes, and intersections, to determine, at each planning step, which inter-agent coupling constraints are necessary. Constraints between agents that are spatially separated or travelling in non-conflicting lanes are omitted; discrete decisions such as yielding at intersections replace continuous coupling constraints where applicable. As illustrated in Fig. 1c, this reduces the complexity of the distributed coordination problem, which can improve task-level performance under a limited ADMM iteration budget, or reduce the computational effort required for coordination. The remaining panels of Fig. 1 further motivate the role of discrete decision-making. Fig. 1a shows how a yielding decision prevents a resource deadlock in which agents would

* Corresponding author.
E-mail addresses: k.d.vos@tue.nl (K. de Vos), t.p.minten@tue.nl (T. Minten), herman.bruyninckx@kuleuven.be (H. Bruyninckx), M.J.G.v.d.Molengraft@tue.nl (R. van de Molengraft), dinesh.krishnamoorthy@ntnu.no (D. Krishnamoorthy), e.torta@tue.nl (E. Torta).

https://doi.org/10.1016/j.conengprac.2026.107135
Received 29 January 2026; Received in revised form 29 May 2026; Accepted 21 June 2026
Available online 26 June 2026
0967-0661/© 2026 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

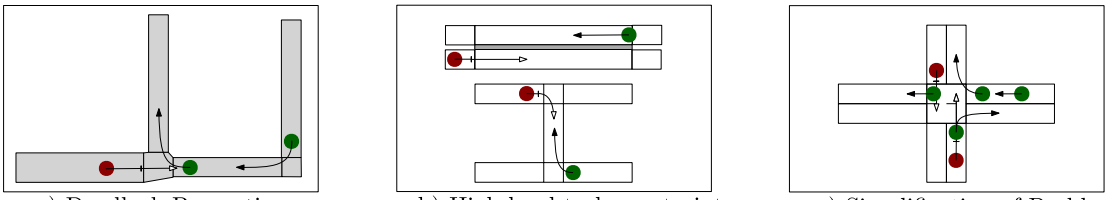

Fig. 1. Motivation for the introduction of discrete actions in the decision making framework of the agent. Yielding agents are illustrated with a bar-terminated line, where the white arrowhead indicates the continuation after yielding, non-yielding agents are illustrated with a conventional arrow; a) Deadlock prevention prohibits the entering of areas in which agents are unable to pass and can thus get stuck; b) High level task constraints can prohibit entering areas in which other agents are present; c) Problem simplification might result in lower communication overhead and computation needs when many agents are present.

mutually block access to a shared area. Fig. 1b illustrates the enforcement of high-level task constraints, such as prohibiting simultaneous bidirectional passage or preventing entry into an area already occupied by an oncoming agent. Together, these scenarios demonstrate that effective multi-robot navigation in structured environments could benefit from a framework that combines continuous trajectory optimisation with discrete, context-aware coordination, supported by a structured knowledge representation. The remainder of this paper is structured as follows. Section 1.1 reviews related work. Section 2 presents the proposed framework. Section 3 describes the validation methodology. Section 3.4 presents the experimental results. Finally, Section 4 concludes the paper.

1.1. Related work

Planning approaches for multi-robot systems are often categorized into coupled and decoupled methods (LaValle, 2006; Zhu, 2022). Coupled methods globally plan the trajectories of all agents jointly, optimizing over the full state of the multi-robot system. By reasoning over long horizons and accounting for interactions between agents along that horizon, such methods can provide guarantees of completeness and optimality, and reason on the interactions between agents over long horizons. However, this comes at the cost of scalability. As the number of agents or the complexity of the environment increases, computational requirements grow rapidly (Stern et al., 2019). In addition, because coupled plans are optimized over long horizons, small unpredicted disturbances or execution delays early in execution can propagate through the plan and compound over the length of the horizon. Planning methods assume that robots have been assigned goals and organised into ordered sequences of locations, a problem collectively referred to as multi-robot task allocation (MRTA) (Bai et al., 2024; Gerkey & Matarić, 2004). This work similarly assumes such assignments are provided as input. Decoupled planning methods plan routes for each agent independently, either as geometric paths (van den Berg & Overmars, 2005) or at a topological level as sequences of areas representing intersections and corridors (Roszkowska et al., 2023; de Vos et al., 2024). Topological representations reduce the search space and leave greater flexibility in execution, making plans less sensitive to disturbances and modelling inaccuracies (Fredriksson et al., 2025). Since decoupled methods do not account for inter-agent interactions explicitly, they rely on conflict resolution at runtime to prevent collisions. Reactive methods for conflict resolution have been widely studied (e.g. (van den Berg et al., 2008; Fiorini & Shiller, 1998; Fox et al., 1997)). These approaches prioritize computational efficiency by reasoning over short time horizons, typically based on the agent's current state, local sensor information, and the immediate effect of candidate actions. While effective in certain scenarios, including hardware implementations (Fox et al., 1997), reactive methods such as artificial potential fields are known to suffer from local minima and trap situations. Addressing these limitations generally requires access to more global information or the integration of additional planning mechanisms (Koren & Borenstein, 1991; Rimon & Koditschek, 1992). Model Predictive Control (MPC) is a well-established approach for runtime conflict resolution in multi-agent navigation (Ferranti et al., 2018; Firoozi et al., 2024; Mercy et al., 2016; de Vos et al., 2025a), combining reactivity through its receding-horizon scheme with a limited look-ahead that allows the planner to anticipate future interactions. Centralized MPC formulations, in which a single node optimises the joint trajectory of all agents, do not scale well as the number of agents grows. Distributing the computation can be achieved through higher-level reasoning, as in de Vos et al. (2025a) where area-occupancy constraints decouple the individual MPC problems, or through distributed optimization techniques such as ADMM (Shorinwa et al., 2024). The area-occupancy approach yields independent subproblems but produces conservative trajectories, since no two agents may occupy overlapping areas simultaneously. ADMM, by contrast, maintains coupled constraints and targets the solution quality of the centralised formulation, at the cost of iterative communication between agents and a coordinator. The use of ADMM for multi-agent trajectory generation has been studied in Ferranti et al. (2023, 2018), Van Parys and Pipeleers (2016), covering applications ranging from vessel coordination at canal intersections to spline-based trajectory optimisation in dynamic environments. However, especially when there is a large set of agents, when communication bandwidth is limited, or the latency of the communication channel is large, the latency of the communication channel can have a significant effect on either the solve time or the achievable solution quality. Reducing the communication requirements of ADMM-based methods has also been explored: Serra-Gómez et al. (2023) learns a communication policy using reinforcement learning, while Best et al. (2018) triggers communication only when uncertainty about a neighboring agent's actions exceeds a threshold. Within robotics, world models (WM) are internal knowledge representations that contain information about the environment in which a robot operates and that is relevant to the execution of its tasks (Sakagami et al., 2023). Semantic world models represent environments at a higher level of abstraction than geometric maps by explicitly encoding entities and their relationships (Crespo et al., 2020; Sakagami et al., 2023). This high-level knowledge enables navigation methods to reason about regions and interactions in the environment in a context-aware manner. The authors of McBeth et al. (2023) propose a motion planning approach in which the planner is guided by a topological skeleton of the environment, biasing exploration towards structurally relevant regions such as corridors and intersections. The authors of Wang and Lin (2026) propose a reinforcement learning-based approach to MAPF that exploits a topological representation of the environment. The authors of Fredriksson et al. (2025) demonstrated that structural-semantic maps can simplify multi-agent path finding problems.

1.2. Contributions

This paper presents a knowledge-informed ADMM-based distributed MPC framework for multi-agent trajectory generation in structured environments. The framework uses a structured world model of areas, lanes, and intersections to automatically configure inter-agent coupling constraints and discrete coordination actions at each planning step, reducing the computational and communication demands of the

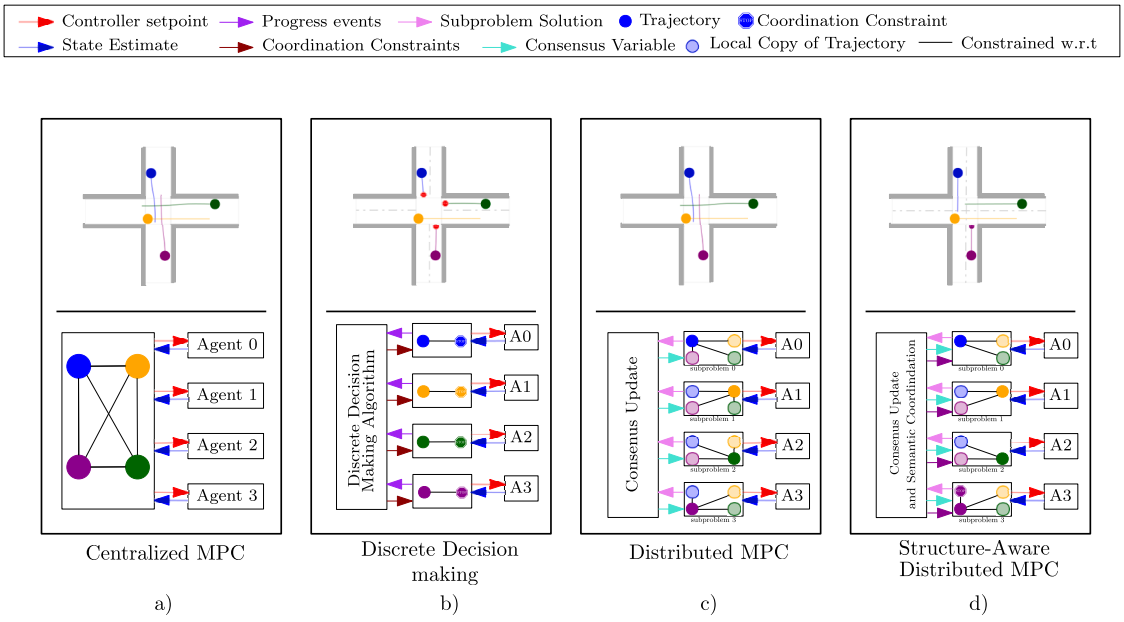

Fig. 2. Illustration of compared methods.

underlying distributed optimisation. The contributions of this paper are as follows. First, we present a systematic method for configuring the constraint structure of an ADMM-based MPC formulation using a strucutred world model. By omitting coupling constraints between agents operating in non-conflicting areas or lanes, the method reduces the complexity of the coordination problem and, consequently, the computational effort required to obtain satisfactory task-level performance. Second, we integrate discrete decision-making into the ADMM coordination mechanism, enabling agents to reason about high-level actions such as yielding at intersections. This hybrid formulation allows the system to prevent resource deadlocks and enforce area-occupancy task constraints without resorting to the conservative motion behavior of purely discrete coordination approaches. Third, we demonstrate through simulation and hardware-in-the-loop experiments on distributed compute nodes that the proposed framework reduces computation time and improves task execution performance compared to conventional ADMM baselines, confirming its feasibility for deployment on embedded hardware.

2. Method

The proposed framework augments a consensus ADMM-based distributed MPC formulation with a knowledge-informed configuration layer that determines, at each planning step, which inter-agent coupling constraints are active. Agents operate within a structured environment representation, organised into areas, lanes, and intersections, and the configuration layer queries this representation to assess whether a given pair of agents requires an explicit coordination constraint. When coordination is not required, the corresponding coupling constraint is omitted; when a discrete action such as yielding is applicable, it is enforced through a priority rule that replaces the continuous coupling constraint. Fig. 2 illustrates the four formulations discussed in this paper: centralised MPC (a), discrete decision-making with decoupled MPC (b), standard ADMM-based distributed MPC (c), and the proposed knowledge-informed ADMM formulation (d).

The principal advantages of this approach over a standard ADMM formulation are threefold. First, by omitting coupling constraints between agents that are sufficiently separated or operating in non-conflicting lanes, the framework reduces the number of constraints that must be enforced in each local subproblem. Second, because fewer consensus constraints are present, the number of ADMM iterations needed to reach a feasible solution is reduced, lowering both computation time and communication overhead per planning cycle. Third, these reductions in per-iteration cost enable the deployment of the ADMM scheme in larger-scale environments, in terms of number of agents, where a standard ADMM formulation would fail to converge within the available computation and communication budget. In addition, the discrete decision-making layer allows the framework to satisfy higher-level occupancy constraints, such as deadlock prevention and area-exclusivity rules, that cannot be expressed directly within common continuous MPC formulations.

2.1. Centralized MPC formulation

The decentralized MPC formulation is derived from the centralized problem. Therefore, in this section, we first present the centralized formulation. In a centralized MPC setting, the states and inputs of the system, which are the states $\mathbf{x}_i$ and inputs $\mathbf{u}_i$ of all agents $a_i \in \mathcal{A}$, are jointly optimized over a given finite prediction horizon N, given a shared objective J and set of constraints. The optimization problem is formulated as a single, centralized problem that cannot be directly decomposed per agent due to the presence of non-collision constraints between the trajectories of each agent. Consequently, it must be solved by a single solver, typically executed on a single processing unit. Specifically, in this case, we consider the following optimization problem:

$$\min_{\mathbf{x},\mathbf{u}} \quad J = \sum_{i\in\mathcal{A}} \sum_{k=0}^{N-1} J_{i,k}(\mathbf{x}_i(k), \mathbf{u}_i(k)), \tag{1a}$$

$$\text{s.t.} \quad \mathbf{x}_i(k+1) = \mathbf{f}_i\big(\mathbf{x}_i(k), \mathbf{u}_i(k)\big), \quad \forall i \in \mathcal{A},\ k = 0, \dots, N-1, \tag{1b}$$

$$\mathbf{x}_i(k) \in \mathcal{X}_i, \quad \mathbf{u}_i(k) \in \mathcal{U}_i, \quad \forall i \in \mathcal{A},\ k = 0, \dots, N-1, \tag{1c}$$

$$\mathbf{x}_i(0) = \mathbf{x}_{i,\text{init}}, \qquad \forall i \in \mathcal{A}, \tag{1d}$$

$$\delta_{i,j}\ \operatorname{dist}\big(\mathbf{x}_i(k), \mathbf{x}_j(k)\big) \geq 0, \quad \forall i, j \in \mathcal{A},\ i \neq j,\ k = 0, \dots, N. \tag{1e}$$

in which $\mathcal{X}_i$ and $\mathcal{U}_i$ denote the admissible (safe) state and input spaces of agent a_i, respectively; $\mathbf{x}_{i,\text{init}}$ represents the initial (i.e., current) state of agent a_i; $\mathbf{f}_i$ the dynamical model of agent a_i; $\delta_{i,j} \in \{0, 1\}$ a binary switch

variable for disabling the inter-agent constraint between agents a_i and a_j, note that $\delta_{i,j}$ is not a decision variable but a parameter set at the task level. Furthermore, dist$(\cdot,\cdot)$ denotes a function that computes the distance the states along the trajectories of two agents. The admissible state and input spaces of the agent result from the task description and physical limitations of the agents. For example, $\mathcal{X}_i$ may be constrained by the areas and obstacles defined in the world model; $\mathcal{U}_i$ results from the actuation limits of the agent. Since it is assumed that agents can be modeled as having circular footprints, the inter-agent no-collision constraint is expressed in terms of the Euclidean distance between the center positions of the agents. More specifically, the constraint is expressed as:

$$\text{dist}\big(\mathbf{x}_i(k),\mathbf{x}_j(k)\big) = ||\mathbf{G}x_i - \mathbf{G}x_j||_2 - r_i - r_j \geq 0 \tag{2}$$

with $\mathbf{G}$ a selection matrix that selects the position components from the state vector, and r_i the radius of agent a_i.

Constraint representations

The presented MPC framework is general with respect to the chosen representations of objectives and constraints. Two key considerations in selecting a representation are the resulting computational complexity and the expressiveness of the formulation. In this work, and throughout the experimental validation, we focus primarily on multi-agent navigation problems that arise in corridor and intersection-crossing scenarios. Consequently, we consider two types of constraint formulation: a polytopic representation of the corridor constraints and the separating hyperplane-based formulation. A polytopic representation describes feasible regions as convex polygons, defined by the intersection of a finite set of linear inequalities. In the context of multi-agent navigation, corridors and motion zones can be encoded as convex polytopes, where each edge of the polytope corresponds to a linear constraint that bounds the agent's position. For the polytopic case, we define the admissible (safe) state space of agent a_i as

$$\mathcal{X}_i \;=\; \Big\{ x \in \mathbb{R}^2 | a_{i,j}^\top x \leq b_{i,j},\; j = 1,\dots,M_i \Big\}, \tag{3}$$

where each pair $(a_{i,j}, b_{i,j})$ encodes one boundary of the corridor. The polytopic representation's main advantage is the relatively low complexity of both implementation and computation, since the constraints can be implemented as linear constraints in NLP or QP solvers. However, this comes at the cost of expressiveness: because linear constraints enforce convex feasible regions, not all relevant navigation scenarios can be captured using a purely polytopic representation. For example, executing a right turn at an intersection typically requires traversing the union of multiple convex regions, and this union is not (necessarily) convex. Such scenarios could be accommodated by dynamically switching between constraint sets at runtime. However, this mechanism is inherently brittle, as unforeseen disturbances may cause agents to deviate from the expected area transition times, leading to incorrect constraint areas and, consequently, a risk of infeasibility in the optimization problem. A second approach to constraint representation is the separating-hyperplane formulation. Rather than explicitly describing the admissible safe state space, this approach introduces constraints that enforce a minimum distance between the agent and surrounding obstacles, which are modeled as convex shapes. By composing multiple of such obstacles constraints, it becomes possible to construct closed areas. An important advantage of this approach is that it enables the construction of non-convex admissible state spaces, e.g. the right turn scenario explained above. Although each individual obstacle is convex, their combined effect can result in non-convex free spaces for the agent to navigate, as demonstrated in, e.g. (de Vos et al., 2025a).

The main disadvantages of the approach are two-fold. Firstly, the formulation requires constructing a separating hyperplane between the agent a_i, commonly approximated by a circular footprint of radius r_i, and an obstacle $o \in \mathcal{O}$, represented as a convex polygon with vertices $\mathcal{V}_o$. This is enforced through the following constraints:

$$\mathbf{a}_{o,k}^\top \mathbf{v}_{o,j} - \mathbf{b}_{o,k} \geq 0, \qquad \forall\, k \in [0,N],\ \forall\, \mathbf{v}_{o,j} \in \mathcal{V}_o, \tag{4a}$$

$$\mathbf{a}_{o,k}^\top \mathbf{G}\mathbf{x}_k - b_{o,k} \leq -r_i, \qquad \forall\, k \in [0,N], \tag{4b}$$

$$\|\mathbf{a}_{o,k}\|_2 \leq 1, \qquad \forall\, k \in [0,N]. \tag{4c}$$

where $\mathbf{a}_{o,k}$ and $\mathbf{b}_{o,k}$ denote, respectively, the normal vector and offset of the separating hyperplane for obstacle o at timestep k. The enforcement of these constraints requires the introduction of additional optimization variables, namely three additional scalar decision variables for every agent-obstacle pair at each timestep. Furthermore, the formulation introduces a quadratic, rather than linear, inequality constraint. Both observations lead to a larger optimization problem, in terms of the number of decision variables, and a more complex one to solve when compared to the polytopic formulation. In addition to that, observations show that the resulting problem is less robust and more sensitive to the choice of discretization timestep. Larger time steps can lead to infeasibility, as the solver may converge to a solution in which the agent's continuous trajectory intersects the obstacle, while the discretized states lie outside of the obstacle region.

Therefore, we employ a combination of the two constraint formulations. A polytopic representation is used to describe convex, structured regions such as corridors and lanes, due to its low computational complexity and stability. To capture more complex, non-convex geometries, such as intersections, turns, and generic obstacles , we augment this representation with a separating hyperplane-based formulation. This hybrid approach allows the framework to retain computational efficiency in structured regions while maintaining sufficient expressiveness to handle non-convex constraints when required.

2.2. Distributed MPC formulation

Starting from the centralized MPC formulation, we derive an equivalent distributed formulation. Concretely, in such a distributed MPC scheme, the centralized optimization problem is decomposed into a set of local subproblems that are solved by distributed computation nodes, e.g. the agent's onboard computation hardware. Although intuitively simple, a complicating factor when considering the optimization problem in (1) is the coupling that arises from the non-collision constraints in (1e). Decomposing the problem naively into $|\mathcal{A}|$ subproblems that are solved independently, thus removing the non-collision constraint coupling, would result in trajectories that intersect one another.

To decompose the optimization problem while still enforcing and reaching consensus over the non-collision constraints, we employ the Alternating Direction Method of Multipliers (ADMM). At a high level, ADMM solves local subproblems that arise from decomposing the centralized optimization problem independently and in parallel. This is achieved through introducing local copies of the coupled variables in the local optimization problems that are solved in parallel by the agents. The agents share their local solutions with a central coordinator, which updates and redistributes a consensus variable. The consensus variable keeps track of the current estimate of the optimal solution of these coupled variables. Each agent then updates its local problem to minimize deviation from this consensus, encouraging agreement across the agents' local subproblems. Through an iterative process, the distributed optimization algorithm converges to a solution.

To derive the distributed optimization scheme, consider the following equivalent reformulation of the centralized optimization problem in (1a)-(1e):

$$\min_{\mathbf{x},\mathbf{u}} \quad J = \sum_{i\in\mathcal{A}} \sum_{k=0}^{N-1} J_{i,k}(\mathbf{x}_i(k),\mathbf{u}_i(k)), \tag{5a}$$

$$\text{s.t.} \quad \mathbf{x}_i \in \bar{\mathcal{X}}_i, \quad \mathbf{u}_i \in \bar{\mathcal{U}}_i, \qquad \forall i \in \mathcal{A}, \tag{5b}$$

$$\delta_{i,j}\ \text{dist}\big(\mathbf{x}_i(k),\mathbf{x}_j(k)\big) \geq 0, \quad \forall\, i,j \in \mathcal{A},\ i \neq j,\ \ k = 0,\dots,N. \tag{5c}$$

in which, for ease of notation, $\bar{\mathcal{X}}$ and $\bar{\mathcal{U}}$ capture the original state and input constraints $\mathcal{X}, \mathcal{U}$, the dynamics, hardware limits, and initial conditions. The main insight in the derivation of the ADMM scheme is the

introduction of local copies $\mathbf{q}_{ij}$ and global consensus variables $\mathbf{z}_i$ for the trajectories of agents $a_i \in \mathcal{A}$, which results in the following equivalent formulation:

$$\min_{\mathbf{x},\mathbf{u},\mathbf{q},\mathbf{z}} \quad J = \sum_{i\in\mathcal{A}} \sum_{k=0}^{N-1} J_{i,k}(\mathbf{x}_i(k), \mathbf{u}_i(k)), \tag{6a}$$

$$\text{s.t.} \quad \mathbf{x}_i \in \bar{\mathcal{X}}_i, \quad \mathbf{u}_i \in \bar{\mathcal{U}}_i, \quad \forall i \in \mathcal{A},\ k = 0, \dots, N-1, \tag{6b}$$

$$\delta_{i,j}\ \mathrm{dist}\big(\mathbf{x}_i(k), \mathbf{q}_{ij}(k)\big) \geq 0, \quad \forall\, i, j \in \mathcal{A},\ i \neq j,\ \ k = 0, \dots, N. \tag{6c}$$

$$\mathbf{q}_{ij} = \mathbf{z}_j, \quad \mathbf{G}_j \mathbf{x}_j = \mathbf{z}_j. \tag{6d}$$

By relaxing the coupling constraints (6d) and moving them into the objective through the definition of an augmented Lagrangian, a quadratic penalty on the disagreement between the local copies of the trajectories and the global consensus variables is introduced. As the coupling constraint is no longer strictly enforced, the overall optimization problem decomposes into $|\mathcal{A}|$ independent subproblems per agent for fixed values of the dual variables $\mathbf{y}$ and the consensus variable $\mathbf{z}$:

$$\begin{aligned}\min_{\mathbf{x},\mathbf{u},\mathbf{q},\mathbf{z}} \ \mathcal{L}_p = &\sum_{i\in\mathcal{A}} \sum_{k=0}^{N-1} J_{i,k}\big(\mathbf{x}_i(k), \mathbf{u}_i(k)\big) \\ &+ \mathbf{y}^\top\big(\mathbf{h}_i(\mathbf{x}_i, \mathbf{q}_{ij}) - \mathbf{z}\big) \\ &+ \tfrac{1}{2}\rho \left\|\mathbf{h}_i(\mathbf{x}_i, \mathbf{q}_{ij}) - \mathbf{z}\right\|_2^2,\end{aligned} \tag{7a}$$

$$\text{s.t.} \quad \mathbf{x}_i \in \bar{\mathcal{X}}_i, \quad \mathbf{u}_i \in \bar{\mathcal{U}}_i, \quad \forall i \in \mathcal{A},\ k = 0, \dots, N-1, \tag{7b}$$

$$\delta_{i,j}\ \mathrm{dist}\big(\mathbf{x}_i, \mathbf{q}_{ij}\big) \geq 0, \quad \forall\, i, j \in \mathcal{A},\ i \neq j,\ \ k = 0, \dots, N. \tag{7c}$$

In this formulation, ρ configures the trade-off between reaching consensus on the coupling constraint and achieving optimality on the local objectives. The local copies $\mathbf{q}_{ij}$ and the agent trajectories $\mathbf{x}_i$ are grouped into the concise notation $\mathbf{h}_i(\mathbf{x}_i, \mathbf{q}_{ij})$, defined as

$$\mathbf{h}_i(\mathbf{x}_i, \mathbf{q}_{ij}) = \big[\mathbf{h}_{i,0}^\top, \mathbf{h}_{i,1}^\top, \dots, \mathbf{h}_{i,|\mathcal{A}|}^\top\big]^\top, \tag{8}$$

with

$$\mathbf{h}_{i,j} = \begin{cases} \mathbf{G}_i \mathbf{x}_i, & \text{if } i = j, \\ \mathbf{q}_{ij}, & \text{otherwise.} \end{cases} \tag{9}$$

Given the expression for the augmented Lagrangian, the corresponding ADMM-based optimization scheme can be formulated and is summarized in Algorithm 1.

The first step, outlined in line 13, minimizes the augmented Lagrangian $\mathcal{L}_p$ with respect to the local variables $\mathbf{x}_i, \mathbf{u}_i$, and $\mathbf{q}_{ij}$, for a given value $\mathbf{z}^r$ for the global variable, thereby solving the local subproblems independently for each agent in parallel. In the second step, in line 16, each agent updates its dual variables $\mathbf{y}_i$ via gradient ascent, based on the results from optimizing the local subproblems in the first step:

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \rho\ \nabla_{\mathbf{y}} \mathcal{L}_p(\mathbf{x}_i^{k+1}, \mathbf{u}_i^{k+1}, \mathbf{q}_{ij}^{k+1}, \mathbf{y}, \mathbf{z}^k), \tag{10}$$

$$= \mathbf{y}^k + \rho\ \big(\mathbf{h}_i(\mathbf{x}_i^{k+1}, \mathbf{q}_{ij}^{k+1}) - \mathbf{z}^k\big). \tag{11}$$

Finally, in the third step in line 19, all agents communicate their updated trajectories and dual variables to the central coordinator, which computes the new consensus variables $\mathbf{z}$ by solving:

$$\begin{aligned}\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} &\sum_{i\in\mathcal{A}} \mathbf{y}_i^{k+1^T}\Big(\mathbf{h}_i(\mathbf{x}_i^{k+1}, \mathbf{q}_{ij}^{k+1}) - \mathbf{z})\Big) \\ &+ \tfrac{1}{2}\rho\ ||\mathbf{h}_i(\mathbf{x}_i^{k+1}, \mathbf{q}_{ij}^{k+1}) - \mathbf{z}||_2^2\end{aligned} \tag{12}$$

which can be shown to be equivalent to the closed-form update rule

$$\mathbf{z}^{k+1} = \frac{1}{\rho\ |\mathcal{A}|} \sum_{i\in\mathcal{A}} \big(\mathbf{y}_i^{k+1} + \rho\ \mathbf{h}_i(\mathbf{x}_i^{k+1}, \mathbf{q}_{ij}^{k+1})\big). \tag{13}$$

Algorithm 1 Structure-informed alternating direction method of multipliers (ADMM) with discrete yielding actions (DA).

Require: Initial guesses: $\mathbf{x}_i^0, \mathbf{u}_i^0,\ \mathbf{q}_{ij}^0,\ \mathbf{z}^0,\ \mathbf{y}_i^0$
Require: Inter-agent Constraint set: $\mathbf{c}$
Require: Agent plans, index: $(\mathbf{p}_i, k_i) \quad \forall a_i \in \mathcal{A}$
1: $\boldsymbol{\delta} = 0$
2: **for all** $c = (a_i, a_j) \in \mathbf{c}$ **do**
3: $\quad \delta_{i,j} \leftarrow 1$
4: *Environmental constraint configuration*
5: $\mathbf{N} \leftarrow$ Compute_Area_Hor.$(\mathbf{p}, \mathbf{k})$ ▷ Algorithm 4
6: **for all** $a_i \in \mathcal{A}$ **do**
7: $\quad \mathcal{X}_i \leftarrow$ Select Constraints$(\mathbf{p}_i, k_i, N_i)$ ▷ Algorithm 2
8: *ADMM Loop (up to $k_{\max}$ iterations)*
9: **for** $k = 0 : k_{\max}$ **do**
10: $\quad$ **for all** $a_i \in \mathcal{A}$ **do**
11: $\quad\quad$ **Local primal update:**
12: $\quad\quad \mathbf{x}_i^{k+1}, \mathbf{u}_i^{k+1}, \mathbf{q}_{ij}^{k+1} \leftarrow$
13: $\quad\quad \arg\min_{\mathbf{x}_i,\mathbf{u}_i,\mathbf{q}_{ij}} \mathcal{L}_p(\mathbf{x}_i, \mathbf{u}_i, \mathbf{q}_{ij}, \mathbf{y}_i^k, \mathbf{z}^k)$
14: $\quad\quad$ **Dual update:**
15: $\quad\quad \mathbf{y}_i^{k+1} \leftarrow \mathbf{y}^k +$
16: $\quad\quad \rho\ \nabla_{\mathbf{y}} \mathcal{L}_p(\mathbf{x}_i^{k+1}, \mathbf{u}_i^{k+1}, \mathbf{q}_{ij}^{k+1}, \mathbf{y}, \mathbf{z}^k)$
17: $\quad$ **Consensus update:**
18: $\quad\quad \mathbf{z}_i^{k+1} \leftarrow$
19: $\quad \arg\min_{\mathbf{z}} \mathcal{L}_p(\mathbf{x}_i^{k+1}, \mathbf{u}_i^{k+1}, \mathbf{q}_{ij}^{k+1}, \mathbf{y}^{k+1}, \mathbf{z})$
20: $\quad$ **Check Convergence:**
21: $\quad (r_k, s_k) \leftarrow$ compute residuals (14) and (15)
22: $\quad$ **if** $r^k < r_c$ and $s^k < s_c$ **then**
23: $\quad\quad$ **break**
24: *Structure-informed reconfiguration*
25: **Determine current area for each agent**
26: **for all** $a_i \in \mathcal{A}$ **do**
27: $\quad k_i \leftarrow$ CurrentArea$(\mathbf{x}_i^r)$
28: **Update inter-agent constraints**
29: $\mathbf{c} \leftarrow$ Inter-agent constraint selection$(\mathbf{p},\ \mathbf{k},\ \mathbf{N})$ ▷ Algorithm 3
30: **Detect changes in constraint set**
31: $C_{\text{add}} \leftarrow \tilde{\mathbf{c}} \setminus \mathbf{c}^{\text{old}}, C_{\text{rem}} \leftarrow \mathbf{c}^{\text{old}} \setminus \tilde{\mathbf{c}},$
32: $C_{\text{change}} \leftarrow C_{\text{add}} \cup C_{\text{rem}}$
33: **for all** $c = (a_i, a_j) \in C_{\text{change}}$ **do**
34: $\quad \delta_{ij} \leftarrow [\, c \in C_{\text{add}}\,], \quad \mathbf{y}_{ij} \leftarrow 0, \quad \mathbf{q}_{ij} \leftarrow 0$
35: $\mathbf{c} \leftarrow \tilde{\mathbf{c}}$

After each ADMM iteration, convergence is checked against a series of heuristics. The first, the primal residual, expresses the violation of the coupling constraints (6d) and thus indicates the amount of disagreement between local subproblems. Mathematically, the primal residual at iteration k is defined as:

$$r^k := \max_{i\in\mathcal{A}} \left\|\mathbf{h}_i(\mathbf{x}_i^k, \mathbf{q}_{ij}^k) - \mathbf{z}_i^k\right\|_2, \tag{14}$$

The second heuristic, the dual residual, quantifies the change in the consensus variable between consecutive iterations. It reflects the evolution of the dual variables and thus provides an indication of the progress and stability of convergence and is defined as:

$$s^k := \rho ||\mathbf{z}^k - \mathbf{z}^{k-1}||_2. \tag{15}$$

The primal and dual residuals are both computed in each ADMM iteration in line 21 and are used as metrics to monitor the convergence of the algorithm. When both metrics are below a certain threshold, r_c and s_c respectively, or when the maximum amount of iterations $k_{\max}$ is reached, iterations are stopped and the optimized agent' trajectories are returned.

2.3. Structure-aware decision making and configuration

Although ADMM-based distributed MPC methods can be effective as a local planner approach, we argue that deployment in structured real life environments requires higher level decision-making to either make them more robust, more computationally efficient, or to allow for admissable behavior in narrow areas, or for tasks which involve respecting higher level task constraints. This subsection details both the knowledge representation that is used to model the environment, the decision making algorithms, and the integration of the decision into the distributed optimization framework as was shown in Algorithm 1.

Knowledge representation

The world model (WM) is a description of the knowledge that agents have about the environment. We assume that this view is accurate, up-to-date, and that all agents have access to (a copy of) the WM for updates initiated by the individual agents. The vertices of the graph represent elements, which we refer to as map primitives. In this work, we distinguish three types of primitives: Areas, Interfaces, and Boundaries.

To formalize this representation, we model the WM as a property graph, where vertices represent map primitives, and edges describe the relationships between them:

Definition 1 (World model)**.** The world model is a *property graph* $G = (V, E, \tau, \alpha)$, where

- V is a set of vertices,
- $E \subseteq V \times V$ is a set of directed edges,
- $\tau : V \to \{\mathit{Area}, \mathit{Boundary}, \mathit{Interface}, \mathit{Object}, \ldots\}$ is a typing function assigning a entity type to each vertex,
- $\alpha : V \to \mathcal{A}$ assigns vertex properties (identity, geometry, accessibility, traversability …)

Edges in E encode structural relationships between vertices (e.g. adjacency, containment, connectivity, overlap).

The WM models, among other aspects, the existence and properties of areas within the environment in which agents operate. These areas may correspond to physically present regions, such as rooms, or may represent virtual areas that are not directly bounded by physical objects. Within an area, a further spatial decomposition may be defined in the form of lanes; for example, a corridor may be partitioned into two subspaces to require agents to remain within their respective lanes, thereby enabling movement in both directions, an example of which is shown in Fig. 1c.

From a motion planning perspective, the (sub-)area decomposition enforces a structured partitioning of the environment. In particular, by assigning agents to specific sub-areas, coupling constraints between agents can be restricted to those that occupy the same or adjacent areas. This reduces the number of active inter-agent constraints that must be considered at each planning step, thereby lowering the computational burden of the distributed optimization. The construction of such area decompositions is discussed in our previous work (de Vos et al., 2025b), while in this paper we focus on how the resulting structure is exploited for constraint configuration.

The world model is not only a static representation but is queried at each possible configuration change to extract the knowledge and constraints that are relevant to an agent's current task. Based on this information, the planner derives environment constraints, and inter-agent coupling constraints, determined from relational information such as shared areas, interfaces, or potential conflicts. The precise mapping from the world model to these constraint sets is detailed in the remainder of this section. While the specific query mechanisms and data structures used to perform this mapping are described in detail in our previous work (de Vos et al., 2024, 2025a), in this paper we focus on how the resulting structured information is used to configure the distributed MPC problem.

Relevant constraint selection

Each agent generates its plan independently of other agents by using conventional planning methods, e.g. A^*, on the connectivity graph that is derived from the WM. At this stage, the planning process assigns each agent a sequence of areas and lanes to traverse, based on the connectivity graph derived from the WM. The resulting plan is represented by the sequence of ids that correspond to the areas that an agent should traverse to complete its task. However, for an agent to traverse these areas, a plan with greater detail is necessary that specifies how each area should be traversed. As mentioned above, the generation of greater detail, but short-horizon, plans is the responsibility of the *local planner*. However, with the large amount of knowledge on the environment in the WM it becomes important to configure the planner in such a way that only constraints that are relevant to the current situation of the agent are taken into account. Adding a large amount of constraints, that may not all influence the final plan, leads to an increase in computational needs to compute the local plans. Retrieval of relevant constraints, that relate to the knowledge of the environment stored within the WM, is performed based on the plan and current state of each agent. Consider an agent a_i with plan $\mathbf{p}_i$, and assume it is currently at stage $\mathbf{p}_i^j$ of its plan. The boundaries that bound the area $\mathbf{p}_i^j$, i.e. $\mathcal{B}_{i,j}$, are included within the set of relevant constraints, however, are on their own not sufficient to constrain the motion of the agent. Specifically, the agent must also be prevented from entering areas for which the corresponding boundary constraints have been intentionally omitted. Such areas include not only those unrelated to the agent's plan, but also the areas associated with both past and future stages of the agent's own plan, the geometric description of the interfaces connected to area $\mathbf{p}_i^j$ and the area to which the agent should not navigate provide the knowledge necessary to describe those constraints. However, when inspecting the set of constraints constructed according to the description above, it becomes evident that the agent a_i would not be able to make any progress along its plan. Restricting the agent to the boundaries of the current area $\mathcal{B}_{i,j}$, while simultaneously prohibiting it from entering any other area, effectively prevents it from transitioning to the next stage of its plan. To address this, we extend the set of considered constraints beyond the current area. We incorporate the boundary constraints of a sequence of upcoming Areas along the plan, as described in Algorithm 2. Note that the constraint set reconfiguration is event-triggered and updated only upon transition to a new plan stage, i.e., when the agent enters a new area; between transitions the constraint set remains fixed, reducing the amount of necessary retrievals and reconfigurations.

Algorithm 2 Selection of relevant environment constraints.

Require: Agent plan and current plan index $(\mathbf{p}_i, k_i)$
Require: Area horizon length N
$\mathcal{X}_i \leftarrow \emptyset$
for $j = k_i : k_i + N$ **do**
  1. Retrieve boundaries of the current area
  $\mathcal{B}_{i,j} \leftarrow$ all Boundary vertices connected to area $\mathbf{p}_{i,j}$
  2. Determine areas within the Area horizon
  $\mathcal{N}_{i,j} \leftarrow \emptyset$
  if $j > k_i$ **then**
    $\mathcal{N}_{i,j} \leftarrow \mathcal{N}_{i,j} \cup \{\mathbf{p}_{i,j-1}\}$
    ▷ previous area within horizon
  if $j < k_i + N$ **then**
    $\mathcal{N}_{i,j} \leftarrow \mathcal{N}_{i,j} \cup \{\mathbf{p}_{i,j+1}\}$
    ▷ next area within horizon
  3. Retrieve relevant interfaces
  $\mathcal{I}_{i,j} \leftarrow$ all Interface vertices connected to
  $\mathbf{p}_{i,j}$ and not connected to any area in $\mathcal{N}_{i,j}$
  4. Add constraints for this plan step
  $\mathcal{X}_i \leftarrow \mathcal{X}_i \cup \mathcal{B}_{i,j} \cup \mathcal{I}_{i,j}$
return $\mathcal{X}_i$

Relevant agent selection

In addition to the constraints that are derived from the WM knowledge about the environment, a second reason to constrain the motion of agents is to prevent collision between agents. However, similarly to the previous sections, the inclusion of constraints between all agents present in the environment could result in high computation and, in this case, a high communication overhead. Therefore, consider an agent a_i that is currently at stage $\mathbf{p}_i^j$ of its plan $\mathbf{p}_i$. Furthermore, assume that its area horizon, the horizon over which constraints from the WM are considered, is of constant length N. Assume that we only constrain the motion of the agent against any other agent contained within the same area, i.e., the area that corresponds to $\mathbf{p}_i^j$. This would be sufficient in most scenarios, however, the planner could get into a situation in which the agent enters $\mathbf{p}_i^{j+1}$, or another agent enters $\mathbf{p}_i^j$, at a moment when a collision is unavoidable, leading to collision and an infeasible planning problem. To prevent this, agents constrain themselves against the trajectories of agents which are contained within their current area, any area that is within their area horizon, and any agent whose area horizon intersects their area horizon. Formally, this set of constraints is constructed according to Algorithm 3. Since the safety of the constructed set of constraints relies on the earlier assumed correctness of the world model, any inaccuracy propagates directly into the correctness of the constraint set. Specifically, if area boundaries are incorrectly specified, adjacency relations are missing, area-transition updates are delayed, or agent localization is incorrect, two agents may be classified as occupying non-adjacent areas, omitting the inter-agent collision avoidance constraint while in fact being in proximity. Therefore, monitoring the safety of the actions taken by agents remains important, even when agents are not directly constrained through the motion planning problem.

Algorithm 3 Relevant inter-agent constraint selection.

Require: Agent plan and plan index $(\mathbf{p}_i,\ k_i) \quad \forall a_i \in \mathcal{A}$
Require: Area horizon lengths N_i
$\mathbf{c} \leftarrow \emptyset$
for all $a_i \in \mathcal{A}$ **do**
  $\mathbf{h}_i = \{\mathbf{p}_i^{k_i} \dots \mathbf{p}_i^{k_i+N_i}\}$
  for all $a_j \in \{\mathcal{A} - \{a_i\}\}$ **do**
    $\mathbf{h}_j = \{\mathbf{p}_j^{k_j} \dots \mathbf{p}_j^{k_j+N_j}\}$
    if $\mathbf{h}_i \cap \mathbf{h}_j \neq \emptyset$ **then**
      $\mathbf{c} \leftarrow \mathbf{c} \cup \{(a_i, a_j)\}$
Return: The set of inter-agent constraints: $\mathbf{c}$

Discrete coordination constraints

A final source of constraints on agent behavior originates from discrete coordination decisions. Integrating such discrete decision-making mechanisms into the overall navigation framework enables the enforcement of higher-level behaviors that are difficult to encode directly within a conventional MPC formulation. In the framework of de Vos et al. (2025a), coordination, and consequently collision avoidance, is enforced by limiting the number of agents permitted to occupy a single area to at most one. A central coordinator maintains knowledge of the current areas and planned routes of the agents and issues permissions allowing agents to enter specific areas. The (lack of) permission is then incorporated as constraints within each agent's local MPC planner, by shrinking the area horizon of the agent to the necessary size such that it only enters areas for which it has permission. This approach effectively enforces the desired behaviors and, due to the decoupling introduced by the coordinator, scales better computationally than fully centralized MPC schemes. However, the resulting motion plans tend to be conservative, as the strict occupancy constraints can unnecessarily restrict agent motion even in cases where collisions could be avoided without stopping but through optimization of the agents trajectory in the continuous space. However, by extending the *permission to enter* check to not be limited by single agent occupancy but, for instance, to check whether a corridor can support agents moving in opposite directions, more complex behaviors can be achieved. The configuration of the *permission to enter* check algorithm is achieved by reasoning over the knowledge that is stored about each area in the WM, based on the specific requirements of the task and environment. The resulting overlap in area horizons is resolved using distributed MPC. Since the implementation details and priority rules of the discrete coordination mechanism are application-specific, we assume, for the remainder of this paper, the simple coordination strategy presented in Algorithm 4. Non-priority agents are not allowed to enter areas that overlap with a priority agent; instead, their area horizon is shortened so that they may continue moving but must stop just before the first overlapping area. This static priority assignment is a deliberate simplification, chosen to keep the presentation focused; the framework is not limited to this strategy. More expressive coordination policies, including dynamic claim-based approaches that compute area reservations at runtime and ensure deadlock freeness without requiring pre-assigned priorities, are demonstrated in e.g. (de Vos et al., 2025a). To retain recursive feasibility of the trajectory planner under dynamically changing permissions, however, care must be taken when updating an agent's allowed area. In particular, either the accessible area should not shrink along the agent's motion, or it must be ensured that the agent can still safely come to a stop within the remaining permitted region after a change in permissions, for instance by verifying that the agent's predicted trajectory has not already committed to entering a newly restricted area.

Algorithm 4 Example coordination strategy.

Require: Maximum area horizon $N_{\max}$
Require: Agent plans and current indices $(\mathbf{p}, \mathbf{k})$
Require: Priority agents $\mathcal{P}$
Require: Overlap areas $\mathcal{O} = \{(area, a_i, a_j)\}$

Initialization
for all agents i **do**
  $N_i \leftarrow \min\left(N_{\max},\ |\mathbf{p}_i| - k_i\right)$
Apply waiting rule for non-priority agents
for all $(area, a_i, a_j) \in \mathcal{O}$ **do**
  if $a_i \in \mathcal{P}$ **and** $a_j \notin \mathcal{P}$ **then**
    $k_{\max,j} \leftarrow \min\{\ n \mid n \geq k_j,\ \mathbf{p}_{a_j}[n] = area\ \}$
    $N_j \leftarrow \min\left(N_j,\ k_{\max,j} - k_j - 1\right)$ ▷ a_j must stop before *area*
  else if $a_j \in \mathcal{P}$ **and** $a_i \notin \mathcal{P}$ **then**
    $k_{\max,i} \leftarrow \min\{\ n \mid n \geq k_i,\ \mathbf{p}_{a_i}[n] = area\ \}$
    $N_i \leftarrow \min\left(N_i,\ k_{\max,i} - k_i - 1\right)$ ▷ a_i must stop before *area*
for all agents i **do**
  $N_i \leftarrow \max\left(0,\ \min(N_i, N_{\max})\right)$
Return: Area horizons $\mathbf{N} = \{N_i\}$

2.4. Optimizations for (Wireless) communication networks

A large part of the effective computation time of the framework, when considering distributed computation hardware communicating over a wireless network, is related to the latency of communication of that network, the amount of iterations necessary to reliably reach a constraint admissible solution is an important metric for the applicability of the approach. When many iterations are necessary to reach a feasible solution, the computation, due to the communication latency, is large even before taking the actual solver time into account. Although the local ADMM subproblems retain the constraints of the original centralized formulation, they enforce them in a relaxed manner and are coupled through local copies of the shared state variables. The primal residual measures the lack of consensus between these copies and is therefore a metric for the magnitude of possible constraint violations.

Within the implemented framework, the inter-agent collision constraints are therefore tightened based on the value of the primal residual from the previous iteration, ensuring that the generated trajectories remain admissible even when the algorithm has not yet converged. Formally, we redefine the inter-agent collision avoidance constraint in (1e) as

$$\delta_{i,j}\left(\operatorname{dist}\left(\mathbf{x}_i(k),\mathbf{x}_j(k)\right)-t_{\text{prim}}(k)\right)\geq 0, \tag{16}$$

where

$$t_{\text{prim}}(k)=\operatorname{clip}\left(\alpha\left\|\left(\mathbf{h}_i(\mathbf{x}_i^k,\mathbf{q}_{ij}^k)-\mathbf{z}^k\right)\right\|_\infty,\ t_{\min},\ t_{\max}\right), \tag{17}$$

and the clipping operator is defined as

$$\operatorname{clip}(x,\ t_{\min},\ t_{\max})=\max\left(t_{\min},\ \min(x,\ t_{\max})\right). \tag{18}$$

3. Validation

This section validates the proposed methods in two complementary settings. First, the methods are evaluated in simulation to assess trajectory quality across a set of validation scenarios. Second, to provide a more realistic evaluation that better reflects the intended application context, a hardware-in-the-loop (HIL) validation is performed. In the HIL setting, representative computation hardware, wireless networking, and communication protocols are employed to assess the impact of practical deployment factors, including communication latency and computational overhead.

3.1. Implementation

The framework described in this paper is implemented in C++. All ADMM-based MPC controllers are constructed using CasADi (Andersson et al., 2019) and solved using the OSQP quadratic programming solver (Stellato et al., 2020). The distributed computation nodes communicate using ZeroMQ sockets, a deployment diagram is depicted in Fig. 3.

For the hardware-in-the-loop (HIL) validation, a distributed computation setup consisting of multiple Raspberry Pi 5 units (Fig. 4, details in Table 1) is used. Each Raspberry Pi represents a single agent in the multi-agent system and executes an independent instance of the local MPC solver. In contrast to a fully decentralized implementation, the ADMM consensus step is executed on a central workstation. This workstation aggregates local solutions from all agents, performs the consensus update, and redistributes the updated variables to the agent nodes. The simulation results are executed entirely on this workstation. Wireless communication between the Raspberry Pi nodes and the workstation is carried out over a 5 GHz Wi-Fi network (IEEE 802.11ax). Communication follows a star topology, where each agent exchanges information exclusively with the central consensus node.

The performed experiments, furthermore, use polytopic constraint formulation where possible. The separating hyperplane formulation is used when agents have non-convex drivable space configuration, this occurs when agents turn a corner, or have to evade an obstacle. The communication time between iterations is reduced by optimizing the serialization method used for messages exchanged between agents. In addition to the constant network latency, the communication time is influenced by the processing required to translate the variables stored in memory into a communication message, as well as by the amount of data that must be transmitted. To reduce communication latency, a binary JSON-compatible format was used instead of plain JSON. In particular, CBOR was selected, as the survey in Viotti and Kinderkhedia (2022) shows that such binary formats provide a more compact representations and lower serialization overhead over plain JSON. The resulting reduction in payload size and encoding time decreases the communication time between iterations.

3.2. Experimental design

The goal of the validation is to demonstrate both the feasibility of the proposed framework and the computational benefits of integrating structural knowledge into ADMM-based decision making for robotic agents. To this end, three ADMM-based decision-making methods are compared. The first method is a conventional ADMM framework. The second augments this conventional framework with predefined lanes, thereby reducing the number of inter-agent constraints. The third is the full proposed method, which combines a lane-based structure with discrete yielding actions. As a baseline for trajectory quality, these methods are additionally compared against a centralized nonlinear MPC formulation with a long prediction horizon, solved using IPOPT. Performance is evaluated using four metrics: successful task completion rate, path length, task completion time, and average computation time. The task success rate is further decomposed to distinguish between failures caused by collisions and failures caused by stagnation, such as deadlocks or situations in which agents reach configurations from which further progress becomes locally infeasible.

To validate the methods, three sets of scenarios are introduced. The first set considers four scenarios in which agents pass each other head-on in a rectangular corridor. In scenario 1A, one agent departs from each end of the corridor heading to the opposite side. Scenario 1B increases the amount of agents by placing one agent on one side and two on the

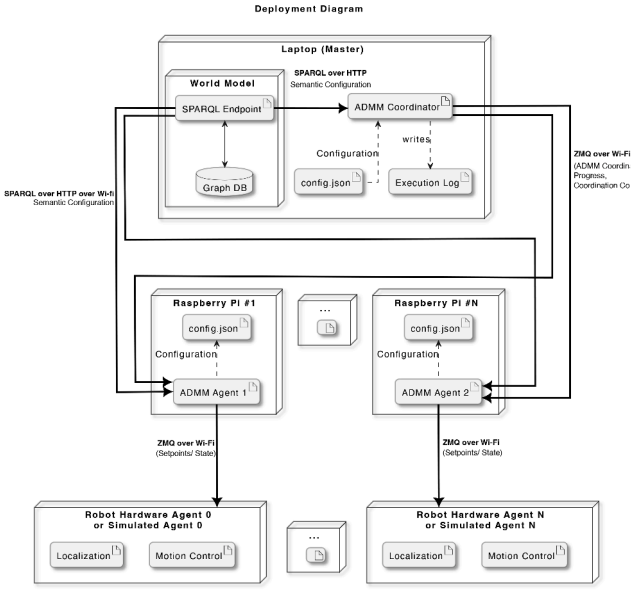

Fig. 3. Deployment diagram of the presented framework.



Fig. 4. Hardware-in-the-loop (HIL) setup with distributed Raspberry Pi 5 agent nodes executing local MPC optimization.

Table 1
Hardware and software specifications for simulation and HIL experiments.

Property	Distributed Nodes	Workstation
System	Raspberry Pi 5	Desktop PC
CPU & RAM	Cortex-A76, 8 GB	i5-13500, 32 GB
OS	Ubuntu 24.04	Windows 11
WI-FI	Wi-Fi 6 (802.11ax), 5 GHz	
Middleware	ZeroMQ	
Solver	CasADi v3.7.2, OSQP v0.6.3	

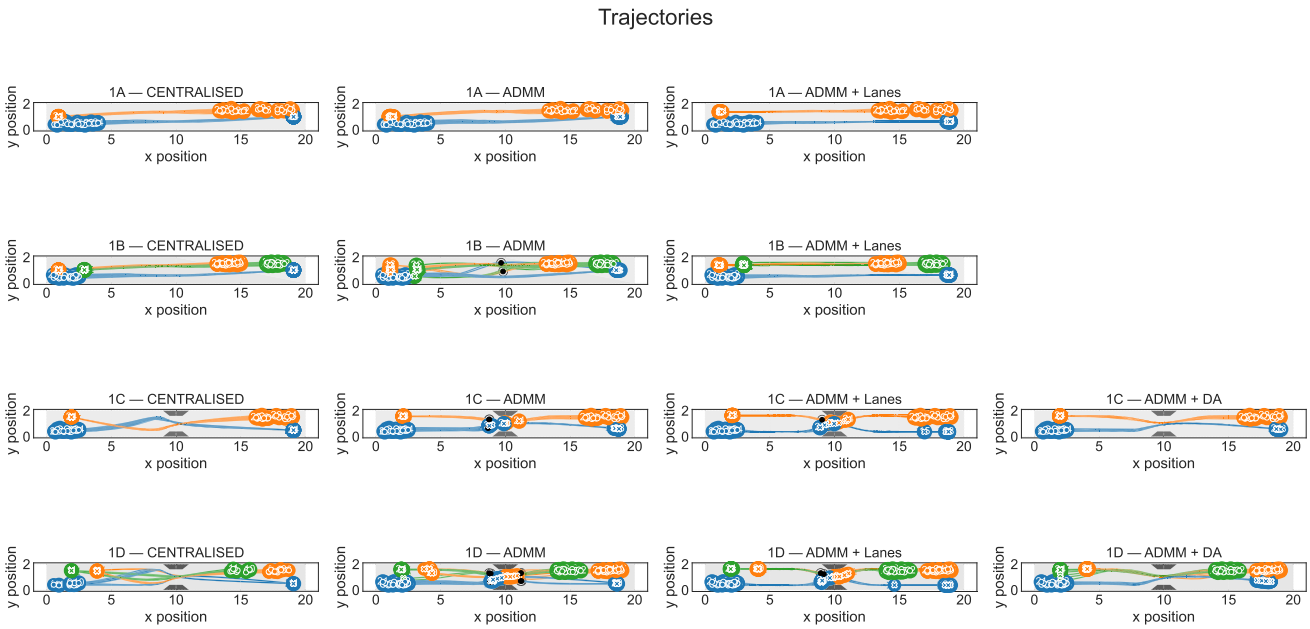

Fig. 5. Trajectories of agents across 25 different runs for each method of scenarios 1A through 1D. Agents start at the circle, and end their trajectories at the cross. Agents not reaching the same goal configuration as the centralized solver are classified as stagnation. Collisions are depicted using black circles.

Table 2
Parameters used in the validation of the four presented methods. A conventional ADMM-based distributed MPC formulation, a Structurally configured (SC) variant, a Structurally configured (SC) variant with Discrete Yielding Actions (DA), and the centralized baseline.

Symbol	ADMM Methods	Centralised Baseline
N	15	45
ΔT	0.25	0.25
r_i	0.70	0.70
$r_{i,soft}$	0.75	0.75
w_{obj}	0.5/1.0 [1]	0.5/1.0 [1]
solver	OSQP	IPOPT
α	1.0	
$t_{\min}$	0.0	
$t_{\max}$	1.0	
ρ	25.0	
$k_{\max}$	5	
r_c	0.25	
s_c	1.0	

[1] Experiments showed that agents required to execute a cornering maneuver to reach their goal benefit from a higher objective weight compared to the normal configuration.

other. Scenarios 1C and 1D mirror 1A and 1B respectively, but introduce a static obstacle that narrows the corridor at its midpoint, forcing agents to negotiate a bottleneck while avoiding a collision. The second set considers four scenarios in which agents navigate a four-arm intersection. In scenarios 2A and 2B, all agents travel straight through to the opposite arm: 2A places one agent in each of the 4 arms, while 2B increases traffic density by placing two agents in two of the arms. Scenarios 2C and 2D replicate these configurations, but introduce turning behavior, with one agent in 2C and two agents in 2D turning right at the intersection rather than continuing straight. The final set considers a single scenario, 3A, in which 10 agents navigate two four-arm intersections arranged in series, requiring agents to successfully resolve conflicts at two consecutive decision points. The details of each scenario are summarized in Figs. 5–7. To reduce the effect that a specific realization of initial conditions can have on the quality of the obtained solution, 25 runs of each scenario are performed with initial conditions randomized around a nominal initial condition. In the DA variant of the algorithm that includes yielding actions, priority rules are assigned as follows: scenarios 1A and 1B are excluded from this experiment, as the geometry does not have a natural configuration in which yielding decisions make sense. In 1C and 1D the agent traveling from the left must yield to the one traveling from the right. In 2A and 2B agents traveling vertically must yield to those traveling horizontally. In 2C and 2D the agent whose path is crossed by a turning agent must yield to that turning agent. In 3A agents traveling horizontally required to yield to those traveling vertically, thereby showing the algorithm's flexibility to generalize on figure different priority assignments.

3.3. Parameter selection

In principle, for convex optimization problems, the centralized formulation and the distributed ADMM formulation are theoretically equivalent. Given sufficient iterations, ADMM converges to the same optimal solution as the centralized solver, however, in practice this often requires a substantial number of iterations. Furthermore, particularly in the case of nonconvex problems, the algorithm configuration parameters need to be selected carefully to balance convergence stability against convergence speed. For example, choosing a large value of ρ can induce oscillatory or unstable convergence behavior, whereas very small values of ρ typically result in slow convergence. Intermediate values tend to provide convergence properties that allow for real-time implementations that we target in this paper. Table 2 lists the main parameter values used in the validation experiments. The maximum number of ADMM iterations was selected such that all methods were able to generate constraint-admissible trajectories in Scenarios 1A and 1B. Experimental observations showed that higher objective weights yielded smoother trajectories, in particular with respect to the velocity profiles. At the same time, increasing w_{obj} led to a larger

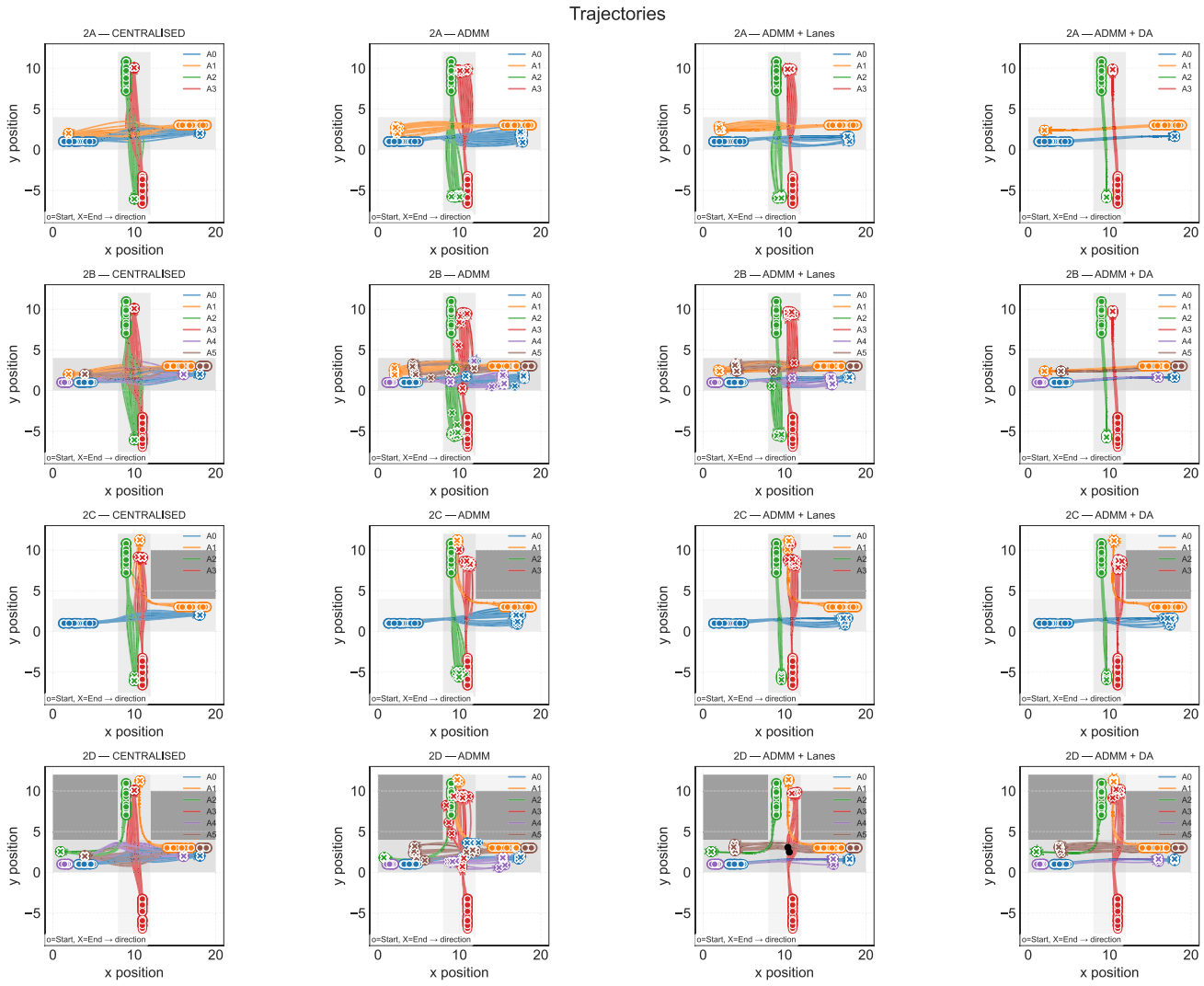

Fig. 6. Trajectories of agents across 25 different runs for each method of scenarios 2A through 2D. Agents start at the circle, and end their trajectories at the cross. Agents not reaching the same goal configuration as the centralized solver are classified as stagnation. Collisions are depicted using black circles.

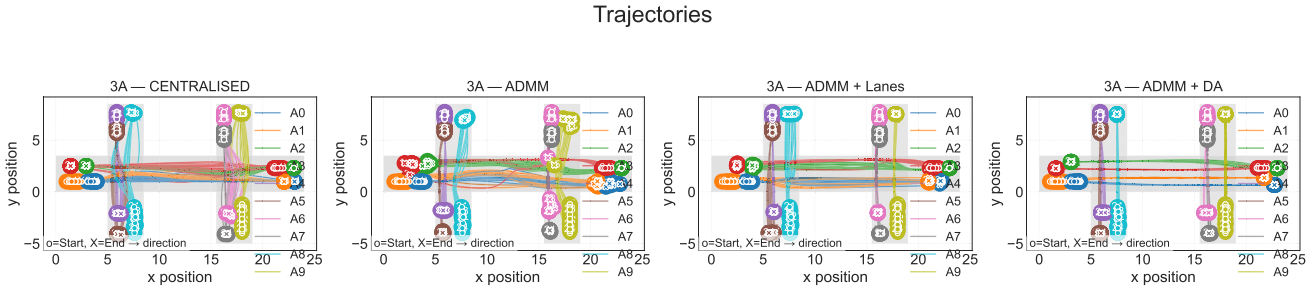

Fig. 7. Trajectories of agents across 25 different runs for each method of scenario 3A. Agents start at the circle, and end their trajectories at the cross. Agents not reaching the same goal configuration as the centralized solver are classified as stagnation. Collisions are depicted using black circles.

number of iterations required to maintain satisfactory collision rates. Similarly, the ADMM parameter ρ influenced the convergence behavior: larger values of ρ improved consensus by more strongly penalizing disagreement between agents, but could lead to conservative or oscillatory updates, whereas smaller values resulted in slower convergence. Therefore, both w_{obj} and ρ were fixed to constant values that provided the best trade-off between motion performance and avoiding constraints violations in all methods in Scenarios 1A and 1B.

3.4. Results

The trajectories and computed metrics for the runs of the first set of scenarios are presented in Figs. 3 and 5. The metrics indicate that

Table 3
Metrics averaged over runs and agents ($\mu \pm \sigma$).

Scenario	Metric	Method			
		Centralised MPC	ADMM	ADMM w/ Lanes	ADMM w/ Lanes & DA
1A	Path Length (m)	15.690±1.703	15.621±1.636	15.603±1.698	–
	Time to Completion (s)	29.460±3.409	29.484±3.284	29.465±3.408	–
	Success Rate (%)	100	100	100	–
	Collision Rate (%)	0	0	0	–
1B	Path Length (m)	15.048±1.830	14.972±1.803	14.957±1.823	–
	Time to Completion (s)	28.183±3.658	28.297±3.571	28.280±3.560	–
	Success Rate (%)	100	100	100	–
	Collision Rate (%)	0	0	0	–
1C	Path Length (m)	16.584±1.237	16.378±1.994	16.471±1.418	16.376±1.208
	Time to Completion (s)	32.022±2.994	31.000±4.000	31.847±3.215	35.255±6.369
	Success Rate (%)	100	8	36	100
	Collision Rate (%)	0	48	52	0
1D	Path Length (m)	14.926±2.040	14.445±1.444	14.727±1.791	14.767±1.987
	Time to Completion (s)	27.962±3.240	28.491±3.979	28.972±5.665	29.800±6.509
	Success Rate (%)	100	16	32	100
	Collision Rate (%)	0	8	4	0
2A	Path Length (m)	15.230±1.092	14.800±1.041	14.943±1.056	14.953±1.105
	Time to Completion (s)	28.620±2.202	29.797±2.425	28.972±2.347	34.003±6.168
	Success Rate (%)	100	100	100	100
	Collision Rate (%)	0	0	0	0
2B	Path Length (m)	14.607±1.022	14.044±0.868	14.225±0.864	14.303±0.933
	Time to Completion (s)	27.463±2.088	30.525±3.895	29.020±3.043	31.623±7.285
	Success Rate (%)	100	76	96	100
	Collision Rate (%)	0	0	0	0
2C	Path Length (m)	14.408±1.403	13.939±0.967	14.068±1.128	14.034±1.215
	Time to Completion (s)	26.914±2.905	28.523±2.463	27.451±2.929	28.349±3.276
	Success Rate (%)	100	96	100	100
	Collision Rate (%)	0	0	0	0
2D	Path Length (m)	13.949±1.503	13.582±1.030	13.887±1.317	13.846±1.339
	Time to Completion (s)	25.958±3.038	30.389±2.858	26.643±3.233	26.995±3.758
	Success Rate (%)	100	52	96	100
	Collision Rate (%)	0	0	4	0
3A	Path Length (m)	14.005±4.962	13.263±4.615	13.467±4.577	13.722±4.930
	Time to Completion (s)	26.732±9.469	40.357±11.506	31.989±12.129	30.952±16.187
	Success Rate (%)	100	78	100	100
	Collision Rate (%)	0	0	0	0

the chosen parameter set yields satisfactory task completion for both ADMM-based methods in scenarios 1A and 1B. Discrete yielding actions are not explicitly validated in these scenarios, as there is limited overlap between agent trajectories. However, Scenarios 1C and 1D introduce an obstruction within the rectangular corridor, increasing the level of interaction between agents. In these cases, both ADMM-based methods without discrete yielding actions struggle to adequately resolve the scenario, with the lane-constrained variant performing better, but still failing in more than half of the runs. In contrast, the method incorporating discrete yielding actions is able to successfully solve the scenario in all runs with only a relatively small task performance penalty compared to the centralized solution.

The results of scenario set 2 is presented in Fig. 6, and Tables 3 and 4. The computed metrics show the trade-off in MPC between computational performance and trajectory quality. The centralized solver achieves a perfect task completion rate across all scenarios. In contrast, the conventional ADMM approach shows reduced performance, particularly in the six-agent scenarios (2B and 2D), where task completion rates decrease significantly. Introducing a lane-based structure significantly improves performance, with ADMM with imposed lane structure achieving near-perfect task completion across all scenarios. The full proposed method, incorporating both lane structure and discrete yielding actions, achieves perfect task completion in all cases, eliminating both collisions and stagnation-related failures. These improvements, however, come at the cost of trajectory efficiency. Both the imposed lane structure and discrete decision layer lead to increases in path length and task completion time. This reflects the additional coordination constraints and conservative conflict resolution strategies introduced by the framework. In particular, the magnitude of this performance penalty is scenario-dependent and varies with the specific realization of initial conditions. Notably, this effect is only partially visible in the reported mean values, as a subset of agents exhibit substantially longer completion times, which are primarily reflected in the increased variance rather than the mean.

Comparing computation times, in the upper section of Table 4, for scenarios 2A through 2D, which are deemed representative of the general trend,shows that the ADMM-based methods exhibit largely comparable computation times on average, primarily due to the imposed cap on the number of iterations. While increasing the iteration limit for the conventional ADMM approach would likely improve task completion performance, this would come at the cost of increased computation time, possibly reducing its real-time applicability. The middle section of Table 4 reports statistics on the convergence of the ADMM algorithm. The addition of discrete actions yields a clear improvement in scenarios 2A and 2B, where both the percentage of time steps reaching the maximum iteration limit and the average residuals are reduced. This suggests that resolving part of the interaction logic at the discrete level can reduce the complexity of the continuous ADMM coordination problem. These same observations do not, however, hold for scenarios 2C and 2D. Importantly, however, the convergence statistics should be

Table 4
Computation-time and convergence statistics for scenario set 2 in simulation and for the HIL implementation of scenario 2B. Simulation computation times are reported as minimum, average, and maximum values in milliseconds. Convergence statistics report the average number of ADMM iterations, the percentage of time steps reaching the maximum iteration limit, and the average primal and dual residuals after admm iterations.

	Scenario	Conventional ADMM	ADMM w/ Lanes	ADMM w/ Lanes + DA		
		Simulation computation time: min / avg. / max. (ms)				
	2A	7 / 19 / 34	10 / 17 / 31	3 / 7 / 21		
	2B	6 / 26 / 52	6 / 23 / 44	5 / 15 / 43		
	2C	7 / 22 / 47	7 / 19 / 37	9 / 16 / 31		
	2D	8 / 30 / 57	14 / 25 / 58	7 / 24 / 52		
		Convergence statistics				Task-level
Scenario	Method	Avg. iter.	At max iter. (%)	Avg. primal res.	Avg. dual res.	Success (%)
2A	Conventional ADMM	4.17	71	0.49	2.37	100
	ADMM w/ Lanes	4.16	71	0.33	2.05	100
	ADMM w/ Lanes + DA	2.54	11	0.00	0.25	100
2B	Conventional ADMM	4.36	78	1.38	3.52	76
	ADMM w/ Lanes	4.21	73	0.83	2.92	96
	ADMM w/ Lanes + DA	3.87	40	0.16	1.57	100
2C	Conventional ADMM	4.71	90	0.50	2.89	96
	ADMM w/ Lanes	4.71	89	0.35	2.57	100
	ADMM w/ Lanes + DA	4.83	93	0.33	2.51	100
2D	Conventional ADMM	4.45	81	1.63	3.42	52
	ADMM w/ Lanes	5.00	100	15.56	20.45	96
	ADMM w/ Lanes + DA	4.94	99	12.92	20.02	100
	HIL statistics, scenario 2B	Conventional ADMM	ADMM w/ Lanes	ADMM w/ Lanes + DA		
	Minimum computation time (ms)	28	27	12		
	Average computation time (ms)	91	91	43		
	Maximum computation time (ms)	347	370	162		
	Average overhead (ms)	68	71	36		

interpreted in conjunction with the task-level metrics. In scenario 2D, the baseline method does not achieve satisfactory task-level performance; therefore, the acceptable convergence statistics of the baseline method do not lead to successful closed-loop coordination. Scenario 2D further shows that, even with lanes and discrete actions, the optimization problem remains difficult to solve: both lane-based methods reach the maximum iteration limit in almost all time steps and exhibit large residuals. The lower section of Table 4 reports the results of the Hardware-in-the-Loop (HIL) experiments, in which scenario 2B was replicated on distributed computation hardware, with inter-agent communication carried out over a Wi-Fi connection. The resulting trajectories that were obtained from this run were equivalent; however, from the table it is evident that overall computation times significantly degrade. Although this degradation is partly attributable to the lower processing performance of the Raspberry Pi hardware compared to the simulation platform, the results indicate that the dominant contribution arises from a substantial increase in communication overhead. This highlights that the number of ADMM iterations required to reach consensus becomes a critical tuning parameter, particularly under realistic deployment conditions such as Wi-Fi-based communication, where each additional iteration incurs non-negligible communication overhead.

Similar observations can be made for the metrics obtained in the third scenario set, shown in Table 3. While the centralized and structured approaches achieve near-perfect task completion across the runs, it is again evident that the conventional ADMM struggles to consistently reach consensus within the imposed maximum number of iterations, as reflected by the reduced success rate. However, instead of only resulting in collisions or deadlocks, unsuccessful or slow-converging runs are also characterized by increased task completion times. In contrast, the proposed methods retain full reliability while remaining relatively close to the completion time of the baseline method.

4. Conclusions

This paper presents a framework for structurally configuring distributed MPC-based controllers for mobile robot systems. Both simulation and hardware-in-the-loop experiments demonstrate a clear computational advantage over conventional distributed approaches by exploiting task- and environment-level knowledge encoded in a world model. The proposed hybrid formulation combines continuous space coordination combined with reasoning over discrete actions, allowing the controller to capture both continuous space coordination and high-level task constraints. This leads to a reduced search space and, consequently, lower computational complexity, while simultaneously the ability to enforce task-level rules. At the same time, in contrast to our earlier work where a single-agent occupancy constraint effectively forced all agents to yield, the proposed hybrid formulation removes this global restriction. As a result, agents can continue progressing toward their goal unless explicitly constrained by the task-level decision layer, leading to less conservative behavior. However, the current approach suffers from the following limitations. While we aim to reduce the amount of iterations, since communication overhead is a large part of the total execution time, in the hardware-in-the-loop experiments overhead is still a significant factor. In particular, synchronization between solver processes, caused by heterogeneous local problem solve times, together with network communication latency, can dominate the overall runtime and limit the practical scalability of the approach. Techniques such as asynchronous ADMM could be investigated to mitigate these limitations. Furthermore, the performance of the proposed algorithms is sensitive to the choice of parameters. A key limitation of the proposed approach is that feasibility and safety are not formally guaranteed under incomplete convergence of the ADMM iterations. In practice, constraint satisfaction depends on the degree of convergence achieved within the available computation

time, which in turn is influenced by the choice of algorithm parameters. Although the selected parameter settings were sufficient to produce constraint-admissible trajectories in most realizations of the evaluated scenarios, no systematic procedure for determining these values was established. As a result, appropriate parameter values for a given scenario cannot be predicted a priori and must currently be tuned empirically. Consequently, both convergence behavior and safety performance may degrade under suboptimal parameter choices or limited communication budgets. While empirical results indicate that constraint violations are infrequent in the evaluated scenarios, this behavior cannot be guaranteed in general. This limitation, however, is shared with the baseline centralized and distributed MPC formulations used for comparison. Developing systematic tuning strategies and providing formal safety guarantees under limited convergence remain important directions for future work. Finally, all controller configurations presented in this work rely on the same world model mechanism to configure their constraints and objective functions. As a consequence, they share the sensitivity to inaccuracies or incompleteness in the structural knowledge encoded in the world model: an incorrectly specified area boundary, missing lane partition, or erroneous adjacency relation will affect constraint generation uniformly across all configurations. A systematic analysis of controller performance with respect to inaccuracies in the world model, as well as mechanisms for detecting and correcting such inaccuracies at runtime, remain interesting directions for future work.

CRediT authorship contribution statement

Koen de Vos: Conceptualization, Methodology, Software, Validation, Writing – original draft, Writing – review & editing; **Tom Minten:** Conceptualization, Methodology, Conceptualization, Writing – review & editing; **Herman Bruyninckx:** Conceptualization, Supervision, Writing – review & editing; **René van de Molengraft:** Supervision, Writing – review & editing; **Dinesh Krishnamoorthy:** Conceptualization, Methodology, Supervision, Writing – review & editing; **Elena Torta:** Conceptualization, Supervision, Writing – review & editing.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used generative AI in order to improve the readability and language of the manuscript. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work has been executed as part of the Semantic navigation for Teams of Open World Robots (TOWR) research project and is part of the ICAI EAISI FAST lab.

References

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., & Diehl, M. (2019). CasADi: A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, *11*(1), 1–36. https://doi.org/10.1007/s12532-018-0139-4

Bai, X., Li, C., Zhang, B., Wu, Z., & Ge, S. S. (2024). Efficient performance impact algorithms for multirobot task assignment with deadlines. *IEEE Transactions on Industrial Electronics*, *71*(11), 14373–14382. https://doi.org/10.1109/TIE.2024.3374366

van den Berg, J., Lin, M., & Manocha, D. (2008). Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE International conference on robotics and automation* (pp. 1928–1935). https://doi.org/10.1109/ROBOT.2008.4543489

van den Berg, J. P., & Overmars, M. H. (2005). Prioritized motion planning for multiple robots. In *2005 IEEE/RSJ International conference on intelligent robots and systems* (pp. 430–435). https://doi.org/10.1109/IROS.2005.1545306

Best, G., Forrai, M., Mettu, R. R., & Fitch, R. (2018). Planning-aware communication for decentralised multi-robot coordination. In *2018 IEEE International conference on robotics and automation (ICRA)* (pp. 1050–1057). https://doi.org/10.1109/ICRA.2018.8460617

Crespo, J., Castillo, J. C., Mozos, O. M., & Barber, R. (2020). Semantic information for robot navigation: A survey. *Applied Sciences*, *10*(2). https://doi.org/10.3390/app10020497

Ferranti, L., Lyons, L., Negenborn, R. R., Keviczky, T., & Alonso-Mora, J. (2023). Distributed nonlinear trajectory optimization for multi-robot motion planning. *IEEE Transactions on Control Systems Technology*, *31*(2), 809–824. https://doi.org/10.1109/TCST.2022.3211130

Ferranti, L., Negenborn, R. R., Keviczky, T., & Alonso-Mora, J. (2018). Coordination of multiple vessels via distributed nonlinear model predictive control. In *2018 European control conference (ECC)* (pp. 2523–2528). https://doi.org/10.23919/ECC.2018.8550178

Fiorini, P., & Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *The international journal of robotics research*, *17*(7), 760–772. https://doi.org/10.1177/027836499801700706

Firoozi, R., Ferranti, L., Zhang, X., Nejadnik, S., & Borrelli, F. (2024). A distributed multi-vehicle coordination algorithm for navigation in tight environments. *IEEE Transactions on Vehicular Technology*, *73*(10), 14499–14509. https://doi.org/10.1109/TVT.2024.3409687

Fox, D., Burgard, W., & Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, *4*(1), 23–33. https://doi.org/10.1109/100.580977

Fredriksson, S., Bai, Y., Saradagi, A., & Nikolakopoulos, G. (2025). Multi-agent path finding using conflict-based search and structural-semantic topometric maps. In *2025 IEEE International conference on robotics and automation (ICRA)* (pp. 4229–4235). https://doi.org/10.1109/ICRA55743.2025.11128758

Gerkey, B. P., & Matarić, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, *23*(9), 939–954. https://doi.org/10.1177/0278364904045564

Koren, Y., & Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings. 1991 IEEE international conference on robotics and automation* (pp. 1398–1404 vol.2). https://doi.org/10.1109/ROBOT.1991.131810

LaValle, S. M. (2006). Planning algorithms. USA: Cambridge University Press. https://doi.org/10.1017/CBO9780511546877

McBeth, C., Motes, J., Uwacu, D., Morales, M., & Amato, N. M. (2023). Scalable multi-robot motion planning for congested environments with topological guidance. *IEEE Robotics and Automation Letters*, *8*(11), 6867–6874. https://doi.org/10.1109/LRA.2023.3312980

Mercy, T., Van Loock, W., & Pipeleers, G. (2016). Real-time motion planning in the presence of moving obstacles. In *2016 European control conference (ECC)* (pp. 1586–1591). https://doi.org/10.1109/ECC.2016.7810517

Rimon, E., & Koditschek, D. E. (1992). Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, *8*(5), 501–518. https://doi.org/10.1109/70.163777

Roszkowska, E., Makowski-Czerski, P., & Janiec, L. (2023). Multi-level control for multiple mobile robot systems. *Discrete Event Dynamic Systems*, *33*(4), 425–453. https://doi.org/10.1007/s10626-023-00383-x

Sakagami, R., Lay, F. S., Dömel, A., Schuster, M. J., Albu-Schäffer, A., & Stulp, F. (2023). Robotic world models-conceptualization, review, and engineering best practices. *Frontiers in Robotics and AI*, *10*, 1253049. https://doi.org/10.3389/frobt.2023.1253049

Serra-Gómez, Á., Zhu, H., Ferreira de Brito, B. F., Böhmer, W., & Alonso-Mora, J. (2023). Learning scalable and efficient communication policies for multi-robot collision avoidance. *Autonomous Robots*, *47*(8), 1275–1297. https://doi.org/10.1007/s10514-023-10127-3

Shorinwa, O., Halsted, T., Yu, J., & Schwager, M. (2024). Distributed optimization methods for multi-robot systems: Part 2-a survey. *IEEE Robotics & Automation Magazine*, *31*(3), 154–169. https://doi.org/10.1109/MRA.2024.3352852

Siegwart, R., Nourbakhsh, I. R., & Scaramuzza, D. (2011). Introduction to autonomous mobile robots. Cambridge, MA, USA: MIT Press.

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., & Boyd, S. (2020). OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, *12*(4), 637–672. https://doi.org/10.1007/s12532-020-00179-2

Stern, R., Sturtevant, N., Felner, A., Koenig, S., Ma, H., Walker, T., Li, J., Atzmon, D., Cohen, L., Kumar, T. K. et al. (2019). Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the international symposium on combinatorial search* (pp. 151–158). (*vol. 10*). https://doi.org/10.1609/socs.v10i1.18510

Van Parys, R., & Pipeleers, G. (2016). Online distributed motion planning for multi-vehicle systems. In *2016 European control conference (ECC)* (pp. 1580–1585). https://doi.org/10.1109/ECC.2016.7810516

Viotti, J. C., & Kinderkhedia, M. (2022). A survey of JSON-compatible binary serialization specifications. *CoRR, abs/2201.02089*.

de Vos, K., Torta, E., Bruyninckx, H., López Martínez, C. A., & van de Molengraft, M. J. G. (2024). Automatic configuration of multi-agent model predictive controllers based on semantic graph world models. In *2024 IEEE International conference on robotics and automation (ICRA)* (pp. 7194–7200). https://doi.org/10.1109/ICRA57147.2024.10610708

de Vos, K., Torta, E., Bruyninckx, H., Martínez, C. A. L., & van de Molengraft, M. J. G. (2025a). Hybrid decision making for scalable multi-agent navigation: Integrating

semantic maps, discrete coordination, and model predictive control. In *2025 IEEE International conference on robotics and automation (ICRA)* (pp. 5380–5386). https://doi.org/10.1109/ICRA55743.2025.11128657

de Vos, K., Torta, E., Bruyninckx, H., & van de Molengraft, R. (2025b). Automatic configuration of multi-robot MPC controllers from metric and semantic digital twin data. SSRN. https://doi.org/10.2139/ssrn.5374773

Wang, A., & Lin, Z. (2026). Multi-agent reinforcement learning with spatial structure awareness for topological map-based path-finding. *IEEE Robotics and Automation Letters*, *11*(1), 618–625. https://doi.org/10.1109/LRA.2025.3635371

Zhu, H. (2022). Probabilistic motion planning for multi-robot systems. Dissertation (TU Delft) Delft University of Technology. https://doi.org/10.4233/uuid:52038194-6cdf-4098-8723-852eb68dfd00