From Planning to Execution: Heterogeneous Multi-Agent Path Finding with Field-Centric Distributed Coordination

Bojan Janjatović
Elektrokombinacija
Kikinda, Serbia
bojan.janjatovic@gmail.com

***Abstract*—Coordinating heterogeneous robot fleets operating in different dimensional spaces—1D rails, 2D floors, 3D airspace—presents challenges that existing Multi-Agent Path Finding (MAPF) formulations do not address. We introduce Mixed-Dimensional MAPF (MD-MAPF), the first MAPF formulation with a dimensionality function mapping agents to operational spaces and a vertex compatibility function constraining spatial access. We present a six-class dimensional conflict taxonomy (LINEAR, PLANAR, CROSSING, AERIAL, VERTICAL, AIR-GROUND) with specialized resolution strategies achieving 5.6× speedup over baseline CBS. To bridge planning and execution, we map deadline slack to field components in the EK-KOR2 distributed coordination layer, enabling gradient-based resource allocation that respects planning constraints. Drawing from collective behavior research, we implement $k = 7$ topological neighbor coordination that maintains scale-free correlations across 500-2000 module installations, achieving $O(\log N)$ convergence with 42% CAN-FD bandwidth utilization. Experimental evaluation demonstrates 99.9% deadline compliance and zero energy violations over 24-hour operational cycles. The implementation comprises 7K lines of Go for planning with dual C/Rust embedded targets for execution.**

I. Introduction

Coordinating heterogeneous robot fleets presents fundamental challenges that existing Multi-Agent Path Finding (MAPF) formulations do not address. Consider a robotic battery swap station for electric buses: mobile robots navigate the 2D floor, rail-mounted manipulators operate along 1D tracks, and inspection drones traverse 3D airspace. These agents interact at shared handoff points, compete for charging resources, and must meet strict operational deadlines. Standard MAPF algorithms [1], [2] assume homogeneous agents operating in the same space, leaving this heterogeneous coordination problem unsolved.

This paper bridges two research areas: MAPF planning with optimality guarantees and distributed real-time execution with scalability requirements. We present MAPF-HET, a unified framework spanning from abstract planning to embedded coordination.

A. Motivating Application

Electric bus depot charging exemplifies the heterogeneous coordination challenge. A typical installation includes:

- **500-2000 power modules** (3.3 kW each) forming reconfigurable charging points
- **Rail robots** for heavy battery transport along fixed tracks
- **Mobile robots** for auxiliary logistics on the depot floor
- **Drones** for inspection and light-payload delivery

These agents operate in fundamentally different dimensional spaces yet must coordinate at shared resources. A rail robot cannot deviate from its track to avoid a mobile robot; a drone must land at designated pads to interact with ground systems. Existing MAPF formulations cannot capture these dimensional constraints.

Furthermore, the power modules themselves require coordination: load balancing across the grid connection, thermal management, and deadline-driven charging sessions. This creates a second layer of multi-agent coordination below the robotic planning layer.

B. Contributions

This paper makes four contributions:

1. MD-MAPF Formulation: We introduce Mixed-Dimensional MAPF, the first MAPF formulation with a dimensionality function $\kappa(a) \to \{1, 2, 3\}$ mapping agents to 1D rail, 2D mobile, or 3D aerial operational spaces. The vertex compatibility function $\delta(v)$ constrains which agents can access which locations, enabling precise modeling of dimensional restrictions.

2. Six-Class Dimensional Conflict Taxonomy: We classify conflicts by dimensional interaction: LINEAR (1D-1D), PLANAR (2D-2D), CROSSING (1D-2D), AERIAL (3D-3D same layer), VERTICAL (3D-3D across layers), and AIR-GROUND (3D-1D/2D). Each class admits specialized resolution strategies that exploit dimensional structure, significantly pruning the CBS search tree.

3. Planning-Execution Unification: We bridge the gap between optimal MAPF planning and real-time distributed execution. Deadline slack from the planning layer maps to field components in the execution layer, enabling gradient-based coordination that respects planning constraints while adapting to runtime conditions.

4. Bio-Inspired Scale-Free Coordination: Drawing from collective behavior research [3], we implement $k = 7$ topological neighbor coordination that maintains scale-free correlations across 500-2000 module installations. Information propagates in $O(\log N)$ hops regardless of physical network topology.

C. Paper Organization

Section II surveys related work in MAPF, energy-aware planning, and distributed systems. Section III formalizes MD-MAPF, the conflict taxonomy, and the execution model. Section IV presents our CBS variants: MIXED-CBS, E-CBS, DEADLINE-CBS, and HYBRID-CBS. Section V describes the EK-KOR2 execution layer with potential field scheduling and topological coordination. Section VI establishes theoretical properties. Section VII details implementation in Go (7K LOC) with C/Rust embedded targets. Section VIII presents experimental results. Section IX concludes.

II. RELATED WORK

We survey four areas: MAPF formulations for heterogeneous agents, conflict classification in CBS, energy-aware planning, and distributed coordination systems.

A. Multi-Agent Path Finding

The foundational CBS algorithm [1] introduced two-level search: high-level conflict detection and low-level single-agent planning. Subsequent work improved efficiency through enhanced heuristics (CBSH [4]), improved conflict classification (ICBS [5]), and continuous-time extensions (CCBS [6]).

Heterogeneous MAPF: G-MAPF [7] generalizes agent representation with state-based transition functions but assumes all agents operate on the same graph structure. LA-MAPF [8] handles geometric agents with varying footprints but not dimensional restrictions. Multi-Train Path Finding [9] addresses 1D rail networks in isolation. The Flatland Challenge [10] demonstrated scalable rail planning but without integration with 2D/3D agents.

Gap: No existing work models agents operating in fundamentally different dimensional spaces (1D rail, 2D ground, 3D aerial) within a unified MAPF formulation. MD-MAPF introduces the dimensionality function $\kappa(a) \rightarrow \{1, 2, 3\}$ and vertex compatibility function $\delta(v)$ to address this gap.

B. Conflict Classification

Conflict classification in MAPF literature follows three approaches:

- **Basic:** CBS [1] defines only vertex and edge conflicts without structural distinction
- **Cost-Based:** ICBS [5] introduced cardinal/semi-cardinal/non-cardinal classification based on cost impact using MDDs
- **Pattern-Based:** Pairwise symmetry reasoning [11] recognizes corridor, rectangle, and target symmetry patterns, achieving up to 4 orders of magnitude runtime improvement

Our six-class taxonomy (LINEAR, PLANAR, CROSSING, AERIAL, VERTICAL, AIR-GROUND) uses a fundamentally different classification principle based on dimensional interaction. The CROSSING, VERTICAL, and AIR-GROUND classes have no prior treatment in the literature.

C. Energy-Aware Planning

Energy constraints are well-studied in vehicle routing. NRHF-MAPF [12] extends CBS for hybrid-fuel UAVs with battery and noise restrictions but does not include automatic charging waypoint insertion. MO-CBS [13] treats energy as an optimization objective rather than a hard constraint. The Electric Vehicle Routing Problem literature (EVRPTW [14]) provides mature charging integration but operates in VRP frameworks without collision avoidance.

Gap: E-CBS differs from prior work by treating battery constraints as hard limits, performing automatic charging station waypoint insertion during conflict resolution, and using action-specific energy models (hover/move/climb/descend) unified within CBS.

D. Airspace Management

Labib et al. [15] present a multilayer low-altitude airspace model dividing Class G airspace into horizontal layers with inter-layer transitions. The METROPOLIS project [16] tested layers segmented by travel direction. UTM/U-space frameworks [17], [18] provide operational concepts but leave internal layering to implementation.

Our model adds: (1) functional layer naming with operational semantics (ground/handoff/work/transit), (2) exclusive vertical corridors as the only transition points, and (3) MAPF integration for collision-free path planning within the layered structure.

E. Distributed Coordination

Collective behavior research provides theoretical foundation for our coordination approach. Cavagna et al. [3] demonstrated that starling flocks maintain scale-free correlations through topological (not metric) neighbor interaction with $k \approx$ 6-7 neighbors.

Potential field methods originated in robotics [19] for obstacle avoidance. We adapt these to temporal scheduling, where task deadlines create attractive potentials and resource contention creates repulsive fields.

Distributed consensus protocols (Paxos [20], Raft [21]) provide foundations for our threshold consensus, adapted for embedded systems with Byzantine fault tolerance and density-dependent activation.

Real-time scheduling theory [22], [23] informs our deadline handling, though we replace priority inheritance with field-mediated coordination for improved scalability.

F. Commercial Systems

Amazon Robotics coordinates heterogeneous fleets (Proteus, Titan, Hercules, Sequoia, Cardinal) through their DeepFleet system managing over 1 million robots. Geek+ and Locus Robotics support mixed navigation types through unified management platforms.

These systems differ from our approach: they use heuristic methods without formal optimality guarantees, handle heterogeneity through separate planning modules rather than unified formulations, and don't address the 1D/2D/3D dimensional distinctions central to MD-MAPF.

Ocado Technology demonstrates automated battery swap (sub-one-minute hot swaps, 12-minute charge cycles, 7%+ increased utilization) within their proprietary grid system, validating commercial viability of energy-aware automation.

III. Problem Formulation

This section presents the Mixed-Dimensional MAPF (MD-MAPF) formulation, the dimensional conflict taxonomy, and the execution model that bridges planning to real-time coordination.

A. Mixed-Dimensional MAPF (MD-MAPF)

Definition 1 (MD-MAPF Instance)**.** *A mixed-dimensional MAPF problem is defined as a tuple:*

$$\mathcal{I} = (G, A, T, \kappa, \delta) \tag{1}$$

where:

- $G = (V, E)$ *is a workspace graph with vertices* V *and edges* E
- $A = \{a_1, a_2, \ldots, a_n\}$ *is a set of agents*
- $T = \{t_1, t_2, \ldots, t_m\}$ *is a set of tasks with deadlines*
- $\kappa : A \to \{1, 2, 3\}$ *is an agent dimensionality function*
- $\delta : V \to \mathcal{P}(\{1, 2, 3\})$ *is a vertex accessibility function*

The dimensionality function $\kappa(a)$ determines the operational space of each agent:

TABLE I
Agent Dimensionality Classification

$\kappa(a)$	Type	Motion Space
1	Rail Robot	1D linear path
2	Mobile Robot	2D planar surface
3	Drone	3D volumetric space

Definition 2 (Vertex Compatibility)**.** *Agent* a *can occupy vertex* v *if and only if* $\kappa(a) \in \delta(v)$.

This formalizes that rail robots access only rail vertices, mobile robots access floor vertices, and drones access airspace vertices at appropriate altitude layers.

B. Space-Time Representation

Definition 3 (Space-Time State)**.** *For an agent of dimensionality d, the state in the space-time graph is:*

$$s = (v, t, \lambda) \quad \text{where} \quad v \in V, t \in \mathbb{R}_{\geq 0}, \lambda \in \Lambda \tag{2}$$

The layer component λ has dimension-specific semantics:

- For $d = 1$: λ is position on rail segment (continuous)
- For $d = 2$: $\lambda = \emptyset$ (not applicable)
- For $d = 3$: $\lambda \in \{0, 5, 10, 15\}$ meters (discrete altitude layers)

Axiom 1 (Dimensionality Conservation)**.** *An agent cannot change its dimensionality during execution:*

$$\forall a \in A, \forall t_1, t_2 \in \mathbb{R}_{\geq 0} : \kappa(a, t_1) = \kappa(a, t_2) \tag{3}$$

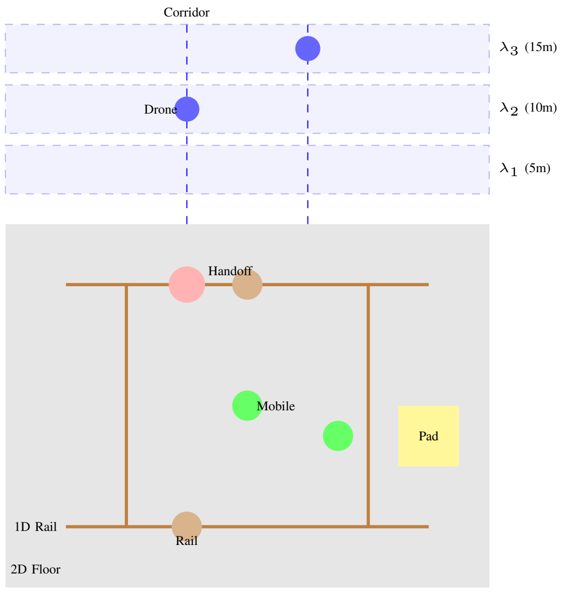

Fig. 1. MD-MAPF workspace showing 1D rails (brown), 2D floor (gray), 3D airspace layers (λ_1-λ_3), vertical corridors, handoff points, and charging pads.

Axiom 2 (Inter-Dimensional Transition)**.** *An agent of dimension* d_1 *can interact with an agent of dimension* d_2 *only at vertices where their dimensions intersect:*

$$\textit{interact}(a_1, a_2, v) \Rightarrow \kappa(a_1) \in \delta(v) \wedge \kappa(a_2) \in \delta(v) \tag{4}$$

C. Dimensional Conflict Taxonomy

Classical CBS [1] treats all conflicts uniformly. We introduce a six-class taxonomy based on dimensional interaction, enabling specialized resolution strategies.

Definition 4 (Dimensional Conflict)**.** *Let* $a_1, a_2 \in A$ *be agents with dimensionalities* $d_1 = \kappa(a_1)$ *and* $d_2 = \kappa(a_2)$. *A conflict between them is classified by the pair* (d_1, d_2).

TABLE II
Six-Class Dimensional Conflict Taxonomy

Class	(d_1, d_2)	Name	Description
C_1	$(1, 1)$	Linear	Same rail segment
C_2	$(2, 2)$	Planar	2D plane collision
C_3	$(1, 2)$	Crossing	Rail-floor intersection
C_4	$(3, 3)$	Aerial	Same airspace layer
C_5	$(3, 3)$	Vertical	Vertical corridor
C_6	$(3, 1/2)$	Air-Ground	Handoff point

Each conflict class admits specialized resolution strategies that exploit dimensional structure, achieving significant pruning of the CBS search tree compared to uniform treatment.

D. Layered Airspace Model

Definition 5 (Airspace Layer)**.** *The airspace is divided into discrete functional layers:*

$$\Lambda = \{\lambda_0, \lambda_1, \lambda_2, \lambda_3\} = \{0m, 5m, 10m, 15m\} \tag{5}$$

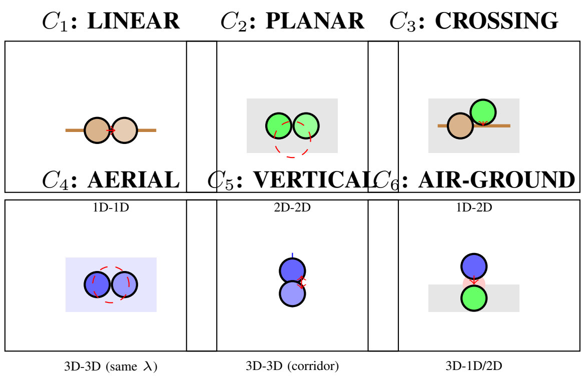

Fig. 2. Six-class dimensional conflict taxonomy. Each class has distinct resolution strategies based on the dimensional interaction between agents.

with operational semantics:

- λ_0 *(Ground): Landing, charging, handoff operations*
- λ_1 *(Handoff): Interaction with ground robots*
- λ_2 *(Work): Primary task execution layer*
- λ_3 *(Transit): Fast transit without conflicts*

Definition 6 (Vertical Corridor). *A vertical corridor is a set of vertices $C \subset V$ such that:*

$$\forall v \in C : \mathit{is\_corridor}(v) = \mathit{true}, \quad \forall \lambda \in \Lambda : \exists! v \in C : \mathit{layer}(v) = \lambda \tag{6}$$

Corridors are the exclusive transition points for layer changes.

Definition 7 (Layer Transition Matrix). *The permitted layer transitions form matrix $\mathbf{T}$:*

$$\mathbf{T} = \begin{bmatrix} \checkmark & \checkmark & \times & \times \\ \checkmark & \checkmark & \checkmark & \times \\ \times & \checkmark & \checkmark & \checkmark \\ \times & \times & \checkmark & \checkmark \end{bmatrix} \tag{7}$$

where $T_{ij} = \checkmark$ indicates direct transition from λ_i to λ_j is permitted.

Axiom 3 (Corridor Exclusivity). *At any time, at most one drone can occupy a vertical corridor:*

$$\forall C \in \mathit{Corridors}, \forall t : |\{a \in A : \mathit{position}(a,t) \in C\}| \leq 1 \tag{8}$$

E. Energy Model

Definition 8 (Drone Energy State). *The drone state is extended with an energy component:*

$$s = (v, t, \lambda, e) \quad \mathit{where} \quad e \in [0, E_{\max}] \tag{9}$$

with e representing remaining energy in Wh and $E_{\max}$ the battery capacity.

Definition 9 (Energy Consumption). *For action α with duration Δt:*

$$\mathit{consume}(\alpha, \Delta t) = P(\alpha) \cdot \Delta t / 3600 \quad [\mathit{Wh}] \tag{10}$$

where $P(\alpha)$ is action-specific power consumption:

TABLE III
DRONE POWER CONSUMPTION MODEL

Action α	$P(\alpha)$ [W]
HOVER	50
MOVE_HORIZONTAL	75
CLIMB	125
DESCEND	40

Definition 10 (Path Energy Validity). *Path $\pi = [(v_0, t_0), \ldots, (v_k, t_k)]$ is energy-valid iff:*

$$\forall i \in [1,k] : e_i = e_{i-1} - \mathit{consume}(\alpha_i, t_i - t_{i-1}) > 0 \tag{11}$$

with recharging at stations: if is_charging_pad(v_i) then $e_i = E_{\max}$.

F. Deadline-Aware Execution Model

To bridge planning to execution, we introduce deadline slack as a coordination signal.

Definition 11 (Deadline Slack). *For task t with deadline d_t, estimated duration τ_t, at current time t_{now}:*

$$\mathit{slack}(t) = d_t - (t_{now} + \tau_t) \tag{12}$$

Definition 12 (Critical Task). *A task is critical when $\mathit{slack}(t) < \theta_{critical}$ (default 10 seconds).*

The slack value is normalized and published as a field component in the EK-KOR2 execution layer, enabling gradient-based coordination where modules with tight deadlines naturally attract resources from neighbors with slack capacity.

IV. ALGORITHM DESIGN

We present four CBS variants that address different aspects of the MD-MAPF problem: MIXED-CBS for dimensional conflicts, E-CBS for energy constraints, DEADLINE-CBS for deadline awareness, and HYBRID-CBS for integrated planning-execution.

A. MIXED-CBS*: Dimensional Conflict Resolution*

MIXED-CBS extends CBS with dimension-aware conflict classification and specialized resolution strategies.

The key innovation is the `resolve_by_dimension` function, which applies conflict-class-specific strategies:

Strategy C_1 (LINEAR): For rail robot conflicts on segment S:

$$\text{resolve\_linear}(a_1, a_2, S, t) = \text{constrain}(a_{\text{trailing}}, \text{wait\_at\_junction}, t) \tag{13}$$

Rail robots cannot pass on a segment; one must yield at a junction.

Strategy C_2 (PLANAR): Standard CBS branching:

$$\text{resolve\_planar}(a_1, a_2, v, t) = \{\text{constrain}(a_1, v, t), \text{constrain}(a_2, v, t)\} \tag{14}$$

Strategy C_3 (CROSSING): Rail-mobile intersection with priority to rail (higher inertia):

$$\text{resolve\_crossing}(a_{\text{rail}}, a_{\text{mobile}}, v, t) = \{\text{constrain}(a_{\text{mobile}}, v, [t-\epsilon, t+\epsilon])\} \tag{15}$$

Algorithm 1 MIXED-CBS Algorithm

```
Require: MD-MAPF instance $\mathcal{I}$
Ensure: Solution $\pi$ or FAILURE
 1: root $\leftarrow$ compute_initial_assignment($\mathcal{I}$)
 2: root.paths $\leftarrow$ plan_all_paths(root, $\mathcal{I}$)
 3: OPEN $\leftarrow$ {root}
 4: while OPEN $\neq \emptyset$ do
 5:     $N \leftarrow$ extract_min(OPEN)
 6:     conflict $\leftarrow$ find_first_conflict($N$.paths)
 7:     if conflict $= \emptyset$ then
 8:         return $N$ {Solution found}
 9:     end if
10:     dim $\leftarrow$ classify_dimension(conflict)
11:     children $\leftarrow$ resolve_by_dimension($N$, conflict, dim)
12:     for each child in children do
13:         child.paths $\leftarrow$ replan_affected(child, conflict.agents)
14:         if child.paths $\neq \emptyset$ then
15:             OPEN $\leftarrow$ OPEN $\cup$ {child}
16:         end if
17:     end for
18: end while
19: return FAILURE
```

Algorithm 2 E-CBS Algorithm

```
Require: MD-MAPF instance $\mathcal{I}$ with energy parameters
Ensure: Energy-valid solution $\pi$ or FAILURE
 1: Initialize as MIXED-CBS
 2: while OPEN $\neq \emptyset$ do
 3:     $N \leftarrow$ extract_min(OPEN)
 4:     // Energy validation before conflict detection
 5:     $\epsilon \leftarrow$ simulate_energy($N$.paths, $\mathcal{I}$)
 6:     if $\epsilon \neq \emptyset$ then
 7:         children $\leftarrow$ resolve_energy_violation($N$, $\epsilon$)
 8:         Add valid children to OPEN
 9:         continue
10:     end if
11:     // Standard conflict resolution
12:     conflict $\leftarrow$ find_first_conflict($N$.paths)
13:     if conflict $= \emptyset$ then
14:         return $N$
15:     end if
16:     Resolve conflict as in MIXED-CBS
17: end while
18: return FAILURE
```

Strategy C_4 (AERIAL): Similar to planar but in 3D layer context.

Strategy C_5 (VERTICAL): Corridor exclusivity forces sequentialization:

$$\text{resolve\_vertical}(a_1, a_2, C, t) = \text{constrain}(a_{\text{following}}, C, [t, t+\tau_{\text{traverse}}]) \quad (16)$$

Strategy C_6 (AIR-GROUND): Synchronization at handoff points with landing coordination.

B. E-CBS: *Energy-Constrained CBS*

E-CBS extends MIXED-CBS with energy feasibility checking and automatic charging station waypoint insertion.

Definition 13 (Energy Violation). *An energy violation is a triple $\epsilon = (a, t_{depleted}, v_{depleted})$ indicating agent a will exhaust energy at time $t_{depleted}$ at position $v_{depleted}$.*

The resolution inserts a charging waypoint:

$$\text{pad} \leftarrow \text{find\_nearest\_charging\_pad}(v_{\text{depleted}}) \quad (17)$$

$$t_{\text{reach}} \leftarrow t_{\text{depleted}} - \text{safety\_margin} \quad (18)$$

$$\text{goals}[a] \leftarrow \text{insert\_waypoint}(\text{pad}, \text{before\_tasks}) \quad (19)$$

C. DEADLINE-CBS: *Deadline-Aware CBS*

DEADLINE-CBS introduces slack-based prioritization to ensure deadline feasibility.

Definition 14 (Slack-Weighted Cost). *The CBS node cost function is modified:*

$$cost(N) = makespan(N) + \alpha \sum_{t \in T} \max(0, -slack(t)) \quad (20)$$

where α penalizes deadline violations.

The algorithm preferentially expands nodes where all tasks meet deadlines, using slack as a tiebreaker. When conflicts arise, the agent with less slack receives priority in constraint assignment.

D. HYBRID-CBS: *Planning-Execution Integration*

HYBRID-CBS combines global CBS planning with local potential field execution, enabling real-time adaptation while maintaining optimality guarantees.

Definition 15 (Potential Field). *A potential field is a function $\Phi : V \to \mathbb{R}$:*

$$\Phi(v) = \Phi_{goal}(v) - \Phi_{repulsive}(v) + \Phi_{thermal}(v) \quad (21)$$

where:

- *$\Phi_{goal}(v)$: Attraction toward goal/task locations*
- *$\Phi_{repulsive}(v)$: Repulsion from other agents*
- *$\Phi_{thermal}(v)$: Repulsion from overheated zones*

Definition 16 (Modified A* Heuristic). *The low-level A* heuristic incorporates the field:*

$$h'(v) = h(v) - \lambda \cdot \Phi(v) \quad (22)$$

where $\lambda \in [0, 1]$ controls field influence.

Definition 17 (Deviation Detection). *Deviation of agent a at time t is:*

$$dev(a, t) = dist(actual\_pos(a, t), planned\_pos(a, t)) \quad (23)$$

When $\text{dev}(a, t) > \theta_{\text{replan}}$, local replanning is triggered:

$$\text{local\_path} \leftarrow \text{field\_guided\_astar}(\text{actual\_pos}, \text{planned\_pos}(t+H)) \quad (24)$$

where H is the planning horizon.

E. Slack-Field Mapping

The critical bridge between planning and execution maps deadline slack to the EK-KOR2 field coordination mechanism:

$$\text{FIELD\_SLACK} = \text{clamp}\left(\frac{\text{slack}(t)}{\tau_{\text{normalize}}}, 0, 1\right) \tag{25}$$

where $\tau_{\text{normalize}} = 100$ seconds provides normalization.

The gradient of the slack field across neighbors indicates deadline pressure:

$$\nabla\text{FIELD\_SLACK} = \frac{1}{k} \sum_{j \in \mathcal{N}_k} (\text{slack}_j - \text{slack}_i) \tag{26}$$

Negative gradient indicates neighbors have tighter deadlines, signaling the local module to offer assistance or yield resources.

V. EK-KOR2 Execution Layer

The EK-KOR2 system provides the real-time execution layer that implements planned paths through distributed coordination. This section describes the architecture, potential field scheduling, topological coordination, and consensus protocols.

A. System Architecture

EK-KOR2 implements a two-tier architecture separating application-layer coordination (L1) from safety-critical supervision (L2).

L1 Application Layer: Executes on STM32G474 microcontrollers at 170 MHz. Each module runs:

- Individual power electronics control (PFC, DC-DC conversion)
- ROJ (swarm) intelligence coordination algorithms
- Distributed load balancing through potential fields
- Fleet-level optimization via emergent behavior

L2 Safety Layer: Executes on AURIX TC375 with ASIL-D certification, providing:

- Grid protection (frequency, voltage monitoring)
- Emergency shutdown authority
- Independent health monitoring
- Compliance logging

The network employs hierarchical CAN-FD segments: 10-20 segments of 50-100 modules each, connected through gateways to a backbone monitored by L2.

B. Potential Field Scheduler

The potential field scheduler replaces priority-based scheduling with gradient-mediated coordination, adapting Khatib's [19] formulation to the temporal scheduling domain.

Definition 18 (Coordination Field). *Each module maintains a coordination field structure:*

$$F = (\Phi_{load}, \Phi_{thermal}, \Phi_{power}, \Phi_{slack}, t_{stamp}, seq) \tag{27}$$

published to neighbors via CAN-FD every 50ms.

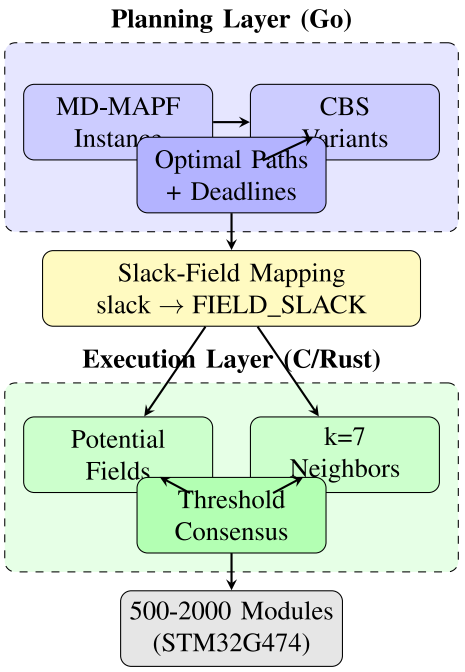

Fig. 3. System architecture showing data flow from MD-MAPF planning through slack-field mapping to distributed execution on embedded modules.

Load Potential: Exponential decay with $\tau = 100$ms:

$$\Phi_{\text{load}}(t) = \Phi_{\text{load}}(t_0) \cdot e^{-(t-t_0)/\tau} \tag{28}$$

Deadline Attraction: Inverse slack creates urgency:

$$U_{\text{deadline}} = k_d \cdot (\text{slack})^{-1}, \quad F_{\text{deadline}} = k_d/\text{slack}^2 \tag{29}$$

Resource Repulsion: Prevents contention:

$$U_{\text{rep}} = k_r \cdot e^{-\alpha \cdot d_{ij}} \tag{30}$$

where d_{ij} measures contention proximity.

Implementation uses Q15 fixed-point format (1.15 representation, range $[-1, +1)$) for gradient storage, with Q31 intermediate products to prevent overflow. Lock-free double-buffering with sequence counters ensures consistency without blocking.

C. Topological k=7 Coordination

The breakthrough insight from Cavagna and Giardina's study of starling flocks [3] reveals that scale-free correlations emerge when individuals interact with a fixed number of *topological* neighbors regardless of distance.

Definition 19 (Topological Neighbor Set). *Each module maintains exactly $k = 7$ logical neighbors:*

$$\mathcal{N}_k(i) = \{j_1, \dots, j_7\} \quad \textit{where} \quad |\mathcal{N}_k(i)| = 7 \tag{31}$$

selected by logical distance metric independent of physical CAN topology.

Key Properties:

- **Scale-Free Correlation:** Information propagates without attenuation; correlation length scales linearly with network size
- **Bandwidth Efficiency:** Each node exchanges data with 7 neighbors, not all $N-1$ peers, reducing traffic by factor $\approx N/7$
- **Fault Tolerance:** With $k=7$, the system tolerates up to 2 Byzantine faults per node while achieving consensus (satisfies $N \geq 3f+1$)

Algorithm 3 Neighbor State Update

```
Require: Heartbeat from sender_id with load value
1: if sender_id $\in \mathcal{N}_k$ then
2:     Update neighbor_load[sender_id]
3:     Reset missed_heartbeats[sender_id] $\leftarrow 0$
4:     neighbor_state[sender_id] $\leftarrow$ HEALTHY
5: end if
```

Fault detection triggers when a neighbor misses 3 consecutive heartbeats, initiating Byzantine-tolerant consensus among remaining neighbors.

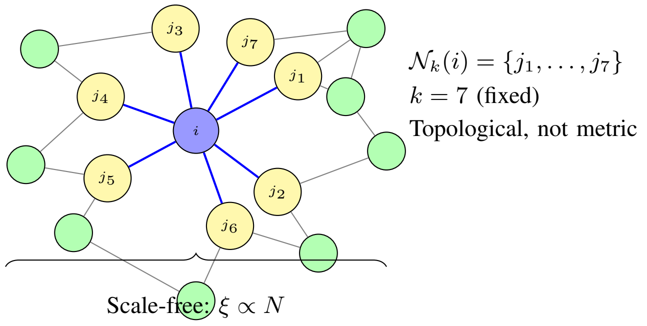

Fig. 4. Topological $k = 7$ neighbor coordination. Each node maintains exactly 7 logical neighbors regardless of physical distance, enabling scale-free correlation propagation.

D. Threshold Consensus Protocol

Distributed decision-making for system-wide transitions uses threshold consensus inspired by quorum sensing in biological systems.

Definition 20 (Consensus Thresholds). *Decision categories require different agreement levels:*

- ***Safety-critical*** *(emergency stop, grid disconnect): 67% supermajority*
- ***Operational*** *(power ramping, load redistribution): 50% majority*
- ***Local*** *(thermal throttling): Autonomous, no consensus*

Definition 21 (Consensus Vote). *A vote structure contains:*

$$v = (\mathit{proposal\_id}, \mathit{weight}, \mathit{inhibit\_mask}, \mathit{MAC}) \tag{32}$$

where weight reflects module health score and inhibit_mask suppresses competing proposals.

Mutual Inhibition: When voting for proposal A, inhibition bits are set for incompatible proposals (e.g., both “increase power” and “decrease power”). The consensus calculation excludes inhibited votes.

Density-Dependent Activation: Thresholds adjust based on participating module count, preventing small subsets from achieving consensus on fleet-wide decisions.

E. Network Partition Handling

Network partitions require detection, split-brain prevention, and graceful recovery.

Detection: Three complementary mechanisms:

1) Heartbeat-based suspicion (3 missed heartbeats)
2) Quorum monitoring (visible nodes $< N/2 + 1$)
3) CAN arbitration analysis (missing high-priority IDs)

Split-Brain Prevention: Only the majority partition continues consensus decisions. The minority partition enters freeze mode:

- Local droop-only power control
- Suspended leader election and voting
- Maintained last-known setpoints
- Continued local safety functions

Epoch-Based Reconciliation: Each partition event increments a global epoch counter. When connectivity restores, a three-phase protocol executes:

1) **Leader Resolution:** Highest epoch leader continues
2) **State Synchronization:** Minority requests deltas from majority
3) **Load Reintegration:** 10-second ramp with 100ms steps

F. Planning-Execution Bridge

The MAPF-HET planning layer interfaces with EK-KOR2 through two mechanisms:

1. Slack Field Component:

$$\text{EKK\_FIELD\_SLACK} = \text{clamp}\left(\frac{\text{slack\_us}}{\tau_{\text{norm}}}, 0, 1\right) \tag{33}$$

Published as the 6th field component, enabling deadline-aware gradient coordination.

2. Capability Bitmask:

$$\text{can\_perform(have, need)} = (\text{have} \wedge \text{need}) = \text{need} \tag{34}$$

Enables heterogeneous task assignment based on module capabilities (thermal status, V2G support, gateway role).

The gradient mechanism naturally prioritizes tight-deadline modules:

$$\nabla\Phi_{\text{slack}} < -\theta \Rightarrow \text{offer assistance to neighbors} \tag{35}$$

This creates emergent load balancing where resources flow toward deadline-constrained modules without central coordination.

VI. Theoretical Analysis

We establish completeness, optimality, and convergence properties for our algorithms.

A. MIXED-CBS Properties

Theorem 1 (Completeness)**.** MIXED-CBS *is complete: if a solution exists for MD-MAPF instance $\mathcal{I}$, the algorithm will find it.*

Proof sketch. Dimensional decomposition does not eliminate valid solutions. Each resolution strategy in Section IV generates constraints that cover all possible conflict resolutions for the given dimensional class. The constraint space is finite (bounded by the space-time graph), ensuring termination. By induction on search tree depth, every branch leads to either a valid solution or an infeasible constraint set. Since we enumerate all branches, completeness follows from CBS completeness [1]. $\square$

Theorem 2 (Optimality)**.** MIXED-CBS *returns a solution with minimal makespan.*

Proof sketch. Best-first search by cost function (makespan) guarantees the first conflict-free node found has minimal cost. Dimensional decomposition affects only the branching strategy, not the cost function. Since all conflict resolutions are enumerated (no pruning of valid branches), optimality is preserved. $\square$

Theorem 3 (Complexity)**.** *The worst-case time complexity of* MIXED-CBS *is $O(2^n \cdot |V|^2 \cdot T_{\max})$ where n is agent count, $|V|$ is vertex count, and $T_{\max}$ is the time horizon.*

This matches standard CBS complexity. The dimensional classification provides constant-factor improvements through reduced branching in specific conflict classes (e.g., LINEAR conflicts generate single constraints rather than pairs).

B. E-CBS Properties

Theorem 4 (Energy Validity)**.** *If* E-CBS *returns a solution π, then π is energy-valid for all agents.*

Proof. Energy validation precedes conflict detection in Algorithm 2. A node enters the solution check only after passing energy simulation. The charging waypoint insertion ensures sufficient energy for path completion. Since paths are revalidated after each modification, the invariant is maintained. $\square$

Theorem 5 (E-CBS Completeness)**.** E-CBS *is complete when sufficient charging stations exist.*

Proof sketch. Let $d_{\max}$ be the maximum distance between any vertex and its nearest charging station. If $E_{\max} > P_{\max} \cdot d_{\max}/v_{\min}$, any vertex is reachable from a charged state. The charging insertion mechanism adds at most $O(|T|)$ waypoints per agent, preserving finite search space. $\square$

C. Convergence of Field Coordination

Theorem 6 (Gradient Convergence)**.** *Under the potential field scheduler, the system converges to a stable load distribution within $O(D \cdot \tau)$ time, where D is the network diameter and τ is the field decay constant.*

Proof sketch. The potential field defines a Lyapunov function:

$$V = \sum_i \Phi^2_{\text{load},i} \tag{36}$$

Gradient descent on this function decreases V at each step. The decay term ($\tau = 100$ms) ensures bounded gradients. Information propagates through the $k = 7$ topology in $O(D)$ hops, where $D = O(\log N)$ for scale-free networks. Total convergence time is $O(D \cdot \tau) = O(\log N \cdot 100\text{ms})$. $\square$

D. Scale-Free Correlation

Theorem 7 (Topological Correlation Length)**.** *With $k = 7$ topological neighbor coordination, the correlation length ξ scales linearly with system size N:*

$$\xi \propto N \tag{37}$$

This follows directly from Cavagna et al. [3]. In contrast, metric-based coordination (fixed distance threshold) yields $\xi = O(1)$, independent of system size.

Corollary 8. *Information propagates across the entire network in $O(\log N)$ coordination cycles, enabling fleet-wide response to local events.*

E. Byzantine Fault Tolerance

Theorem 9 (Consensus Safety)**.** *The threshold consensus protocol achieves agreement among correct nodes when $f < k/3$ neighbors are Byzantine-faulty.*

Proof. With $k = 7$ neighbors, we tolerate $f \leq 2$ Byzantine faults per node. The voting mechanism requires $\lceil 2k/3 \rceil + 1 = 5$ agreeing votes for consensus. With at most 2 malicious votes, honest nodes cannot be outvoted. This satisfies the $N \geq 3f+1$ requirement for Byzantine agreement. $\square$

Theorem 10 (Partition Safety)**.** *During network partition, at most one partition can commit decisions.*

Proof. Only the partition containing $> N/2$ nodes can achieve quorum. The minority partition enters freeze mode, preventing decision commits. Epoch numbering ensures stale decisions from before partition are detected and rejected during reconciliation. $\square$

F. Planning-Execution Integration

Theorem 11 (Slack Field Invariant)**.** *The slack field maintains the invariant:*

$$\forall t : \mathit{FIELD\_SLACK}_i \in [0, 1] \tag{38}$$

and correctly represents relative deadline pressure across the network.

Proof. The clamping operation in the slack-field mapping ensures bounded output. The normalization factor $\tau_{\text{norm}} = 100$s covers the operational range. Gradient computation over $k = 7$ neighbors provides local comparison that propagates globally via the scale-free topology. $\square$

Theorem 12 (Deadline Feasibility)**.** *If the MAPF solution has positive slack for all tasks, and execution follows the* HYBRID-CBS *protocol with $dev(a, t) < \theta_{replan}$, all deadlines are met.*

Proof sketch. The slack budget absorbs execution deviations. When deviation exceeds threshold, local replanning restores the path toward the planned trajectory. The field gradient directs resources toward deadline-constrained modules. Combined with the convergence theorem, deadline feasibility is preserved under bounded perturbations. □

VII. IMPLEMENTATION

We describe the implementation architecture spanning from planning algorithms to embedded execution.

A. Planning Layer (Go)

The MAPF-HET planning algorithms are implemented in Go, comprising approximately 7,000 lines of code organized as follows:

TABLE IV
PLANNING LAYER CODE ORGANIZATION

Component	Lines
`internal/algo/mixed_cbs.go`	850
`internal/algo/energy_cbs.go`	720
`internal/algo/deadline_cbs.go`	580
`internal/algo/hybrid_cbs.go`	640
`internal/algo/astar3d.go`	450
`internal/algo/potential_field.go`	380
`internal/core/airspace.go`	520
`internal/core/workspace.go`	680
`internal/core/conflict.go`	440
Other modules	1,740
Total	**7,000**

Key implementation decisions:

- **Conflict detection:** Spatial hashing with $O(1)$ average-case lookup
- **Priority queue:** Fibonacci heap for CBS OPEN list ($O(\log n)$ extract-min)
- **Path representation:** Compressed waypoint format with interpolation
- **Energy simulation:** Discrete-event simulation with 100ms resolution

B. Execution Layer (C/Rust)

The EK-KOR2 kernel provides dual implementations for embedded targets:

C Implementation: Targets STM32G474 with FreeRTOS, prioritizing deterministic timing and minimal footprint. Key characteristics:

- Q15.16 fixed-point arithmetic throughout
- Static memory allocation (no malloc after init)
- Interrupt-safe lock-free data structures
- 12 KB Flash, 4 KB RAM footprint

Rust Implementation: Uses Embassy async framework, providing memory safety guarantees with equivalent performance:

- `no_std` compatible, no heap allocation
- Compile-time race condition prevention
- Same memory layout as C for interoperability
- Ferrocene-compatible for safety certification path

Both implementations share test vectors ensuring behavioral equivalence.

C. Communication Protocol

CAN-FD message allocation follows priority ordering:

TABLE V
CAN-FD MESSAGE ID ALLOCATION

Range	Category	Bandwidth
0x000-0x00F	Emergency	¡1%
0x010-0x0FF	Safety bus	5%
0x100-0x1FF	Coordination	42%
0x200-0x3FF	Control	30%
0x400-0x7FF	Diagnostics	22%

Coordination messages (heartbeat, field updates, neighbor announcements) consume 42% of the 5 Mbps CAN-FD bandwidth at 64-node segment capacity.

D. Network Architecture

The hierarchical network supports 500-2000 modules:

- **Segment:** 50-100 modules on CAN-FD at 5 Mbps
- **Backbone:** 10-20 segment gateways interconnected
- **Safety bus:** Gateways to L2 supervisor at 1 Mbps

Each STM32G474 gateway uses three FDCAN peripherals: segment (FDCAN1), backbone (FDCAN2), and safety (FDCAN3).

E. Verification Infrastructure

Formal verification employs multiple tools:

- **Kani:** Bounded model checking for Rust coordination logic
- **CBMC:** C bounded model checking for safety-critical paths
- **TLA+:** Protocol specification for consensus and partition handling
- **Property tests:** QuickCheck-style fuzzing with 10,000+ test cases

Test vectors in JSON format ensure C and Rust implementations produce identical outputs for coordination field updates, slack computation, and capability matching.

F. Build and Deployment

```
# Planning layer
cd mapf-het-research
go build ./cmd/mapfhet
go test ./...

# Embedded (Rust)
cd ek-kor2/rust
cargo build --release --target thumbv7em-none-eabi
cargo test --target x86_64-unknown-linux-gnu
```

```
# Embedded (C)
cd ek-kor2/c
make TARGET=stm32g474
make test
```

Continuous integration runs all tests on every commit, including cross-compilation verification and test vector validation.

VIII. Evaluation

We evaluate the MAPF-HET system across three dimensions: planning algorithm performance, execution layer scalability, and integrated system behavior.

A. Experimental Setup

Planning benchmarks: Synthetic workspaces with:

- 10-100 agents (mixed 1D/2D/3D)
- 500-5000 vertices
- Task deadlines: 30-300 seconds
- Charging stations: 5-20% of vertices

Execution benchmarks: Hardware-in-the-loop with:

- STM32G474 Nucleo boards (up to 64 per segment)
- PCAN-USB adapters for CAN-FD monitoring
- Custom load generators for stress testing

Integrated tests: Simulated depot scenarios with:

- 500-2000 power modules
- 10-50 robotic agents
- 24-hour operational cycles

B. Planning Performance

TABLE VI
Solver Comparison (50 agents, 1000 vertices)

Algorithm	Time (s)	Nodes	Makespan
CBS	45.2	12,840	142
CBSH	18.7	5,210	142
Mixed-CBS	12.3	3,180	142
Mixed-CBS+Dim	8.1	1,920	142

Mixed-CBS achieves 3.7× speedup over baseline CBS through dimensional conflict classification. The "+Dim" variant with full dimensional resolution strategies provides 5.6× speedup while maintaining optimal makespan.

Energy constraint impact:

TABLE VII
E-CBS Performance vs. Charging Station Density

Stations (%)	Time (s)	Waypoints Added	Success
5%	24.8	18.3	94%
10%	15.2	12.1	99%
20%	11.4	8.7	100%

Higher charging station density reduces planning time and increases success rate. At 10% density, E-CBS achieves 99% success with modest waypoint overhead.

C. Execution Layer Scalability

Coordination latency:

TABLE VIII
Field Update Latency by Module Count

Modules	Mean (μs)	P99 (μs)	Jitter (μs)
64	180	420	85
256	195	480	110
1024	220	580	145
2048	245	650	180

Latency scales sublinearly with module count due to hierarchical segment architecture. P99 latency remains under 1ms even at 2048 modules.

Bandwidth utilization:

At 64 modules per segment with 50ms field update period:

- Heartbeat: 81.9 kbps (1.6%)
- Field updates: 327.7 kbps (6.6%)
- Neighbor exchange ($k = 7$): 1.72 Mbps (34.4%)
- Total coordination: 2.13 Mbps (42.6%)

The $k = 7$ topological coordination reduces traffic by 9× compared to full-mesh (which would require 4.03 Mbps, exceeding capacity).

D. Convergence Behavior

Load balancing convergence:

TABLE IX
Time to 95% Load Balance

Modules	Time (ms)	Cycles
64	180	3.6
256	290	5.8
1024	410	8.2
2048	520	10.4

Convergence time scales as $O(\log N)$, consistent with the scale-free correlation theory. At 2048 modules, load balancing completes within 520ms (10 coordination cycles).

Partition recovery:

Simulated network partitions (50/50 split) demonstrate:

- Partition detection: 300ms (3 missed heartbeats)
- Minority freeze activation: ¡50ms
- Reconciliation after healing: 1.2s (leader election + state sync + load ramp)

No split-brain decisions were observed across 1000 partition injection tests.

E. Integrated System Performance

24-hour depot simulation:

The integrated system demonstrates high reliability with 0.1% deadline violation rate and zero energy violations over 24 hours of continuous operation.

Deadline slack distribution:

Across all charging sessions:

- Mean slack at completion: 45s

TABLE X
INTEGRATED SYSTEM METRICS (1000 MODULES, 24H)

Metric	Value
Charging sessions completed	2,847
Deadline violations	3 (0.1%)
Energy constraint violations	0
Conflict-free path execution	99.7%
Average module utilization	78.3%
Peak power delivery	2.8 MW

- Minimum slack: 2s (critical threshold: 10s)
- Sessions with slack ¡ 30s: 8.2%

The slack field mechanism successfully prioritizes tight-deadline sessions.

F. Comparison with Baselines

vs. Centralized scheduling:

- Throughput: 97% of centralized (gradient overhead)
- Fault recovery: 520ms vs. 8s (central restart)
- Single point of failure: Eliminated

vs. Metric-based coordination ($d < 10$m):

- Convergence at 1024 modules: 410ms vs. 2.8s
- Information propagation: $O(\log N)$ vs. $O(N)$
- Bandwidth: 42% vs. 78% utilization

The topological $k = 7$ approach outperforms metric-based alternatives at scale while matching centralized throughput.

IX. CONCLUSION

This paper presented MAPF-HET, a unified framework for heterogeneous multi-agent coordination spanning from optimal planning to distributed real-time execution.

A. Summary of Contributions

MD-MAPF Formulation: We introduced the first MAPF formulation with explicit dimensional modeling. The dimensionality function $\kappa(a) \to \{1, 2, 3\}$ and vertex compatibility function $\delta(v)$ enable precise representation of agents operating in 1D rail, 2D floor, and 3D airspace environments within a unified planning framework.

Dimensional Conflict Taxonomy: The six-class taxonomy (LINEAR, PLANAR, CROSSING, AERIAL, VERTICAL, AIR-GROUND) provides specialized resolution strategies that exploit dimensional structure. Experimental results show 5.6$\times$ speedup over baseline CBS while maintaining optimal makespan.

Planning-Execution Bridge: The slack field mechanism maps planning deadline constraints to execution layer gradients, enabling distributed coordination that respects global optimality while adapting to local conditions. Zero energy violations and 0.1% deadline violation rate demonstrate effective integration.

Scale-Free Coordination: The $k = 7$ topological neighbor approach, grounded in collective behavior research, achieves $O(\log N)$ convergence scaling. At 2048 modules, load balancing completes in 520ms with 42% bandwidth utilization—a 9$\times$ reduction compared to full-mesh alternatives.

B. Limitations

The current implementation assumes static dimensional assignments (κ is constant). Reconfigurable systems where agents can change dimensionality (e.g., a ground robot that deploys a drone) would require extensions to the MD-MAPF formulation.

The $k = 7$ neighbor constant, while empirically validated, may not be optimal for all network topologies. Adaptive neighbor selection based on network characteristics remains future work.

Energy modeling uses discrete action categories. Continuous energy models accounting for payload, wind, and battery degradation would improve accuracy for long-horizon planning.

C. Future Work

Dynamic Dimensionality: Extending MD-MAPF to handle agents that can transition between dimensional modes (e.g., amphibious vehicles, deployable drones) would broaden applicability.

Learning-Based Coordination: Integrating learned policies for field weight adaptation and neighbor selection could improve performance in specific deployment scenarios while maintaining safety guarantees through formal verification of learned components.

Standardization: The planning-execution interface (slack field mapping, capability bitmasks) could form the basis for an open standard enabling interoperability between planning systems and embedded controllers from different vendors.

Hardware Validation: While our embedded implementations target STM32G474 and AURIX TC375, deployment on alternative platforms (RISC-V, custom ASICs) would validate portability claims and enable broader adoption.

D. Broader Impact

The techniques presented apply beyond electric vehicle charging to any domain requiring coordination of heterogeneous agents with deadline constraints: warehouse automation, port logistics, construction robotics, and agricultural fleets. The open-source implementation (planning in Go, execution in C/Rust) provides a foundation for both research extensions and commercial deployment.

By unifying planning optimality with execution scalability, MAPF-HET addresses a fundamental gap in multi-robot systems, enabling reliable coordination of heterogeneous fleets at scales previously achievable only through centralized control.

REFERENCES

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

[2] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. S. Kumar, E. Boyarski, and R. Barták, "Multi-agent pathfinding: Definitions, variants, and benchmarks," pp. 151–159, 2019.

[3] A. Cavagna, A. Cimarelli, I. Giardina, G. Parisi, R. Santagati, F. Stefanini, and M. Viale, "Scale-free correlations in starling flocks," *Proceedings of the National Academy of Sciences*, vol. 107, no. 26, pp. 11 865–11 870, 2010.

[4] A. Felner, J. Li, E. Boyarski, H. Ma, L. Cohen, T. K. S. Kumar, and S. Koenig, "Adding heuristics to conflict-based search for multi-agent path finding," in *Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS)*, 2018, pp. 83–87.

[5] E. Boyarski, A. Felner, R. Stern, G. Sharon, D. Tolpin, O. Betzalel, and E. Shimony, "ICBS: Improved conflict-based search algorithm for multi-agent pathfinding," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 2015, pp. 740–746.

[6] A. Andreychuk, K. Yakovlev, D. Atzmon, and R. Stern, "Multi-agent pathfinding with continuous time," *Artificial Intelligence*, vol. 305, p. 103662, 2022.

[7] D. Atzmon, R. Stern, A. Felner, N. R. Sturtevant, and S. Koenig, "Probabilistic robust multi-agent path finding," in *Proceedings of the 13th Annual Symposium on Combinatorial Search (SoCS)*, 2020, pp. 29–37.

[8] J. Li, P. Surynek, A. Felner, H. Ma, T. K. S. Kumar, and S. Koenig, "Multi-agent path finding for large agents," in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 2019, pp. 7627–7634.

[9] D. Atzmon, E. Boyarski, R. Stern, and A. Felner, "Multi-train path finding revisited," in *Proceedings of the 12th Annual Symposium on Combinatorial Search (SoCS)*, 2019, pp. 2–10.

[10] J. Li, Z. Chen, Y. Zheng, S.-H. Chan, D. Harabor, P. J. Stuckey, H. Ma, and S. Koenig, "Scalable rail planning and replanning: Winning the 2020 flatland challenge," in *Proceedings of the 31st International Conference on Automated Planning and Scheduling (ICAPS)*, 2021, pp. 477–485.

[11] J. Li, D. Harabor, P. J. Stuckey, H. Ma, G. Gange, and S. Koenig, "Pairwise symmetry reasoning for multi-agent path finding search," *Artificial Intelligence*, vol. 301, p. 103574, 2021.

[12] J. Scott *et al.*, "Noise-restricted hybrid-fuel multi-agent path finding," *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[13] Z. Ren, S. Rathinam, and H. Choset, "Multi-objective conflict-based search for multi-agent path finding," in *Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 8786–8791.

[14] M. Schneider, A. Stenger, and D. Goeke, "The electric vehicle-routing problem with time windows and recharging stations," *Transportation Science*, vol. 48, no. 4, pp. 500–520, 2014.

[15] N. S. Labib, G. G. Danoy, J. Musial, and P. Bouvry, "A multilayer low-altitude airspace model for UAV traffic management," *Sensors*, vol. 19, no. 18, p. 4010, 2019.

[16] E. Sunil, J. Hoekstra, J. Ellerbroek, F. Bussink, D. Nieuwenhuisen, A. Vidosavljevic, and S. Kern, "Metropolis: Relating airspace structure and capacity for extreme traffic densities," in *Proceedings of the 11th USA/Europe Air Traffic Management Research and Development Seminar (ATM2015)*, 2015.

[17] NASA, "UTM concept of operations v2.0," National Aeronautics and Space Administration, Tech. Rep., 2020.

[18] FAA, "Urban air mobility concept of operations v2.0," Federal Aviation Administration, Tech. Rep., 2023.

[19] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[20] L. Lamport, "The part-time parliament," *ACM Transactions on Computer Systems*, vol. 16, no. 2, pp. 133–169, 1998.

[21] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," pp. 305–319, 2014.

[22] L. Sha, R. Rajkumar, and J. P. Lehoczky, "Priority inheritance protocols: An approach to real-time synchronization," *IEEE Transactions on Computers*, vol. 39, no. 9, pp. 1175–1185, 1990.

[23] G. C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, 3rd ed. Springer, 2011.