Modeling and Optimizing the Provisioning of Exhaustible Capabilities for Simultaneous Task Allocation and Scheduling

Jinwoo Park
Georgia Institute of Technology
Atlanta, GA, USA
jinwoop@gatech.edu

Harish Ravichandar
Georgia Institute of Technology
Atlanta, GA, USA
harish.ravichandar@cc.gatech.edu

Seth Hutchinson
Northeastern University
Boston, MA, USA
s.hutchinson@northeastern.edu

ABSTRACT

Deploying heterogeneous robot teams to accomplish multiple tasks over extended time horizons presents significant computational challenges for task allocation and planning. In this paper, we present a comprehensive, time-extended, offline heterogeneous multi-robot task allocation framework, TRAITS, which we believe to be the first that can cope with the provisioning of exhaustible traits under battery and temporal constraints. Specifically, we introduce a nonlinear programming-based trait distribution module that can optimize the trait-provisioning rate of coalitions to yield feasible and time-efficient solutions. TRAITS provides a more accurate feasibility assessment and estimation of task execution times and makespan by leveraging trait-provisioning rates while optimizing battery consumption—an advantage that state-of-the-art frameworks lack. We evaluate TRAITS against two state-of-the-art frameworks, with results demonstrating its advantage in satisfying complex trait and battery requirements while remaining computationally tractable.

KEYWORDS

Heterogeneous Multi-Robot Systems; Task Allocation; Scheduling

ACM Reference Format:
Jinwoo Park, Harish Ravichandar, and Seth Hutchinson. 2026. Modeling and Optimizing the Provisioning of Exhaustible Capabilities for Simultaneous Task Allocation and Scheduling. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/NGQQ5993

1 INTRODUCTION

Given their ability to collectively execute tasks, Multi-Robot Systems (MRSs) enhance productivity and robustness [3], leading to their increasing deployment in logistics, agriculture, and manufacturing. Heterogeneous MRSs, composed of robots with different *traits* (capabilities), offer greater flexibility and efficiency in multi-agent coordination, particularly when tasks involve complex requirements that cannot be satisfied by homogeneous agents [38]. Such systems must simultaneously reason about the intertwined problems of task allocation (*who*), scheduling (*when*), and motion planning (*how*) while satisfying multidimensional task requirements [29]. However, identifying even a feasible plan for this problem is NP-hard due to its combinatorial nature [28].

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/NGQQ5993

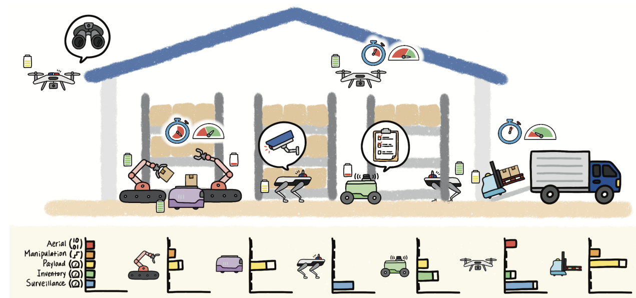

Figure 1: A motivating scenario: An autonomous warehouse with heterogeneous robots (top), characterized by traits that have provisioning and exhaustibility attributes (bottom), collaborating on complex tasks subject to temporal and battery constraints.

While prior works have proven effective in handling challenges associated with coordinating heterogeneous MRSs, they ignore crucial practical considerations that exacerbate computational challenges. See Figure 1 for an illustrative example. First, robot traits may be either *exhaustible* (e.g., deployable resources) or *inexhaustible* (e.g., sensing radius). Ignoring the exhaustible nature of traits can lead to overly optimistic solutions that fail to achieve adequate throughput. Second, tasks might require that exhaustible traits be *provisioned* at a particular rate (e.g., chemical dispensing). Ignoring provisioning can lead to suboptimal task performance or failures. Third, it is necessary to account for *energy* consumption (e.g., battery drain). Plans that ignore energy considerations risk being overly optimistic and violating important constraints such as battery capacity or fuel supply. Together, these factors considerably expand the search space, increase the computational burden, and complicate modeling and optimization.

In this work, we introduce ***TR**ait Throughput and **A**llocation **I**nformed **T**eam **S**cheduling* (TRAITS), a new modeling and algorithmic framework that extends the capabilities of existing heterogeneous Multi-Robot Task Allocation (MRTA) approaches to include the dynamics of exhaustible traits and their allocation within teams operating under battery constraints. Introducing trait-provisioning rates and battery optimization facilitates more accurate estimation of task durations and sustained robot operations. To this end, TRAITS simultaneously solves the trait-provisioning problem to generate a plan that satisfies multiple constraints.

In summary, we contribute

(i) a novel modeling framework that formalizes the provisioning problem for exhaustible traits, while maintaining compatibility with existing approaches to trait-based task allocation,

(ii) a comprehensive time-extended task allocation algorithm capable of handling traits with attributes of exhaustibility, provisioning, and varying forms of aggregation,
(iii) an offline MRTA framework that models battery usage by linking trait provision to current draw, incorporating Peukert's law to optimize robot utilization.

We evaluated the performance of TRAITS across various scenarios in simulated warehouse environments with different trait requirements. We compared TRAITS against two state-of-the-art heterogeneous MRTA frameworks that can only handle a subset of traits that TRAITS can manage and do not incorporate battery considerations. The results show that TRAITS can effectively handle a broader range of traits and generate battery-optimized, feasible plans with computational tractability. The source code is publicly available at https://github.com/GT-STAR-Lab/TRAITS.

2 RELATED WORK

The robot coordination problem is typically categorized into instantaneous allocation (IA) and time-extended allocation (TA) [14]. Numerous IA frameworks for heterogeneous task allocation have demonstrated high computational efficiency in coordinating robots under various compounding factors, such as energy [25, 32], distributed optimization [40], differing capabilities and resources [39, 44], making them well-suited for online computation and practical deployment. However, these approaches lack the ability to reason about task execution order and are incapable of producing viable solutions when the number of robots is fewer than the number of tasks. In contrast, many TA frameworks simultaneously address the combinatorial coordination problem while additionally accounting for uncertainties [13, 27, 29], tighter temporal constraints [34, 45] alongside the increased complexity of scheduling in a higher-dimensional search space. Nevertheless, these frameworks typically assume time-invariant resource availability.

MRTA architectures must cope with numerous criteria, such as competence, energy, resource, spatial, and temporal constraints. The market-based [8, 9, 20] and optimization-based [1, 15, 22, 35] approaches have effectively addressed task allocation problems. Furthermore, probabilistic [10], game-theoretical [7], and learning [2, 41] methods were proposed to tackle interdependent subproblems simultaneously. Despite their novelties and capability of finding feasible solutions, both resource and energy management were outside the scope of their research.

Resource management under energy constraints for MRS has been the focus of various research efforts. However, the methods that account for energy either involve instantaneous allocations without scheduling [12, 32], or abstract energy consumption in terms of time [5]. Moreover, the resource constraint problem addressed in [4, 6, 37] has either been demonstrated only with a small set of robots or does not account for energy constraints. A decentralized, auction-based approach [23] effectively managed resources and battery consumption, though it was limited to homogeneous robots. In contrast, the algorithm for heterogeneous coalitions [43] does not require all tasks to be completed.

Recent works [13, 18, 24, 29, 30, 34, 36] leverage the notion of *traits* to address coordination problems in heterogeneous multi-robot systems. However, these approaches have modeled only traits that are inexhaustible and instantaneously provisioned. In particular, studies [13, 18, 24, 29, 30, 34] consider only inexhaustible traits that are either non-provisioned or provisioned instantaneously. Although [13] and [16] formulate task allocation as a network-flow problem, task duration is still treated as constant. The concept of *resource-spending velocity*, or *resource burn rate* was introduced in [19] from a game-theoretic perspective, employing reinforcement learning to address decentralized, time-extended multi-robot foraging problem. However, all robots were assumed to be homogeneous and relied solely on fuel as their resource. The concept of a dynamic trait model, where trait values can change, was introduced in [31]; however, traits are ultimately transformed into constant values for computational purposes. The approach in [18] incorporates robot capabilities, collision avoidance, and energy constraints; however, it does not account for scheduling and trait exhaustibility. While this trait-based MRTA research has introduced novel approaches to address challenging spatio-temporal constraints, these studies still assume traits to be constant, without incorporating time-dependent rates. A shortcoming of assuming constant traits is that such solutions cannot capture the reduction in task duration that occurs when additional robots are added to a coalition.

3 TRAITS: MODELING FRAMEWORK

Our problem of interest is a variant of XD [ST-MR-TA]:SP and TW problems, defined by the established MRTA taxonomies [14, 21, 33]. We consider scenarios with K tasks and N robots. Each robot's capabilities are defined by a distinct trait composition, drawn from a total of U traits, where the set and magnitude of traits vary across robots. Let us define the index sets for *robots*, *tasks*, and *traits* as $\mathcal{N}, \mathcal{K}$, and $\mathcal{U}$, respectively.

Problem Statement: Given a set of robot traits, desired provisioning rates, battery capacities, and operational constraints, the objective is to determine a feasible plan that specifies how traits should be provisioned across tasks in a way that minimizes overall task execution time and satisfies task specifications while honoring battery constraints.

To address the above problem, we first introduce our comprehensive new modeling framework in this section. We begin with formal definitions of fundamental quantities to capture the notion of trait provisioning (Sec. 3.1), then discuss our new model and expanded taxonomy of traits (Sec. 3.2), followed by a discussion of how our model influences trait aggregation (Sec. 3.3) and scheduling (Sec. 3.4), and conclude the section with a model of how battery consumption can impact task allocation (Sec. 3.5).

3.1 Modeling Trait Provisioning

While past research has treated traits as irreducible quantities that can be instantly deployed [28, 35, 36], we reformulate them as *dynamic* quantities that need to be regulated over time and optimized based on task-specific constraints.

We denote the initial traits (e.g., resource, sensing radius) and the maximum trait-provisioning rates of the n-th robot by $Q_0^{(n)} \in \mathbb{R}_{\geq 0}^{1\times U}$ and $\dot{Q}_{\max}^{(n)} \in \mathbb{R}_{\geq 0}^{1\times U}$, respectively. The scalar entries $Q_{0,u}^{(n)}$ and $\dot{Q}_{\max,u}^{(n)}$ represent the initial amount and the maximum provisioning rate of the u-th trait, respectively.

Let the team be collectively responsible for completing K tasks. The desired trait matrix $\mathbf{Y}^* \in \mathbb{R}_{\geq 0}^{K \times U}$ and desired trait-provisioning rate matrix $\dot{\mathbf{Y}}^* \in \mathbb{R}_{\geq 0}^{K \times U}$ specify the minimum trait quantities and provisioning rates required to successfully execute all tasks, respectively. We use Y_{ku}^* and $\dot{Y}_{ku}^*$ to denote the required u-th trait and its provisioning rate for the k-th task, respectively.

The core of our problem is to determine *how each trait should be provisioned* by each robot, subject to the definitions above. Let $q_{ku}^{(n)}$ represent a decision variable denoting how much of n-th robot's u-th trait should be provisioned for the k-th task, yielding a decision vector for each robot:

$$q_k^{(n)} = \left[q_{k1}^{(n)}, q_{k2}^{(n)}, \ldots, q_{kU}^{(n)} \right] \in \mathbb{R}_{\geq 0}^{1 \times U} \quad \forall n \in \mathcal{N} \; \forall k \in \mathcal{K}.$$

$q_{ku}^{(n)} \in \mathbb{R}_{\geq 0}$ for continuous or discrete traits, and $q_{ku}^{(n)} \in \{0, 1\}$ for binary traits. If the robot lacks the trait, then $q_{ku}^{(n)} = 0$.

Similarly, we define a trait-provisioning rate vector $\dot{q}_k^{(n)} \in \mathbb{R}_{\geq 0}^{1 \times U}$ as a decision variable which represents the rate at which the n-th robot must provide its traits for the k-th task. The provisioning rate $\dot{q}_{ku}^{(n)}$ is positive only when the trait can be provisioned gradually; otherwise, it is zero.

3.2 A New Trait Taxonomy

Multi-robot tasks (MR) require multiple traits, where some traits support cooperative contribution from the coalition, while others demand a minimum level of proficiency from each individual robot. Additionally, the total contribution from any robot across tasks may be limited by its initial trait capacity. Despite their practical significance, no existing taxonomy or model captures these considerations [13, 30, 36].

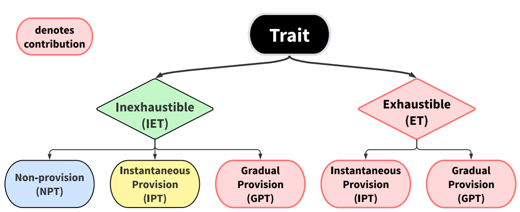

Figure 2: The proposed model captures various trait aspects and the framework handles all categories in the diagram.

Based on the definitions in Sec. 3.1, we propose a new taxonomy for traits (see Figure 2):

3.2.1 Exhaustible Traits. The exhaustibility attribute of a trait indicates whether the total provision of a robot's traits across multiple tasks is limited by its initial trait levels. An **inexhaustible trait (IET)** is not depleted through use and typically represents inherent properties or infinitely replicable entities. In contrast, an **exhaustible trait (ET)** is consumed with each task and generally corresponds to quantitatively measurable, finite resources carried by the robot. Examples of IETs include maximum payload capacity, monitoring capability, and sensor range, whereas examples of ETs include the volume of chemicals, battery level, and other consumable resources. Thus, the following constraint must hold:

$$Q_{0,u}^{(n)} \geq \begin{cases} \max\limits_{k \in A_\tau(n)} q_{ku}^{(n)} & \text{if } IET, \\ \sum\limits_{k \in A_\tau(n)} q_{ku}^{(n)} & \text{if } ET, \end{cases} \tag{1}$$

where $A_\tau(n)$ is the set of task indices allocated to the n-th robot.

3.2.2 Provisioning Traits. A provisioning attribute of a trait indicates whether it can be provisioned and, if so, defines the feasible provisioning rate. As such, we categorize traits into three groups:

(i) **Non-provisioning traits (NPTs)**: These traits represent constant or qualitative properties (e.g., robot type, regulation compliance) that cannot be provisioned.
(ii) **Instantaneous-provisioning traits (IPTs)**: These traits can be transferred instantaneously (i.e., $\dot{q}_{ku}^{(n)} = \infty$), such as occupying a seat, engaging an emergency stop, and transferring a token. They are generally discrete, single, or one-time-use in nature.
(iii) **Gradual-provisioning traits (GPTs)**: These traits require time to be transferred, i.e., $\dot{q}_{ku}^{(n)} \in \left(0, \dot{Q}_{\max,u}^{(n)}\right]$, and typically represent continuous processes such as gas refueling or water provisioning for fire suppression.

To make the problem tractable, we define the gradual provisioning rate $\dot{q}_{ku}^{(n)}$ as the constant rate at which the n-th robot provides the u-th trait to complete the k-th task during its execution. The desired trait-provisioning rate is set to zero $\left(\dot{Y}_{ku}^* = 0\right)$ for all NPTs and IPTs.

Whether a trait is continuous or discrete depends on the level of granularity in its definition—such as whether it is considered a distinct unit, a flow, or a bulk entity. Moreover, certain traits, such as the LiDAR scanning angle, exhibit an inexhaustible nature yet may impose temporal requirements, such as the scanning frequency. These can be modeled through gradual provisioning rates, indicating that trait provision is not necessarily constrained by exhaustibility.

We define $d_{ku}^{q(n)} \in \mathbb{R}_{\geq 0}$ to be the time required for the n-th robot to apply the u-th trait to the k-th task

$$d_{ku}^{q(n)} = \begin{cases} 0 & \text{if } NPT \text{ or } IPT, \\ q_{ku}^{(n)} / \dot{q}_{ku}^{(n)} & \text{if } GPT. \end{cases} \tag{2}$$

3.3 Trait Aggregation

To enhance adaptability across diverse task requirements, we define aggregation functions that determine how multiple robots can make partial contributions to a collective task, using decision variables that enable task-specific adaptation. For instance, certain tasks may necessitate the aggregation of traction force at the team level, whereas others may require ensuring a minimum traction capability at the individual level, highlighting the importance of task-specific aggregation strategies.

A *cumulative* attribute of a trait indicates whether a task's required characteristic must be individually satisfied by each robot within a coalition or can be collectively achieved through the combined output of the team. We define the former as a **noncumulative trait (NCT)**, which pertains to individual robot capabilities, and the latter as a **cumulative trait (CT)**, which pertains to team-level capabilities. Examples of NCTs include monitoring capability, emergency stop functionality, and the number of manipulator arms per robot. In contrast, examples of CTs include traction force and storage volume, though such traits can also be modeled as NCTs depending on the task requirements.

Hence, if the k-th task involves both cumulative and noncumulative traits, then for every u-th trait in the k-th task, we define the aggregation function

$$y_{ku} = \begin{cases} \min\limits_{n \in A_r(k)} q_{ku}^{(n)} & \text{if } NCT, \\ \sum\limits_{n \in A_r(k)} q_{ku}^{(n)} & \text{if } CT, \end{cases} \tag{3}$$

where $A_r(k)$ is the set of indices of robots allocated to the k-th task. The time required to fulfill the u-th trait of the k-th task by robot coalition $A_r(k)$ is given by $d_{ku}^{q} = \max\limits_{n \in A_r(k)} d_{ku}^{q(n)}$. Accordingly, the simultaneous trait-provisioning rate of the coalition for the u-th trait in the k-th task is defined as

$$\dot{y}_{ku} = \begin{cases} 0 & \text{if } NPT \text{ or } IPT, \\ y_{ku}/d_{ku}^{q} & \text{if } GPT. \end{cases} \tag{4}$$

Non-gradual, inexhaustible provisioning traits (NPT, IPT, and IET) have been used in prior works [28, 30, 34] to represent volume, water capacity, and maximum sensor range. *In contrast, TRAITS is, to the best of our knowledge, the only framework that models ET and GPT, enabling more accurate makespan estimation and the formation of meaningful coalitions under resource and temporal constraints.*

3.4 Task and Transition Durations

The duration of the k-th task is defined $d_k = d_k^s + d_k^q + \phi_{\mapsto}^k$, where d_k^s is the user-defined static task duration representing additional task duration unrelated to trait provisions, $d_k^q = \max\limits_{u \in \mathcal{U}} d_{ku}^q$ is the lower bound on the time required for all trait provisions of the k-th task, and $\phi_{\mapsto}^k$ is the intra-task transition duration from the initial position s_i^k to terminal position s_t^k of the k-th task. Furthermore, the initial transition duration for the k-th task, ϕ_0^k, represents the lower bound on the time required for all robots allocated to the k-th task to travel from their respective initial positions I_c to s_i^k. The inter-task transition duration ϕ_{ij} is the lower bound on the time required for all robots assigned to both the i-th and j-th tasks to travel from s_t^i to s_i^j.

3.5 Battery Consumption

Battery consideration is essential in identifying a feasible task allocation and schedule, as robots must have enough energy to execute their assigned tasks. A plan that considers only trait fulfillment, without accounting for energy constraints, may lead to unrealistic outcomes—such as robots being assigned an infinite number of tasks. Rather than modeling energy as an additional u-th trait, the proposed framework explicitly defines the relationship between trait values $(q, \dot{q})$, electrical current I [A], and battery consumption [J]. Inspired by [26], which models power linearly with sensor frequency and motor speed, we instead model current as proportional to trait values and provisioning rates, while capturing the nonlinear relationship between current and power. In our framework, battery current during trait delivery is modeled as a function proportional to the trait quantities and the rates at which they are provided.

The electrical current $I_{ku}^{(n)}$ drawn by the n-th robot to provide the u-th trait for the k-th task varies as a function of the trait value $q_{ku}^{(n)}$ and its provisioning rate $\dot{q}_{ku}^{(n)}$

$$I_{ku}^{(n)} = c_{q,u}^{(n)} \cdot q_{ku}^{(n)} + c_{\dot{q},u}^{(n)} \cdot \dot{q}_{ku}^{(n)} \in \mathbb{R}, \tag{5}$$

where $c_{q,u}^{(n)} \in \mathbb{R}_{\geq 0}$ and $c_{\dot{q},u}^{(n)} \in \mathbb{R}_{\geq 0}$ are independent coefficients representing the relationships between current draw and trait value, and between current draw and trait-provisioning rate, respectively. The coefficient values can be determined with domain expertise, guided by robot specification sheets (e.g., motor efficiency, torque constants, and sensor/actuator power characteristics). Either coefficient may be set to zero, and both may be zero when the corresponding effects are negligible. In this paper, the coefficients in the battery current model are derived from the battery's maximum deliverable current, as specified by its C-rating, and the maximum provisioning rates or magnitudes associated with each trait. This ensures that current estimates remain grounded in the physical constraints of the hardware. In our experiments, we compute these coefficients based on specifications of real-world robot platforms [11].

Battery current is drained even when the robot is idle ($I_{idle}^{(n)}$) as well as when it is moving while executing a task. The speed of this movement is referred to as the intra-task transition speed of the coalition for the k-th task, $v_{\mapsto}^k$ that must hold

$$v_{\mapsto}^k \leq \min_{n \in A_r(k)} v_k^{(n)} \leq \min_{n \in A_r(k)} v_{\max}^{(n)}. \tag{6}$$

The speed of n-th robot while executing the k-th task is denoted by $v_k^{(n)} \in \left[0, v_{\max}^{(n)}\right]$, and $c_v^{(n)}$ represents the relationship between battery current and robot speed. Therefore, the average current drawn by the n-th robot to accomplish the k-th task can be modeled as

$$I_k^{(n)} = I_{idle}^{(n)} + c_v^{(n)} \cdot v_{\mapsto}^k + \sum_{u \in \mathcal{U}} I_{ku}^{(n)}, \tag{7}$$

where it must hold that $I_k^{(n)} \leq I_{\max}^{(n)}$ as the current must not exceed the maximum current that the battery can supply based on the C-rating, $I_{\max}^{(n)}$.

The Peukert exponent $p^{(n)}$ of the n-th robot is a power coefficient that characterizes how battery drain accelerates as electrical current increases. This current increase is driven by higher trait values, greater provisioning rates, and the additional energy required for faster transitions. Hence, the total battery consumption $b^{(n)}$ [J] by the n-th robot must satisfy

$$B_0^{(n)} \geq b^{(n)} = b_{\leftrightarrows}^{(n)} + \sum_{k \in A_\tau(n)} V^{(n)} \cdot \left(I_k^{(n)}\right)^{p^{(n)}} \cdot d_k, \tag{8}$$

where $B_0^{(n)}$ is the robot's initial battery capacity, $b_{\leftrightarrows}^{(n)}$ is the energy consumed during inter-task transitions, $V^{(n)}$ is its operating voltage, and d_k is the duration of the k-th task.

4 TRAITS: OPTIMIZATION FRAMEWORK

The high-level overview of the TRAITS framework is shown in Figure 3 (refer Algorithm 1), consisting of four different layers. The *task allocation* layer incrementally adds robots to tasks and conveys allocation to trait distribution and scheduling layers. The *trait distribution* layer computes feasible trait provisions under battery

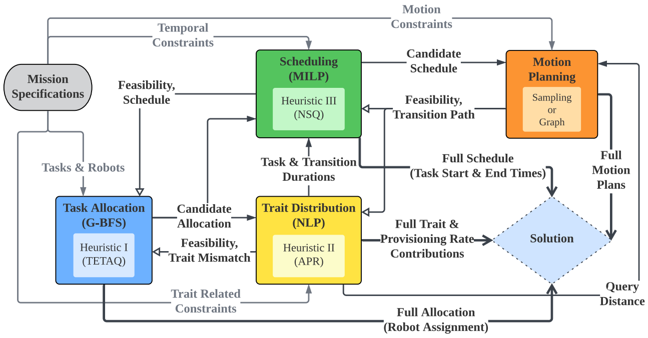

Figure 3: High-level architecture of the TRAITS framework.

constraints and communicates the estimated task durations, obtained from the nonlinear program (NLP), to the scheduling layer. The *scheduling* layer queries the *motion planning* layer for feasible paths and their corresponding transition durations, and computes an efficient makespan schedule using a mixed-integer linear program (MILP). This section provides an in-depth examination of each layer and its components. Collision avoidance is abstracted away in our framework and is assumed to be handled by a low-level control layer at deployment.

Algorithm 1: TRAITS

Input: $\mathcal{P}, T_{abs}, T_{rel}, \mathrm{Y}^*, \dot{\mathrm{Y}}^*, Q_0, \dot{Q}_{\max}, B_0, I_c, \alpha, \gamma$
Output: $\langle \mathrm{A}, q, \dot{q}, \sigma, \Pi \rangle$

```
pq ← PriorityQueue ({root})
while pq not empty do
    node ← pq.pop()
    if node.apr = 0 and node.nsq ≤ 1 then
        return ⟨A, q, q̇, σ, Π⟩
    for A ∈ generateSuccessor(node) do
        q, q̇, E, Ė, d, v⇆ ← traitDistributor (A)
        apr ← f_apr(E, Ė, γ)
        C, σ, Π ← scheduler (A, 𝒫, T_abs, T_rel, d, v⇆)
        nsq ← f_nsq(C)
        tetaq ← f_tetaq(α, apr, nsq)
        child ← ⟨tetaq, apr, nsq, A, q, q̇, σ, Π⟩
        pq.push(child)
return Null
```

4.1 Heuristic-based Task Allocation

The task allocation layer performs a greedy best-first search that incrementally assigns robots to tasks based on a heuristic evaluation. The search employs the following three heuristics. The Allocation Percentage Remaining (APR) is defined as

$$f_{apr}\left(E, \dot{E}\right) \triangleq \frac{\gamma \cdot E}{\sum_k \sum_u Y^*_{ku}} + \frac{(1-\gamma)\dot{E}}{\sum_k \sum_u \dot{Y}^*_{ku}} \in [0,1]\,, \tag{9}$$

where $\gamma \in (0, 1)$ is a user-defined hyperparameter that balances trait and trait-provisioning rate mismatches. The variables E and $\dot{E}$ represent the total trait and trait-provisioning rate mismatches, respectively, and are defined as

$$E = \sum_{k\in\mathcal{K}} \sum_{u\in\mathcal{U}} \underbrace{\max(0, Y^*_{ku} - y_{ku})}_{E_{ku}}, \tag{10}$$

$$\dot{E} = \sum_{k\in\mathcal{K}} \sum_{u\in GPT} \underbrace{\max(0, \dot{Y}^*_{ku} - \dot{y}_{ku})}_{\dot{E}_{ku}}. \tag{11}$$

A trait (provisioning rate) mismatch is defined as the non-negative shortfall between the required and provided trait (provisioning rate), with any surplus treated as zero mismatch.

The Normalized Schedule Quality (NSQ) is defined as

$$f_{nsq}(C) \triangleq \frac{C - C_{lb}}{C_{ub} - C_{lb}} \in [0,1]\,, \tag{12}$$

where C is the makespan of the given schedule, $C_{lb} = \sum_{k\in\mathcal{K}} d^s_k$, and $C_{ub} = \sum_{k\in\mathcal{K}} \left[d^s_k + \max_{u\in GPT} \left(\frac{Y^*_{ku}}{\dot{Y}^*_{ku}} \right) + \frac{2l_{\max}}{v_{\min}} \right]$ where $l_{\max}$ is the distance of the longest possible path in the map and $v_{\min}$ is the slowest robot's speed.

The Time-Extended Task Allocation Quality (TETAQ) is the heuristic with user-defined hyperparameter α, which is defined as

$$f_{tetaq} \triangleq \alpha f_{apr}\left(E, \dot{E}\right) + (1-\alpha) f_{nsq}\left(C\right) \in [0,1]\,. \tag{13}$$

The task allocation layer returns a solution when allocation satisfies $f_{apr} = 0$ and $C \leq C_{ub}$; otherwise, the search continues until a feasible solution is found or the user-defined timeout is reached.

4.2 Incorporating Battery Consumption

While many existing heterogeneous multi-robot task allocation approaches assume that robots always travel at their maximum speeds [24, 30], we relax this assumption by computing each robot's maximum transition speed based on its initial battery capacity. The distance function $l(s_i, s_t, n)$ returns the length of a traversable path from the initial position s_i to the terminal position s_t for robot n.

4.2.1 Consumption during task execution. This paper assumes the coalition moves at a uniform speed during the task execution. Accordingly, the intra-task transition speed $v^k_{\mapsto}$ for robots assigned to the k-th task is obtained by maximizing it subject to Constraint (6). We further assume that the path is constrained by the robot with the widest bounding radius, denoted as n^k_{wide}. Thus, the intra-task transition duration of the k-th task is defined as

$$\phi^k_{\mapsto} = \frac{l\left(s^k_i, s^k_t, n^k_{wide}\right)}{v^k_{\mapsto}}. \tag{14}$$

4.2.2 Consumption during task transitions. The two inter-task transitions of concern are: (i) from a robot's initial position to its first task ϕ^k_0, and (ii) between consecutive tasks ϕ_{ij}. The average inter-task transition speed $v^{(n)}_{\leftrightarrows} \in [0, v^{(n)}_{\max}]$ is the highest feasible speed under the battery constraint. Robots may use all remaining energy after task execution, as shown in Constraint (8), as long as their speed does not exceed their maximum limit $v^{(n)}_{\max}$. To this end, let $b^{(n)}_{\leftrightarrows}$ denote the total battery energy consumed by n-th robot during inter-task transitions over its schedule. Then

$$b^{(n)}_{\leftrightarrows} \geq V^{(n)} \cdot \left(I^{(n)}_{idle} + c^{(n)}_v \cdot v^{(n)}_{\leftrightarrows} \right)^{p^{(n)}} \cdot \phi^{(n)}_{\leftrightarrows}, \tag{15}$$

where $\phi_{\leftrightarrows}^{(n)} = \frac{l_{\leftrightarrows}^{(n)}}{v_{\leftrightarrows}^{(n)}}$ is the total inter-task transition duration of the n-th robot.

Nonetheless, determining the exact value of $\phi_{\leftrightarrows}^{(n)}$ is computationally intractable, as the precise total traveling distance $l_{\leftrightarrows}^{(n)}$ depends on task orderings—an NP-hard combinatorial problem. Therefore, TRAITS employs an overestimation of inter-task transition distance $\left(\text{i.e., } l_{\leftrightarrows}^{(n)} \leq \hat{l}_{\leftrightarrows}^{(n)}\right)$, where

$$\hat{l}_{\leftrightarrows}^{(n)} = \sum_{k \in A_{\tau}(n)} l\left(I_c^{(n)}, s_i^k, n\right) + l\left(s_t^k, I_c^{(n)}, n\right), \tag{16}$$

that is the sum of distances between the robot's initial position $I_c^{(n)}$ and its assigned tasks' start s_i^k and end s_t^k positions. Once $v_{\leftrightarrows}^{(n)}$ is obtained from Constraint (15), the initial transition duration for the k-th task is calculated as $\phi_0^k = \max_{n \in A_r(k)} \left\{ \frac{l\left(I_c^{(n)}, s_i^k, n\right)}{v_{\leftrightarrows}^{(n)}} \right\}$ while the inter-task transition duration between the i-th and j-th tasks is given by $\phi_{ij} = \max_{n \in A_r(i) \bigcap A_r(j)} \frac{l\left(s_t^i, s_i^j, n\right)}{v_{\leftrightarrows}^{(n)}}$, effectively establishing the lower bounds on the time required for robot transitions.

4.3 Temporal Constraints

The temporal relationship between tasks can be modeled using synchronization and precedence (SP) constraints, as well as time window (TW) constraints [33]. This framework incorporates three forms of temporal constraints: precedence $\mathcal{P}$, mutex $\mathcal{M}$, and deadlines T_{abs}, T_{rel}. A precedence constraint $\left(\tau_i \prec \tau_j\right) \in \mathcal{P}$ specifies that the i-th task must be completed before the j-th task begins. A mutex constraint $\left(\tau_i \leftrightarrow \tau_j\right) \in \mathcal{M}$, with $\mathcal{P} \cap \mathcal{M} = \emptyset$, ensures that the two tasks cannot be executed concurrently. An absolute deadline constraint $(p < t_{abs}) \in T_{abs}$ requires the task's start or finish time $p \in \mathbb{R}_{\geq 0}$ to fall within a specified bound. A relative deadline constraint $\left(p_i, p_j, t_{rel}\right) \in T_{rel}$ enforces that the time difference between two specified time points—p_i of i-th task and p_j of j-th task—must be less than t_{rel}, assuming $\tau_i \prec \tau_j$. The values of p_i and p_j can represent either the start or end time of their respective tasks.

4.4 Nonlinear Programming (NLP) Trait Distributor

A key contribution of our work is that, unlike prior frameworks relying on simple matrix operations, TRAITS formulates trait provisioning jointly per task and per robot, incorporating both provisioning rate and battery constraints into the allocation process. The problem of determining robot-to-task trait provisions is formulated as a nonlinear program with hierarchical objectives and constraints.

$$\text{minimize } f_{obj} \qquad \forall k \in \mathcal{K}, \forall n \in \mathcal{N}$$

$$\begin{aligned}
s.t.\ & Q_{0,u}^{(n)} \geq q_{ku}^{(n)} && \forall u && \text{(17a)}\\
& Q_{0,u}^{(n)} \geq \sum_{k \in A_{\tau}(n)} q_{ku}^{(n)} && \forall u \in ET && \text{(17b)}\\
& y_{ku} \leq q_{ku}^{(n)} && \forall u \in NCT && \text{(17c)}\\
& y_{ku} = \sum_{n \in A_r(k)} q_{ku}^{(n)} && \forall u \in CT && \text{(17d)}\\
& \dot{Q}_{\max,u}^{(n)} \geq \dot{q}_{ku}^{(n)} && \forall u && \text{(17e)}\\
& \dot{y}_{ku} \leq \dot{q}_{ku}^{(n)} && \forall u \in NCT && \text{(17f)}\\
& \dot{y}_{ku} = y_{ku} / d_{ku}^q && \forall u \in GPT && \text{(17g)}\\
& d_{ku}^{q(n)} = 0 && \forall u \notin GPT && \text{(17h)}\\
& d_{ku}^{q(n)} = q_{ku}^{(n)} / \dot{q}_{ku}^{(n)} && \forall u \in GPT && \text{(17i)}\\
& d_{ku}^q \geq d_{ku}^{q(n)} && \forall u && \text{(17j)}\\
& d_k = d_k^s + d_k^q + \phi_{\mapsto}^k && && \text{(17k)}\\
& b^{(n)} \leq B_0^{(n)} && && \text{(17l)}
\end{aligned}$$

where the hierarchical objective f_{obj} is to: ① minimize f_{apr} in Equation (9), ② minimize task duration d_k, and ③ maximize transitions velocities $v_{\leftrightarrows}^{(n)}$. This formulation facilitates the identification of feasible solutions that satisfy trait requirements while minimizing the makespan, subject to battery constraints. Objectives with higher priority (i.e., those indicated by smaller numbers inside circles) must not be compromised to improve lower-priority objectives.

4.4.1 Trait Provision. Constraints (17a)–(17d) define the trait provision requirements. Constraint (17a) ensures that a robot's trait provision does not exceed its individual capacity, while (17b) limits total use of exhaustible traits to the robot's initial levels. Constraints (17c) and (17d) enforce the coalition's trait provision for NCT and CT, respectively.

4.4.2 Trait-Provisioning Rate. Constraints (17e)–(17g) address trait-provisioning rates across assigned robots. Constraint (17e) enforces each robot's maximum rate limit, while (17f) ensures that the team-level capability for an NCT is determined by the least capable robot in the team. The coalition-level simultaneous provisioning rate model, consistent with Equation (4), is enforced by (17g).

4.4.3 Task Duration and Energy. Constraints (17h)–(17k) establish task durations based on trait provision and intra-transition durations, and (17l) ensures that energy consumption does not exceed the robot's initial battery capacity.

4.5 Mixed-Integer Linear Programming (MILP) Scheduler

Once the NLP-based trait distributor determines d_k and $v_{\leftrightarrows}^{(n)}$, the MILP scheduler computes ϕ_0^k and ϕ_{ij} as part of the scheduling process and searches for a schedule that minimizes the makespan C as follows:

$$\text{minimize } C$$

$$\begin{aligned}
s.t.\ & C \geq t_f^k && \forall k \in \mathcal{K} && \text{(18a)}\\
& t_f^k \geq t_s^k + d_k && \forall k \in \mathcal{K} && \text{(18b)}\\
& t_s^k \geq \phi_0^k && \forall k \in \mathcal{K} && \text{(18c)}\\
& t_s^j \geq t_f^i + \phi_{ij} && \forall (i,j) \in \mathcal{P} && \text{(18d)}\\
& t_s^j \geq t_f^i + \phi_{ij} - M(1 - \delta_{ij}) && \forall (i,j) \in \mathcal{M} && \text{(18e)}\\
& t_s^i \geq t_f^j + \phi_{ji} - M\delta_{ij} && \forall (i,j) \in \mathcal{M} && \text{(18f)}\\
& \delta_{ij} \in \{0,1\} && \forall (i,j) \in \mathcal{M} && \text{(18g)}\\
& p \leq t_{abs} && \forall\ (p, t_{abs}) \in T_{abs} && \text{(18h)}\\
& p_j - p_i \leq t_{rel} && \forall\ \left(p_i, p_j, t_{rel}\right) \in T_{rel} && \text{(18i)}
\end{aligned}$$

Table 1: Comparison of TRAITS, ITAGS, and CTAS across key attributes. While uncertainty is not the focus of this work, it is considered exclusively in CTAS.

Considerations	TRAITS (ours)	ITAGS [30]	CTAS [13]
Uncertainty control	✗	✗	✓
Motion planning	✓	✓	✗
Time-varying trait	✓	✗	✗
Trait-provisioning rate	✓	✗	✗
Variable task duration	✓	✗	✗
Battery depletion	✓	✗	✗
Deadline constraint	✓	✗	✗

Table 2: Performance comparison of TRAITS, ITAGS, and CTAS. Values are μ ($\pm\sigma$). Bold indicates the best performance, which improves in the direction of the arrow. Results are based on 300 experiments with $[5, 25]$ tasks and $[5, 25]$ robots.

	TRAITS (ours)	ITAGS [30]	CTAS [13]
Plan feasibility [%] (↑)	**100.0** (±0.0)	51.7 (±17.8)	63.6 (±16.4)
Task trait insufficiency [%] (↓)	**0.0** (±0.0)	11.5 (±17.9)	0.2 (±2.0)
Provisioning rate insufficiency [%] (↓)	**0.0** (±0.0)	41.6 (±14.3)	36.2 (±16.5)
Under-resourced robots [%] (↓)	**0.0** (±0.0)	26.7 (±13.7)	14.3 (±9.3)
C-rating violations [%] (↓)	**0.0** (±0.0)	47.5 (±14.9)	37.7 (±16.4)
Battery-capacity violations [%] (↓)	**0.0** (±0.0)	29.7 (±10.1)	23.9 (±11.1)
Deadline violations [%] (↓)	**0.0** (±0.0)	5.0 (±12.8)	22.3 (±18.9)
Computation time [s] (↓)	200.77(±253.26)	**31.12**(±27.75)	207.81(±258.32)

The makespan of the schedule is defined as the finish time of the last task (18a). Each task must finish at least d_k after it begins execution (18b). A task may commence only after the initial robot transitions have completed (18c). If a precedence constraint applies, it may commence only after the transitions from its predecessor tasks have also completed (18d). The Boolean indicator $\delta_{ij} = 1$ (18g) encodes a mutex constraint reduced to $\tau_i \prec \tau_j$, and $\delta_{ij} = 0$ otherwise. This introduces combinatorial complexity, making the scheduling problem an instance of an NP-hard class. Finally, (18h) and (18i) enforce absolute and relative deadline constraints, respectively.

4.6 Solution

A solution is considered feasible when the following three conditions are satisfied: (i) the trait mismatch $E = 0$, (ii) the trait-provisioning rate mismatch $\dot{E} = 0$, and (iii) the makespan $C \leq C_{ub}$. If these conditions are met, the solution is represented as the tuple $\langle \mathbf{A}, q, \dot{q}, \sigma, \Pi \rangle$, where: ① $\mathbf{A} \in \{0,1\}^{K\times N}$ is the task allocation matrix, where $A_{kn} = 1$ if the n-th robot is assigned to the k-th task, and 0 otherwise. ② q and $\dot{q}$ denote the optimized trait values and provisioning rates, respectively, for every trait of each robot assigned to each task. ③ σ is the task schedule, containing each task's start time t_s and finish time t_f. ④ Π contains the motion plans for all robots, including both inter- and intra-task transitions.

5 EVALUATION

We evaluate proposed framework across three different aspects. First, we compare TRAITS against two state-of-the-art trait-based MRTA frameworks, ITAGS [30] and CTAS [13] (see Table 1). Second, we conduct stress tests by varying the number of tasks K and robots N. Third, we analyze the impact of hyperparameters on search performance. The evaluation is based on 400 randomized experiments across various scenarios in a simulated warehouse environment with diverse trait requirements. In each trial, parameters are sampled as follows: $N \in [5, 30]$, $K \in [5, 40]$, and $p^{(n)} \in [1.01, 1.15]$. Additionally, $Q_{0,u}^{(n)}, \dot{Q}_{\max,u}^{(n)}, B_0^{(n)}$ are sampled $(0, 1]$ relative to each robot's respective maximum values. Each experiment includes at least one instance of precedence $\mathcal{P}$, absolute deadline T_{abs}, and relative deadline T_{rel} constraints.

All experiments were designed such that robots possess sufficient traits and battery to handle assigned tasks while forming meaningful coalitions. Each scenario includes both exhaustible and gradual-provisioning traits, and each task requires at least one cumulative or one non-cumulative trait. To observe the hyperparameter effects, the robots were diversified to include low trait-provisioning rate with large capacity, standard, and high trait-provisioning rate with limited capacity. Battery current coefficients were modeled using technical specifications from Clearpath Robotics' Husky [11]. It is often difficult to even establish the existence of a solution. To address this, we incorporated a necessary condition into the framework: the initial set of robot teams must collectively possess sufficient trait to cover the requirement of the tasks. If this condition is not satisfied at the outset, the problem is immediately declared infeasible. TRAITS framework leverages off-the-shelf libraries for motion planning [42] and optimizations [17]. All experiments were conducted on an AMD Ryzen 5950X CPU and 128 GB of RAM.

5.1 Comparison with Baselines

We evaluated TRAITS against two state-of-the-art trait-based MRTA frameworks CTAS [13] and ITAGS [30]. Specifically, we chose CTAS-O, a deterministic variant of the framework proposed in [13]. As shown in Table 2 and Figure 4, TRAITS outperforms both CTAS and ITAGS across multiple metrics, largely because they do not model trait exhaustibility, provisioning rates, battery constraints, and deadline constraints. Since CTAS and ITAGS lack trait-provisioning rates, their trait-provisioning durations are computed as $d_k^q = \max_{u\in GPT}\left(\frac{Y_{ku}^*}{\dot{Y}_{ku}^*}\right)$. Moreover, CTAS is an anytime algorithm, meaning its solution quality improves as computation time increases. To accommodate this, the experiment was structured so that CTAS returns a solution upon finding the optimal one, returns the best solution found at the TRAITS timeout if a feasible one exists, or continues running until a solution is found.

TRAITS incurs greater computation time due to its modeling of rate-dependent trait provision, deadlines, and battery-aware scheduling. ITAGS and CTAS achieve 51.7% and 63.6% plan feasibility, respectively, often generating solutions that TRAITS rejects due to constraint violations—such as trait and provisioning rate insufficiencies, under-resourced robots, C-rating violations, and battery-capacity violations—which are rigorously handled by our proposed TRAITS framework. Figure 4 corroborates that TRAITS consistently avoids overcommitment by ensuring that robots have sufficient traits to execute their assigned tasks, unlike ITAGS and CTAS, which tend to assign tasks beyond the robots' capabilities.

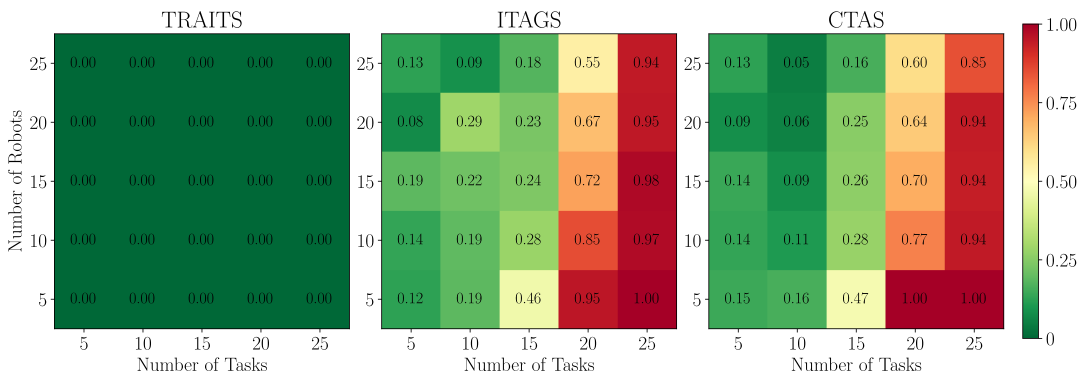

Figure 4: Fraction of under-resourced robots (robots assigned to tasks without sufficient traits). Lower values (green) indicate better performance. TRAITS consistently avoids assigning robots to tasks that exceed their trait capacities, whereas ITAGS and CTAS tend to assign insufficiently resourced robots as the task load grows for a fixed team size. This behavior partly stems from the assumption of irreducible traits in ITAGS and CTAS. Under-resourcing declines with increasing team size.

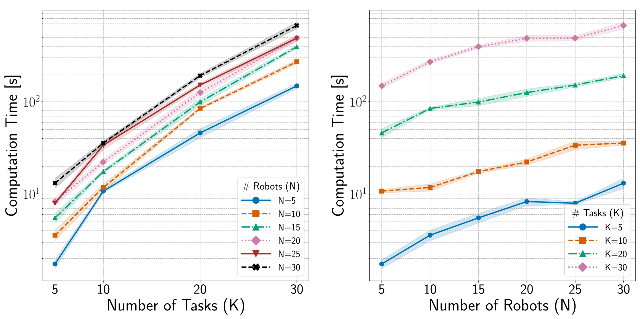

Figure 5: Computation time as a function of the number of tasks K (left) and the number of robots N (right). Solid dots indicate the mean, and the shaded regions represent the standard error computed from 120 experiments. The results highlight that computation time increases with both parameters, with task count having a more pronounced impact.

5.2 Scalability and Hyperparameter Sensitivity

Stress tests were conducted to analyze computation time as a function of the number of tasks K and robots N. As shown in Figure 5, the number of tasks has a significantly greater impact on computation time than the number of robots, with differences reaching up to an order of magnitude. In general, computation time increases as both K and N grow, primarily due to the expansion in the number of variables in the trait distribution and scheduling modules. Owing to its NP-hard combinatorial structure, computational effort grows rapidly with the instance size, making increased runtime unavoidable as the problem scales. Furthermore, when the task-to-robot ratio is high—meaning each robot is assigned to more than one task—computation time further increases. This is attributed to the scheduling layer, which requires more time to resolve mutex constraints under high task-load conditions.

In this experiment, we also observed that the hyperparameters α and γ both affect the makespan and computation times, but in distinct ways. α influences total task duration by favoring faster-provisioning robots as $\alpha \to 0$, which shortens task execution times. It also significantly impacts computation time: smaller α values broaden the search over robot coalitions, increasing runtimes, while larger values lead to greedier solutions with shorter computation times by prioritizing feasibility over optimality—a trade-off noted in prior work [30]. γ affects transition durations by favoring robots closer to the tasks as $\gamma \to 0$ and reduces trait-provisioning durations as γ increases. Although generally less impactful on computation time than α, extreme γ values can increase runtimes due to an imbalance in satisfying the trait mismatch E and the trait-provisioning rate mismatch $\dot{E}$. If one is satisfied significantly earlier than the other, additional robot assignments may be needed, increasing search effort.

6 CONCLUSION

This paper presents an offline heterogeneous time-extended multi-robot task allocation and scheduling framework that addresses the provision of exhaustible traits under battery constraints. We introduce the notions of trait provision and exhaustibility, which capture the temporal dynamics of traits and enable more expressive categorization of traits. Experimental results demonstrate our framework's enhanced comprehensiveness and superior performance compared to state-of-the-art baselines, while also analyzing trade-offs between search efficiency and solution quality as influenced by two hyperparameters. Despite these strengths, our framework assumes constant trait-provisioning rates and presumes that paths remain collision-free regardless of the number of robots—limitations we aim to address in future work. Overall, the proposed approach effectively manages a broader range of traits while maintaining computational tractability.

ACKNOWLEDGMENTS

This work was supported in part by the Army Research Lab under Grant W911NF-17-2-0181 (DCIST CRA). We gratefully acknowledge Lae Young Kim for visualizing the motivating scenario.

REFERENCES

[1] Ashay Aswale and Carlo Pinciroli. 2023. Heterogeneous Coalition Formation and Scheduling with Multi-Skilled Robots. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 5402–5409.

[2] Jacopo Banfi, Andrew Messing, Christopher Kroninger, Ethan Stump, Seth Hutchinson, and Nicholas Roy. 2022. Hierarchical planning for heterogeneous multi-robot routing problems via learned subteam performance. *IEEE Robotics and Automation Letters* 7, 2 (2022), 4464–4471.

[3] M. Bernardine Dias, M. Zinck, R. Zlot, and A. Stentz. 2004. Robust multirobot coordination in dynamic environments. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, Vol. 4. 3435–3442 Vol.4.

[4] Aurélie Beynier and Abdel-Illah Mouaddib. 2007. Decentralized Markov Decision Processes for Handling Temporal and Resource constraints in a Multiple Robot System. In *Distributed Autonomous Robotic Systems 6*, Rachid Alami, Raja Chatila, and Hajime Asama (Eds.). Springer Japan, Tokyo, 191–200.

[5] Álvaro Calvo and Jesús Capitán. 2024. Optimal Task Allocation for Heterogeneous Multi-robot Teams with Battery Constraints. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*. 7243–7249. https://doi.org/10.1109/ICRA57147.2024.10611147

[6] Jian Chen and Dong Sun. 2011. Resource constrained multirobot task allocation based on leader–follower coalition methodology. *The International Journal of Robotics Research* 30, 12 (2011), 1423–1434.

[7] Shengkang Chen, Tony X. Lin, and Fumin Zhang. 2023. Game-Theoretical Approach to Multi-Robot Task Allocation Using a Bio-Inspired Optimization Strategy. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 5388–5393. https://doi.org/10.1109/IROS55552.2023.10341947

[8] Han-Lim Choi, Luc Brunet, and Jonathan P. How. 2009. Consensus-Based Decentralized Auctions for Robust Task Allocation. *IEEE Transactions on Robotics* 25, 4 (2009), 912–926. https://doi.org/10.1109/TRO.2009.2022423

[9] Han-Lim Choi, Andrew K Whitten, and Jonathan P How. 2010. Decentralized task allocation for heterogeneous teams with cooperation constraints. In *Proceedings of the 2010 American Control Conference*. IEEE, 3057–3062.

[10] Shushman Choudhury, Jayesh K Gupta, Mykel J Kochenderfer, Dorsa Sadigh, and Jeannette Bohg. 2022. Dynamic multi-robot task allocation under uncertainty and temporal constraints. *Autonomous Robots* 46, 1 (2022), 231–247.

[11] Clearpath Robotics. 2025. *Husky UGV Spec Comparison*. https://clearpathrobotics.com/husky-spec-comparison/

[12] F. Dressler and G. Fuchs. 2005. Energy-aware operation and task allocation of autonomous robots. In *Proceedings of the Fifth International Workshop on Robot Motion and Control, 2005. RoMoCo '05*. 163–168. https://doi.org/10.1109/ROMOCO.2005.201418

[13] Bo Fu, William Smith, Denise M. Rizzo, Matthew Castanier, Maani Ghaffari, and Kira Barton. 2023. Robust Task Scheduling for Heterogeneous Robot Teams Under Capability Uncertainty. *IEEE Transactions on Robotics* 39, 2 (2023), 1087–1105. https://doi.org/10.1109/TRO.2022.3216068

[14] Brian P Gerkey and Maja J Matarić. 2004. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International journal of robotics research* 23, 9 (2004), 939–954.

[15] Matthew C. Gombolay, Ronald J. Wilcox, and Julie A. Shah. 2018. Fast Scheduling of Robot Teams Performing Tasks With Temporospatial Constraints. *IEEE Transactions on Robotics* 34, 1 (2018), 220–239.

[16] Walker Gosrich, Siddharth Mayya, Saaketh Narayan, Matthew Malencia, Saurav Agarwal, and Vijay Kumar. 2023. Multi-Robot Coordination and Cooperation with Task Precedence Relationships. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 5800–5806. https://doi.org/10.1109/ICRA48891.2023.10160998

[17] Gurobi Optimization, LLC. 2025. Gurobi Optimizer Reference Manual. https://www.gurobi.com

[18] Takumi Ito, Riku Funada, Mitsuji Sampei, and Gennaro Notomista. 2025. Energy-Aware Task Allocation for Teams of Multi-mode Robots. *arXiv preprint arXiv:2503.12787* (2025).

[19] Gal A. Kaminka, Dan Erusalimchik, and Sarit Kraus. 2010. Adaptive multi-robot coordination: A game-theoretic perspective. In *2010 IEEE International Conference on Robotics and Automation*. 328–334.

[20] Sven Koenig, Craig Tovey, Michail Lagoudakis, Vangelis Markakis, David Kempe, Pinar Keskinocak, Anton Kleywegt, Adam Meyerson, and Sonal Jain. 2006. The power of sequential single-item auctions for agent coordination. In *proceedings of the 21st national conference on Artificial intelligence-Volume 2*. 1625–1629.

[21] G Ayorkor Korsah, Anthony Stentz, and M Bernardine Dias. 2013. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research* 32, 12 (2013), 1495–1512.

[22] Kevin Leahy, Zachary Serlin, Cristian-Ioan Vasile, Andrew Schoer, Austin M. Jones, Roberto Tron, and Calin Belta. 2022. Scalable and Robust Algorithms for Task-Based Coordination From High-Level Specifications (ScRATCHeS). *IEEE Transactions on Robotics* 38, 4 (2022), 2516–2535.

[23] Dong-Hyun Lee. 2018. Resource-based task allocation for multi-robot systems. *Robotics and Autonomous Systems* 103 (2018), 151–161.

[24] Jiazhen Liu, Glen Neville, Jinwoo Park, Sonia Chernova, and Harish Ravichandar. 2026. Learning and Optimizing the Efficacy of Spatio-Temporal Task Allocation under Temporal and Resource Constraints. arXiv:2601.02505 [cs.RO] https://arxiv.org/abs/2601.02505

[25] Siddharth Mayya, Diego S. D'antonio, David Saldaña, and Vijay Kumar. 2021. Resilient Task Allocation in Heterogeneous Multi-Robot Systems. *IEEE Robotics and Automation Letters* 6, 2 (2021), 1327–1334.

[26] Yongguo Mei, Yung-Hsiang Lu, Y.C. Hu, and C.S.G. Lee. 2005. A case study of mobile robot's energy consumption and conservation techniques. In *ICAR '05. Proceedings., 12th International Conference on Advanced Robotics, 2005*. 492–497. https://doi.org/10.1109/ICAR.2005.1507454

[27] Andrew Messing, Jacopo Banfi, Martina Stadler, Ethan Stump, Harish Ravichandar, Nicholas Roy, and Seth Hutchinson. 2023. A Sampling-Based Approach for Heterogeneous Coalition Scheduling with Temporal Uncertainty. In *Robotics: Science and Systems*.

[28] Andrew Messing, Glen Neville, Sonia Chernova, Seth Hutchinson, and Harish Ravichandar. 2022. Grstaps: Graphically recursive simultaneous task allocation, planning, and scheduling. *The International Journal of Robotics Research* 41, 2 (2022), 232–256.

[29] Glen Neville, Sonia Chernova, and Harish Ravichandar. 2023. D-ITAGS: a dynamic interleaved approach to resilient task allocation, scheduling, and motion planning. *IEEE Robotics and Automation Letters* 8, 2 (2023), 1037–1044.

[30] Glen Neville, Andrew Messing, Harish Ravichandar, Seth Hutchinson, and Sonia Chernova. 2021. An Interleaved Approach to Trait-Based Task Allocation and Scheduling. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 1507–1514. https://doi.org/10.1109/IROS51168.2021.9636569

[31] Glen Neville, Harish Ravichandar, Kenneth Shaw, and Sonia Chernova. 2020. Approximated dynamic trait models for heterogeneous multi-robot teams. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 7978–7984.

[32] Gennaro Notomista, Siddharth Mayya, Yousef Emam, Christopher Kroninger, Addison Bohannon, Seth Hutchinson, and Magnus Egerstedt. 2022. A Resilient and Energy-Aware Task Allocation Framework for Heterogeneous Multirobot Systems. *IEEE Transactions on Robotics* 38, 1 (2022), 159–179.

[33] Ernesto Nunes, Marie Manner, Hakim Mitiche, and Maria Gini. 2017. A taxonomy for task allocation problems with temporal and ordering constraints. *Robotics and Autonomous Systems* 90 (2017), 55–70. https://doi.org/10.1016/j.robot.2016.10.008 Special Issue on New Research Frontiers for Intelligent Autonomous Systems.

[34] Jinwoo Park, Andrew Messing, Harish Ravichandar, and Seth Hutchinson. 2023. Risk-Tolerant Task Allocation and Scheduling in Heterogeneous Multi-Robot Teams. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 5372–5379. https://doi.org/10.1109/IROS55552.2023.10341837

[35] Amanda Prorok, M. Ani Hsieh, and Vijay Kumar. 2017. The Impact of Diversity on Optimal Control Policies for Heterogeneous Robot Swarms. *IEEE Transactions on Robotics* 33, 2 (2017), 346–358. https://doi.org/10.1109/TRO.2016.2631593

[36] Harish Ravichandar, Kenneth Shaw, and Sonia Chernova. 2020. STRATA: unified framework for task assignments in large teams of heterogeneous agents. *Autonomous Agents and Multi-Agent Systems* 34 (2020), 1–25.

[37] Philipp Schillinger, Mathias Bürger, and Dimos V Dimarogonas. 2017. Multi-objective search for optimal multi-robot planning with finite ltl specifications and resource constraints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 768–774.

[38] Onn Shehory and Sarit Kraus. 1998. Methods for task allocation via agent coalition formation. *Artificial intelligence* 101, 1-2 (1998), 165–200.

[39] Pedro M. Shiroma and Mario F. M. Campos. 2009. CoMutaR: A framework for multi-robot coordination and task allocation. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 4817–4824.

[40] Ola Shorinwa, Ravi N. Haksar, Patrick Washington, and Mac Schwager. 2023. Distributed Multirobot Task Assignment via Consensus ADMM. *IEEE Transactions on Robotics* 39, 3 (2023), 1781–1800. https://doi.org/10.1109/TRO.2022.3228132

[41] Anusha Srikanthan and Harish Ravichandar. 2022. Resource-Aware Adaptation of Heterogeneous Strategies for Coalition Formation. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems* (Virtual Event, New Zealand) *(AAMAS '22)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1732–1734.

[42] Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. 2012. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine* 19, 4 (December 2012), 72–82. https://doi.org/10.1109/MRA.2012.2205651 https://ompl.kavrakilab.org.

[43] Danfeng Wu, Guangping Zeng, Lingguo Meng, Weijian Zhou, and Linmin Li. 2017. Gini coefficient-based task allocation for multi-robot systems with limited energy resources. *IEEE/CAA Journal of Automatica Sinica* 5, 1 (2017), 155–168.

[44] Zirui Xu and Vasileios Tzoumas. 2022. Resource-Aware Distributed Submodular Maximization: A Paradigm for Multi-Robot Decision-Making. In *2022 IEEE 61st Conference on Decision and Control (CDC)*. 5959–5966.

[45] Lixuan Zhang, Jianzhuang Zhao, Edoardo Lamon, Yabin Wang, and Xiaopeng Hong. 2025. Energy Efficient Multi-Robot Task Allocation Constrained by Time Window and Precedence. *IEEE Transactions on Automation Science and Engineering* 22 (2025), 18162–18173.