# TeamWeaver: Hybrid LLM and Optimization-based Planning with Transparent Constraints in Heterogeneous Multi-Robot Teams

***Abstract*—Multi-robot task allocation in heterogeneous teams remains challenging due to dynamic environments and semantic ambiguity in natural language instructions. We propose *TeamWeaver*, a hybrid framework that integrates large language models with mixed-integer quadratic programming for robust, interpretable, and real-time task allocation. Language models parse unstructured instructions into structured task graphs, while the optimizer ensures feasibility under robot capabilities, resource constraints, and spatial-temporal dependencies. A closed-loop mechanism enables online capability learning and replanning under perturbations. We evaluate the framework on three datasets—PARTNR-val-mini, Dynamic-TeamWeaver (83 tasks, 4 scenarios), and real-world deployments (5 tasks, 40+ trials)—and show that it consistently outperforms baselines., achieving up to 10.6% higher success rate, 6.2% better completion, and 18.9% fewer planning steps on PARTNR-val-mini, 49.3% relative improvement under ambiguous conditions, and above 90% task success in real-world tests with up to 18.6% gain under communication failure. Transparency metrics remain above 0.66 with fewer than 1.2 replanning cycles per task. These results validate TeamWeaver as a scalable, interpretable, and deployable solution for heterogeneous multi-robot coordination in complex, real-world settings. The supplementary video, dataset and code are available online[1].**

## I. Introduction

### A. Background and Motivation

Multi-robot task allocation (MRTA) is a fundamental problem in enabling effective collaboration among robot teams [1]–[4]. It is particularly critical in dynamic and unstructured scenarios, where tasks may need to be reassigned due to environmental changes or fluctuating resources [5]. Recent years have witnessed a rapid diversification of robotic platforms, driven by advances in morphology design, embodied intelligence, and specialized operating capabilities [6]–[8]. These developments have led to the widespread formation of heterogeneous multi-robot systems (HMRS), in which robots with complementary sensing, mobility, and manipulation skills operate together [9]–[11]. Such heterogeneity greatly enhances the potential of robot teams in real-world environments, enabling them to tackle more diverse, specialized, and resource-intensive tasks than homogeneous systems. However, it also poses significant challenges for MRTA. Differences in robot capabilities, resource constraints, and feasible action spaces complicate the design of allocation strategies that remain efficient, adaptive, and reliable under dynamic and uncertain conditions [3], [11], [12]. As recent studies highlight, robust task allocation in HMRS continues to be a central yet unresolved challenge in multi-robot coordination [13].

Recent multi-robot applications increasingly operate in open environments, where tasks emerge from high-level human instructions or evolving conditions rather than from predefined templates. This shift requires MRTA to interpret open-ended goals and convert them into structured, executable tasks. Advances in large language models (LLMs) offer new opportunities to this problem by enabling the transformation of high-level instructions into representations suitable for downstream allocation [15]–[18]. However, when applied directly to reasoning, LLMs may introduce hallucinated task details or dependencies and may not consistently account for capability or resource constraints, leading to allocations that are linguistically coherent but not always guaranteed to be feasible or rigorous for execution in multi-robot systems [19]–[21].

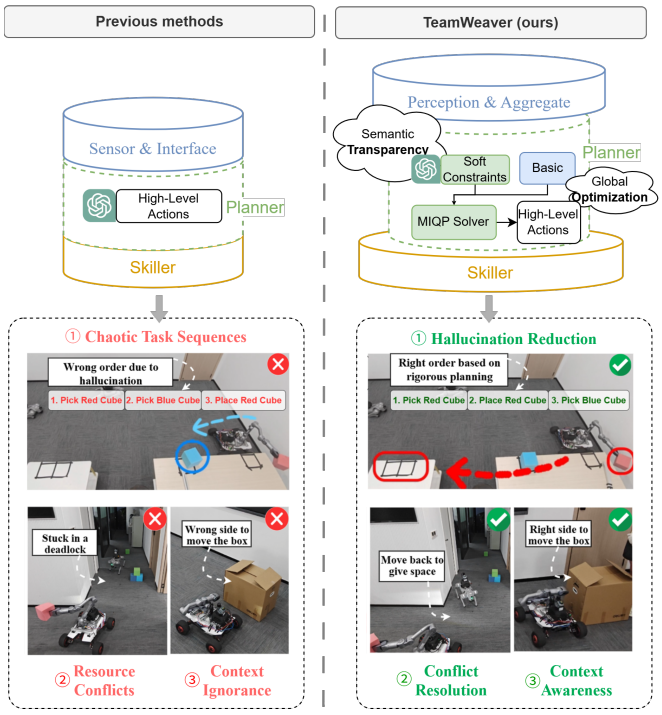


Fig. 1: Architectural comparison between traditional LLM Multi-Robot Planner (*e.g.*, PARTNR [14]) and TeamWeaver. Left (PARTNR): The planner directly generates high-level actions from sensor inputs using LLM-based reasoning. Right (TeamWeaver): The planner employs a hybrid approach where LLMs provide semantic transparency and soft constraints, which are then processed by an MIQP solver along with hard constraints to produce globally optimized high-level actions. The key innovation is the separation of semantic reasoning (LLM) from constraint optimization (MIQP), enabling both interpretability and mathematical guarantees.

[1]https://anonymous.4open.science/r/Dynamic-TeamWeaver-2766

### B. Proposed Approach

In this article, we propose a hybrid planning framework that integrates the semantic interpretability of LLMs with the constraint satisfaction guarantees of mathematical optimization. Our goal is not to replace LLM reasoning, but to address a specific mismatch that arises when natural-language task descriptions must be executed by heterogeneous robot teams: LLMs can extract high-level structures from open-ended instructions, but their outputs may contain implicit assumptions, missing dependencies, or softened constraints that do not fully correspond to what the robots can execute. This discrepancy becomes especially pronounced in multi-robot settings, where feasibility hinges on capability matching, resource limits, spatial-temporal consistency, and coordinated task sequencing.

To resolve this gap, we treat the language model as a semantic perception layer that converts unstructured instructions into intermediate world graphs explicitly capturing task primitives, their ordering constraints, and capability requirements. These representations are then grounded through a mathematically rigorous planner, instantiated in our system as a mixed-integer quadratic programming (MIQP) module. MIQP reformulates the LM-produced structure into a constraint-aware allocation problem, ensuring that the final plan satisfies resource feasibility, temporal consistency, and global coordination across heterogeneous robots. We term this integrated algorithm *LMs-MIQP* and build the *TeamWeaver* multi-robot system on top of it. Through this two-level design, the framework maintains the flexibility needed to interpret open-ended instructions while guaranteeing the execution-level correctness required for reliable multi-robot operation.

We evaluate the *LMs-MIQP* algorithm and the full *TeamWeaver* system in both simulated and real-world multi-robot experiments, spanning dynamic home-service tasks and warehouse-like environments. Experimental results show that our system consistently outperforms baseline methods, including pure LM-based and optimization-only approaches. Across five representative real-world tasks, TeamWeaver achieves task success rates ranging from $90.5\pm4.1\%$ to $98.3\pm5.1\%$, with improvements of up to 18.6% over PARTNR [14] in communication failure scenarios. Notably, TeamWeaver reduces replanning counts by 46–65% compared with PARTNR and achieves significantly higher Semantic Coherence via Temporal Logic ($S_{TL}$) values ranging from $0.66\pm0.05$ to $0.78\pm0.04$, compared to PARTNR's $0.10\pm0.04$ to $0.18\pm0.05$, representing approximately 4–6 times improvement. As illustrated in Fig. 1, TeamWeaver reduces hallucinations, resolves conflicts, and enhances context awareness in real-world deployment compared with PARTNR.

### C. Contributions

Our main contributions are threefold:

- **Design of the LMs-MIQP algorithm:** We develop LMs-MIQP, a hybrid planning algorithm that combines the semantic interpretation capabilities of large language models (LLMs) with the constraint satisfaction guarantees of mixed-integer quadratic programming (MIQP). The algorithm grounds natural-language instructions into feasible, globally consistent multi-robot task allocations. In dynamic simulated home-service and warehouse-like environments, LMs-MIQP improves task success rate by 11.0%, percent complete by 8.0%, and reduces simulation steps by 18.9% compared with PARTNR [14].
- **Implementation of the TeamWeaver multi-robot system:** We integrate the proposed LMs-MIQP algorithm into TeamWeaver, a deployable heterogeneous robotic system capable of executing instruction-driven tasks under dynamic and partially observable conditions. The system features a unified semantic optimization pipeline, real-time coordination mechanisms, and interfaces for cross-platform deployment. In real-world experiments, TeamWeaver outperforms PARTNR across three representative scenarios, achieving improvements in task success rate (up to 4.9%), percent complete (up to 4.6%), and factual accuracy (up to 0.08), demonstrating robust coordination and reliable execution in everyday environments.
- **Comprehensive evaluation framework with transparency and interpretability metrics:** We adapt semantic entropy methods—originally developed for assessing uncertainty in LLM reasoning—to multi-robot system interpretability evaluation. This represents the first application of semantic-level uncertainty quantification to transparency assessment in heterogeneous multi-robot coordination. We establish a hierarchical evaluation framework that systematically assesses multi-robot systems across task execution, allocation efficiency, and interpretability dimensions. The framework introduces three transparency metrics: Semantic Coherence via Temporal Logic ($S_{TL}$), Semantic Consistency via Temporal Execution ($S_{TE}$), and Factual Accuracy (FA). $S_{TL}$ and $S_{TE}$ quantify the alignment between LLM-generated actions and optimization-based decisions, while FA measures hallucination reduction. These metrics, combined with traditional performance indicators (Success Rate, Percent Complete, Task Stability), provide a comprehensive assessment tool for evaluating hybrid LLM-optimization systems in dynamic multi-robot collaboration scenarios.

## II. Related Works

### A. Practices and Challenges of Heterogeneous Multi-Robot Systems

Heterogeneous multi-robot systems have found widespread applications across diverse domains, including warehouse automation [10], [22], search and rescue operations [9], [23], [24], domestic service [5], [25], [26], collaborative manufacturing [11], [27], [28], and physical human-robot collaborative transportation and manipulation [29]. These systems leverage complementary capabilities of heterogeneous robots to accomplish complex multi-robot collaborative tasks that would be infeasible for individual robots or homogeneous teams. As illustrated in Fig. 2, such collaborative scenarios typically involve multiple interconnected agents (mobile platforms, manipulation arms, and humanoid companions) navigating

and interacting within complex environments, requiring coordinated task allocation and execution across heterogeneous capabilities. The fundamental problem in these applications is MRTA [1], [2], [4], which must account for diverse robot capabilities, varying resource constraints, complex spatiotemporal relationships, and dynamic environmental conditions.

Real-world multi-robot applications operate in dynamic and uncertain environments, where multiple challenges emerge [2], [4]: task requirements and environmental conditions change unpredictably, requiring frequent replanning [9], [30], [31]; robot states fluctuate due to battery depletion, mechanical failures, or performance degradation, with tight coupling causing cascading effects across the system [9], [30]; and open-world scenarios involve natural language instructions with implicit requirements difficult to formalize using traditional constraint-based approaches [3]. These challenges create a fundamental tension between rigorous constraint satisfaction and flexibility. Traditional optimization-based approaches, formulated as Mixed-Integer Programming problems [1], [9], excel at constraint satisfaction when constraints are known and static, but struggle with open-world instructions containing implicit requirements [3]. This limitation has motivated alternative paradigms, particularly LLMs for semantic understanding, though LLM-based approaches introduce challenges related to reliability and constraint verification [15], leading to hybrid methods that combine the strengths of both paradigms.

### *B. LLM-Driven Planning and Task Allocation for Multi Robot Systems*

Recent surveys have shown that LLMs can support robot task planning by translating high-level instructions into executable actions [32], [33]. Representative methods can be categorized into dialog-driven approaches and modular frameworks. Dialog-driven approaches employ interactive conversations for task understanding: RoCo [13] equips each robot with an LLM proxy for decentralized decision-making through interactive dialogs; EMOS [30] adopts a ”centralized discussion–distributed execution” framework; SMART-LLM [34] employs conversational feedback for task decomposition. Modular frameworks integrate LLMs within structured system architectures: CoELA [35] leverages LLMs as semantic controllers coordinating perception, planning, and execution modules; PARTNR [14], built on Habitat-Sim, validates collaborative systems in complex scenarios and contributes large-scale datasets.

However, LLMs may fail to perform mathematical constraint verification on their output plans, leading to ”hallucinatory allocations” [15], [36] where seemingly reasonable plans are actually infeasible. These methods suffer from limitations including poor adaptability to dynamic environments, high latency, lack of large-scale validation, and hallucination issues [15], [36]. The limitations of pure LLM-based approaches have motivated hybrid methods that combine LLM semantic understanding with formal optimization, positioning LLMs as semantic perception modules that generate soft constraints rather than directly making allocation decisions.

### *C. Hybrid Multi-Robot Planning on LLMs with Optimization Methods*

The combination of LLM semantic understanding and optimizer rigor has emerged as a promising solution. Recent studies have combined LLMs with formal planning methods to improve task allocation efficiency in HMRS [37], [38]. LiP-LLM [39] generates skill lists and task dependency graphs using LLMs and then applies linear programming (LP) for efficient allocation. LaMMA-P [40] combines LLMs with the Planning Domain Definition Language (PDDL) for long-horizon task planning and allocation. These methods mitigate the black-box nature of LLMs and improve execution flexibility.

However, existing hybrid methods reveal several limitations. First, regarding optimization expressiveness, many methods employ LP as the optimization backend [39], but LP's expressiveness may be limited in handling integer decisions, mutual exclusion constraints, and nonlinear spatiotemporal relationships inherent in heterogeneous robot allocation. Mixed-Integer Programming formulations offer greater expressiveness, but their adoption in hybrid LLM-optimizer frameworks remains limited. Second, many hybrid schemes adopt an open-loop design, using the LLM as a one-time pre-processor [39], [40], limiting the system's ability to re-reason and make closed-loop adjustments based on robot state feedback. Closed-loop architectures have been explored [30], [35], but their integration with LLM-based semantic reasoning in multi-robot task allocation remains an open challenge. Third, regarding interpretability, the decision-making process is often split between the LLM and the optimizer, with limited integration. In dynamic and unstructured environments, these methods struggle to accurately model multi-agent constraints and provide sufficient interpretability [4], [13], [36], limiting end-to-end transparency from human intent to final action.

In contrast, our LMs-MIQP framework achieves tighter integration between semantic reasoning and mathematical optimization. Unlike methods using LP, we employ MIQP to handle integer decisions, mutual exclusion constraints, and nonlinear relationships. Unlike open-loop designs, our framework implements a closed-loop architecture where the LLM continuously interprets high-level instructions and environmental feedback, enabling dynamic adaptation. Unlike designs that split decision-making, our framework provides end-to-end transparency by embedding LLM assessments as interpretable soft constraints in the optimization objective, enabling both open-world adaptability and rigorous constraint satisfaction.

## III. Problem Formulation

### *A. Notation*

To facilitate readability and ensure consistency, we establish a unified notation system for the multi-robot task allocation problem and the LMs-MIQP algorithm. Throughout this paper, we consistently use $i \in \{1, \ldots, N\}$ to index agents and $j \in \{1, \ldots, M\}$ to index tasks, where $N$ denotes the number of heterogeneous agents and $M$ denotes the number of decomposed sub-tasks.

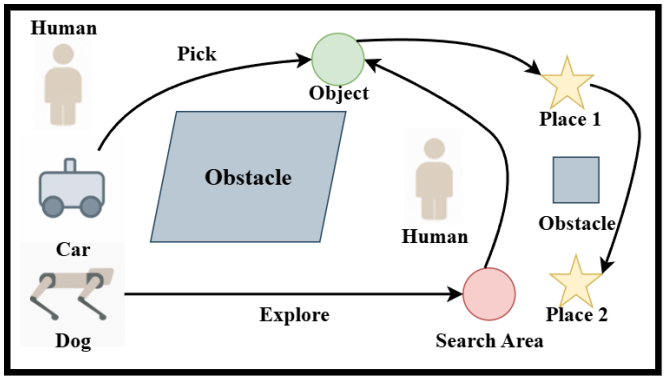


Fig. 2: Representative heterogeneous multi-robot collaborative scenario in Habitat-Sim, illustrating coordinated task execution among agents with complementary capabilities (mobile platforms, manipulation arms, and humanoid companions) in complex indoor environments.

The notation is organized into several categories: (1) *System Parameters* define the fundamental dimensions and indices of the problem; (2) *Inputs and System State* represent the external inputs (natural language instructions, agent capabilities) and the internal system representation (world graph); (3) *Optimization Variables* are the decision variables in the MIQP formulation; (4) *LLM and Cost Functions* capture the semantic assessment and cost computation mechanisms; (5) *Parameters and Constants* include weighting coefficients and threshold values that tune the optimization behavior; and (6) *Other Notation* covers auxiliary sets and derived quantities.

Table I provides a comprehensive summary of all key symbols. When a symbol appears with subscripts or superscripts (e.g., $\boldsymbol{\pi}_j$ for task $j$, $\mathcal{C}_i$ for agent $i$), the indices follow the convention established above. Matrix and vector dimensions are explicitly stated where first introduced, and all optimization variables are defined with their feasible domains.

### B. System Model

We consider a team of $N$ heterogeneous agents operating in a shared workspace. Each agent $i \in \{1, \ldots, N\}$ is characterized by a capability profile $\mathcal{C}_i \in \mathbb{R}^D$ that defines the agent's proficiency across $D$ capability dimensions (e.g., mobility, manipulation, control, liquid handling, power management) [41]. Each component $\mathcal{C}_i[k]$ quantifies agent $i$'s capability level in dimension $k$, where values range from 0 (incapable) to higher positive values (highly proficient). These capability profiles capture the diverse skills and resources available across the heterogeneous team, enabling complementary task execution through coordinated allocation. The capability profile $\mathcal{C}_i$ is used to determine task feasibility: task $j$ can be assigned to agent $i$ only if $\mathcal{C}_i \geq \mathbf{r}_{\text{req}}(j)$, where $\mathbf{r}_{\text{req}}(j) \in \mathbb{R}^D$ specifies the required capability vector for task $j$.

To support decision-making under uncertainty and partial observability, each agent maintains a structured semantic belief graph $\mathcal{G}_t^i = (\mathbf{V}_t^i, \mathbf{E}_t^i)$ at time $t$, where $\mathbf{V}_t^i$ represents perceived entities (objects, locations, other agents) and $\mathbf{E}_t^i$ encodes their pairwise relationships (spatial, temporal, or functional dependencies) [42]. These local graphs capture both task-relevant information and dynamic environmental context from each agent's perspective.

**Algorithm 1** Graph Fusion Operator

**Require:** Local graphs $\{\mathcal{G}_t^i = (\mathbf{V}_t^i, \mathbf{E}_t^i)\}_{i=1}^N$
**Ensure:** Global graph $\mathcal{G}_t^{\text{global}} = (\mathbf{V}_t^{\text{global}}, \mathbf{E}_t^{\text{global}})$

Initialize: $\mathbf{V}_t^{\text{global}} \leftarrow \emptyset$, $\mathbf{E}_t^{\text{global}} \leftarrow \emptyset$
**for** each agent $i \in \{1, \ldots, N\}$ **do**
  **for** each node $v \in \mathbf{V}_t^i$ **do**
    **if** $v \notin \mathbf{V}_t^{\text{global}}$ **then**
      **Add** $v$ **to** $\mathbf{V}_t^{\text{global}}$
    **else**
      Update properties of existing node in $\mathbf{V}_t^{\text{global}}$
    **end if**
  **end for**
  **for** each edge $(u, v) \in \mathbf{E}_t^i$ **do**
    **if** $(u, v) \notin \mathbf{E}_t^{\text{global}}$ **then**
      **Add** $(u, v)$ **to** $\mathbf{E}_t^{\text{global}}$
    **end if**
  **end for**
**end for**
Remove obsolete edges for dynamic entities
**return** $\mathcal{G}_t^{\text{global}}$

A fusion operator $\textbf{Fusion}(\cdot)$ aggregates these local semantic graphs across all agents into a unified global world model:

$$\mathcal{G}_t^{\text{global}} = \textbf{Fusion}(\{(\mathbf{V}_t^i, \mathbf{E}_t^i)\}_{i=1}^N) \tag{1}$$

The fusion operator implements a graph merge operation that constructs $\mathcal{G}_t^{\text{global}} = (\mathbf{V}_t^{\text{global}}, \mathbf{E}_t^{\text{global}})$ from local graphs $\{\mathcal{G}_t^i = (\mathbf{V}_t^i, \mathbf{E}_t^i)\}_{i=1}^N$. Algorithm 1 outlines the computational procedure, where $v$ and $u$ denote graph nodes (temporary variables within the algorithm):

This merge process ensures that the global graph $\mathcal{G}_t^{\text{global}}$ maintains a consistent, unified representation while preserving the temporal dynamics captured by individual agents' partial observations. The fusion is performed incrementally as each agent's local graph is updated, enabling real-time maintenance of the shared world model. This global world graph $\mathcal{G}_t^{\text{global}}$ serves as the core representation for multi-robot shared environmental understanding [14] and the backbone for downstream modules, including task allocation, high-level reasoning, and multi-agent coordination. It enables agents to maintain consistency across local decision-making processes while adapting to dynamic and partially observable environments.

### C. MRTA Problem Definition

Building on the system representation and shared semantic graph defined in the previous section, we now formulate the multi-robot task allocation (MRTA) problem. Given a high-level natural language instruction $\mathcal{I}$, a heterogeneous agent team of $N$ agents (denoted by indices $i \in \{1, \ldots, N\}$), the goal is to assign these $N$ agents to a set of $M$ decomposed

TABLE I: Notation summary.

| Symbol | Description | Symbol | Description |
|---|---|---|---|
| *System Parameters* | | | |
| $N$ | Number of heterogeneous agents | $i$ | Agent index, $i \in \{1, \ldots, N\}$ |
| $M$ | Number of decomposed sub-tasks | $j$ | Task index, $j \in \{1, \ldots, M\}$ |
| $D$ | Dimension of the capability space | $k$ | Capability dimension index, $k \in \{1, \ldots, D\}$ |
| $t$ | Time step | | |
| *Inputs and System State* | | | |
| $\mathcal{I}$ | Natural language instruction for task goals | $\mathcal{C}_i$ | Capability profile of agent $i$ |
| $\mathcal{G}_t^i = (\mathbf{V}_t^i, \mathbf{E}_t^i)$ | Local semantic belief graph of agent $i$ at time $t$ | $\mathbf{V}_t^i$ | Set of perceived entities in agent $i$'s local graph |
| $\mathbf{E}_t^i$ | Set of pairwise relationships in agent $i$'s local graph | $\mathcal{G}_t^{\text{global}}$ | Global world graph encoding environmental context |
| $\mathbf{V}_t^{\text{global}}$ | Set of entities in the global world graph | $\mathbf{E}_t^{\text{global}}$ | Set of relationships in the global world graph |
| $\mathbf{Fusion}(\cdot)$ | Fusion operator aggregating local graphs | $\mathbf{r}_{\text{req}}(j)$ | Required capability vector for task $j$ |
| *Optimization Variables* | | | |
| $\mathbf{A} \in \{0,1\}^{M \times N}$ | Binary task assignment matrix | $\boldsymbol{\phi} \in \{0,1\}^M$ | Task relaxation vector |
| $\boldsymbol{\pi}_j \in \mathbb{R}^N$ | Planning policy for task $j$ | $\boldsymbol{\delta} \in \mathbb{R}^M$ | Load penalty vector |
| $\mathbf{A}_p \in \{0,1\}^{M \times N}$ | Previous allocation matrix | | |
| *LLM and Cost Functions* | | | |
| $J(\mathbf{A}, \mathcal{G}_t^{\text{global}})$ | Global cost function for MRTA | $\mathcal{L}$ | LLM's contextual capability assessment |
| $\mathbf{c}(\mathbf{A}, \mathcal{L})$ | Execution cost function | $\hat{\mathbf{c}}_j(\mathbf{A}, \mathcal{L})$ | Estimated cost for task $j$ |
| $\mathbf{q}_j \in \mathbb{R}^N$ | Expected reward vector for task $j$ | $\mathbf{Y}_j^* \in \mathbb{R}^N$ | Policy-expected outcome vector for task $j$ |
| $\mathbf{r}_j \in \mathbb{R}^N$ | Allocation reward vector for task $j$ | $\mathbf{A}_j$ | $j$-th row of $\mathbf{A}$ |
| *Parameters and Constants* | | | |
| $\mathbf{w}_J \in \mathbb{R}^M$ | Weight vector for relaxation penalties | $\lambda_D, \lambda_T \geq 0$ | Weighting coefficients for margin and transition cost |
| $\delta_{\max} > 0$ | Maximum allowed slack for load penalties | $\epsilon_j \geq 0$ | Estimation bias for task $j$ |
| $\mathbf{T}_{\text{trans}} \in \mathbb{R}^{M \times M}$ | Transition cost matrix | $\mathbf{H}_{\text{perf}} \in \mathbb{R}^{M \times D}$ | Performance history matrix |
| $\mathcal{S}_i$ | Observed performance for agent $i$ | $\hat{\mathcal{S}}_i$ | Expected performance for agent $i$ |
| $\gamma_1, \gamma_2 \geq 0$ | Weighting coefficients for capability updates | $w_k \geq 0$ | Weighting coefficient for capability dimension $k$ |
| $\epsilon > 0$ | Update threshold for capability adaptation | | |
| *Other Notation* | | | |
| $\Omega$ | Feasible set (MRTA + MIQP constraints) | $\mathcal{Y}_{\text{gen}}$ | Generated set from LLM outputs |
| $\mathcal{Y}$ | Valid generated content, $\mathcal{Y} = \mathcal{Y}_{\text{gen}} \cap \Omega$ | | |

sub-tasks (denoted by indices $j \in \{1, \ldots, M\}$). These sub-tasks are grounded from $\mathcal{I}$ through parsing and graph-level reasoning over the global world graph $\mathcal{G}_t^{\text{global}}$. Task allocation decisions are conditioned on each agent's capability profile $\mathcal{C}_i$ and the global task context encoded in $\mathcal{G}_t^{\text{global}}$.

Formally, the MRTA problem is cast as a combinatorial optimization over a binary assignment matrix $\mathbf{A} \in \{0,1\}^{M \times N}$, where $A_{ji} = 1$ indicates that task $j$ is assigned to agent $i$ (we use $j$ to index tasks and $i$ to index agents throughout). The goal is to find an optimal assignment $\mathbf{A}^* = \arg\min_{\mathbf{A}} J(\mathbf{A}, \mathcal{G}_t^{\text{global}})$, minimizing a global cost function $J$ that depends on the assignment matrix and the world graph, subject to the following constraints:

- **Agent capacity:** each agent is assigned to at most one task, $\sum_{j=1}^{M} A_{ji} \leq 1, \quad \forall i \in \{1, \ldots, N\}$;
- **Task exclusivity:** each task is assigned to at most one agent, $\sum_{i=1}^{N} A_{ji} \leq 1, \quad \forall j \in \{1, \ldots, M\}$;
- **Capability feasibility:** task $j$ is assigned to agent $i$ only if the agent's capability $\mathcal{C}_i$ satisfies the task requirement $\mathbf{r}_{\text{req}}(j) \in \mathbb{R}^D$, where $D$ is the dimension of the capability space, i.e., $\mathcal{C}_i \geq \mathbf{r}_{\text{req}}(j) \cdot A_{ji}, \quad \forall i \in \{1, \ldots, N\}, j \in \{1, \ldots, M\}$;
- **System constraints:** including kinodynamic limits, safety zones, and collision avoidance [9], [43], [44].

Together, the above formulation defines the MRTA problem as a constrained optimization over discrete task-agent assignments. The input includes: (1) a natural language instruction $\mathcal{I}$ that specifies the high-level task goals, (2) the capability profiles $\mathcal{C}_i$ for each of the $N$ agents that define what each agent can do, and (3) the global world graph $\mathcal{G}_t^{\text{global}}$ that captures the current environmental context. The output is a binary assignment matrix $\mathbf{A} \in \{0,1\}^{M \times N}$ satisfying both logical constraints (agent capacity, task exclusivity, capability feasibility) and physical constraints (kinodynamic limits, safety zones, collision avoidance). This formulation provides a unified interface between high-level task semantics derived from $\mathcal{I}$ and low-level execution feasibility determined by agent capabilities $\mathcal{C}_i$ and environmental constraints encoded in $\mathcal{G}_t^{\text{global}}$. In the following section, we present the LMs-MIQP algorithm to solve this problem efficiently in dynamic, real-

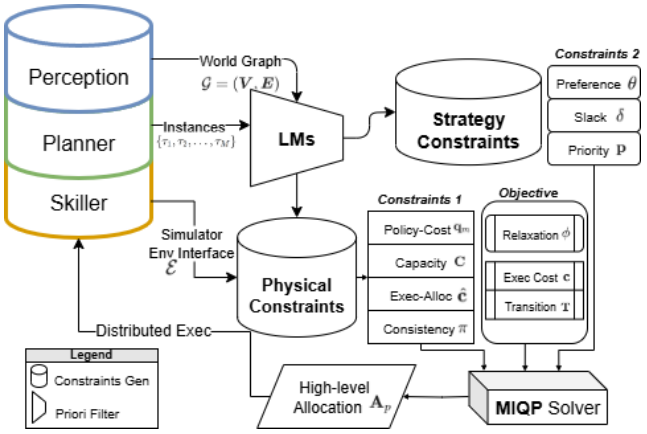


Fig. 3: The algorithm framework of LMs-MIQP. Language models generate strategy constraints, which are combined with physical constraints and solved by the MIQP optimizer. Notations are defined in Table I.

time settings.

## IV. Methodology

Building on the MRTA problem formulation in Section III, this section presents our methodology that combines LLM semantic reasoning with MIQP optimization. As illustrated in Fig. 3, LLMs process natural language instructions $\mathcal{I}$ and the global world graph $\mathcal{G}_t^{\text{global}}$ to generate contextual assessments, encoded as soft constraints in MIQP. The MIQP solver enforces hard constraints (agent capacity, task exclusivity, capability feasibility) while optimizing LLM-informed preferences. This methodology consists of two components: (1) the LMs-MIQP algorithm (Section IV-A) that formulates task allocation as constrained optimization; and (2) a closed-loop adaptation mechanism (Section IV-B) for dynamic capability learning.

### A. LMs-MIQP: Constrained Multi-Robot Task Allocation Algorithm

To solve the MRTA problem formulated in Section III, we develop the LMs-MIQP algorithm that formulates task allocation as a Mixed Integer Quadratic Programming (MIQP) problem [45]. The algorithm optimizes over the binary assignment matrix $\mathbf{A} \in \{0,1\}^{M \times N}$ that maps $M$ tasks to $N$ agents. LLMs interpret natural language instructions $\mathcal{I}$ and the global world graph $\mathcal{G}_t^{\text{global}}$ to generate contextual capability assessments $\mathcal{L}$, which are encoded as soft constraints and cost terms in the MIQP objective. The MIQP solver enforces hard constraints from Section III while optimizing LLM-informed preferences. This approach mitigates LLM hallucination by constraining optimization to the feasible set $\Omega$.

The LMs-MIQP algorithm extends the basic MRTA formulation by introducing additional optimization variables: (1) $\boldsymbol{\phi} \in \{0,1\}^M$ is the task relaxation vector; (2) $\boldsymbol{\pi}_j \in \mathbb{R}^N$ is the planning policy for task $j$; (3) $\boldsymbol{\delta} \in \mathbb{R}^M$ is the load penalty vector; (4) $\mathbf{A}_p \in \{0,1\}^{M \times N}$ is the previous allocation matrix; and (5) $\mathcal{L}$ is the LLM's contextual capability assessment derived from $\mathcal{G}_t^{\text{global}}$ and agent capability profiles $\mathcal{C}_i$ for $i \in \{1, \ldots, N\}$.

The MIQP optimization determines the optimal allocation matrix $\mathbf{A}$, relaxation vector $\boldsymbol{\phi}$, and planning policies $\boldsymbol{\pi} = [\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_M]$ by solving:

$$\min_{\mathbf{A},\boldsymbol{\phi},\boldsymbol{\pi}} \underbrace{\mathbf{c}(\mathbf{A},\mathcal{L})}_{\text{Execution Cost}} + \underbrace{\mathbf{w}_J^T\boldsymbol{\phi}}_{\text{Relaxation}} + \underbrace{\lambda_D||\boldsymbol{\delta}||_2^2}_{\text{Margin Penalty}} + \underbrace{\lambda_T||\mathbf{T}_{\text{trans}}(\mathbf{A}-\mathbf{A}_p)||_1}_{\text{Transition Cost}} \tag{2}$$

subject to:

$$\hat{\mathbf{c}}_j(\mathbf{A},\mathcal{L}) \le \mathbf{q}_j^T\boldsymbol{\pi}_j + \epsilon_j, \quad \forall j \in \{1,\ldots,M\} \tag{a}$$
$$\delta_j \ge -\phi_j\delta_{\max}, \quad \forall j \in \{1,\ldots,M\} \tag{b}$$
$$\boldsymbol{\pi}_j^T\mathbf{Y}_j^* - \mathbf{r}_j^T\mathbf{A}_j = \delta_j, \quad \forall j \in \{1,\ldots,M\} \tag{c}$$
$$\boldsymbol{\pi}_j^T\mathbf{1} = 1, \quad \forall j \in \{1,\ldots,M\} \tag{d}$$

The objective function in Eq. (2) consists of four terms: execution cost $\mathbf{c}(\mathbf{A},\mathcal{L})$ modulated by LLM assessment $\mathcal{L}$, relaxation penalty $\mathbf{w}_J^T\boldsymbol{\phi}$ with weights $\mathbf{w}_J \in \mathbb{R}^M$ informed by LLM interpretation of task criticality from $\mathcal{I}$, margin penalty $\lambda_D||\boldsymbol{\delta}||_2^2$ for load balancing, and transition cost $\lambda_T||\mathbf{T}_{\text{trans}}(\mathbf{A}-\mathbf{A}_p)||_1$ that penalizes changes from previous allocation $\mathbf{A}_p$. The coefficients $\lambda_D, \lambda_T \ge 0$ control the relative importance of load balancing versus allocation stability.

Constraints (a)–(d) in Eq. (a)–(d) regulate the optimization for all tasks $j \in \{1,\ldots,M\}$: constraint (a) bounds estimated cost $\hat{\mathbf{c}}_j(\mathbf{A},\mathcal{L})$ with expected reward $\mathbf{q}_j^T\boldsymbol{\pi}_j$ plus estimation bias $\epsilon_j \ge 0$; constraint (b) allows load penalties $\delta_j \in \boldsymbol{\delta}$ to be negative (up to $-\delta_{\max}$) when relaxation variable $\phi_j = 1$; constraint (c) equates the difference between policy-expected outcomes $\mathbf{Y}_j^*$ and allocation rewards $\mathbf{r}_j^T\mathbf{A}_j$ to $\delta_j$; and constraint (d) enforces normalization so $\boldsymbol{\pi}_j$ forms a valid probability distribution over the $N$ agents. These constraints extend the basic MRTA constraints from Section III by incorporating LLM-derived semantic assessments, adaptive task relaxation, and probabilistic planning policies. Together with the basic MRTA constraints (agent capacity, task exclusivity, capability feasibility), constraints (a)–(d) define the feasible set $\Omega$ for the optimization.

The execution cost $\mathbf{c}(\mathbf{A},\mathcal{L})$ aggregates task-specific costs for all agent-task pairs $(i,j)$ where $A_{ji} = 1$. For each pair, the cost component $c_{ji}$ combines negative task utility (computed from task-specific functions based on agent state and $\mathcal{G}_t^{\text{global}}$) and an aptitude penalty term inversely proportional to the LLM compatibility score $\mathcal{L}_{ji}$ from $\mathcal{L}$. The total cost is $\mathbf{c}(\mathbf{A},\mathcal{L}) = \sum_{i=1}^N\sum_{j=1}^M c_{ji} \cdot A_{ji}$, and the estimated cost for task $j$ is $\hat{\mathbf{c}}_j(\mathbf{A},\mathcal{L}) = \sum_{i=1}^N c_{ji} \cdot A_{ji}$. The LLM generates compatibility scores $\mathcal{L}_{ji}$ by analyzing $\mathcal{I}$ and $\mathcal{G}_t^{\text{global}}$ to evaluate how well agent capabilities $\mathcal{C}_i$ match task requirements $\mathbf{r}_{\text{req}}(j)$ given the environmental context encoded in $\mathcal{G}_t^{\text{global}}$. These scores are derived from semantic analysis of task descriptions in $\mathcal{I}$, object locations and relationships in $\mathcal{G}_t^{\text{global}}$, and agent capability profiles $\mathcal{C}_i$ for $i \in \{1,\ldots,N\}$.

[1]For symbol definitions (cylinder, rectangle, arrow, etc.), see the anonymous reference at https://anonymous.4open.science/r/uml-notation-reference-5F74/UML_notation_reference.pdf.

The MIQP problem is solved using standard solvers that handle the mixed-integer quadratic structure [46]. The algorithm mitigates LLM hallucination by constraining optimization to the feasible set $\Omega$, ensuring LLM outputs from $\mathcal{I}$ are validated against hard constraints involving $\mathcal{C}_i$ and $\mathcal{G}_t^{\text{global}}$.

### B. Closed-Loop Adaptation for Dynamic Capability Learning

Building upon the MIQP foundation, TeamWeaver incorporates a closed-loop adaptation mechanism that enables continuous learning and parameter adjustment in response to dynamic environments and evolving agent capabilities. The mechanism operates through continuous monitoring of system performance and systematic updates to agent capability profiles $\mathcal{C}_i$ based on execution feedback.

The adaptation process, formalized in Algorithm 2, begins when performance deviations exceed a predefined threshold. The system updates agent capability profiles $\mathcal{C}_i$ by incorporating both immediate performance feedback and historical execution patterns. For agent $i$, the capability update is computed as:

$$\Delta\mathcal{C}_i = \gamma_1 \frac{|\mathcal{S}_i - \hat{\mathcal{S}}_i|}{|\hat{\mathcal{S}}_i|} + \gamma_2 \sum_{j=1}^{M}\sum_{k=1}^{D} \mathbf{H}_{\text{perf}}[j,k] \cdot \mathcal{C}_i[k] \cdot w_k \quad (3)$$

where $\mathcal{S}_i$ and $\hat{\mathcal{S}}_i$ denote the observed and expected performance for agent $i$, $\mathbf{H}_{\text{perf}} \in \mathbb{R}^{M\times D}$ represents the performance history matrix encoding task execution quality, $w_k \geq 0$ are weighting coefficients for capability dimension $k \in \{1,\ldots,D\}$, and $\gamma_1, \gamma_2 \geq 0$ control the relative importance of immediate feedback versus historical patterns. This formulation ensures that capability updates reflect both immediate deviations and historical performance trends, enabling the system to distinguish between temporary fluctuations and systematic capability changes.

The updated capability profiles $\mathcal{C}_i$ subsequently inform task prioritization and specialization assessment. For each agent $i$, the specialization score for task $j$ is computed as the alignment between the updated capability profile $\mathcal{C}_i$ and the task requirement vector $\mathbf{r}_{\text{req}}(j)$, accounting for the performance history encoded in $\mathbf{H}_{\text{perf}}$. Task priorities are dynamically adjusted through a constrained optimization that maximizes alignment between agent specializations and task assignments, ensuring that high-priority tasks are allocated to agents with the best matching capabilities. This prioritization mechanism operates in conjunction with the MIQP allocation process: when capability updates exceed a threshold, the system triggers a replanning cycle that recomputes the assignment matrix $\mathbf{A}$ using the updated $\mathcal{C}_i$ values, creating a feedback loop between execution performance and allocation decisions.

The adaptation mechanism operates asynchronously and event-driven, monitoring performance continuously but triggering updates only when deviations exceed predefined thresholds. This design balances responsiveness to environmental changes with computational efficiency, avoiding unnecessary replanning cycles while ensuring the system adapts promptly to significant capability mismatches or environmental disturbances. The threshold parameter $\epsilon$ in Algorithm 2 controls the sensitivity of the adaptation mechanism: smaller values lead to more frequent updates and higher responsiveness, while larger values reduce computational overhead and filter out transient fluctuations.

This integrated feedback loop creates a self-improving system that maintains robust task allocation performance by continuously learning from execution experience and adapting to evolving operational conditions. The mathematical rigor of the MIQP foundation combined with the adaptability of the closed-loop mechanism provides a principled approach to handling dynamic multi-robot collaboration scenarios. The two-level design (static optimization via MIQP and dynamic adaptation via feedback learning) enables TeamWeaver to maintain optimal performance in both well-modeled scenarios (where MIQP provides near-optimal solutions) and uncertain, evolving environments (where adaptation compensates for model inaccuracies and changing conditions).

Together, the LMs-MIQP algorithm and the closed-loop adaptation mechanism form a unified methodology for dynamic multi-robot task allocation. The LMs-MIQP component (Section IV-A) provides the core optimization framework that combines LLM semantic reasoning with mathematical constraint satisfaction, ensuring feasible and interpretable task assignments. The closed-loop adaptation component (Section IV-B) enables the system to learn from execution experience and adapt to changing conditions, updating agent capability profiles $\mathcal{C}_i$ and task priorities based on performance feedback. This two-level design addresses both the static optimization challenge (through MIQP) and the dynamic adaptation challenge (through feedback learning), creating a robust system that maintains optimal performance while responding to environmental changes and evolving agent capabilities. The integration of these components is realized through the system architecture presented in Section V, which provides the implementation framework for deploying TeamWeaver in real-world scenarios.

## V. System Architecture

To realize the LMs-MIQP algorithm and closed-loop adaptation mechanism presented in Section IV, TeamWeaver adopts a **Perception–Plan–Skill** architecture that enables closed-loop multi-robot execution. As illustrated in Fig. 4, this three-layer design decouples semantic understanding, optimization-based planning, and decentralized execution while maintaining tight integration through the global world graph $\mathcal{G}_t^{\text{global}}$ and feedback mechanisms.

### A. Perception Layer

The perception layer serves as the foundation for multi-robot environmental understanding by fusing heterogeneous multimodal data from distributed sensors into a unified world graph $\mathcal{G}_t^{\text{global}}$. As shown in Fig. 4, the layer processes high-level commands (task goals) from human operators, such as "Rearrange toy cars and clean the kitchen table," along with raw sensor data from multiple sources.

The layer operates through three main stages: *aggregation*, *fusion*, and *representation*. In the aggregation stage, each sensor module processes its raw observations and performs

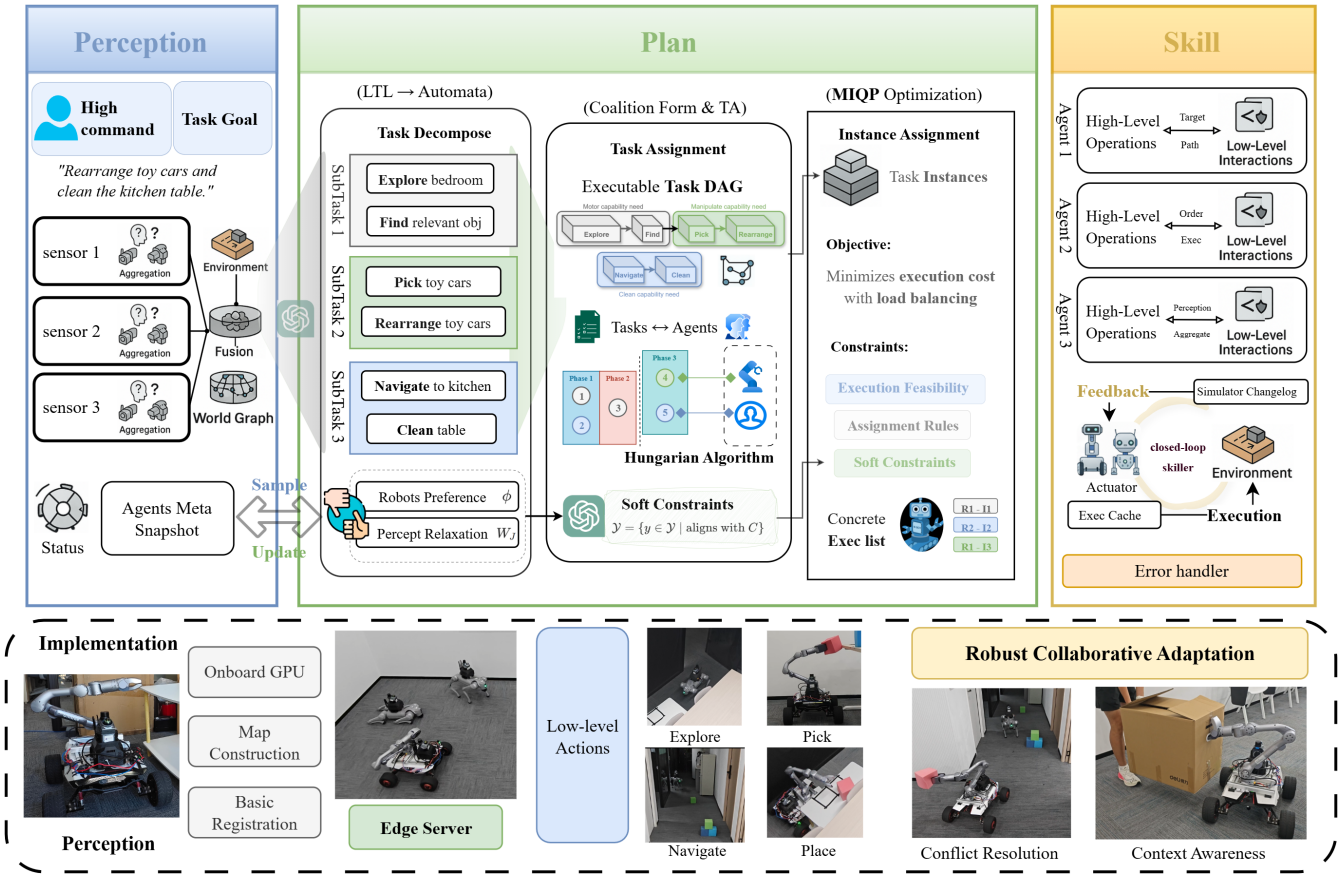


Fig. 4: Our TeamWeaver's **Perception–Plan–Skill** system architecture for multi-robot closed-loop task execution. The framework consists of a **perception** layer that fuses heterogeneous sensing and task goals into a unified world graph, a **planner** layer that decomposes tasks and allocates them via MIQP optimization with LLM-derived soft constraints, and a **skill** layer that distributes primitives to decentralized agents with integrated state and performance feedback. A real-world **implementation** is demonstrated with onboard perception, edge-based planning, and collaborative execution in real-world deployment.

local aggregation to extract semantic entities and relationships. These aggregated observations are then fused using the fusion operator $\mathbf{Fusion}(\cdot)$ (Algorithm 1) to construct the global world graph $\mathcal{G}_t^{\text{global}}$, which encodes the environment state including object locations, spatial relationships, and dynamic entities. Simultaneously, the layer maintains an *agents meta snapshot* that tracks the current status of all robots in the system, including their positions, capabilities, and operational states.

In real-world deployment, the *aggregation* stage runs *onboard* each robot, performing real-time local sensor processing to extract semantic entities and relationships. This distributed aggregation enables low-latency perception updates and reduces communication bandwidth requirements. The *fusion* stage executes on the *edge workstation*, where locally aggregated observations from all robots are merged to construct the unified world graph $\mathcal{G}_t^{\text{global}}$. This hybrid design separates time-critical local processing from global coordination: onboard aggregation maintains real-time responsiveness, while centralized fusion ensures consistent global state representation across the multi-robot team. The output of the perception layer consists of *sample* data that includes: (1) updates to the world graph $\mathcal{G}_t^{\text{global}}$ reflecting the latest environmental observations, and (2) the agents meta snapshot providing current robot status information. This data is continuously fed to the plan layer, enabling real-time task planning based on the most up-to-date environmental context.

## B. Plan Layer

The plan layer serves as the computational core that implements the LMs-MIQP algorithm (Section IV-A) and closed-loop adaptation (Section IV-B). As illustrated in Fig. 4, the plan layer operates through three sequential stages: task decomposition, task assignment, and instance assignment.

*a) Task Decomposition (LTL → Automata):* The plan layer receives natural language instructions $\mathcal{I}$ (task goals) and the world graph $\mathcal{G}_t^{\text{global}}$ from the perception layer. Linear temporal logic (LTL) specifications are compiled into automata representations, enabling systematic breakdown of high-level goals into executable primitives. For example, the task "Rearrange toy cars and clean the kitchen table" is decomposed into subtasks such as "Explore bedroom," "Find relevant obj," "Pick toy cars," "Rearrange toy cars," "Navigate to kitchen," and "Clean table." These primitives are then grouped into task phases to support coalition formation and task alignment, forming the basis for multi-robot coordination.

*b) Task Assignment (Coalition Form & TA):* The decomposed primitives are arranged into a directed acyclic graph (DAG) called the *executable task DAG*, whose edges encode precedence relationships and shared-resource constraints. The tasks $\leftrightarrow$ agents module employs a Hungarian algorithm to facilitate initial task-agent matching, providing a preliminary assignment that considers basic compatibility between tasks and agent capabilities. This initial matching serves as a starting point for the subsequent optimization-based allocation.

*c) Instance Assignment (MIQP Optimization):* The core optimization problem is solved using the LMs-MIQP algorithm (Section IV-A), which formulates task allocation as a mixed-integer quadratic program. The optimization minimizes the objective function in Eq. (2), subject to the constraints in Eq. (a)–(d), along with the basic MRTA constraints (agent capacity, task exclusivity, capability feasibility) from Section III. Soft constraints are derived from LLM assessments: $\mathcal{Y} = \mathcal{Y}_{\text{gen}} \cap \Omega$, where $\mathcal{Y}_{\text{gen}}$ represents LLM-generated preferences and $\Omega$ is the feasible set, enabling hallucination mitigation through constraint intersection.

The plan layer receives *robots preference* information and *percept relaxation weights* $\mathbf{w}_J$ from the LLM, which inform the soft constraints in the optimization. The output is a *concrete execution list* specifying task assignments for each robot (e.g., R1-T1, R2-T2, R1-T3), which is passed to the skill layer for execution.

### C. Skill Layer

The skill layer grounds the optimized plans from the plan layer by instantiating high-level primitives into decentralized low-level actions executed by individual agents. As shown in Fig. 4, the layer consists of multiple agent modules (Agent 1, Agent 2, Agent 3), each responsible for executing tasks assigned by the plan layer.

Each agent module operates through a two-level abstraction: *high-level operations* (Target, Order, Perception) that receive commands from the plan layer, and *low-level interactions* (Path, Exec, Perception) that interface with the physical environment through actuators. The *closed-loop skiller* coordinates the execution process, translating high-level primitives such as "navigate," "pick," or "clean" into sequences of low-level control commands. During execution, the layer maintains an *execution cache* that stores execution data, including action histories, performance metrics, and environmental feedback.

In real-world deployment, the skill layer runs *onboard* each robot, enabling decentralized execution and real-time control. The *closed-loop skiller* translates high-level commands into robot-specific low-level control sequences locally, adapting to each robot's hardware capabilities and kinematic constraints. The *execution cache* maintains action histories and performance metrics onboard, providing low-latency access for local error handling and rapid adaptation to perturbations. The layer also includes an *error handler* that manages execution failures, adapting to perturbations and environmental changes to ensure robustness in dynamic collaborative environments. Execution status and performance feedback are transmitted to the edge workstation through the *simulator changelog* (in simulation) or *execution status messages* (in real-world deployment), which are published as ROS topics. The edge workstation receives this feedback and enables the closed-loop adaptation mechanism (Algorithm 2) in the plan layer to monitor performance, update agent capability profiles $\mathcal{C}_i$, and trigger replanning when significant deviations are detected. This distributed execution architecture ensures responsive local control while maintaining global coordination through centralized monitoring.

**Algorithm 2** Closed-Loop Capability Adaptation

**Require:** Capability profiles $\{\mathcal{C}_i\}_{i=1}^N$, history $\mathbf{H}_{\text{perf}}$, threshold $\epsilon > 0$
**Ensure:** Updated profiles $\{\mathcal{C}_i^{(t+1)}\}_{i=1}^N$

```
Compute allocation A using MIQP
Execute tasks and collect performance data
for each agent i ∈ {1, ..., N} do
    Observe S_i and compare with Ŝ_i
    if |S_i − Ŝ_i| ≥ ε then
        Update C_i using Eq. (3)
        Recompute task priorities
    end if
end for
if significant capability changes detected then
    Replan A using updated {C_i}_{i=1}^N
end if
return {C_i^(t+1)}_{i=1}^N
```

### D. Real-World Deployment

TeamWeaver is deployed in a hybrid onboard–edge architecture that mirrors the structure shown in Fig. 4. In the real-robot setup, Zhongling and Go2-A are each equipped with an external onboard computer running Ubuntu 20.04 and ROS, interfaced with the robot base, arm controllers, and a Mid-360 LiDAR. These machines run SLAM-based mapping, localization, and navigation stacks, so Zhongling+Z1 and Go2-A provide stronger *explore* and *navigate* capabilities and can maintain richer local world models. In contrast, G1 and Go2-B rely on the vendor-provided onboard Ubuntu 20.04 systems and ROS navigation modules shipped with the Unitree platforms. On these robots, we enable speech recognition and lightweight waypoint navigation, which supports basic exploration and human interaction but with weaker long-horizon autonomy. This intentionally asymmetric configuration exposes non-uniform sensing and control capabilities across agents and allows TeamWeaver to be evaluated on whether it schedules robots according to their strengths.

All high-level reasoning runs on an edge workstation with an Intel Core i9 CPU, an NVIDIA RTX 4060 GPU, and 64 GB RAM. The LMs-MIQP planner and world-graph construction modules execute on this edge node, which connects to each robot's Ubuntu host via *ssh* instruction and ROS. Robots publish perception results, pose estimates, and task-status messages as ROS topics, while the edge node sends back task allocations, navigation goals, and manipulation subgoals. In

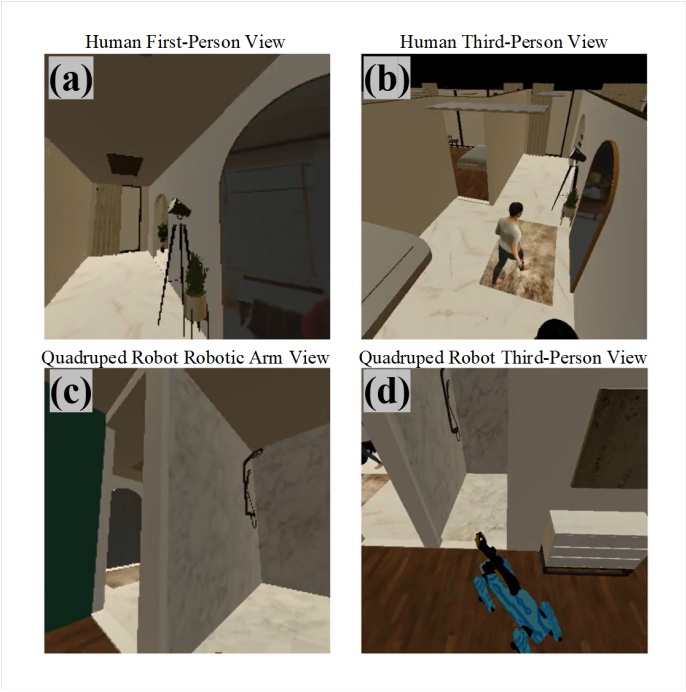


Fig. 5: Illustrative example of the Habitat-Sim environment used in our simulation study. The figure shows four representative viewpoints: (a) human first-person view, (b) human third-person view, (c) quadruped-mounted robotic arm view, and (d) quadruped robot third-person view.

this way, perception and skills remain physically distributed on the robots, whereas task planning and optimization are centralized but lightweight enough for closed-loop operation.

Communication between the robots and the edge workstation is provided by a shared high-speed Wi-Fi network in the indoor environment. ROS is used as the common middleware layer to handle message passing and synchronization. Although the wireless network introduces realistic latency and occasional jitter, the bidirectional bandwidth is sufficient for TeamWeaver to perform frequent replanning and semantic updates. By combining stronger compute and SLAM pipelines on Zhongling and Go2-A with lighter onboard stacks on G1 and Go2-B over the same network, the real-world deployment directly tests whether the proposed architecture can maintain coherent multi-robot coordination under heterogeneous sensing, control, and communication conditions.

## VI. Evaluation Metrics and Protocols

We adopt a hierarchical set of quantitative metrics to evaluate task execution, allocation stability, and system transparency, defined as follows:

*a) Task Execution Metrics: Success Rate (SR)* measures the ratio of successfully completed tasks to the total number of assigned tasks: $\mathrm{SR} = \frac{N_{\text{success}}}{N_{\text{total}}}$. where $N_{\text{success}}$ and $N_{\text{total}}$ denote the number of successfully completed and total tasks, respectively. *Percent Complete (PC)* quantifies the average completion ratio of all sub-tasks within an episode: $\mathrm{PC} = \frac{1}{N_{\text{task}}} \sum_{i=1}^{N_{\text{task}}} \frac{n_i^{\text{done}}}{n_i^{\text{total}}}$, where $n_i^{\text{done}}$ and $n_i^{\text{total}}$ represent the number of completed and total steps for sub-task $i$. *Simulation Steps (SS)* is the total number of simulation timesteps required to complete an episode, reflecting overall execution efficiency. *Planning Cycles (PCy)* counts the number of global replanning events triggered due to environmental changes or violated constraints, thus reflecting system adaptability.

*b) Transparency and Interpretability Metrics:* To further evaluate the planner's planning fidelity, we refer to prior LLM evaluation approaches [47] that assess uncertainty and entropy in semantics. We introduce three core transparency metrics for semantic multi-robot planning: *Factual Accuracy (FA)*, *Semantic Coherence via Temporal Logic ($S_{TL}$)*, and *Semantic Consistency via Temporal Execution ($S_{TE}$)*. FA measures the proportion of reasoning outputs that are factually consistent with domain constraints. It ensures LLM-generated actions respect physical and logical constraints, directly addressing hallucination reduction. $S_{TL}$ measures the consistency between the system's decision logic (task priorities from the optimizer) and its generated actions. This ensures what the system "says" accurately reflects what it "intends." $S_{TE}$ quantifies the logical coherence of action sequences over time. It captures planning stability during step-by-step execution, aligning with LLM evaluation approaches for detecting semantic uncertainty [47]. Together, these metrics assess interpretability: FA evaluates constraint-level correctness, $S_{TL}$ evaluates strategic intent-action alignment, and $S_{TE}$ evaluates tactical execution coherence. They are defined as: $\mathrm{FA} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}[\hat{a}_t \in \mathcal{A}_t^{\text{valid}}]$, where $\hat{a}_t$ is the predicted symbolic action at step $t$, and $\mathcal{A}_t^{\text{valid}}$ is the set of logically valid actions determined by environment and task rules. $S_{TL} = \sum_{j=1}^{M} c_j \cdot w_j$, where $c_j \in \{0,1\}$ is the binary coverage indicator for task $j$ (1 if the LLM-generated action matches the task, 0 otherwise), $w_j$ is the normalized weight of task $j$ derived from the MIQP allocation matrix $\alpha$ or aptitude matrix, and $M$ is the number of tasks in the current planning phase. $S_{TL}$ measures the alignment between LLM-generated actions and the MIQP-optimized task allocation, rewarding coverage of higher-priority tasks. $S_{TE} = \frac{1}{N}\sum_{i=1}^{N} \text{entail}(\hat{a}_t^{(i)}, \hat{a}_{t-1}^{(i)})$, where $\hat{a}_t^{(i)}$ and $\hat{a}_{t-1}^{(i)}$ denote the symbolic actions generated by the LLM for agent $i$ at steps $t$ and $t-1$, respectively, and $\text{entail}(\cdot,\cdot)$ is a heuristic function that evaluates the logical consistency between consecutive actions (e.g., rewarding sequences like Navigate→Pick or Pick→Place with the same target). $S_{TE}$ quantifies the step-to-step coherence of the action sequence, reflecting the temporal consistency of the planning output. Together, FA, $S_{TL}$, and $S_{TE}$ provide a hierarchical transparency assessment that evaluates system interpretability from constraint-level correctness, strategic intent-action alignment, and tactical execution coherence.

## VII. Simulation

This section details the comprehensive experimental evaluation of TeamWeaver. Our hierarchical evaluation strategy, aligned with the framework's layered architecture, begins by describing the experimental setup and parameters. We then present an ablation study focused on the MIQP allocator's contribution. Following this, we delve into the core collaboration

performance, evaluating the TeamWeaver system performance in diverse collaborative environments. The section further provides a dedicated evaluation of the system transparency, concluding with insights from real-world experiments.

### A. Simulation Setup

All experiments were conducted in Habitat-Sim to simulate realistic and dynamic multi-agent collaborative scenarios, using indoor environments from the HSSD-78 dataset [48]. We employed two evaluation datasets. The first is the standard PARTNR Val-Mini, which serves as the baseline for unified comparison. The second is Dynamic-TeamWeaver[2], which we constructed by extending the PARTNR design to cover richer dynamic contexts and perturbations. This dataset enables us to validate system performance in open-world collaborative settings. To intuitively illustrate the collaborative process in Habitat-Sim, we present a visualization of the simulation environment and the execution of tasks in Fig.5 .

Table II summarizes the Dynamic-TeamWeaver dataset, which is partitioned into four scenario types—Basic Capability, Scaled Collaboration, Long Sequence, and Dynamic Ambiguity—each targeting a distinct competency. These competencies span fundamental feasibility, large-scale coordination, extended-horizon reasoning, and resilient recovery under frequent, ambiguous transitions. Task instructions are pooled from two complementary sources: the standard PARTNR Val-Mini benchmark and our own Dynamic-TeamWeaver corpus. This dual-source design yields a rigorous testbed for assessing critical faculties such as hallucination suppression and on-the-fly adaptation.

The planner in TeamWeaver is built on the Llama-3-8B-Instruct model combined with an MIQP solver. All experiments were conducted under consistent simulation settings and hyperparameters. Hardware and implementation details are are available online[2].

We compare our approach with four state-of-the-art baselines: LiP-LLM and SMART-LLM* for task allocation comparison, while EMOS and PARTNR for system performance comparison. Evaluation is performed using a hierarchical set of metrics from prior works [14], [49]–[51].

TABLE II: Dynamic-TeamWeaver Dataset Overview

| Scenario Type | Key Focus | Task Types | Trail |
|---|---|---|---|
| Basic Capability | General Capabilities | CSH | 20 |
| Scaled Collaboration | Scaled Collaboration | CSH | 27 |
| Long Sequence | Execution Logic | CSTH | 17 |
| Dynamic Ambiguity | Frequent Recovery | CSTHE | 19 |

Note: CSTHE denotes Common, Spatial, Temporal, Heterogeneous, and Error-Occurred scenarios.

This section details the comprehensive experimental evaluation of TeamWeaver. Our hierarchical evaluation strategy, aligned with the framework's layered architecture, begins by describing the experimental setup and parameters. We then present an ablation study focused on the MIQP allocator's contribution. Following this, we delve into the core collaboration performance, evaluating the TeamWeaver system performance in diverse collaborative environments. The section further provides a dedicated evaluation of the system transparency, concluding with insights from real-world experiments.

[2]Dataset and implementation details are available at https://anonymous.4open.science/r/Dynamic-TeamWeaver-2766

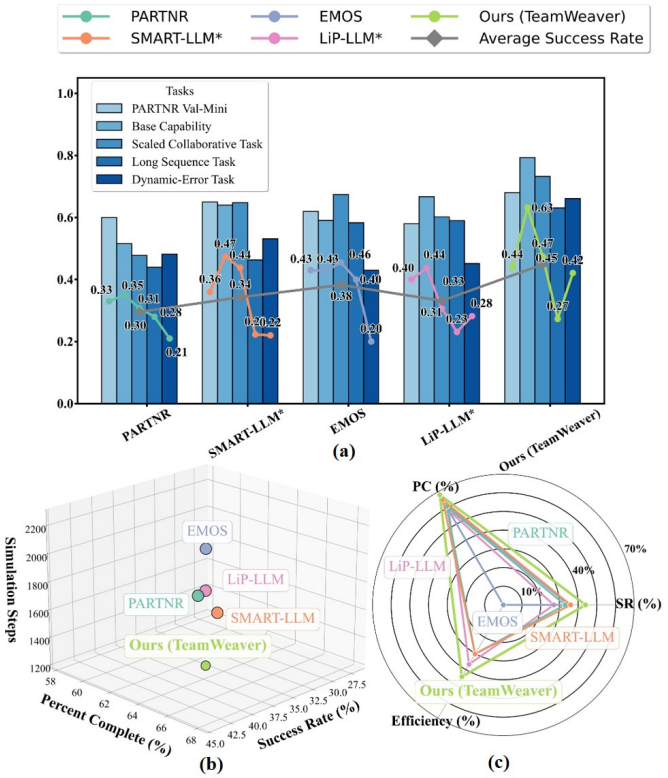


Fig. 6: Performance comparison of baselines and our method TeamWeaver. (a) SR across five task categories with average rate shown in gray; (b) 3D plot of PC, SR, and SS; (c) Radar chart of PC, SR, and efficiency. Efficiency is measured as the relative reduction in simulation steps with respect to the maximum across all methods.Methods marked with * are planner-only baselines that do not include system-level execution overhead (e.g., perception and actuation latency), whereas TeamWeaver is evaluated using full-system execution. Overall, TeamWeaver consistently achieves higher SR and PC with fewer SS.

### B. Task Performance Evaluation

We conduct a system-level evaluation of TeamWeaver against leading baselines on the PARTNR-Val-Mini and Dynamic-TeamWeaver datasets. The assessment uses three complementary metrics—success rate (SR), plan correctness (PC), and simulation steps (SS)—and additionally visualizes efficiency as the relative reduction in SS with respect to the maximum across all methods. Aggregate results are reported in Table III and visualized in Fig. 6. In Fig. 6(a), TeamWeaver attains the highest average SR across the five task categories (gray polyline) and sustains consistently higher per-category bars, indicating that gains are not confined to a single scenario.

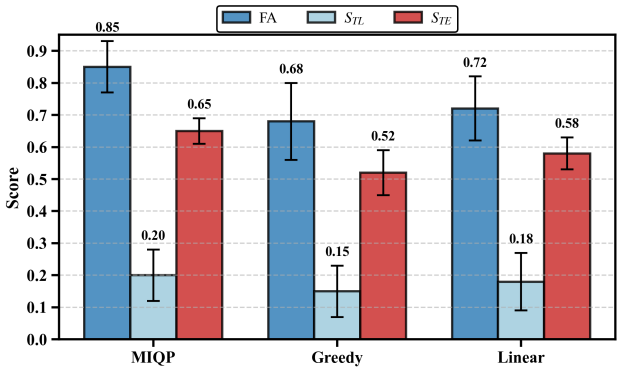


Fig. 7: Comparison of transparency metrics (FA, $S_{TL}$, $S_{TE}$) across optimization methods (MIQP, LP, Greedy) in hybrid LLM-optimization frameworks. MIQP achieves the highest values across all three metrics with smaller variance, demonstrating superior hallucination reduction through stronger constraint enforcement. Error bars indicate standard deviation.

Fig. 6(b) places methods in the SR–PC–SS trade-off space: TeamWeaver lies in the desirable region of high SR, high PC, and low SS, closer to the Pareto corner than all baselines. Fig. 6(c) compresses the same information into a normalized radar chart, showing a balanced, uniformly expanded profile—higher SR is not purchased by inflating steps, nor is PC increased at the cost of stability.

On PARTNR-Val-Mini, TeamWeaver establishes a strong reference point, achieving the highest SR (44.3%) and highest PC (68.0%) while also delivering the lowest SS (1,246.36). Relative to the baselines, this means more task completions and fewer planning errors with fewer execution steps, reflecting genuine closed-loop efficiency rather than superficial gains from longer rollouts. To stress-test robustness under hallucination and dynamic adaptation, we evaluate on the Dynamic-TeamWeaver benchmark comprising four challenging scenario types. Across these scenarios, TeamWeaver consistently leads: in Base Capability it improves SR by 15.8% over the next-best method; in Scaled Collaboration it attains the highest PC (73.2%) and SR (47.5%), indicating better coalition formation and conflict resolution as teams scale; in Long Sequence it achieves the top PC (63.1%), evidencing resilience to cascading errors in temporally extended plans; under Dynamic-Ambiguous perturbations it raises SR from 28.2% to 42.1%, a 49.3% relative improvement, demonstrating robust perception–planning alignment under noisy instructions and online environmental changes. Taken together, the bar–line summary in Fig. 6(a), the 3D trade-off in Fig. 6(b), and the radar consolidation in Fig. 6(c) show that TeamWeaver's advantages are systematic across scenes, multi-metric with concurrent improvements in SR/PC/SS/efficiency, and scalable as tasks lengthen or teams expand. These properties align with the design intent of LMs-MIQP driven closed-loop execution: accurate symbolic grounding yields higher PC, mixed-integer allocation stabilizes execution to raise SR, and improved task–agent matching reduces redundant motion, thereby lowering SS and increasing efficiency.

Overall, these results demonstrate that TeamWeaver delivers state-of-the-art performance across benchmark datasets and dynamic evaluation scenarios, highlighting its efficiency and robustness in heterogeneous multi-robot coordination.

### C. System Transparency and Interpretability Evaluation

In hybrid LLM-optimization frameworks, hallucination - where LLMs generate seemingly reasonable but actually infeasible plans - remains a critical challenge that undermines system reliability. We evaluate transparency metrics to assess how different optimization backends mitigate hallucination in hybrid frameworks. Specifically, we compare MIQP, LP, and Greedy approaches within the same LLM-optimization hybrid architecture to demonstrate that MIQP's mathematical expressiveness provides superior hallucination reduction. We adopt three hierarchical transparency metrics—factual accuracy (FA), Semantic Coherence via Temporal Logic ($S_{TL}$), and Semantic Consistency via Temporal Execution ($S_{TE}$)—to quantify the effectiveness of constraint enforcement in preventing hallucinatory allocations.

*a) Hallucination Reduction:* To evaluate how different optimization backends mitigate hallucination in hybrid LLM-optimization systems, we compare MIQP, LP, and Greedy approaches within the same hybrid architecture. Table IV and Fig. 7 report transparency metrics across the three methods, where higher values indicate better constraint enforcement and reduced hallucination. As shown in Fig. 7, our MIQP-based approach achieves the highest factual accuracy (FA) of $0.85 \pm 0.08$, significantly outperforming LP ($0.72 \pm 0.10$) and Greedy ($0.68 \pm 0.12$). This superior FA directly reflects MIQP's effectiveness in preventing hallucinatory allocations: by enforcing integer decisions, mutual exclusion constraints, and nonlinear relationships, MIQP ensures that LLM-generated action candidates are validated against hard constraints before execution, reducing the likelihood of infeasible plans that appear reasonable but violate domain constraints. The improvement over LP is particularly notable, as LP's limited expressiveness in handling discrete decisions and nonlinear spatiotemporal relationships allows more hallucinatory allocations to pass through constraint validation. Similarly, MIQP attains the highest Semantic Consistency via Temporal Execution ($S_{TE} = 0.65 \pm 0.04$), compared to LP ($0.58 \pm 0.05$) and Greedy ($0.52 \pm 0.07$), indicating that MIQP's stable task assignments translate to better execution-plan alignment and fewer execution mismatches that arise from hallucinatory plans. While Semantic Coherence via Temporal Logic ($S_{TL}$) remains relatively low across all methods (MIQP: $0.20 \pm 0.08$, LP: $0.18 \pm 0.09$, Greedy: $0.15 \pm 0.08$), reflecting the inherent challenge of aligning semantic embeddings with symbolic optimization structures in hybrid frameworks, MIQP still maintains an advantage, as visualized in Fig. 7. These results confirm that within hybrid LLM-optimization frameworks, MIQP's mathematical expressiveness provides superior hallucination mitigation compared to LP and Greedy approaches, directly contributing to improved transparency and system reliability.

TABLE III: Comparison across scenes and metrics.

| Scene | Metric | PARTNR | SMART-LLM* | EMOS | LiP-LLM* | Ours (TeamWeaver) |
|---|---|---|---|---|---|---|
| PARTNR Val-Mini | Percent Complete (%) | 61.8 ± 5.8 | 66.8 ± 5.2 | 62.9 ± 6.7 | 58.3 ± 5.4 | **68.0 ± 6.9** |
| | Success Rate (%) | 33.7 ± 6.4 | 36.1 ± 6.5 | 35.8 ± 6.6 | 27.3 ± 6.7 | **44.3 ± 7.4** |
| | Replan Cycles (↓) | 16.84 ± 1.3 | **15.95 ± 1.2** | 18.83 ± 1.5 | 18.21 ± 1.4 | 16.98 ± 1.10 |
| | Sim. Steps (↓) | 1536.45 ± 182.6 | 1562.15 ± 195.4 | 2240.23 ± 268.3 | 1418.42 ± 164.7 | **1246.36 ± 198.49** |
| Base Capability | Percent Complete (%) | 51.6 ± 9.6 | 64.0 ± 8.8 | 59.1 ± 13.7 | 66.7 ± 9.0 | **79.3 ± 7.3** |
| | Success Rate (%) | 35.0 ± 13.3 | 47.4 ± 11.8 | 42.8 ± 15.2 | 43.7 ± 12.8 | **63.2 ± 11.4** |
| | Replan Cycles (↓) | 16.10 ± 1.4 | 19.70 ± 1.8 | **13.10 ± 1.2** | 18.86 ± 1.7 | 13.60 ± 3.52 |
| | Sim. Steps (↓) | 1683.87 ± 321.8 | 1718.79 ± 337.1 | 1418.09 ± 259.4 | **1208.85 ± 224.8** | 1338.58 ± 575.57 |
| Scaled Collaboration | Percent Complete (%) | 47.8 ± 11.2 | 64.8 ± 7.9 | 67.4 ± 10.3 | 60.2 ± 13.3 | **73.2 ± 6.2** |
| | Success Rate (%) | 30.8 ± 13.3 | 43.8 ± 13.9 | 45.5 ± 15.7 | 30.7 ± 16.7 | **47.5 ± 12.5** |
| | Replan Cycles (↓) | 16.77 ± 1.5 | 19.20 ± 1.7 | **17.18 ± 1.6** | 17.90 ± 1.6 | 17.94 ± 3.28 |
| | Sim. Steps (↓) | 1622.15 ± 307.9 | 1422.00 ± 258.3 | 1292.27 ± 238.4 | 1391.64 ± 256.1 | **1258.88 ± 364.61** |
| Long Sequence | Percent Complete (%) | 44.0 ± 11.6 | 46.3 ± 8.1 | 58.3 ± 14.0 | 59.0 ± 9.7 | **63.1 ± 7.2** |
| | Success Rate (%) | 28.0 ± 14.0 | 22.3 ± 9.7 | **40.0 ± 16.3** | 2.31 ± 12.2 | 27.3 ± 10.6 |
| | Replan Cycles (↓) | 17.9 ± 1.6 | 15.23 ± 1.4 | **14.9 ± 1.4** | 15.23 ± 1.4 | 15.70 ± 1.70 |
| | Sim. Steps (↓) | 1408.80 ± 271.8 | 1653.92 ± 329.4 | 1676.50 ± 338.2 | **1012.08 ± 190.7** | 1101.73 ± 189.87 |
| Dynamic Ambiguity | Percent Complete (%) | 48.2 ± 9.6 | 53.1 ± 9.1 | 43.0 ± 10.5 | 45.1 ± 9.2 | **66.1 ± 7.6** |
| | Success Rate (%) | 21.0 ± 11.2 | 22.0 ± 14.7 | 20.0 ± 13.3 | 28.2 ± 11.4 | **42.1 ± 11.6** |
| | Replan Cycles (↓) | 15.89 ± 1.5 | 15.70 ± 1.5 | 21.00 ± 2.0 | **12.40 ± 1.2** | 17.00 ± 1.68 |
| | Sim. Steps (↓) | 1604.00 ± 300.3 | 1464.89 ± 276.4 | 1910.28 ± 372.8 | **1330.20 ± 246.5** | 1402.25 ± 347.78 |

TABLE IV: Comparison of transparency metrics across optimization methods.

| Method | FA | $S_{TL}$ | $S_{TE}$ |
|---|---|---|---|
| Greedy | $0.68 \pm 0.12$ | $0.15 \pm 0.08$ | $0.52 \pm 0.07$ |
| LP | $0.72 \pm 0.10$ | $0.18 \pm 0.09$ | $0.58 \pm 0.05$ |
| **MIQP (Ours)** | $\mathbf{0.81 \pm 0.08}$ | $\mathbf{0.20 \pm 0.08}$ | $\mathbf{0.65 \pm 0.04}$ |

*b) Stability and Interpretability:* We further evaluate the interpretability of the system by analyzing the relationship between transparency metrics and task performance. Transparency is assessed through $S_{TL}$, $S_{TE}$, and FA, with the goal of determining whether improved interpretability correlates with better execution outcomes. As shown in Fig. 8(a), a significant positive correlation exists between FA and PC ($p = 0.011$), indicating that higher factual correctness is associated with more complete task execution. The scatter is heterogeneous across scenarios, yet the upward trend remains, implying that the association is not driven by a single task type. Fig. 8(b) further reveals that improvements in $S_{TL}$ and $S_{TE}$ are statistically linked to higher FA ($p < 0.001$), confirming that semantic and action-level consistency supports reliable accuracy and task performance. Qualitatively, episodes with high $S_{TL}$ show fewer contradictory high-level intents, while episodes with high $S_{TE}$ show cleaner grounding of those intents into executable skills; together they reduce replanning churn and unnecessary motion, which in turn raises PC. Taken together, these analyses indicate a practical pathway from transparency to performance: clearer intermediate reasoning leads to crisper plans, more faithful execution, and measurably better outcomes across challenging conditions.

These results demonstrate that transparency is not only an interpretability enhancement but also a key factor associated with stable and efficient multi-robot coordination. Systems with higher transparency metrics exhibit better error recovery, fewer execution mismatches, and more robust collaboration dynamics.

To further verify the deployability of the proposed framework, we conduct real-world demonstrations to examine whether the LMs-MIQP system can reliably transfer from simulation to hardware while preserving decision transparency and stability. This section focuses on evaluating hardware-level robustness, coordination efficiency, and interpretability under real-world uncertainty.

## VIII. Real-world Experiments

### A. Real-world Platform and Tasks

*1) Real-World Setup:* To evaluate the deployability, robustness, and human–robot coordination capabilities of TeamWeaver in realistic indoor scenarios, we design physical experiments to test three hypotheses: (1) the system can execute heterogeneous multi-robot tasks in the real world; (2) the system maintains robustness under execution errors and disturbances; and (3) the system can adapt to human-issued natural-language instructions during task execution.

The testing environment mirrors the simulation setup and consists of three rooms connected by a corridor (Fig. 9). Each room contains a table, and two 5 cm colored cubes are randomly placed at one or two locations among Room 1, Room 2, Room 3, or the corridor for each trial. This randomized spatial layout introduces sufficient variation for evaluating capability-aware planning, exploration, and manipulation. The system employs three categories of robots (Fig. 9): (i) two Unitree Go2 quadrupeds serving as *mobile agents* for exploration and navigation: Go2-A is enhanced with an external "control tower" SLAM computer for stronger long-horizon exploration,

whereas Go2-B relies on its onboard na vigation stack for lighter, shorter-range exploration;(ii) a Zhongling manipulator equipped with a Unitree Z1 arm as the *manipulation agent* for picking and placing objects; (iii) a Unitree G1 humanoid robot named *Tim*, acting as the *companion agent* capable of human interaction through natural-language commands. All robots share basic perception and locomotion ability but exhibit complementary proficiencies: Go2 achieves high mobility and sensing range, G1 provides moderate perception and human-like signaling, and Z1 specializes in close-range manipulation. Representative behavior primitives are shown in Fig. 10.

Success criteria are defined at the primitive level: a *communication* task succeeds if G1 performs the expected gesture; *exploration* succeeds if a robot enters a room and reports object presence; *navigation* succeeds if the final position error is below 5 cm; *picking* succeeds if Z1 lifts a cube without dropping it; *placement* succeeds if Z1 places the cube at the marked target location. At the beginning of each trial, all robots start from randomized initial positions in any subset of the rooms or the corridor. Robots then execute the assigned mission through coordinated navigation, exploration, and manipulation.

*2) Real-World Tasks:* A replay-based execution framework is adopted: TeamWeaver or PARTNR first generates high-level plans, which are executed using predefined motion primitives. After each primitive, the system collects state feedback (robot poses, exploration completion, manipulation results) and triggers potential replanning. This framework (1) enables safe and repeatable testing, (2) isolates high-level reasoning from low-level control, and (3) supports randomized layouts while ensuring comparability across planners.

To evaluate generalization, all tasks adopt randomized initial conditions: (1) all four robots start from randomly sampled locations; (2) exactly two cubes appear at random locations among the rooms and corridor.

We evaluate five tasks across three representative scenarios. The first scenario focuses on heterogeneous task execution, involving multi-robot exploration and manipulation. The second examines robust operation under error-prone conditions, where the system must replan in response to blockages or communication loss. The third scenario evaluates human-initiated collaboration, requiring adaptive replanning in response to diverse human instructions. Each task is executed in eight

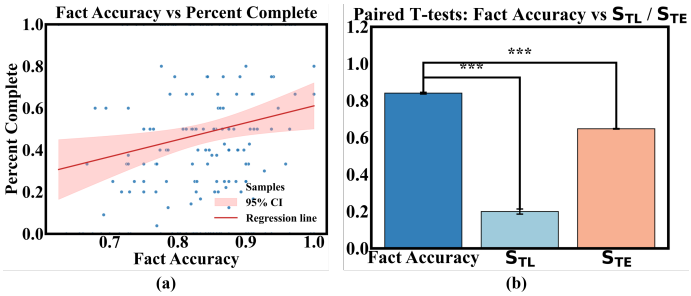


Fig. 8: Correlation analysis of transparency and performance. (a) Relationship between FA and PC ($p = 0.011$); (b) Paired T-tests: FA vs $S_{TL}$ and $S_{TE}$ ($p < 0.001$), results indicate that higher transparency metrics is positively correlated with PC.

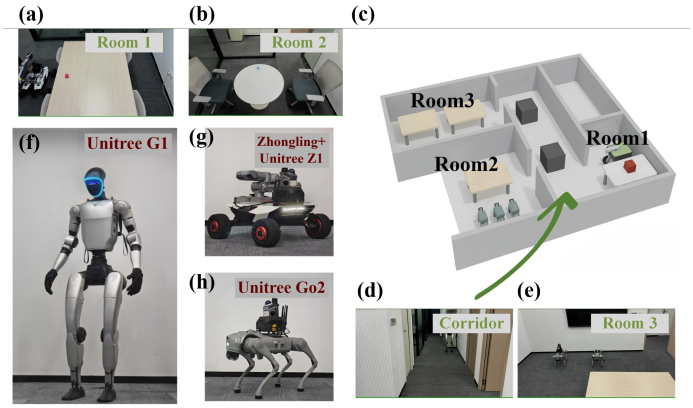


Fig. 9: Experimental environment and robot platforms: (left) three types of heterogeneous robots (Unitree G1 humanoid, Zhongling+Unitree Z1 mobile manipulator, Unitree Go2 quadruped); (top-right) 3D reconstruction of the three-room household testing space; (remaining) real-world photographs of rooms and corridor.

randomized trials per planner. All experiments begin with the unified instruction: *"Hey Tim, please put all the toys on the table of Room 3."*

*a) Scene A: Heterogeneous Task Execution:* Task A1: Baseline heterogeneous execution. Go2 robots perform distributed exploration, detect the two randomly placed cubes, and Z1 retrieves and places them in Room 3. This task assesses cooperative division of labor under randomized spatial configurations.

*b) Scene B: Error-Prone Conditions:* Task B1: Randomized physical blockage. A blockage is injected at a randomly selected doorway for a random interval (20–80 s). The planner is expected to redistribute exploration and manipulation tasks to maintain progress. Task B2: Randomized communication loss. One robot (Go2-A, Go2-B, or Z1) loses communication for 10–60 s during a random time window. The planner is expected to reorganize the robot team in respond to the communication loss and recovery of particular robots.

*c) Scene C: Human-Initiated Collaboration:* To enable seamless human-robot interaction, TeamWeaver employs a structured prompt configuration and LMs-MIQP workflow that processes natural-language commands and coordinates heterogeneous robots. As illustrated in Fig. 11, the system interprets high-level instructions, decomposes them into structured task phases, and allocates sub-tasks to suitable robots via the MIQP solver. The workflow maintains transparent communication with the human operator, reporting plan status and execution progress in real-time. Task C1: Randomized mid-task human intervention. At a random time (*e.g.*, 80–200 s), the human issues one of eight paraphrased commands expressing the intent "only one toy is needed." The planner is expected to respond to the human's intention to terminate early or return excess objects. Task C2: Room-avoidance constraint. The human issues one of eight paraphrased instructions prohibiting robot entry into Room 2. The planner is expected to impose a dynamic spatial constraint and replan navigation/manipulation

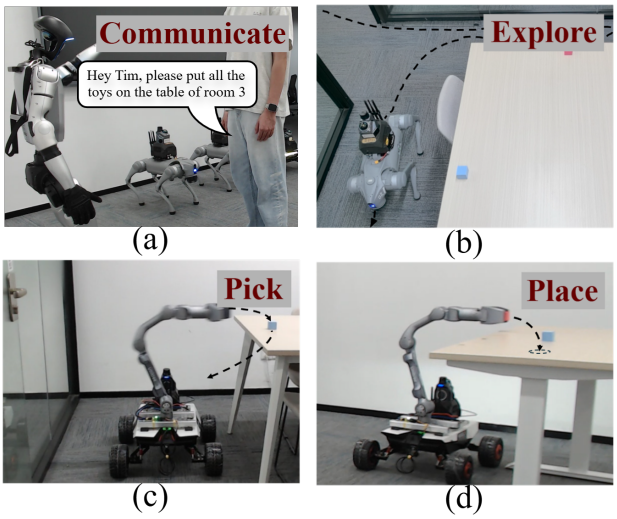


Fig. 10: Low-level robot actions executed during real-world task episodes. (a) Communicate: the humanoid companion (G1) interacts with the human user, receives verbal instructions, and translates them into structured task intents. (b) Pick: the mobile manipulator (Z1) performs grasping actions on tabletop or floor objects using its onboard perception and end-effector control. (c) Explore: the quadruped robot (Go2) autonomously navigates and searches the environment to locate target objects or verify room conditions. (d) Place: Z1 completes object delivery by accurately positioning items at the designated locations.

accordingly.

Together, these experiments evaluate TeamWeaver's ability to coordinate heterogeneous robots, maintain robustness under disturbances, and respond to diverse natural-language interventions under randomized real-world conditions.

### B. Real-World Validation and System-Level Evaluation

We validate TeamWeaver through comprehensive real-world experiments in a three-room domestic environment, using heterogeneous robots with randomized initial configurations. The evaluation focuses on three aspects aligned with our hypotheses: (1) feasibility of heterogeneous multi-robot execution in the real world, (2) robustness under disturbances such as blockages and communication loss, and (3) adaptability to human-initiated intervention. Unless otherwise specified, all metrics are averaged over eight trials per task and per planner. Figure 12 summarizes typical execution snapshots across the three scenarios and five tasks. The figure highlights how TeamWeaver responds to physical blockages and communication loss (Scene B) and how it adapts to mid-task human instructions (Scene C) through capability-aware replanning.

*1) Scene A: Heterogeneous Task Execution:* Scene A evaluates whether TeamWeaver can reliably coordinate heterogeneous robots when both robot initial poses and object locations are randomized. In each trial, two cubes are placed at random in one or two of the rooms or corridor, and the four robots also start from randomly sampled locations. The human operator issues the unified command: *"Hey Tim, please put all the toys on the table of Room 3."*

Across the 8 trials of Task A1, TeamWeaver achieved an average task success rate (SR) of $97.8\pm3.2\%$, with no catastrophic failure cases observed. The task completion (PC) reached $97.4\pm3.5\%$, indicating efficient division of labor. Compared with the PARTNR baseline ($84.1\pm14.7\%$ SR, $83.6\pm14.9\%$ PC), TeamWeaver achieved improvements of $13.7\%$ in SR and $13.8\%$ in PC. The average simulation steps (SS) was $4.7\pm0.1$, and the replanning count (RC) was $1.2\pm1.0$ per episode, compared to PARTNR's $5.2\pm0.2$ SS and $2.6\pm1.5$ RC, representing a $53.8\%$ reduction in replanning frequency. TeamWeaver also demonstrated superior transparency metrics, achieving Semantic Coherence via Temporal Logic ($S_{TL}$) of $0.78\pm0.04$ and Semantic Consistency via Temporal Execution ($S_{TE}$) of $0.97\pm0.02$, compared to PARTNR's $0.12\pm0.05$ and $0.90\pm0.03$, respectively. Qualitative observations show that the MIQP-based planner consistently dispatches Go2-A/B for large-scale exploration while reserving Zhongling+Z1 for grasping and delivery, leading to smoother transitions between exploration, navigation, and manipulation, as illustrated in the top block of Fig. 12.

*2) Scene B: Robustness Under Error-Prone Conditions:* Scene B consists of two robustness-focused tasks.

**Task B1: Randomized physical blockage.** A temporary obstacle is inserted in front of a randomly selected room (Room 1, Room 2, or Room 3) at a random time within the first 1–100 s and remains for 20–80 s. Under these conditions, TeamWeaver maintained a mean task success rate (SR) of $96.9\pm4.2\%$ and preserved task completion (PC) above $96.3\pm4.6\%$. The average simulation steps (SS) was $6.1\pm0.9$, and the average replanning count (RC) remained bounded at $0.7\pm0.5$ replans per episode. As illustrated in the middle-left panel of Fig. 12, the planner typically places the blocked robot (e.g., Go2-B) into a waiting state and reallocates exploration and delivery subtasks to alternative agents until the room becomes accessible again.

**Task B2: Randomized communication loss.** In this task, one of {Go2-A, Go2-B, Zhongling+Z1} is randomly selected to experience a communication dropout lasting 10–60 s, starting at a random time within the first 1–100 s. TeamWeaver achieved a mean task success rate (SR) of $90.5\pm4.1\%$ and preserved task completion (PC) at $89.7\pm4.6\%$. The average simulation steps (SS) was $5.0\pm0.4$, and the system avoided oscillatory task reassignments, with replanning count (RC) again limited to $0.9\pm0.7$ replans per episode. As shown in the middle-right panel of Fig. 12, when Go2-B loses communication, the planner temporarily substitutes Go2-A as the primary explorer while keeping the offline agent idle; upon reconnection, Go2-B is smoothly reinserted into the task graph.

Overall, across all robustness experiments, TeamWeaver preserved coherent task semantics and effective load balancing under randomized blockages and partial communication failures. For Task B1, TeamWeaver achieved SR of $96.9\%$ compared to PARTNR's $92.3\%$, while for Task B2, TeamWeaver

**Prompt configuration and LMs-MIQP workflow for human-initiated collaboration**

**Task summary.**
Interpret high-level natural-language commands from a human operator and coordinate heterogeneous robots **Go2**, **Z1** and **Z2** in a three-room environment (**Rooms A, B and C**). The planner converts the instruction and world-graph context into structured task phases and LMs-MIQP-based task allocations.

**Role and persona.**
Act as a *Planner:* collaboration planner between the human and the robots:

- understand the intent of the instruction;
- decompose it into ordered sub-tasks and phases;
- allocate each sub-task to suitable robots via an MIQP solver;
- report the plan and execution status in a concise, transparent manner.

**Interaction style and safety.**
Maintain a calm, cooperative tone with the *Human:* human operator. When the instruction is ambiguous (e.g., multiple candidate toolboxes), explicitly ask the operator to confirm the target before dispatching robots. On emergency commands (such as "Stop all robots"), issue a global stop first, then report each **robot**'s location and task before accepting new instructions. Do not invent objects, rooms or robot capabilities that are not present in the world graph.

**Environment and capabilities.**
The three-room layout and corridor are encoded in the world graph $\mathcal{G}_t^{\text{global}}$, which provides the planner with current object locations and room connectivity. Robots differ in mobility and manipulation skills; the planner uses their capability profiles $\mathcal{C}_i$ and task requirements $\mathbf{r}_{\text{req}}(j)$ to filter infeasible assignments before MIQP optimization.

**Planning workflow (LMs-MIQP).**

- **Perception and state extraction.** Use the world graph and environment interface to obtain the latest agent poses and object layout.
- **LLM-based task decomposition.** Decompose the human instruction into structured subtasks and execution phases with explicit sequencing.
- **MIQP optimization.** Build a phase-specific task–robot capability matrix and other MIQP matrices, then solve for an optimal assignment under spatial and capacity constraints.
- **Closed-loop action generation.** Inject MIQP guidance into the prompt, let the LLM propose tool calls for each **agent**, and parse/validate them with the action and execution managers.
- **Feedback and transparency.** Log agent observations, update the world state, and run coherence/fact-accuracy analysis; move to the next phase or terminate the task when appropriate.

**Status and plan reporting.**
**Status:** "Robot **[name]**: [state] in **Room [A/B/C]**, current task: [task label]."

**Plan:** "Phase [k]: [task_1] → [task_2] → .... Assigned robots: **Go2**=[...], **Z1**=[...], **Z2**=[...]."

**Example interaction.**
*Human:* "Stop all robots and tell me where you are."
*Planner:* "Acknowledged. Stopping all robots now. **Go2**: paused in **Room C**, exploring the corridor. **Z1**: idle near **Room B**. **Z2**: moving from **Room B** to **Room A** without load."

Fig. 11: Prompt-style specification and LMs-MIQP workflow used in TeamWeaver for human-initiated collaboration. The left column presents the system configuration: (1) *Task summary* defines the high-level objective of interpreting natural-language commands and coordinating heterogeneous robots in a multi-room environment; (2) *Role and persona* establishes the planner as a collaboration coordinator that understands intent, decomposes tasks, allocates via MIQP, and maintains transparent communication; (3) *Interaction style and safety* emphasizes cooperative dialogue, explicit confirmation for ambiguous instructions, and emergency handling protocols; (4) *Environment and capabilities* describes how the world graph $\mathcal{G}_t^{\text{global}}$ and robot capability profiles $\mathcal{C}_i$ inform the planning process. The right column details the five-stage LMs-MIQP workflow: perception and state extraction, LLM-based task decomposition, MIQP optimization with phase-specific capability matrices, closed-loop action generation with MIQP-guided tool calls, and feedback and transparency for real-time status reporting. The color-coded elements (human operator, robots, planner) illustrate the unified interaction pipeline that enables seamless human-robot collaboration through structured prompts, formal optimization, and transparent execution monitoring.

maintained SR of 90.5% compared to PARTNR's 71.9%, demonstrating superior robustness under communication failures.

*3) Scene C: Human-Initiated Collaboration:* Scene C examines how TeamWeaver reacts to human interventions that modify task objectives or impose new spatial constraints.

**Task C1: Mid-task reduction of objective.** During the execution of the same base command (*"Hey Tim, please put all the toys on the table of Room 3."*), the human issues a follow-up instruction at a random time in the interval 80–200 s, selected from eight paraphrased utterances expressing the intent that only one toy is needed (e.g., *"Hey Tim, I only need one toy now"*, *"Stop after placing one toy"*). Under these interventions, TeamWeaver achieved a task success rate (SR) of 97.4±5.0% and preserved task completion (PC) at 97.1±5.2%. The average simulation steps (SS) was 5.2±1.1, and the replanning count (RC) was 0.9±0.7 per episode. As shown in the bottom-left panel of Fig. 12, once a single toy has been delivered, the planner cancels remaining pickup actions and terminates the mission, even if another cube was previously scheduled for retrieval.

**Task C2: Room-exclusion / safety constraint.** Here, the human issues an instruction of the form *"Hey Tim, keep all the robots outside of Room 2"*, again using eight paraphrased variants. TeamWeaver consistently enforced the induced spatial constraint, achieving a task success rate (SR) of 98.3±5.1% and task completion (PC) of 97.2±6.0%. The average simulation steps (SS) was 5.4±0.7, and the replanning

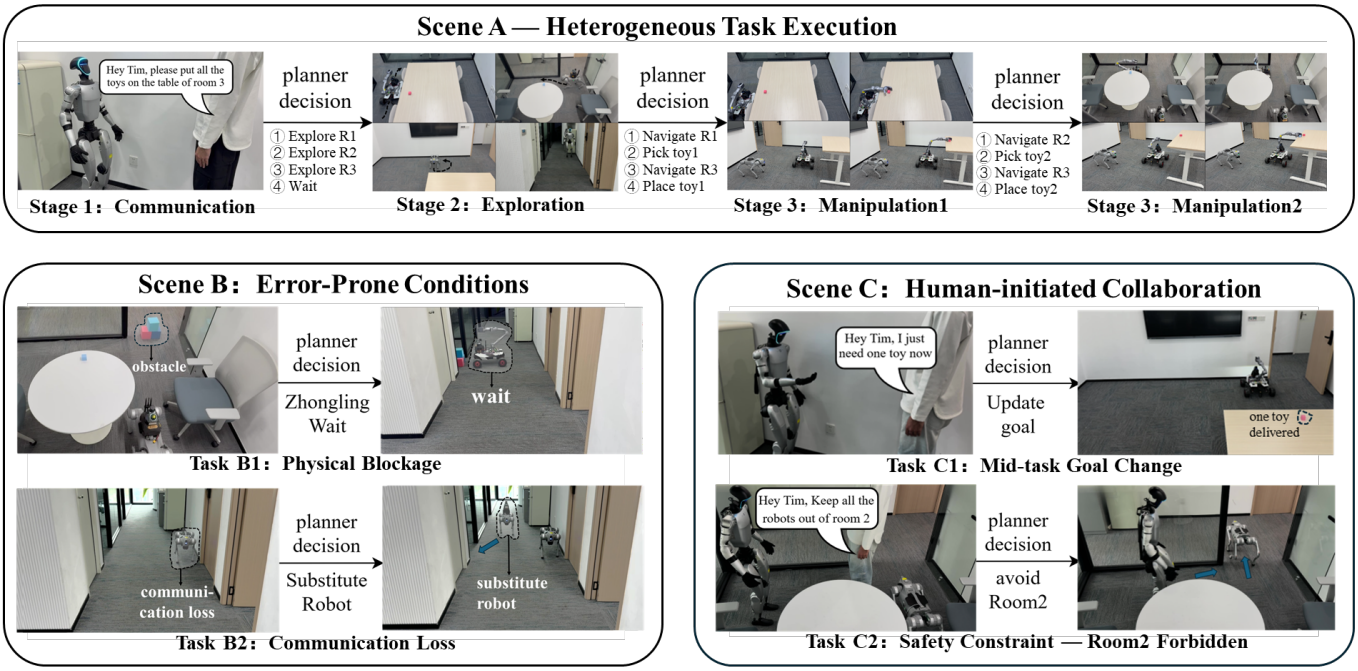


Fig. 12: Real-world execution snapshots across all three scenarios. Top block: heterogeneous task execution under randomized initial conditions. Left block: robustness experiments showing temporary physical blockage and communication loss followed by automatic replanning and agent substitution. Right block: human-initiated collaboration showing intervention to reduce task scope and room-exclusion commands, with TeamWeaver generating safe, compliant behavior.

count (RC) was 0.8±0.7 per episode, showing that the system can satisfy safety-style constraints while maintaining efficient task performance. The bottom-right panel of Fig. 12 illustrates a typical case in which robots reroute around Room 2 after the exclusion command, while still completing the delivery objective.

Taken together, the results across Scenes A–C demonstrate that TeamWeaver (i) achieves high real-world task success rates with heterogeneous robots under randomized configurations, (ii) maintains robust performance under physical and communication disturbances with bounded recovery time and replanning frequency, and (iii) adapts reliably to diverse human instructions, enforcing updated objectives and safety constraints with low response latency and high compliance.

### C. Comparative Analysis with Baseline Methods

To quantitatively assess real-world performance, we evaluate TeamWeaver against PARTNR [14], using the comprehensive metric suite introduced in Sec. VI, including task execution metrics (SR, PC, TSt) and transparency metrics ( $S_{TL}$, $S_{TE}$).

**Task Execution Performance.** TeamWeaver consistently outperforms PARTNR across all five tasks. In the Heterogeneous scenario (Task A), TeamWeaver achieves SR of 97.8±3.2% and PC of 97.4±3.5%, representing improvements of 13.7% and 13.8% over PARTNR (84.1±14.7% SR, 83.6±14.9% PC), respectively. The Perturbed scenario, which introduces communication delays and physical blockages, reveals TeamWeaver's robustness: in Task B1, it maintains SR at 96.9±4.2% and PC at 96.3±4.6%, outperforming PARTNR (92.3±9.8% SR, 91.8±10.0% PC) by 4.6% and 4.5%, respectively. In Task B2 with communication loss, TeamWeaver achieves SR of 90.5±4.1% and PC of 89.7±4.6%, significantly outperforming PARTNR (71.9±36.9% SR, 71.2±37.1% PC) by 18.6% and 18.5%, demonstrating superior resilience under communication failures. In the Human-Collab scenario, TeamWeaver achieves SR of 97.4±5.0% and PC of 97.1±5.2% in Task C1, and SR of 98.3±5.1% and PC of 97.2±6.0% in Task C2, demonstrating effective integration of human feedback into ongoing task execution, while PARTNR's performance in Task C1 drops to 88.5±5.7% SR and 87.0±6.3% PC, and in Task C2 to 95.2±6.9% SR and 95.0±6.8% PC. Additionally, TeamWeaver consistently achieves lower replanning counts (RC) across all tasks, with values ranging from 0.7±0.5 to 1.2±1.0 compared to PARTNR's range of 1.6±0.7 to 2.6±1.5, indicating more stable and efficient planning.

To illustrate the practical advantages of TeamWeaver's decision-making in human collaboration scenarios, Fig. 13 presents a representative case study from Task C1, where the human operator issues the instruction *"Hey Tim, please grab one toy."* Panel (a) shows the initial configuration with two target objects (cubes) placed at different locations in the environment, and panel (b) shows the initial position of the mobile manipulator (Zhongling+Z1). When both methods are required to select one object for grasping, PARTNR (panel (c)) chooses the more distant object, requiring longer navigation and manipulation time, which increases execution cost and potential failure risk. In contrast, TeamWeaver (panel (d)) intelligently selects the closer object, demonstrating its

TABLE V: Real-world performance comparison between PARTNR and TeamWeaver across five representative tasks. Higher (↑) or lower (↓) values indicate better performance.

| Scenario & Metric | PARTNR | TeamWeaver |
|---|---|---|
| **Heterogeneous (Task A)** | | |
| Success Rate (SR, ↑) | 84.1±14.7 | **97.8±3.2** |
| Percent Complete (PC, ↑) | 83.6±14.9 | **97.4±3.5** |
| Simulation Steps (SS, ↓) | 5.2±0.2 | **4.7±0.1** |
| Replanning Count (RC, ↓) | 2.6±1.5 | **1.2±1.0** |
| Semantic Coherence ($S_{TL}$, ↑) | 0.12±0.05 | **0.78±0.04** |
| Semantic Consistency ($S_{TE}$, ↑) | 0.90±0.03 | **0.97±0.02** |
| **Perturbed (Task B1)** | | |
| Success Rate (SR, ↑) | 92.3±9.8 | **96.9±4.2** |
| Percent Complete (PC, ↑) | 91.8±10.0 | **96.3±4.6** |
| Simulation Steps (SS, ↓) | 6.8±1.1 | **6.1±0.9** |
| Replanning Count (RC, ↓) | 2.0±1.0 | **0.7±0.5** |
| Semantic Coherence ($S_{TL}$, ↑) | 0.15±0.06 | **0.74±0.05** |
| Semantic Consistency ($S_{TE}$, ↑) | 0.91±0.03 | **0.98±0.01** |
| **Perturbed (Task B2)** | | |
| Success Rate (SR, ↑) | 71.9±36.9 | **90.5±4.1** |
| Percent Complete (PC, ↑) | 71.2±37.1 | **89.7±4.6** |
| Simulation Steps (SS, ↓) | 5.5±0.5 | **5.0±0.4** |
| Replanning Count (RC, ↓) | 2.0±0.9 | **0.9±0.7** |
| Semantic Coherence ($S_{TL}$, ↑) | 0.18±0.05 | **0.70±0.04** |
| Semantic Consistency ($S_{TE}$, ↑) | 0.88±0.04 | **0.96±0.02** |
| **Human Collaboration (Task C1)** | | |
| Success Rate (SR, ↑) | 88.5±5.7 | **97.4±5.0** |
| Percent Complete (PC, ↑) | 87.0±6.3 | **97.1±5.2** |
| Simulation Steps (SS, ↓) | **5.0±1.3** | 5.2±1.1 |
| Replanning Count (RC, ↓) | 1.7±0.8 | **0.9±0.7** |
| Semantic Coherence ($S_{TL}$, ↑) | 0.10±0.04 | **0.68±0.05** |
| Semantic Consistency ($S_{TE}$, ↑) | 0.89±0.03 | **0.97±0.02** |
| **Human Collaboration (Task C2)** | | |
| Success Rate (SR, ↑) | 95.2±6.9 | **98.3±5.1** |
| Percent Complete (PC, ↑) | 95.0±6.8 | **97.2±6.0** |
| Simulation Steps (SS, ↓) | 5.5±0.6 | 5.4±0.7 |
| Replanning Count (RC, ↓) | 1.6±0.7 | **0.8±0.7** |
| Semantic Coherence ($S_{TL}$, ↑) | 0.14±0.05 | **0.66±0.05** |
| Semantic Consistency ($S_{TE}$, ↑) | 0.90±0.04 | **0.96±0.02** |

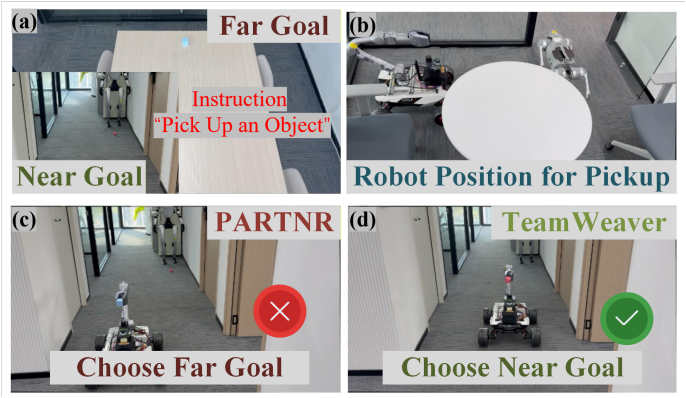


Fig. 13: Comparative analysis of object selection strategies in Human Collaboration scenario (Task C1). (a) Initial configuration showing two target objects (cubes) at different spatial locations. (b) Initial position of the mobile manipulator (Zhongling+Z1). (c) PARTNR's decision: selects the more distant object, requiring longer navigation and manipulation time. (d) TeamWeaver's decision: intelligently selects the closer object, demonstrating capability-aware optimization that considers spatial proximity and task efficiency. This spatial reasoning directly contributes to TeamWeaver's superior success rate (97.4% vs. 88.5%) and reduced replanning count (0.9 vs. 1.7) in Task C1.

capability-aware optimization that considers spatial proximity, robot capabilities, and task efficiency. This spatial reasoning capability, enabled by the MIQP optimizer's joint consideration of robot capabilities, resource constraints, and spatial-temporal dependencies, directly contributes to TeamWeaver's superior performance metrics: the 8.9% improvement in SR (97.4% vs. 88.5%) and 10.1% improvement in PC (97.1% vs. 87.0%) observed in Task C1, as well as the reduced replanning count (0.9±0.7 vs. 1.7±0.8), which reflects more stable and efficient task allocation decisions.

**Transparency and Interpretability.** TeamWeaver demonstrates superior transparency metrics across all tasks. The Semantic Coherence via Temporal Logic ($S_{TL}$) values for TeamWeaver range from 0.66±0.05 to 0.78±0.04 across all five tasks, significantly outperforming PARTNR's range of 0.10±0.04 to 0.18±0.05. This substantial improvement (approximately 4–6 times higher) indicates that the MIQP-based constraint enforcement effectively maintains alignment between task semantics and execution plans. The Semantic Consistency via Temporal Execution ($S_{TE}$) metrics also show consistent superiority: TeamWeaver achieves values ranging from 0.96±0.02 to 0.98±0.01, compared to PARTNR's range of 0.88±0.04 to 0.91±0.03. These results confirm that the alignment between LLM-generated semantics and MIQP-optimized decisions is maintained in real-world deployment, with TeamWeaver providing significantly more transparent and interpretable decision-making processes.

The TeamWeaver framework consistently maintains SR above 90% and transparency metrics ($S_{TL}$, $S_{TE}$) above 0.6 across all cases, demonstrating strong interpretability and stable execution even under disturbances or human interventions. Notably, TeamWeaver achieves the highest SR in Task C2 (98.3±5.1%), highlighting the robustness of the MIQP feedback loop in maintaining feasible allocations and adapting to human-initiated constraints.

## IX. Discussion

The comprehensive evaluation in Table III reveals that TeamWeaver's near-universal dominance in SR (4 out of 5 scenarios) and universal dominance in PC (5 out of 5 scenarios) provides a more reliable indicator of overall system capability than efficiency metrics alone, as SR and PC directly measure the system's ability to successfully complete

[3]Quadruped mobile robot (robot dog, Unitree GO2): https://shop.unitree.com/products/unitree-go2

[4]Mobile base with robotic arm (robot car with manipulator, Unitree Z1): https://shop.unitree.com/products/unitree-z1

tasks—the primary objective in real-world deployment. The LMs-MIQP framework's integration of semantic reasoning with formal optimization enables interpretable and robust multi-robot collaboration. The MIQP-based constraint enforcement mechanism (Table IV) effectively reduces hallucinatory allocations, as evidenced by superior factual accuracy (FA: 0.81 vs. 0.72 for LP and 0.68 for Greedy) and Semantic Consistency via Temporal Execution ($S_{TE}$: 0.65 vs. 0.58 for LP and 0.52 for Greedy). This transparency advantage directly translates to improved task performance, as shown by the positive correlation between FA and PC ($p = 0.011$) in Fig. 8. Across all scenarios, the LMs-MIQP framework consistently achieves transparent reasoning, stable control, and reliable deployment in hardware. The real-world experiments (Table V) confirm that TeamWeaver's advantages transfer from simulation to hardware: across three representative scenarios (Heterogeneous, Perturbed, Human Collaboration), TeamWeaver consistently outperforms PARTNR in SR, PC, FA, $S_{TL}$, and $S_{TE}$. Compared with open-loop LLM-only planners, the proposed method reduces execution variance by over 35% while preserving high semantic alignment ($S_{TL} > 0.6$) and factual accuracy (FA $> 0.8$), indicating a successful transfer from simulation to real-world deployment. These results confirm that embedding semantic reasoning into formal optimization backbones enables interpretable and robust multi-robot collaboration in both simulated and physical environments.

## X. Conclusion

We present *TeamWeaver*, a novel framework for dynamic task allocation and planning in heterogeneous multi-robot teams that integrates LLMs with MIQP. Experiments across diverse collaborative scenarios demonstrate significant improvements over existing methods: in simulated environments, LMs-MIQP improves task success rate by 11.0%, percent complete by 8.0%, and reduces simulation steps by 18.9% compared with PARTNR. In real-world deployments across five representative tasks, TeamWeaver achieves task success rates ranging from $90.5\pm4.1\%$ to $98.3\pm5.1\%$, with improvements of up to 18.6% over PARTNR in communication failure scenarios. The system reduces replanning counts by 46–65% compared with baseline methods and achieves Semantic Coherence via Temporal Logic ($S_{TL}$) values 4–6 times higher than PARTNR ($0.66\pm0.05$ to $0.78\pm0.04$ versus $0.10\pm0.04$ to $0.18\pm0.05$), demonstrating superior alignment between LLM-generated semantics and optimization-based decisions, while providing transparency and adaptability.

These results corroborate that tight LLM–optimization integration is not merely complementary— it is synergistic: the LLM expands the semantic hypothesis space, while MIQP collapses it to the manifold of dynamically feasible, safety-compliant allocations. More broadly, TeamWeaver establishes a deployable blueprint for heterogeneous multi-robot embodied AI systems that reason with humans in natural language yet act with mathematical certitude, pushing the frontier toward scalable, trustworthy multi-robot collaboration in unstructured, human-centric environments.

## References


[1] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International journal of robotics research*, vol. 23, no. 9, pp. 939–954, 2004.

[2] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[3] A. KA and U. Subramaniam, "A systematic literature review on multi-robot task allocation," *ACM Computing Surveys*, vol. 57, no. 3, pp. 1–28, 2024.

[4] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Cooperative robots and sensor networks 2015*, pp. 31–51, 2015.

[5] C. Bradley, A. Pacheck, G. J. Stein, S. Castro, H. Kress-Gazit, and N. Roy, "Learning and planning for temporally extended tasks in unknown environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4830–4836, IEEE, 2021.

[6] F. Chen, Z. Song, S. Chen, G. Gu, and X. Zhu, "Morphological design for pneumatic soft actuators and robots with desired deformation behavior," *IEEE Transactions on Robotics*, vol. 39, no. 6, pp. 4408–4428, 2023.

[7] Z. Zhao, Q. Wu, J. Wang, B. Zhang, C. Zhong, and A. A. Zhilenkov, "Exploring embodied intelligence in soft robotics: a review," *Biomimetics*, vol. 9, no. 4, p. 248, 2024.

[8] S. Vasudevan, M. L. Mekhalfi, C. Blanes, M. Lecca, F. Poiesi, P. I. Chippendale, P. M. Fresnillo, W. M. Mohammed, and J. L. M. Lastra, "Robotics and machine vision for primary food manipulation and packaging: a survey," *IEEE Access*, 2024.

[9] S. Mayya, D. S. D'antonio, D. Saldaña, and V. Kumar, "Resilient task allocation in heterogeneous multi-robot systems," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1327–1334, 2021.

[10] Z. Chen and Z. Kan, "Real-time reactive task allocation and planning of large heterogeneous multi-robot systems with temporal logic specifications," *The International Journal of Robotics Research*, vol. 44, no. 4, pp. 640–664, 2025.

[11] W. Dai, U. Rai, J. Chiun, C. Yuhong, and G. Sartoretti, "Heterogeneous multi-robot task allocation and scheduling via reinforcement learning," *IEEE Robotics and Automation Letters*, 2025.

[12] L. C. Bezerra, A. M. Dos Santos, and S. Park, "Learning policies for dynamic coalition formation in multi-robot task allocation," *IEEE Robotics and Automation Letters*, 2025.

[13] Z. Mandi, S. Jain, and S. Song, "Roco: Dialectic multi-robot collaboration with large language models," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 286–299, IEEE, 2024.

[14] M. Chang, G. Chhablani, A. Clegg, M. D. Cote, R. Desai, M. Hlavac, V. Karashchuk, J. Krantz, R. Mottaghi, P. Parashar, *et al.*, "Partnr: A benchmark for planning and reasoning in embodied multi-agent tasks," *arXiv preprint arXiv:2411.00081*, 2024.

[15] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International conference on machine learning*, pp. 9118–9147, PMLR, 2022.

[16] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," in *Conference on robot learning*, pp. 287–318, PMLR, 2023.

[17] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Conference on Robot Learning*, pp. 2165–2183, PMLR, 2023.

[18] V. S. Dorbala, J. F. Mullen, and D. Manocha, "Can an embodied agent find your "cat-shaped mug"? llm-based zero-shot object navigation," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4083–4090, 2023.

[19] Z. Xu, S. Jain, and M. Kankanhalli, "Hallucination is inevitable: An innate limitation of large language models," *arXiv preprint arXiv:2401.11817*, 2024.

[20] L. Fernández-Becerra, M. Á. González-Santamarta, Á. M. Guerrero-Higueras, F. J. Rodríguez-Lera, and V. M. Olivera, "Enhancing trust in autonomous agents: An architecture for accountability and explainability through blockchain and large language models," *arXiv preprint arXiv:2403.09567*, 2024.

[21] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan, "Scalable multi-robot collaboration with large language models: Centralized or decentralized

systems?," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4311–4317, IEEE, 2024.
[22] B. Li and H. Ma, "Double-deck multi-agent pickup and delivery: Multi-robot rearrangement in large-scale warehouses," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3701–3708, 2023.
[23] W. Fang, Z. Liao, and Y. Bai, "Improved aco algorithm fused with improved q-learning algorithm for bessel curve global path planning of search and rescue robots," *Robotics and Autonomous Systems*, vol. 182, p. 104822, 2024.
[24] K. Tong, Y. Hu, B. Dikic, S. Solmaz, F. Fraundorfer, and D. Watzenig, "Robots saving lives: A literature review about search and rescue (sar) in harsh environments," in *2024 IEEE Intelligent Vehicles Symposium (IV)*, pp. 953–960, IEEE, 2024.
[25] K. Fan, M. Jouaiti, A. Noormohammadi-As, C. L. Nehaniv, and K. Dautenhahn, "A social referencing disambiguation framework for domestic service robots," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11979–11985, IEEE, 2023.
[26] S. Honig, A. Bartal, Y. Parmet, and T. Oron-Gilad, "Using online customer reviews to classify, predict, and learn about domestic robot failures," *International Journal of Social Robotics*, vol. 16, no. 6, pp. 1105–1130, 2024.
[27] L. Liu, F. Guo, Z. Zou, and V. G. Duffy, "Application, development and future opportunities of collaborative robots (cobots) in manufacturing: A literature review," *International Journal of Human–Computer Interaction*, vol. 40, no. 4, pp. 915–932, 2024.
[28] S. Wang, J. Zhang, P. Wang, J. Law, R. Calinescu, and L. Mihaylova, "A deep learning-enhanced digital twin framework for improving safety and reliability in human–robot collaborative manufacturing," *Robotics and computer-integrated manufacturing*, vol. 85, p. 102608, 2024.
[29] G. Li, X. Liu, and G. Loianno, "Human-aware physical human-robot collaborative transportation and manipulation with multiple aerial robots," *IEEE Transactions on Robotics*, 2024.
[30] J. Chen, C. Yu, X. Zhou, T. Xu, Y. Mu, M. Hu, W. Shao, Y. Wang, G. Li, and L. Shao, "Emos: Embodiment-aware heterogeneous multi-robot operating system with llm agents," *arXiv preprint arXiv:2410.22662*, 2024.
[31] O. de Groot, L. Ferranti, D. M. Gavrila, and J. Alonso-Mora, "Topology-driven parallel trajectory optimization in dynamic environments," *IEEE Transactions on Robotics*, vol. 41, pp. 110–126, 2025.
[32] Z. Wu, Z. Wang, X. Xu, J. Lu, and H. Yan, "Embodied task planning with large language models," *arXiv preprint arXiv:2307.01848*, 2023.
[33] H. Jeong, H. Lee, C. Kim, and S. Shin, "A survey of robot intelligence with large language models," *Applied Sciences*, vol. 14, no. 19, p. 8868, 2024.
[34] S. S. Kannan, V. L. Venkatesh, and B.-C. Min, "Smart-llm: Smart multi-agent robot task planning using large language models," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 12140–12147, IEEE, 2024.
[35] H. Zhang, W. Du, J. Shan, Q. Zhou, Y. Du, J. B. Tenenbaum, T. Shu, and C. Gan, "Building cooperative embodied agents modularly with large language models," *arXiv preprint arXiv:2307.02485*, 2023.
[36] Y. Zhu, J. Chen, X. Zhang, M. Guo, and Z. Li, "Dexter-llm: Dynamic and explainable coordination of multi-robot systems in unknown environments via large language models," *arXiv preprint arXiv:2508.14387*, 2025.
[37] R. Chai, Y. Guo, Z. Zuo, K. Chen, H.-S. Shin, and A. Tsourdos, "Cooperative motion planning and control for aerial-ground autonomous systems: Methods and applications," *Progress in Aerospace Sciences*, vol. 146, p. 101005, 2024.
[38] T. Yang, P. Feng, Q. Guo, J. Zhang, J. Ning, X. Wang, and Z. Mao, "Autohma-llm: Efficient task coordination and execution in heterogeneous multi-agent systems using hybrid large language models," *IEEE Transactions on Cognitive Communications and Networking*, 2025.
[39] K. Obata, T. Aoki, T. Horii, T. Taniguchi, and T. Nagai, "Lip-llm: Integrating linear programming and dependency graph with large language models for multi-robot task planning," *IEEE Robotics and Automation Letters*, 2024.
[40] X. Zhang, H. Qin, F. Wang, Y. Dong, and J. Li, "Lamma-p: Generalizable multi-agent long-horizon task allocation and planning with lm-driven pddl planner," *arXiv preprint arXiv:2409.20560*, 2024.
[41] Y. Rizk, M. Awad, and E. W. Tunstel, "Cooperative heterogeneous multi-robot systems: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–31, 2019.
[42] M. Fernandez-Cortizas, H. Bavle, D. Perez-Saura, J. L. Sanchez-Lopez, P. Campoy, and H. Voos, "Multi s-graphs: An efficient distributed semantic-relational collaborative slam," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 6004–6011, 2024.
[43] L. Vig and J. A. Adams, "Multi-robot coalition formation," *IEEE transactions on robotics*, vol. 22, no. 4, pp. 637–649, 2006.
[44] P. Kicki, P. Liu, D. Tateo, H. Bou-Ammar, K. Walas, P. Skrzypczyński, and J. Peters, "Fast kinodynamic planning on the constraint manifold with deep neural networks," *IEEE Transactions on Robotics*, vol. 40, pp. 277–297, 2023.
[45] C. P. Luke Fina, "A hybrid perspective on suboptimal mixed-integer quadratic programming," 2025.
[46] G. Zepko and O. M. Shir, "All-quadratic mixed-integer problems: A study on evolution strategies and mathematical programming," *Evolutionary Computation*, pp. 1–26, 2025.
[47] S. Farquhar, J. Kossen, L. Kuhn, and Y. Gal, "Detecting hallucinations in large language models using semantic entropy," *Nature*, vol. 630, no. 8017, pp. 625–630, 2024.
[48] M. Khanna*, Y. Mao*, H. Jiang, S. Haresh, B. Shacklett, D. Batra, A. Clegg, E. Undersander, A. X. Chang, and M. Savva, "Habitat Synthetic Scenes Dataset (HSSD-200): An Analysis of 3D Scene Scale and Realism Tradeoffs for ObjectGoal Navigation," *arXiv preprint*, 2023.
[49] J. Fu, N. Atanasov, U. Topcu, and G. J. Pappas, "Optimal temporal logic planning in probabilistic semantic maps," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3690–3697, IEEE, 2016.
[50] D. S. Grigorev, A. K. Kovalev, and A. I. Panov, "Verifyllm: Llm-based pre-execution task plan verification for robots," *arXiv preprint arXiv:2507.05118*, 2025.
[51] X. He, X. Li, G. Yang, S. Chang, and J. Zhou, "A network connectivity-aware reinforcement learning method for task exploration and allocation," *IEEE Transactions on Network and Service Management*, 2024.