2025 IEEE International Conference on Robotics and Automation (ICRA)
May 19-23, 2025. Atlanta, USA

Generalized Mission Planning for Heterogeneous Multi-Robot Teams via LLM-constructed Hierarchical Trees

Piyush Gupta[1*] David Isele[1] Enna Sachdeva[1] Pin-Hao Huang[1]
Behzad Dariush[1] Kwonjoon Lee[1] Sangjae Bae[1]

***Abstract*— We present a novel mission-planning strategy for heterogeneous multi-robot teams, taking into account the specific constraints and capabilities of each robot. Our approach employs hierarchical trees to systematically break down complex missions into manageable sub-tasks. We develop specialized APIs and tools, which are utilized by Large Language Models (LLMs) to efficiently construct these hierarchical trees. Once the hierarchical tree is generated, it is further decomposed to create optimized schedules for each robot, ensuring adherence to their individual constraints and capabilities. We demonstrate the effectiveness of our framework through detailed examples covering a wide range of missions, showcasing its flexibility and scalability.**

I. Introduction

In recent decades, there have been substantial research efforts on the development of cooperative multi-agent systems [1,2]. As we look toward the future, we anticipate a society where robots with diverse capabilities become integral to daily life, contributing to a pro-social and harmonious society [3,4]. These robots could assist in various ways, such as helping an elderly person navigate urban streets, performing search and rescue operations, helping in a medical emergency, or ensuring the safety of a child in dangerous situations such as a busy road (see Fig. 1). The heterogeneity of these robots allows them to collaborate and form dynamic teams [5,6], and successfully carry out intricate operations that are required to accomplish complex missions. To realize this potential, it is essential to design generalized mission-planning frameworks that can accommodate the diverse capabilities of heterogeneous multi-robot teams.

Mission and task planning involve breaking down complex objectives into smaller, manageable tasks, ensuring alignment with available resources and adherence to system constraints. In heterogeneous multi-robot teams, each robot, with its own distinct capabilities and limitations, must be strategically assigned tasks that contribute to the mission in a coordinated and efficient manner. While there are many robot- and task-specific planning approaches, ranging from dexterous manipulation [7,8] to trajectory planning for autonomous vehicles [9–11], the literature lacks generalized mission planning approaches for heterogeneous multi-robot teams. Moreover, existing mission planning approaches for heterogeneous multi-robot teams are often tailored to specific scenarios [12], which limits their flexibility and adaptability across a wide range of real-world applications.

A key limitation of conventional mission planning systems is their lack of general intelligence and common-sense reasoning, both of which are essential for managing the dynamic and unpredictable nature of real-world environments. Consequently, these systems often struggle to generalize across different mission types, limiting their effectiveness in diverse contexts. Recent advancements in Large Language Models (LLMs) such as GPT [13], Gemini [14], and LLaMA [15] have demonstrated emergent capabilities that suggest they might be useful in addressing the shortcomings of mission planning systems to reason over diverse and novel situations. Therefore, we hypothesize that incorporating LLMs into mission planning systems enables more robust and adaptable planning for heterogeneous multi-robot teams, significantly increasing their potential for real-world applications.

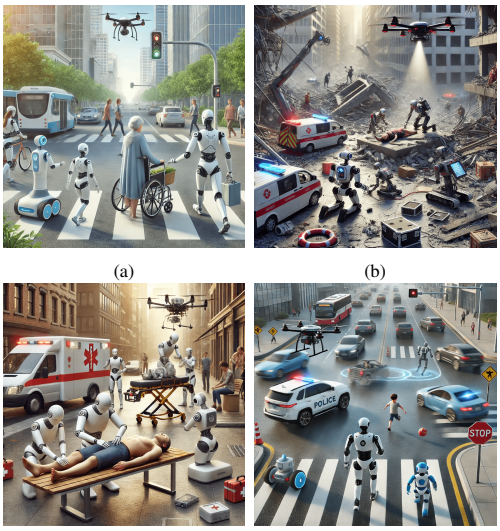

Fig. 1: Diverse scenarios involving heterogeneous multi-robot team missions. (a) Assisting an old lady cross the street safely, (b) Search and Rescue, (c) Medical emergency, (b) Ensuring safety of a child chasing a ball. These images were created with the assistance of DALL-E 3.

The mission in Multi-Robot Task Allocation (MRTA) problems can be categorized as elemental, simple, compound, or complex [16,17]. Among these categories, the most challenging tasks are complex missions with complex task dependencies (CD), where each task can be completed in multiple ways. Our focus is on complex missions of type CD[ST-MR-TA], which involve multiple robots (MR), where each robot performs a single task (ST) at a time. Task allocation and scheduling take into account time-extended assignments (TA), meaning both current and future tasks are considered during planning.

In recent years, significant progress has been made in multi-robot coordination and task allocation, with researchers exploring various approaches such as probabilistic methods based on partially observable Markov decision pro-

[1] Honda Research Institute USA, San Jose, CA, 95134, USA.
[*] Corresponding author {piyush_gupta}@honda-ri.com

2025 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3315-4139-2/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICRA55743.2025.11128711

Authorized licensed use limited to: Consortium - Saudi Arabia SDL. Downloaded on January 30,2026 at 19:10:58 UTC from IEEE Xplore. Restrictions apply.

cesses [18–20], game-theoretic approaches [21,22], auction- and market-based strategies [23,24], and optimization-based techniques [25,26]. Despite their advantages, these methods do not scale and generalize to complex real-world class CD problems, which require task decomposition, allocation, and scheduling under intricate constraints. Moreover, limited studies that are designed for class CD problems, such as those utilizing Generalized Partial Global Planning (GPGP) [27] and hierarchical trees [28], rely heavily on human expertise for task decomposition and treat task allocation and scheduling as separate processes from task decomposition. In contrast, we leverage the common-sense reasoning and function-calling capabilities of LLMs to construct hierarchical trees, utilizing custom-designed subtree routines and APIs that seamlessly integrate robot information. Our approach offers a more cohesive and fully automated control-loop system that does not require any human intervention.

Our mission-planning strategy involves breaking down the complex mission into executable tasks through a hierarchical tree built using an LLM. The representation of multi-robot missions using hierarchical trees is inspired by the language Task Analysis, Environment Modeling and Simulation (TAEMS) [29–31]. Specifically, it utilizes task decomposition, where intricate tasks are incrementally decomposed into simpler tasks, down to the level of actionable tasks. The hierarchical tree structure offers a clear overview of the mission by outlining task dependencies and streamlining the overall mission design. Furthermore, its flexibility and expressiveness also enable the modeling of complex task relationships, making it highly adaptable across diverse domains. After constructing the hierarchical tree using an LLM, we apply a heuristic tree-search algorithm to decompose the tree into multiple MRTA alternatives, ensuring successful mission accomplishment.

This work makes two key contributions. First, we present a generalized mission-planning framework based on hierarchical tree construction, leveraging the common-sense reasoning and function-calling capabilities of LLMs. This approach allows the pipeline to handle diverse, complex missions that are challenging for traditional systems. Second, we introduce a heuristic algorithm to decompose the hierarchical tree into multiple MRTA alternatives, ensuring that missions are completed while adhering to system resources and constraints. Consequently, this decomposition facilitates the formation of multi-robot teams, with tasks efficiently assigned to each robot.

II. Background and Formulation

We present a mission-planning pipeline for a team of heterogeneous robots with diverse constraints and capabilities. This work focuses exclusively on high-level mission planning, assuming the availability of scene understanding [32] and low-level motion planning [33,34] for task execution. The mission-planning module receives relevant scene information from the scene-understanding component, interprets the high-level mission objective, formulates a plan, and assigns tasks to the available robots accordingly. Identifying the overall mission objective from scene understanding can be facilitated by an LLM. In this work, we assume the availability of such a mission objective. Additionally, it is

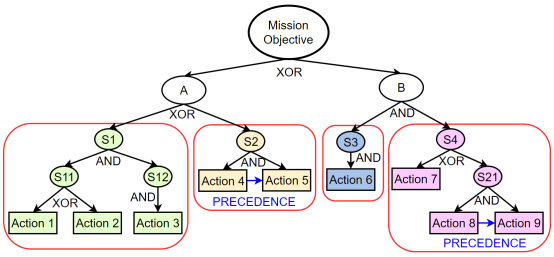

Fig. 2: An example hierarchical tree. Various subtrees of different color are highlighted in red enclosures. The round nodes represent abstract non-primitive tasks and the rectangular nodes represent executable primitive tasks. Children nodes are connected to their parent node with a logical constraint. The blue arrow represents precedence relationship between tasks.

assumed that the low-level planner can successfully execute the task assignments for each robot.

Let $\mathcal{R} = \{r_1, \ldots, r_m\}$ be the set of robots. Each robot $i \in \mathcal{R}$ can be of different types and can have various capabilities and limitations denoted by the sets $\mathcal{C}_i$ and $\mathcal{L}_i$, respectively. For example, a transportation robot may have the capabilities defined as [reach location, follow agent, carry load] and limitations defined as [max speed = 2 m/s, max load capacity = 4 Kg]. Each robot i can perform a set of actions $\mathcal{A}_i$ based on its capabilities $\mathcal{C}_i$. For instance, a robot with the "follow" capability may have an action set that includes actions such as `get_follow_path`, `follow_path`, and `send_location_to_server`. These actions represent various executable sub-routines that can be performed by the robot, and will be referred as primitive actions throughout the paper. The system may have redundancy, and hence, in general $\mathcal{A}_i \cap \mathcal{A}_j \neq \emptyset$ for $i \neq j, i, j \in \mathcal{R}$.

We represent the mission using a hierarchical tree, where the objective is to incrementally decompose the mission objective until the level of primitive actions $a \in \mathcal{A}_i$ that can be executed by the robot $i \in \mathcal{R}$. This approach is inspired by TAEMS, a framework designed to represent large task hierarchies along with the complex relationships between tasks.

Fig. 2 provides an example of such a hierarchical tree. The root node represents the overall mission objective, which is progressively decomposed into simpler tasks, represented as nodes in the tree. These task nodes are classified into two types: primitive tasks (depicted as rectangular nodes) and non-primitive tasks (depicted as round nodes). Primitive tasks are the most basic actions a robot can directly execute based on its capabilities, while non-primitive tasks are abstract tasks that combine with primitive tasks to provide a meaningful structure for achieving the mission objective. At each level, child nodes are linked to their parent nodes via a logical constraint $\{AND, XOR\}$. For AND, all child tasks must be completed to satisfy the parent task, whereas for XOR, exactly one child task must be completed to fulfill the parent task. Additionally, temporal constraints, such as task precedence (illustrated by blue arrows in Fig. 2), may exist between tasks. Collectively, tasks across various levels of the tree contribute to achieving the root-level mission objective.

To evaluate each task, each primitive task $a \in \mathcal{A}_i$ is assigned a utility $u_a(i)$. In the TAEMS framework [30], this utility is often defined as a function of a triple $(q_a(i), d_a(i), c_a(i))$, where $q_a(i)$ represents the quality of the action, $d_a(i)$ represents the time duration of the action, and

Authorized licensed use limited to: Consortium - Saudi Arabia SDL. Downloaded on January 30,2026 at 19:10:58 UTC from IEEE Xplore. Restrictions apply.

$c_a(i)$ estimates the cost for the action (energy expenditure, financial cost, resource consumption etc.) performed by the robot i. These functions are designed by the system designer and can be estimated by the robot based on its current state. The utility of the action a taken by robot i can be computed as:

$$u_a(i) = \alpha q_a(i) - \beta d_a(i) - \gamma c_a(i), \tag{1}$$

where α, β, γ are the system hyper-parameters.

III. Mission Planning Framework

We now discuss the mission-planning framework consisting of hierarchical tree construction using LLMs and tree decomposition into MRTA alternatives.

A. Hierarchical Tree construction via LLM function calling

The goal of the mission planning is to map the mission objective into a set of multi-robot task assignments, consisting of primitive tasks that can be executed by the robots. This becomes particularly challenging in complex class CD problems, where the mission can be accomplished in multiple ways, and tasks may have intricate dependencies. To address this, we leverage the intelligence and common-sense reasoning of LLMs to construct hierarchical trees for a wide range of real-world complex missions.

However, while LLMs are strong in common-sense reasoning, their outputs can be unstructured and prone to hallucinations. This challenge makes it difficult for LLMs to construct a well-organized hierarchical tree with zero-shot prompting. To overcome this, we design various tools and APIs to assist the LLM, utilizing its function-calling capabilities to reliably generate the hierarchical tree.

In the hierarchical tree, all leaf nodes terminate at primitive tasks, which are defined based on the capabilities of the robots. Given a set of robot resources $\mathcal{R}$, we construct a predefined set of subtree routines corresponding to system capabilities, denoted by $\mathcal{C} = \mathcal{C}_1 \cup \ldots \mathcal{C}_m$. Each subtree routine is a function that creates a pre-defined subtree based on system resources, constraints, and other functional arguments, e.g., `FollowSubtree (agent, resources)` routine constructs a follow subtree (Fig. 3) to follow an agent while adhering to robot capabilities and constraints.

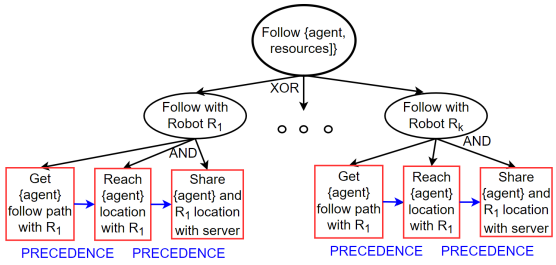

Fig. 3: Follow Subtree constructed by the `FollowSubtree (agent, resources)` routine. Robots that have follow capability in the resources are included in the subtree. The red rectangular nodes represent the primitive actions available to the robots with follow capabilities.

This approach offers multiple advantages. First, for each subtree, we can include a node for all (or a subset of) robots capable of performing a subtree task, thereby integrating the multi-robot team formation (robot selection) process directly into the hierarchical tree construction. Second, multiple subtrees can be merged together to create more complex subtree routines, such as `SearchAndFollowSubtree (agent, resources)`, to search and follow an agent. Lastly, since all subtrees adhere to the hierarchical structure and are common across a wide range of missions (as they are often context-independent), the LLM can leverage these predefined subtree routines to automatically generate the necessary subtrees and seamlessly integrate them into the overall hierarchical tree. Similar to primitive tasks, all hierarchical trees constructed via LLM should now terminate with a subtree attachment.

In addition to the subtree routines, we design a set of APIs such as `HierarchicalTreeConstructor`, `CreateAndAddSubtask`, `addMultiSubtasks`, `attachMultiSubTrees`, `plotTree`, `printTree`, and others to assist the LLM in constructing the hierarchical tree. The function definitions, along with details of their arguments, are provided to the LLM in JSON format. The LLM then leverages its function-calling ability to request these function calls with the appropriate arguments. The system executes the function calls requested by the LLM, ensuring the reliable construction of the hierarchical tree while reducing the LLM's overhead. Fig. 4 shows the overall mission planning pipeline.

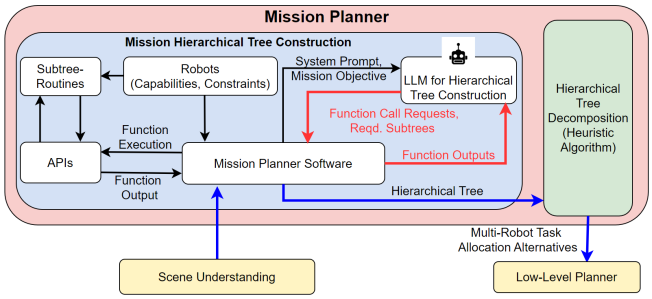

Fig. 4: Overall mission planning pipeline. LLM utilizes the subtree-routines and APIs to create a hierarchical tree which is decomposed to obtain MTRA alternatives.

B. Generating MRTA Alternatives

To decompose the hierarchical tree into several MRTA alternatives, we develop a heuristic algorithm as outlined in Algorithm 1. This recursive algorithm begins at the primitive nodes and progressively builds upwards, concluding at the root node. Let $alt(task) \subseteq \mathcal{A} = \mathcal{A}_1 \cup \ldots \mathcal{A}_m$ represent an unordered set of all actions that, when executed, lead to the completion of $task$.

The cardinality of the alt set is influenced by factors such as the mission tree, available resources, and constraints. For missions with abundant resources and minimal constraints, the MRTA alternative generation process can experience combinatorial explosion, resulting in factorial complexity. To address the potential combinatorial explosion, the heuristic algorithm prunes the minimum-utility results at each step to make the problem tractable. During the procedure, the algorithm computes the utility of the set $task_alt$ by summing the utility of each primitive task in the set, i.e., $u(task_alt) = \sum_{a(i) \in task_alt} u_a(i)$. It is important to note that, while the greedy approach in the pruning stage leads

Authorized licensed use limited to: Consortium - Saudi Arabia SDL. Downloaded on January 30,2026 at 19:10:58 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: MRTA Alternatives

Parameters: ρ - Maximum number of alternatives
Input: Hierarchical tree $\mathcal{T}$, available resources $\mathcal{R}$, task *task*
Output: Alternative decomposition of *task*, consumed resources
Function `get_alternatives(task)`:
  `/* recursion stopping criterion */`
  if $task \in \mathcal{A}$ **then**
    $resource_used \leftarrow$ `check_consumption`$(task, \mathcal{R})$
    return $[[[task]]]$, $[[resource_used]]$
  end
  `/* generate alternatives */`
  $sub_list, sub_resource \leftarrow []$
  for $subtask \in \mathcal{T}.subtasks(task)$ **do**
    $temp_subtask, temp_res \leftarrow$ `get_alternatives`$(subtask)$
    $sub_list \leftarrow sub_list \cup temp_subtask$
    $sub_resource \leftarrow sub_resource \cup temp_res$
  end
  $task_alt \leftarrow []$
  switch $\mathcal{T}.constraints(task)$ **do**
    case *AND* **do**
      $temp_alt \leftarrow$ `cartesian_product`(alt)
      $temp_resource \leftarrow$ `update_resource`$(sub_resource, temp_alt)$
      if *not* `resource_fail`$(temp_resource, \mathcal{R})$ **then**
        $task_alt \leftarrow temp_alt$
        $task_resource \leftarrow temp_resource$
      end
      else
        $task_alt \leftarrow [[]]$
        $task_resource \leftarrow []$
      end
    end
    case *XOR* **do**
      $task_alt \leftarrow [[alt^*]\ for\ alt^* \in alt]$
      $task_resource \leftarrow [[r^*]\ for\ r^* \in sub_resource]$
    end
  end
  `/* pruning */`
  if $|task_alt| > \rho$ **then**
    $utility \leftarrow$ `compute_utility`$(task_alt)$
    $top_ix \leftarrow$ `top_sorted_indices`$(utility, \rho)$
    $task_alt \leftarrow [task_alt(i), i \in top_ix]$
    $task_resource \leftarrow [task_resource(i), i \in top_ix]$
  end
  return $task_alt,\ task_resource$

to a computationally efficient algorithm, it can produce sub-optimal solutions, particularly in resource-constrained systems. However, in most real-world missions, computational efficiency is often preferred over achieving globally optimal allocations.

Once MRTA alternatives are obtained from Algorithm 1, we utilize a topological sort algorithm [35] to satisfy the precedence constraints in each robots task allocation.

IV. Results

We utilize the state-of-the-art GPT-4o-2024-08-06 as our chosen LLM for this work, due to its capability to generate structured outputs and reliably execute function calls. We employ a system message and chain-of-thought prompting [36,37] to obtain reliable hierarchical trees. In our experiments, we focus on mobility tasks and select a set of robots as outlined in Table I. Each robot has its set of capabilities, constraints, and primitive actions. For brevity, we omit the detailed descriptions. For all primitive actions, instead of defining utility for each action individually, we assign a uniform utility based on the robot's capability. Specifically, each primitive action of robot i in a subtree corresponding to a capability receives the same utility, as shown in Table I. We use a simple utility function $u_a(i) = q_a(i)$ and set $d_a(i), c_a(i) = 0$. However, the framework allows for complex utility functions based on the system designer's preferences. The following subtree routines:

(i) `Search (agent, resources)`
(ii) `Follow (agent, resources)`
(iii) `Reach (agent, resources)`
(iv) `SearchAndFollow (agent, resources)`
(v) `Transport (agent, location)`
(vi) `ReachAndTransport (agent, location)`

and following function APIs:

(i) `HierarchicalTree_init (objective)`
(ii) `CreateAndAddSubtask (parentString, isPrimitive, taskName, LogicalConstraint)`
(iii) `addMultiSubtasks (parentString, isPrimitiveList, taskNameList, LogicalConstraint)`
(iv) `attachMultiSubTrees (parentString, treeNames, treeArguments, LogicalConstraint, contraintsPairs)`
(v) `plotTree ()`
(vi) `printTree ()`

were provided to the LLM.

Robot Type	Num. of robots	Capabilities & Primitive Action Utilities
Mobile Scooter	1	Follow: 0.5, Reach: 1.0
Tele-Robot	2	Search: 1.0, Follow: 0.5, Reach: 1.0
Transportation Robot	1	Search: 0.3, Follow: 0.3, Reach: 0.3, Carry: 1.0
Social Robot	1	Follow: 0.3, Reach: 0.3, Message Display: 1.0

TABLE I: Resources

The following is an example of a hierarchical tree generation process by the LLM for the mission objective "Reunite mom with her lost child" (see also Fig. 5a).

System: You are an advanced multi-agent decision-making system responsible for generating a Hierarchical Tree for a given mission objective. The process begins by establishing the mission objective as the root node. From this root, you can add nodes or entire subtrees to build the structure. Each node represents a subtask derived from its parent task, and the children of each node are connected through logical constraints that define their relationships. The hierarchical structure must ensure that all paths terminate in a valid subtree, meaning that every leaf node must be a subtree itself. You have access to a variety of predefined subtrees, which you can select based on the task requirements with appropriate arguments. To assist in creating the optimal tree, you can utilize various functions available through tool calls. After each tool call, you will receive the output from the corresponding function, which you can use to refine your decision-making process and continue building the tree. Your goal is to construct a coherent

Authorized licensed use limited to: Consortium - Saudi Arabia SDL. Downloaded on January 30,2026 at 19:10:58 UTC from IEEE Xplore. Restrictions apply.

and logically consistent tree that effectively represents the mission's objectives and constraints. Before constructing the hierarchical tree by calling functions, think step by step how would you construct this tree. Think about the important agents, available subtree routines, and arguments you would use to create these subtrees. Remember that in the hierarchical tree, each branch must terminate with a subtree. Note that a hierarchical tree can contain multiple ways of solving the mission objective through the use of appropriate logical constraints.

User: Create a Hierarchical Tree with the mission - 'Reunite mom with her lost child'. Think how to use the functions and subTrees to create a logical Hierarchical tree.

---

assistant: To create a Hierarchical Tree for the mission objective "Reunite mom with her lost child", we will need to identify the key agents involved and utilize the available sub-trees in a manner that creates a coherent and optimal decision-making structure.
Key Agents:
1. **Mom** - We need to find the child and potentially transport the child to mom.
2. **Child** - The child is lost, and our primary focus is to reunite them with the mom.
Available Subtrees:
1. **SearchTree(agent)** - To search for a specific agent.
2. **FollowTree(agent)** - To follow a specific agent.
3. **ReachTree(agent)** - To reach a specific agent.
4. **SearchAndFollowTree(agent)** - To search for and then follow an agent.
5. **TransportTree(agent, destination)** - To transport an agent to a destination.
6. **ReachAndTransportTree(agent, destination)** - To reach an agent and then transport them to a destination.
Hierarchical Tree Construction:
We can approach this problem by breaking it into logical subtasks:
1. **Locate the Lost Child** - We first need to find the child.
2. **Transport the Lost Child to Mom** - Once the child is found, we need to transport the child back to mom.
Steps:
1. **Locate the Lost Child**: - **SearchTree** for the child.
2. **Reunite the Child with Mom**: - Once the child is located, use the **ReachAndTransportTree** to reach the child and then transport them to the mom.
Now, let's proceed to use the available functions to create this hierarchical tree. We'll start by initializing the tree and then attaching the respective subtasks and subtrees.

---

assistant: ['function': 'name': 'HierarchicalTree_init', 'arguments': {"objective": "Reunite mom with her lost child"}, 'function': 'name': 'addMultiSubtasks', 'arguments': '{"parentString": "Reunite mom with her lost child", "isPrimitiveList": [false, false], "taskNameList": ["Locate the Lost Child", Reunite the child and mom], "LogicalConstraint": "AND", "constraintPairs: " [["Locate the Lost Child", Reunite the child and mom, Precedence]] }'

---

function: (HierarchicalTree_init): HierarchicalTree constructor was successfully called.
function: (addMultiSubtasks): was called.

---

assistant: ['function': 'name': 'attachMultiSubTrees', 'arguments': '{"parentString": "Locate the Lost Child", "treeNames": ["SearchTree"], "treeArguments": [["Child"]], "LogicalConstraint": "AND", "contraintsPairs": {}}' 'function': 'name': 'attachMultiSubTrees', 'arguments': '{"parentString": "Reunite the Child with Mom", "treeNames": ["ReachAndTransportTree"], "treeArguments": [["Child", "Mom"]], "LogicalConstraint": "AND", "contraintsPairs": {}}']

---

function: (attachMultiSubTrees): was called
function: (attachMultiSubTrees): was called

---

assistant: ['function': 'name': 'plotTree', 'arguments': '{}']

---

function: (plotTree): was called

---

assistant: The hierarchical tree for the mission "`Reunite mom with her lost child" has been constructed and plotted successfully.

For the mission - "Reunite mom with her lost child", the top two task assignments generated by running the heuristic MRTA algorithm are as follows:

A1:
- Tele-robot : [Get search path to find Child, Detect Child, Report detection of Child to server]
- Social robot : [Get Child location to reach, Get to Child location, Display message to Child to follow Social robot to Mom, Get to Mom location]

A2:
- Tele-robot : [Get search path to find Child, Detect Child, Report detection of Child to server]
- Transportation robot : [Get Child location to reach, Get to Child location, Carry Child, Get to Mom location]

The pipeline generates feasible MTRA alternatives to achieve the mission objective. It is important to note that, in practice, when integrated with a low-level planner, tasks allocated to each robot must be scheduled according to the inter-robot task precedence constraints. For instance, in alternative A1, the tasks assigned to the social robot should be scheduled only after the tele-robot has successfully completed its assigned tasks. Figs. 5b, 5c, and 5d show more successful hierarchical trees constructed by the LLM for the missions - "Help the woman walking with heavy luggage", "Save the city from the monster destroying it", and "Rescue cat trapped in a building on fire", respectively. For brevity, we omit the chain-of-thought reasoning and function calls made by the LLM. The mission planning pipeline creates successful MTRA alternatives corresponding to hierarchical trees shown in Figs. 5b and 5c. However, for the mission: "Rescue cat trapped in a building on fire", the top two task assignments generated by running the heuristic MRTA algorithm are as follows:

A1:
- Tele-robot 1 : [Get search path to find firefighter, Detect firefighter, Report detection of Firefighter to server]
- Tele-robot 2 : [Get search path to find cat, Detect cat, Report detection of cat to server]
- Transportation robot : [Get cat location to reach, Get to cat location, Carry cat, Get to safe area location]

A2:
- Tele-robot 1 : [Get search path to find firefighter, Detect firefighter, Report detection of Firefighter to server]
- Tele-robot 2 : [Get search path to find cat, Detect cat, Report detection of cat to server]
- Social robot : [Get cat location to reach, Get to cat

Authorized licensed use limited to: Consortium - Saudi Arabia SDL. Downloaded on January 30,2026 at 19:10:58 UTC from IEEE Xplore. Restrictions apply.

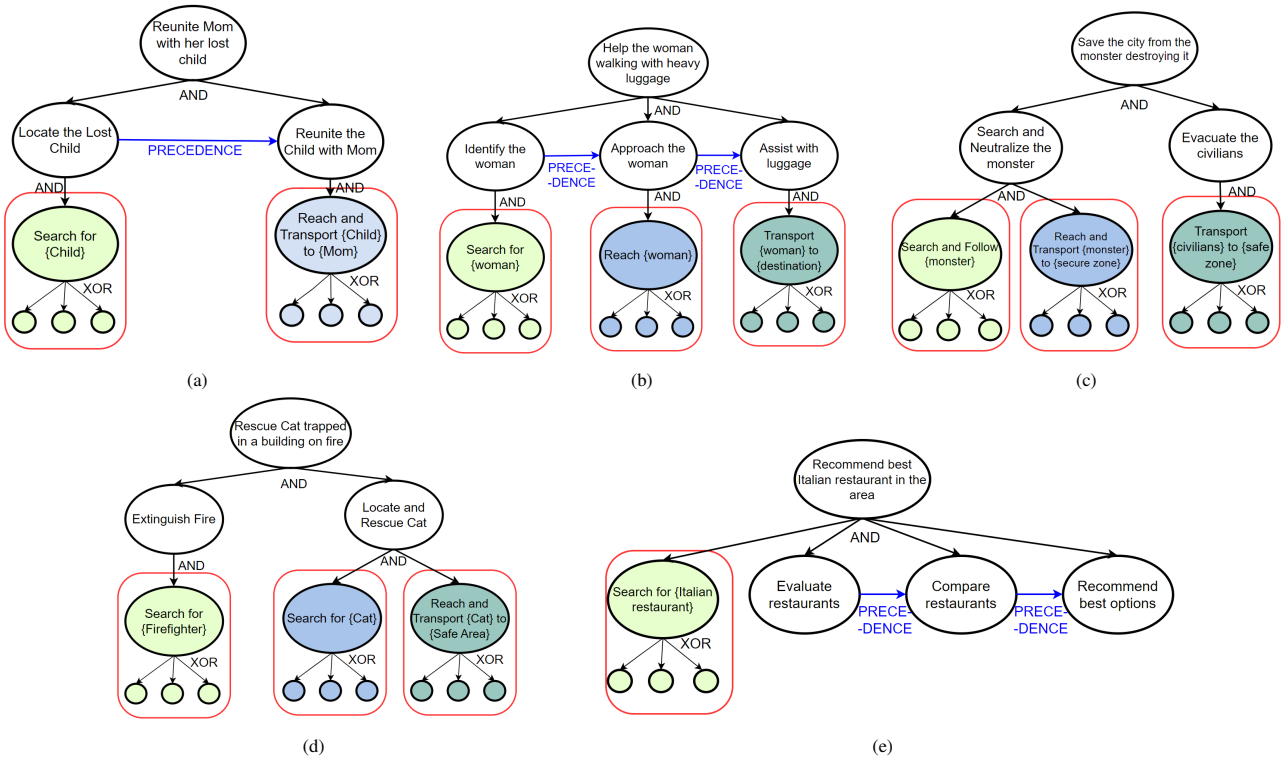

Fig. 5: Hierarchical trees generated by the LLM for different mission goals: (a) Reunite mom with her lost child, (b) Help the woman walking with heavy luggage, (c) Save the city from the monster destroying it, (d) Rescue cat trapped in a building on fire, (e) Recommend best Italian restaurants in the area. The details of the colored subtrees (highlighted with red boxes) used to create the hierarchical tree are omitted for brevity of space.

location, Display message to cat to follow social robot to safe area, Get to safe area location]

In this example, while the first alternative offers a good choice, the second alternative involves a social robot using the primitive action "Display message to cat to follow social robot to safe area," which may not be feasible in the real world. System designers should account for such edge cases when designing subtree routines. These cases can be avoided in several ways, such as by restricting the types of arguments allowed in subroutines or by selecting a different alternative altogether.

Lastly, Fig. 5e shows the incomplete hierarchical tree constructed by the LLM for the mission - "Recommend best Italian restaurants in the area". While the LLM can effectively decompose the mission, it fails to terminate the tree with subtrees that contain the robots' primitive actions. In such cases, where the mission falls outside the scope of the robots' available capabilities and predefined subtree routines, the LLM is unable to generate a complete hierarchical tree.

These examples demonstrate how our framework can leverage the same subtree routines and APIs to produce accurate mission plans for a diverse range of complex missions.

V. Limitations and Future Work

The system has a few limitations. In missions where available robot capabilities are insufficient, the LLM may hallucinate, leading to unrealistic solutions. Naturally, without adequate resources, not all problems are feasible. The performance of the system relies on the available subtree routines. Therefore, it is expected from the system designer to design a comprehensive set of subtree routines that align with the robot capabilities. Another limitation lies in the heuristic tree-search algorithm, which, due to its pruning process, may not always yield globally optimal solutions. Investigating alternative MRTA algorithms with provable optimality properties is an important avenue for future research. Additionally, the system's overall performance is closely tied to the reliability and efficiency of the scene-understanding module and low-level task execution.

In future work, we aim to conduct closed-loop simulations that integrate the scene-understanding and low-level planner modules. Additionally, we would like to fine-tune [38] the LLM for hierarchical tree construction task to obtain better quality trees. Lastly, Developing re-planning strategies for mission failures will also be a key focus.

VI. Conclusions

In this paper, we have introduced a mission-planning framework specifically designed for heterogeneous multi-robot teams. The framework utilizes hierarchical trees to represent and systematically decompose complex missions into manageable tasks. We developed various APIs and tools to automate the construction of these hierarchical trees, leveraging the function-calling capabilities of large language models (LLMs). We present a heuristic algorithm that provides multiple alternative multi-robot task assignments to accomplish the mission objective. Our results demonstrate that the framework effectively generates optimized schedules for the robots, ensuring that each task aligns with the unique constraints and capabilities of eachindividual robot.

Authorized licensed use limited to: Consortium - Saudi Arabia SDL. Downloaded on January 30,2026 at 19:10:58 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] A. Torreno, E. Onaindia, A. Komenda, and M. Štolba, "Cooperative multi-agent planning: A survey," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1–32, 2017.

[2] A. Mahajan, M. Samvelyan, T. Gupta, B. Ellis, M. Sun, T. Rocktäschel, and S. Whiteson, "Generalization in cooperative multi-agent systems," *arXiv preprint arXiv:2202.00104*, 2022.

[3] S. Šabanović, "Robots in society, society in robots: Mutual shaping of society and technology as a framework for social robot design," *International Journal of Social Robotics*, vol. 2, no. 4, pp. 439–450, 2010.

[4] A. Paiva, F. Santos, and F. Santos, "Engineering pro-sociality with autonomous agents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[5] P. Gupta, S. D. Bopardikar, and V. Srivastava, "Incentivizing collaboration in heterogeneous teams via common-pool resource games," *IEEE Transactions on Automatic Control*, vol. 68, no. 3, pp. 1902–1909, 2022.

[6] P. Gupta, *Optimal & Game Theoretic Feedback Design for Efficient Human Performance in Human-Supervised Autonomy*. Michigan State University, 2023.

[7] R. Jäkel, S. R. Schmidt-Rohr, S. W. Rühl, A. Kasper, Z. Xue, and R. Dillmann, "Learning of planning models for dexterous manipulation based on human demonstrations," *International Journal of Social Robotics*, vol. 4, pp. 437–448, 2012.

[8] N. Daoud, J.-P. Gazeau, S. Zeghloul, and M. Arsicault, "A real-time strategy for dexterous manipulation: Fingertips motion planning, force sensing and grasp stability," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 377–386, 2012.

[9] P. Gupta, D. Isele, D. Lee, and S. Bae, "Interaction-aware trajectory planning for autonomous vehicles with analytic integration of neural networks into model predictive control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7794–7800.

[10] P. Gupta, D. Coleman, and J. E. Siegel, "Towards Physically Adversarial Intelligent Networks (PAINs) for safer self-driving," *IEEE Control Systems Letters*, vol. 7, pp. 1063–1068, 2022.

[11] P. Gupta, D. Isele, and S. Bae, "Towards scalable & efficient interaction-aware planning in autonomous vehicles using knowledge distillation," in *2024 IEEE Intelligent Vehicles Symposium (IV)*, 2024, pp. 2735–2742.

[12] M. Roberts, T. Apker, B. Johnson, B. Auslander, B. Wellman, and D. W. Aha, "Coordinating robot teams for disaster relief," in *The Twenty-Eighth International Flairs Conference*, 2015.

[13] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[14] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth *et al.*, "Gemini: A family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.

[15] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[16] 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.

[17] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Cooperative Robots and Sensor Networks*, pp. 31–51, 2015.

[18] P. Morere, R. Marchant, and F. Ramos, "Sequential bayesian optimization as a pomdp for environment monitoring with uavs," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 6381–6388.

[19] P. Gupta and V. Srivastava, "Optimal fidelity selection for improved performance in human-in-the-loop queues for underwater search," *arXiv preprint arXiv:2311.06381*, 2023.

[20] B. Floriano, G. A. Borges, and H. Ferreira, "Planning for decentralized formation flight of uav fleets in uncertain environments with dec-pomdp," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2019, pp. 563–568.

[21] J. Bredin, R. T. Maheswaran, C. İmer, T. Başar, D. Kotz, and D. Rus, "A game-theoretic formulation of multi-agent resource allocation," in *Proceedings of the fourth international conference on autonomous agents*, 2000, pp. 349–356.

[22] P. Gupta, S. D. Bopardikar, and V. Srivastava, "Achieving efficient collaboration in decentralized heterogeneous teams using common-pool resource games," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 6924–6929.

[23] M. Otte, M. Kuhlman, and D. Sofge, "Multi-robot task allocation with auctions in harsh communication environments," in *2017 international symposium on multi-robot and multi-agent systems (MRS)*. IEEE, 2017, pp. 32–39.

[24] E. Nunes, M. McIntire, and M. Gini, "Decentralized multi-robot allocation of tasks with temporal and precedence constraints," *Advanced Robotics*, vol. 31, no. 22, pp. 1193–1207, 2017.

[25] L. Luo, N. Chakraborty, and K. Sycara, "Distributed algorithms for multirobot task assignment with task deadline constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 876–888, 2015.

[26] I. Sghir, I. B. Jaafar, and K. Ghédira, "A multi-agent based optimization method for combinatorial optimization problems," *International Journal on Artificial Intelligence Tools*, vol. 27, no. 05, p. 1850021, 2018.

[27] J. Motes, R. Sandström, H. Lee, S. Thomas, and N. M. Amato, "Multi-robot task and motion planning with subtask dependencies," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3338–3345, 2020.

[28] B. A. Ferreira, T. Petrović, and S. Bogdan, "Distributed mission planning of complex tasks for heterogeneous multi-robot systems," in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, 2022, pp. 1224–1231.

[29] K. Decker, "TAEMS: A framework for environment centered analysis & design of coordination mechanisms," *Foundations of Distributed Artificial Intelligence*, pp. 429–448, 1996.

[30] B. Horling, V. Lesser, R. Vincent, T. Wagner, A. Raja, S. Zhang, K. Decker, and A. Garvey, "The TAEMS white paper," 1999.

[31] J. Phelps and V. Guralnik, "Coordinators: TAEMS modeling and interfacing for first response." in *AAAI Spring Symposium: AI Technologies for Homeland Security*, 2005, pp. 1–8.

[32] F. Bremond, "Scene understanding: perception, multi-sensor fusion, spatio-temporal reasoning and activity recognition," Ph.D. dissertation, Université Nice Sophia Antipolis, 2007.

[33] V.-A. Le, B. Chalaki, V. Tadiparthi, H. N. Mahjoub, J. D'sa, and E. Moradi-Pari, "Social navigation in crowded environments with model predictive control and deep learning-based human trajectory prediction," *arXiv preprint arXiv:2309.16838*, 2023.

[34] V.-A. Le, V. Tadiparthi, B. Chalaki, H. N. Mahjoub, J. D'sa, E. Moradi-Pari, and A. A. Malikopoulos, "Multi-robot cooperative navigation in crowds: A game-theoretic learning-based model predictive control approach," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 4834–4840.

[35] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.

[36] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[37] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," *arXiv preprint arXiv:2203.11171*, 2022.

[38] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. Smith, "Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping," *arXiv preprint arXiv:2002.06305*, 2020.

Authorized licensed use limited to: Consortium - Saudi Arabia SDL. Downloaded on January 30,2026 at 19:10:58 UTC from IEEE Xplore. Restrictions apply.