Robust Task Scheduling for Heterogeneous Robot Teams Under Capability Uncertainty

Bo Fu, William Smith, Denise M. Rizzo, Matthew Castanier, Maani Ghaffari, and Kira Barton

Abstract—This article develops a stochastic programming framework for multiagent systems, where task decomposition, assignment, and scheduling problems are simultaneously optimized. The framework can be applied to heterogeneous mobile robot teams with distributed subtasks. Examples include pandemic robotic service coordination, explore and rescue, and delivery systems with heterogeneous vehicles. Owing to their inherent flexibility and robustness, multiagent systems are applied in a growing range of real-world problems that involve heterogeneous tasks and uncertain information. Most previous works assume one fixed way to decompose a task into roles that can later be assigned to the agents. This assumption is not valid for a complex task where the roles can vary and multiple decomposition structures exist. Meanwhile, it is unclear how uncertainties in task requirements and agent capabilities can be systematically quantified and optimized under a multiagent system setting. A representation for complex tasks is proposed: agent capabilities are represented as a vector of random distributions, and task requirements are verified by a generalizable binary function. The conditional value at risk is chosen as a metric in the objective function to generate robust plans. An efficient algorithm is described to solve the model, and the whole framework is evaluated in two different practical test cases: capture-the-flag and robotic service coordination during a pandemic (e.g., COVID-19). Results demonstrate that the framework is generalizable, is scalable up to 140 agents and 40 tasks for the example test cases, and provides low-cost plans that ensure a high probability of success.

Index Terms—Heterogeneous multiagent systems, pandemic robotic services, scheduling and coordination, stochastic vehicle routing problem, task allocation.

I. INTRODUCTION

TECHNOLOGICAL advances in sensing and control have enabled robotic applications in an ever-growing scope. On the one hand, the growing complexity and requirements of the applications soon exhaust the capability of a single robot: limited by design rules and actuator/sensor power, even the most competent robot is not able to handle all real-world tasks alone. This trend fosters the recent proliferation of multiagent system applications in agriculture [1], warehouse management [2], construction [3], defense [4], exploration [5], [6], [7], and surveillance [8], [9]. The advantages of replacing a large omnipotent robot with a team of smaller and less powerful robots include the robustness to agent failures, resilience of team configuration, and lower maintenance costs [10].

The reason that a team of less powerful robots can achieve the same or more complex tasks is that the functional heterogeneity within the team, i.e., distinct sensor and actuator capabilities of the robots, can complement each other during a task. In contrast to structural heterogeneity (e.g., maximum velocity or energy capacity), functional heterogeneity usually leads to fundamental differences between task capabilities (e.g., the ability to fly or transport people) [11]. The concepts of agent capabilities and task requirements are described in Section I-C.

The framework can be applied to heterogeneous mobile robot teams with distributed subtasks. Examples include pandemic robotic service coordination, explore and rescue, and delivery systems with heterogeneous vehicles. Owing to their inherent flexibility and robustness, multiagent systems are applied in a growing range of real-world problems that involve heterogeneous tasks and uncertain information. Most previous works assume one fixed way to decompose a task into roles that can later be assigned to the agents. This assumption is not valid for a complex task where the roles can vary and multiple decomposition structures exist. Meanwhile, it is unclear how uncertainties in task requirements and agent capabilities can be systematically quantified and optimized under a multiagent system setting. A representation for complex tasks is proposed: agent capabilities are represented as a vector of random distributions, and task requirements are verified by a generalizable binary function. The conditional value at risk is chosen as a metric in the objective function to generate robust plans. An efficient algorithm is described to solve the model, and the whole framework is evaluated in two different practical test cases: capture-the-flag and robotic service coordination during a pandemic (e.g., COVID-19). Results demonstrate that the framework is generalizable, is scalable up to 140 agents and 40 tasks for the example test cases, and provides low-cost plans that ensure a high probability of success.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TRO.2022.3216068.

Digital Object Identifier 10.1109/TRO.2022.3216068

1552-3098 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
framework to optimize capability-based robust task sections. is summarized in Fig. 2 and will be discussed in the following heterogeneous teaming problem (HTP) risk of noncompletion. We define this class of problems as the minimize energy and time costs for the task combined with the be robots/vehicles/humans) and complex tasks; the proposed practical scenarios. Consider a set of heterogeneous agents (can within a stochastic model, which can generalize to multiple task decomposition, assignment, and scheduling simultaneously tradeoff. In Section III, we will introduce how we encode this robustness in the optimization and determine a tradeoff is required. In Section III, we will introduce how we encode this robustness in the optimization and determine a tradeoff.

A. Contributions

In [14], we dealt with a deterministic variation of such a problem, where we assumed exact information (instead of distributions) of the agent capabilities and task requirements was known. This article develops a generalizable framework for task assignment and scheduling that systematically represents heterogeneous and uncertain task requirements and agent capabilities. This article provides the following contributions:

1) the development of a modeling framework that captures uncertainties within the task requirements and agent capabilities;
2) the derivation of a cost function that incorporates the concept of risk within the minimization;
3) the reformulation of the HTP to provide a more scalable algorithm that is solved by using a flow decomposition subproblem;
4) the implementation of a capture-the-flag game simulation to compare the task assignment performance to a baseline algorithm.

The problem size depends on the number of tasks, agent species, and agents per species. With the reformulation in this article (compared to [14]), the routing of individual agents is decoupled and postponed to a flow decomposition subproblem, which reduces the size of the optimization program for the HTP. The HTP only considers the behavior of an entire agent species, and its size no longer depends on the number of agents per species. The complexity of the framework is still exponential, as we use mixed-integer programming to find the exact optimal solution [15]. But the reformulation suppresses the growth such that the framework can solve problems with up to 140 agents and 40 tasks.

The open-source code is available at: https://brg.engin.umich.edu/publications/robust-task-scheduling/

B. Outline

The rest of this article is organized as follows. Section II briefly introduces the related work of multiagent task allocation and concludes with research gaps that this work investigates. Section III describes the general problem mathematically and
Fig. 2. System architecture. Input: cost (energy) maps (an example is in Section V-C), agent capabilities, and task requirements. Output: agent routes, schedules, and team formations. There are two major components. A routing, scheduling, and risk minimization model (see Section III) generates a network flow for each agent species. Then, these networks are further decomposed into individual agent plans through a flow decomposition model (see Section IV).

presents a risk minimization framework. Section III-E introduces an algorithm to solve the risk minimization problem and output a general routing plan described as network flows of agent movements. Section IV proposes a network flow decomposition problem to generate routing plans and schedules for individual agents. The two-step process in Sections III-E and IV improves the scalability of the framework compared to a solution method that outputs individual plans directly within one optimization. Section V discusses the experiments and results. Finally, Section VI concludes this article and provides ideas for future work based on current limitations.

C. Concepts and Terminology

This section provides definitions of concepts used throughout this article. For some of the terminology, we refer to [12], but simplify the description to provide only the core meaning.

1) *Agent capabilities*: A numeric vector that describes the agent’s fitness to conduct a task.

2) **Team capabilities**: A set of fitness values aggregated from agent capabilities in the team.

3) **Cumulative capabilities**: A capability that sums the agent capabilities across an entire team to get the team capability.

4) **Noncumulative capabilities**: a capability that does not sum. In this article, noncumulative team capability is the minimum of the agent capabilities in the team.\(^1\)

5) **Task requirement**: A set of constraints that relate the task and agents. Usually, it is specified as a number of required agents, agent actions, or agent capabilities that have to be satisfied. In this article, it is defined as a set of required team capabilities and will be described in Section III.

6) **Task decomposition**: Representing a task as a set of elemental tasks (there can be constraints between these subtasks), such that completing the subtasks completes the original task.

7) **Task assignment**: Identifying which specific agent should handle an elemental task.

8) **Elemental task**: A task that is not decomposable and should be assigned to only one agent.

9) **Compound task**: A task that can be decomposed into a set of elemental tasks in only one fixed way.

10) **Complex task**: A task that can be decomposed into elemental tasks in more than one way, i.e., following different decompositions, the resulting sets will contain different numbers or types of elemental tasks.

II. RELATED WORK

According to the taxonomy in [12], [16], [17], [18], [19], and [20], our framework for the HTP deals with a task allocation problem in the category CD-[ST-MR-TA]: complex task dependencies (CD), single-task robot (ST), multirobot tasks (MR), and time-extended assignment (TA). It is one of the most challenging task assignment categories and has been considered in few previous works in literature. The time-extended assignment considers scheduling, in contrast to instantaneous assignments (IA), which only make matches between tasks and robots.

While many previous works about multirobot tasks are in the category [ST-MR-TA], most of their papers do not deal with complex tasks. Their papers decompose a multirobot task into elemental tasks that can be assigned to a subset of single agents. If such a decomposition exists, the task decomposition problem is presolved and decoupled from the assignment and scheduling problem. Some works in vehicle routing [11], [21], job shop scheduling [22], [23], and robotic soccer games [24], [25] deal with such non-complex (or compound) tasks. We refer interested readers to the survey paper [12] for more work with compound tasks.

There are systems dealing with complex tasks, but their models do not generalize and are applied to one specific problem. Examples include fire extinguishing and debris removal [26], [27], and a coverage problem of unmanned aerial/ground vehicles with recharging behaviors [8], where tasks are considered complex because multiple decomposition combinations are possible in the problem space.

Some frameworks for CD-[ST-MR-TA] created in previous work generalize to a type of problem, but most of them have
scalability issues and are only applicable to a small group of robots. For instance, Bayesian network representations consisting of tasks, observations, constraints, and action nodes can be used to represent a task with interdependent elemental subtasks [28], [29]. Hierarchical tree networks are used in [30], [31], and [32] to represent tasks where leaf nodes are roles for assignment. Planning domain definition language (PDDL) can be used to represent a task as a graph of states (nodes) and actions (edges) [33], [34], [35]. Temporal planning can be considered an extension to the basic PDDL [36], [37]. In addition to a graph of predefined actions and state space, temporal planning allows continuous temporal constraints and interdependence between subtasks to be encoded as logical formulas. However, these representations based on actions and states are usually applied to systems with less than five agents. The number of states in their trees/graphs could explode rapidly with respect to the number of agents and available actions. The underlying solver (e.g., graph search algorithms) would take a long time to explore a subset of the high-dimensional space. Such a scalability issue limits its application to larger multiagent systems.

Uncertainty exists widely in the estimation of task requirements and agent capabilities. Some recent works capture such uncertainty explicitly in their models. However, the metrics they optimize do not necessarily result in a high-probability task requirement satisfaction (which is considered robust). For example, the work of [38] models the agent action uncertainty in Markov decision processes and optimizes the expected objective. In [24], [25], and [39], an agent’s capabilities are represented as a Gaussian random vector where the uncertainty is captured in the distribution. The work of [39] then penalizes the variance of the assigned agent capabilities to limit the uncertainty. However, expectation and variance are not well-justified quantitative metrics in such an optimization problem and can result in problematic solutions. For instance, if the required capability of a task is matched precisely, it is reasonable to limit the variances within a small threshold. However, if the team’s capability surpasses the task requirement by a lot, larger variances are acceptable. As another example, optimizing an expected cost (average performance) can be problematic for safety-critical applications. Rudolph et al. [40] directly maximize the probability that enough capabilities are assigned. However, probability is nonconvex in the decision variables, and global optima are hard to obtain for large problem cases.

Our proposed model considers a generalizable framework for allocating agent capabilities for complex tasks under capability and requirements uncertainty, and it can be applied to systems consisting of hundreds of agents. We represent the agent capabilities as a vector of random distributions (not necessarily Gaussian) and the task requirements within a function of the team capabilities (defined in Section III). The task requirement is verified once the aggregated capabilities of the team drive the binary function to one. We then solve the task decomposition, assignment, and scheduling problem simultaneously, where we optimize the time, energy, and robustness of task allocation. As mentioned in the Introduction, a robust allocation should ensure the capability of the team exceeds the task requirement with a high probability. We choose to minimize the conditional value at risk (CVaR) of (requirement − capability), which is consistent with maximizing the probability \( P(\text{capability} \geq \text{requirement}) \) [41]. The CVaR is a provably sensible measurement of the uncertainties in practical applications [42]. It is widely accepted by the finance community and appears with a growing frequency in recent robotic applications for risk-aware single-robot control [43], [44], [45] and multirobot coordination [13], [46]. The definition of CVaR is illustrated in Fig. 3.

Our problem can be considered a coalition formation with spatial and temporal constraints. Multiple models have been proposed [47], [48]. The work of [47] has a similar application to ours, but it assumes that there exists a predefined utility function that outputs a fitness value for a team configuration. While an abstract function generalizes the problem space, explicitly defining the function values for all team configurations could be hard in practice. We provide our task requirement and agent capability model to avoid such an explicit definition. The work of [49] and its extension [50] use a similar requirement and capability model. However, they do not consider the uncertainty in their capability model. A graph search-based algorithm is applied in their task allocation process, limiting the scalability of the number of agents.

STRATA [39], [51] shares many similarities with our proposed approach, CTAS, including stochastic agent capability vectors and task requirements on the team’s capabilities. Therefore, we choose STRATA as our baseline algorithm in one of the experiments in Section V. However, STRATA falls in the area of CD-[ST-MR-IA], where it assumes the tasks happen at the same time, which simplifies the scheduling problem. Meanwhile, our task model, represented using requirement functions, is more expressive. Table I provides a more detailed comparison between

\[ \text{TABLE I COMPARISON BETWEEN STRATA AND CTAS} \]

| Agent capabilities | continuous/Gaussian | continuous/stochastic |
|-------------------|---------------------|-----------------------|
| Task requirements | cumulative and/or cumulative | cumulative and/or cumulative |
| Scheduling        | no                  | yes                   |
| Uncertainty control| limit variance      | minimize CVaR         |
| Optimization type | continuous nonlinear | mixed-integer nonlinear |

Fig. 3. Graphical illustration of CVaR, defined as expected cost of the worst \( \beta \)-proportion of the cases. Denoted as \( \eta_\beta(\cdot) \), it is a function of the random distribution with \( \beta \) as a hyperparameter.

\(^2\)According to [41], maximizing the probability is equivalent to minimizing the value at risk, and the CVaR is a risk metric with additional math properties that facilitate optimization problems.
the two models. The meaning of “and/or” in the table is defined in Section III. Note that the definition of “noncumulative” in STRATA differs from ours. STRATA claims that it can deal with noncumulative capability types (such as the speed of an agent). However, it thresholds on a value and then treats the binary value after thresholding as a cumulative capability. For noncumulative types, we define the team’s capability to be the minimum of all agents and require instead that all agents in the team meet the minimum task requirement. STRATA is not able to enforce such requirements for their noncumulative capabilities.

III. ROUTING, SCHEDULING, AND RISK MINIMIZATION

In this section, we formally define the HTP with uncertain agent capabilities and task requirements. We describe the task requirement functions and agent capability vectors used to represent the task structure, organize them in a graphical model, and then encode the task allocation problem as a stochastic mixed-integer program (MIP) whose objective jointly consists of time, energy, and risk cost.

The modeling assumptions are listed as follows.

1) Tasks are distributed spatially and can be completed after a known time once a team with the required capabilities arrives.
2) The agent capabilities can be modeled as a vector of known continuous random distributions.
3) The task requirement can be verified by a binary function of the agent team’s capabilities. The parameters in the function are task-required capabilities and are known continuous random distributions.
4) Different types of capabilities are independent and do not affect each other.
5) The completion of one task does not depend on other tasks. But the scheduling dependency between tasks is considered, i.e., agents should arrive at the same time for a task, and the delay of one task could delay the subsequent tasks. The model can be extended with other time constraints, e.g., precedence constraints.

A. HTP Description

We simplify the graphical model in [14] for this work. Consider a set of agent species \( V = \{1, \ldots, n_v\} \), capability types \( A = \{1, \ldots, n_a\} \), and tasks \( M = \{1, \ldots, n_m\} \). Note that \( V, A, M \subset \mathbb{N} \) are three integer sets. Each agent species \( k \in V \) is associated with a nonnegative capability vector \( c_k = [c_k1, \ldots, c_kn_a]^T \), where \( c_{ka} \) is a random variable with a known distribution, representing the uncertain task capability of type \( a \in A \) for agent \( k \in V \). Each task \( i \in M \) requires an agent team with appropriate capabilities that drives a task requirement function \( \rho_i(\cdot) \) to 1.

A task requirement function is a binary function of a similar structure as (1). The logical operators \( \geq, \land, \lor \) are “greater than or equal to,” “and,” and “or” that return 1 if their conditions are satisfied, and 0 otherwise. \( \gamma_{i\alpha} \) describes the task’s requirement on capability \( a \) and is modeled as a random variable with a known distribution. We define \( \gamma_{i\alpha} = 0 \) if it does not appear in (1). Note that (1) is an example of a task that requires four types of capabilities. In practice, there can be an arbitrary number of \( \land \) and \( \lor \), theoretically

\[
\rho_1(\alpha_1, \alpha_2, \cdots) = [(\alpha_1 \geq \gamma_{11}) \lor (\alpha_2 \geq \gamma_{12})] \\
\land [\alpha_3 \geq \gamma_{13}] \land [\alpha_4 \geq \gamma_{14}].
\]

(1)

\[
\alpha_a = \left\{ \begin{array}{ll}
\sum_{k \in V} c_{ka} \cdot y_{k1}, & a \text{ is cumulative} \\
\min_{k \in V} c_{ka} \cdot r_{k1}, & a \text{ is noncumulative}
\end{array} \right.
\]

\( \forall a \in A. \)

(2)

\( \alpha_a \) is the capability \( a \) of the agent team. \( y_{k1} \) is the number of agents at task \( i \in M \) with species \( k \in V \). \( r_{k1} \) is binary and indicates whether there is an agent with species \( k \in V \) at task \( i \in M \). Depending on whether the capability is cumulative, we can compute \( \alpha_a \) according to (2). An example of noncumulative capabilities is the speed limit of a team. It equals the speed of the slowest moving agent in the team.

The requirement to drive these functions to 1 could be satisfied with appropriate team formations. This requirement can be encoded as linear constraints according to [14]. Note that the only part that could introduce nonconvexity is the logic \( \lor \), which takes the union of two feasible regions. Reducing \( \lor \) operators is desired as it facilitates the optimization.

The goal is to determine the optimal task schedule for a selected set of agents, such that the energy, time, and path constraints are satisfied, and the energy cost and risk of the task’s noncompletion are jointly minimized.

The task is considered complex for two reasons. First, the logic \( \lor \) operator results in multiple decompositions. Second, there is no fixed way to decompose the requirements in (1) into single-agent elemental tasks; instead, they are optimized together with the assignment and scheduling problem.

There are interschedule dependencies between tasks since the overall time cost for one agent to achieve one task depends on the schedule of other agents. For example, if one agent delays its work at another task and arrives late at the current task, the time cost increases for all agents at the current task.

B. Routing Model

As shown in Fig. 4, we first define a directed graph \( G = (N, E) \), with a set of vertices \( N = S \cup U \cup M \) and edges \( E = M \times \{1, \ldots, n_m\} \) is the set of task nodes. \( S = \{n_m + 1, \ldots, n_m + n_v\} \) and \( U = \{n_m + n_v + 1, \ldots, n_m + 2n_v\} \) are the sets of start and terminal nodes for each agent species \( k \in V \).

The prefix \( n_m \) and \( n_m + n_v \) are added such that the vertex indices in \( M, S, \) and \( U \) are unique integers. The size of the vertices set \( M, S, \) and \( U \) is \( n_m \), \( n_v \), and \( n_v \), respectively. Note
TABLE II
DEFINITION OF THE NOTATION

| Meaning |
|---------|
| $x_{ki,j}$ | The number of agents on edge $(i,j)$ with species $k \in V$, where node $i,j \in N$. For simplicity, $x_{k,i,j}$ and $x_{k,j,i}$ are interpreted as $x_{ki,j}$ and $x_{j,i,k}$ since there is no ambiguity. The colored numbers in Fig. 1 are solutions to $x_{ki,j}$ for that problem. |
| $y_{ki}$ | The number of agents at task $i \in M$ with species $k \in V$. |
| $r_{ki}$ | $r_{ki} = 1$, if $x_{ki} \geq 1$, otherwise 0. (A helper variable indicating whether there are agents with species $k$ on that edge.) |
| $r_k$ | $r_k = 1$, if $y_k \geq 1$, otherwise 0. (A helper variable indicating whether there are agents with species $k$ at that task.) |
| $q_i$ | The time that task $i \in M$ begins. |
| $g_{ki}$ | The maximum cumulative energy that an agent of species $k \in V$ has spent at node $i$ (A helper variable.) |
| $b_{ki}$ | The deterministic energy cost for agent $k \in V$ to travel edge $(i,j) \in E$. |
| $t_{ki}$ | The deterministic energy time for agent $k \in V$ to travel edge $(i,j) \in E$. |
| $t_k$ | The deterministic time for agent species $k \in V$ to complete its part for task $i \in M$. |
| $C_q$ | Time penalty coefficient. |
| $C_{large}$ | A large constant number for the MIP. |
| $B_k$ | The energy capacity of agent $k \in V$. |
| $B_{large}$ | A large constant energy for the MIP. |
| $h_i$ | The conditional value at risk from task $i \in M$. |
| $n_k$ | The number of agents with species $k \in V$. |

1) Variable Bounds:

$$x_{ki,j} \geq 0, \quad y_{ki} \geq 0 \quad \forall i,j \in N, \forall k \in V$$

$$r_{ki}, \quad r_k \in \{0,1\} \quad \forall i,j \in N, \forall k \in V$$

$$q_i \geq 0 \quad \forall i \in M \cup U. \quad q_i = 0 \quad \forall i \in S$$

$$g_{ki} \geq 0 \quad \forall i \in M \cup U. \quad g_{ki} = 0 \quad \forall i \in S, \forall k \in V. \quad (3)$$

The numbers of agents from species $k \in V$ on an edge and at a task node is positive and real. Their helper variables are binary. The time and cumulative energy are nonnegative at all nodes and zero at the start nodes.

2) Helper-Variable Constraints:

$$x_{ki,j} \geq r_{ki}, \quad x_{ki,j} \leq n_k \cdot r_{ki} \quad \forall i,j \in N, \forall k \in V$$

$$y_{ki} \geq r_k, \quad y_{ki} \leq n_k \cdot r_k \quad \forall i \in N, \forall k \in V. \quad (4)$$

Equation (4) encodes the relationship between $\{x_{ki,j}, y_{ki}\}$ and their helper variables $\{r_{ki}, r_k\}$.

3) Network Flow Constraints:

$$x_{km} = \sum_{j \in U \cup M} x_{kmj} \quad \forall m \in M, \forall k \in V \quad (5)$$

$$\sum_{i \in S \cup M} x_{ksi} \leq n_k \quad \forall k \in V \quad (6)$$

$$y_{ki} \leq \sum_{i \in M} x_{ksi} \quad \forall j \in M, \forall k \in V \quad (7)$$

$$y_{ki} = \sum_{i \in S \cup M} x_{ki,j} \quad \forall j \in M, \forall k \in V. \quad (8)$$

Equations (5)–(7) are flow constraints that ensure that the agent numbers are smaller than the upper bound, and that the incoming agent number at a node equals the outgoing number. Constraint (8) reflects the relationship that the agent number at a node equals the sum of the agent flows from all the incoming edges.

4) Energy Constraints:

$$g_{ki} - g_{kj} + b_{ki,j} \leq B_{large}(1 - r_{ki,j}) \quad \forall i,j \in N, \forall k \in V \quad (9)$$

$$g_{ki} \leq B_k \quad \forall i \in N, \forall k \in V. \quad (10)$$

Equation (9) ensures that $g_{ki}$ is the cumulative energy of the agent species—if an edge is traveled, the energy $b_{ki,j}$ is added. And because $g_{ki}$ is the maximum cumulative energy of agent species $k$ at node $i$, (10) ensures that the energy cost for an agent of species $k$ does not exceed its energy capacity $B_k$.

These energy constraints form sufficient conditions as they assure that the most costly path in the flow network of species $k$ is within the capacity limit $B_k$. However, it is possible that the most costly path is not picked during the flow decomposition procedure (see Section IV). This is a compromise made by us along the process of improving the scalability of the model: we replace the variables for individual agents with variables for an entire agent species. With the original variables, necessary and sufficient energy constraints were easily imposed.
5) Time Constraints:
\[ q_i - q_j + t_{kij} + t_{kti} \leq C_{\text{large}}(1 - r_{kij}) \quad \forall i, j \in N, \forall k \in V. \quad (11) \]

Equation (11) is a scheduling constraint: for an agent, the time duration between two consecutive tasks should be larger than the service time at the previous task plus the travel time. This constraint gives all the agents in the team enough time to arrive before the current task starts.

With the task time variables \( q_i \), the optimization problem can be extended with other user-defined time constraints. For instance, precedence constraints between tasks or time window constraints within which a task should complete [52].

6) Task Requirement Constraints:
\[
\begin{align*}
1 &= \rho_i \left( \sum_{k \in V} E(c_{k1})y_{k1}, \sum_{k \in V} E(c_{k2})y_{k2}, \ldots \right) \quad \forall i \in M \quad (12) \\
1 &= \rho_i \left( \min_{k \in V} E(c_{k1})r_{k1}, \min_{k \in V} E(c_{k2})r_{k2}, \ldots \right) \quad \forall i \in M. \quad (13)
\end{align*}
\]

For cumulative capabilities, we add constraint (12), and for noncumulative capabilities, we add constraint (13). \( E(\cdot) \) is the expectation operator. \( c_{k1} \) is the capability value defined in Section III-A. Note that an example of \( \rho_i(\cdot) \) is in (1), and \( \gamma_{ia}(\forall i \in M, a \in A) \) should be replaced with \( E(\gamma_{ia}) \) here. Though the task requirements and the agents’ task capabilities are stochastic, we can add this deterministic constraint that requires the teams’ expected capabilities to satisfy the task requirement by driving the requirement function to 1.

7) Objective Function:
\[
\begin{align*}
\min C_e \sum_{k \in V} \sum_{i,j \in N} b_{kij} \cdot x_{kij} + C_q \sum_{i \in U} q_i + C_h \sum_{i \in M} h_i \\
h_i &= \sum_{a \in A, \gamma_{ia} \neq 0} h_{ia} \quad \forall i \in M \quad (15) \\
h_{ia} &= \max_{k \in V \text{ s.t. } r_{ki} = 1} \eta_{\beta}(-c_{ka} + \gamma_{ia}) \quad a \in A, \forall i \in M \quad (16) \\
h_{ia} &= \eta_{\beta} \left( -\sum_{k \in V} c_{ka} \cdot y_{ki} + \gamma_{ia} \right) \quad a \in A, \forall i \in M. \quad (17)
\end{align*}
\]

In the objective function (14), we want to minimize a weighted combination of the energy cost, task time, and the CVaR of the task’s noncompletion, where \( C_e \), \( C_q \), and \( C_h \) are user-defined weights. For the risk of a task, \( h_i \) in (15), take task 1 and its example requirement function in (1) as an instance. According to (1) and (15), \( h_1 = h_{11} + h_{12} + h_{13} + h_{14} \). To explain this, task 1 requires capabilities 1–4. Assuming that the requirements on these capabilities are independently placed, the total risk is the sum of the risks from each requirement.

Consider \( h_{ia} \), the risk on a single capability \( a \in A \). According to the example requirement function (1), a task requests \( \alpha_a \geq \gamma_{ia} \). Both \( \alpha_a \) and \( \gamma_{ia} \) are stochastic. To maximize the probability \( P(\alpha_a \geq \gamma_{ia}) \), we instead minimize the CVaR of \( \gamma_{ia} - \alpha_a \). Let the function \( \eta_{\beta}(\cdot) \) be the CVaR of a random variable with probability level \( \beta \). Then, \( h_{ia} \) can be computed according to (16) or (17) depending on whether capability \( a \in A \) is cumulative or not.

The objective function (14), together with the deterministic task requirement constraint (12), tries to maximize the probability of task success at a low energy and time cost.

D. Sample Average Approximation (SAA)

Solving the optimization in (3)–(17) requires dealing with stochasticity and nonlinearity. This section shows how to convert this stochastic mixed-integer nonlinear program (MINLP) to a deterministic mixed-integer linear program (MILP).

Notice that \( E(c_{ka}) \) in (12) and \( \eta_{\beta}(-c_{ka} + \gamma_{ia}) \) in (16) do not involve decision variables; these can be computed prior to the optimization, given the distribution of \( c_{ka} \) and \( \gamma_{ia} \) \((k \in V, a \in A)\). The stochasticity is eliminated by the expectation and risk function. The deterministic task requirements in (12) can then be represented with a set of linear constraints according to [14]. Constraint (16) can also be represented as a linear constraint
\[
\begin{align*}
h_{ia} &\geq \eta_{\beta}(-c_{ka} + \gamma_{ia}) \cdot r_{ki} \quad \forall k \in V, a \in A, \forall i \in M. \quad (18)
\end{align*}
\]

The only nonlinearity is in (17) due to the function \( \eta_{\beta}(\cdot) \) and the decision variable \( y_{ki} \). However, we can linearize (17) using the SAA algorithm. Suppose that we can represent the random distribution \( c_{ka} \) and \( \gamma_{ia} \) by samples \( c_{k1}^{(\xi)} \) and \( \gamma_{ia}^{(\xi)} \) \((\xi = 1, \ldots, n_{\xi})\), respectively. Then, we can approximate (17) with a linear equation (19) according to [53]. The approximation converges at the rate of \( O(n_{\xi}^{-1/2}) \) [54]. In the following equation, \( \lambda_{ia} \) is a continuous helper variable for calculating \( h_{ia} \) \((i \in M \text{ and capability type } a \in A)\)
\[
\sum_{\xi=1}^{n_{\xi}} \left[ -\sum_{k \in V} c_{k1}^{(\xi)} \cdot y_{ki} + \gamma_{ia}^{(\xi)} - \lambda_{ia} \right]^+ \quad (19)
\]

Finally, the piecewise linear function in (19) can be represented as linear constraints in (21). Note that we add a series of continuous helper variables \( w_{ia}^{(\xi)} \) to represent the \([\cdot]^+\) function in (20)
\[
\begin{align*}
h_{ia} &= \lambda_{ia} + \frac{1}{n_{\xi}(1 - \beta)} \sum_{\xi=1}^{n_{\xi}} w_{ia}^{(\xi)} \\
w_{ia}^{(\xi)} &\geq -\sum_{k \in V} c_{k1}^{(\xi)} \cdot y_{ki} + \gamma_{ia}^{(\xi)} - \lambda_{ia} \\
w_{ia}^{(\xi)} &\geq 0. \quad (21)
\end{align*}
\]

E. L-Shaped Algorithm

In the previous section, we formulate the stochastic HTP as an MINLP optimization and approximate it as an MILP using SAA. However, the number of variables and constraints in the linear program is a function of the sample number \( n_{\xi} \). When
Algorithm 1 L-Shaped Algorithm for the Model.

1: Input: Parameters of the optimization problem (3)–(21)
2: for i ∈ M and a ∈ A do
3: \( L_{\text{ia}} = 0 \)
4: while True do
5: Solve the first stage problem and let \((y_1, \ldots, y_m, \lambda_ia, \theta_ia, x_{kij})\) be the solution.
6: flag = True
7: for i ∈ M and a ∈ A do
8: // \( L_{\text{ia}} \) is abbreviated to \( L \) in the following lines.
9: \( \lambda_{\text{ia}} = 1 : n_{\xi} \) do
10: Solve the second stage problem (24) and let \( \pi_{\text{ia}}(\xi) \) be the Lagrangian multiplier associated with the solution \( w_{\text{ia}}(\xi) \).
11: Calculate \( D_{\text{ia}}^{[\ell]} \) and \( d_{\text{ia}}^{[\ell]} \) according to (25).
12: if \( D_{\text{ia}}^{[\ell]} [y_{1i}, \ldots, y_{ni, a}, \lambda_{\text{ia}}]^{T} + \theta_{\text{ia}} \geq d_{\text{ia}}^{[\ell]} \) then
13: flag = False
14: end if
15: end for
16: if flag then
17: return the solution \( x_{kij}^{[\ell]} \) // Optimality obtained.
18: end if
19: \( p = p + 1 \) // The problem will contain more cuts (23).

the sample number dominates, the size of the linear program is roughly \( O(n^2 \xi) \), and the computation complexity is \( O(n^2 \xi^2) \).

We explore the sparsity of the problem and decouple it into a two-stage linear program using the L-shaped algorithm [55]. The algorithm returns the same optimal solution but reduces the computation cost from \( O(n^2 \xi^2) \) approximately to \( O(n \xi^2) \) with a larger constant coefficient empirically.

The structure of the algorithm is summarized in Algorithm 1. At the first stage, the algorithm solves a fixed-sized MILP (the size is not a function of the sample number \( n_{\xi} \)). At the second stage, the algorithm solves \( n_{\xi} \) small linear programs and then adds cuts to the first-stage problem according to the second-stage solutions. The two-stage process is iterated until convergence.

The first-stage problem in the algorithm is

\[
\min C_c \sum_{k \in V} \sum_{i,j \in N} b_{kij} \cdot x_{kij} + C_q \sum_{i \in U} q_i + C_h \sum_{i \in M} h_i \tag{14}
\]

s.t. (3)–(12) and

\[
h_i = \sum_{a \in A \text{ s.t. } \gamma_{ia} \neq 0} h_{ia}, \quad \forall i \in M, \tag{15}
\]

\[
h_{ia} \geq \eta_{\beta} \left( -c_{ka} + \gamma_{ia} \right) \cdot r_{ki}, \quad \forall k \in V, a \in A, \forall i \in M, \tag{18}
\]

\[
h_{ia} = \lambda_{ia} + \frac{1}{n_{\xi} (1 - \beta)} \theta_{ia}, \quad a \in A, \forall i \in M, \tag{22}
\]

\[
D_{ia}^{[\ell]} [y_{1i}, \ldots, y_{ni, a}, \lambda_{ia}]^{T} + \theta_{ia} \geq d_{ia}^{[\ell]}, \quad \forall \ell = 1, \ldots, L, \quad a \in A, \forall i \in M. \tag{23}
\]

If capability \( a \in A \) is noncumulative, \( h_{ia} \) is calculated according to (18). If \( a \in A \) is cumulative, the large linear program in (21) is replaced with (22) and (23). \( \theta_{ia} \) is a lower bound helper variable for \( \sum_{\ell=1}^{n_{\xi}} w_{ia}^{(\ell)} \) in (21) and is tightened iteratively with the cuts in (23) from the second stage.

In one iteration, as \( y_{ia} \) and \( \lambda_{ia} \) are determined in the first stage and fixed temporarily, the large program in (21) can be decoupled into \( n_{\xi} \) small linear programs with analytic solutions.

We call the second-stage problem:

\[
\min w_{ia}(\xi) \tag{24}
\]

s.t. \( w_{ia}(\xi) \geq - \sum_{k \in V} c_{ka} \cdot y_{ki} + \gamma_{ia} - \lambda_{ia} \cdot y_{ia} \)

\[
w_{ia}(\xi) \geq 0.
\]

Once the second-stage problem is solved during each iteration, an additional optimality cut can be added in (23) according to the Lagrangian multipliers (simplex multipliers) \( \pi_{ia}(\xi)[p] \) associated with the second-stage solutions. The parameters of the new optimality cut, \( D_{ia}^{[\ell]} \) and \( d_{ia}^{[\ell]} \), are calculated as follows.

Note that \( L \) is an integer label for the new cut

\[
D_{ia}^{[\ell]} = \sum_{\xi=1}^{n_{\xi}} \pi_{ia}(\xi)[p] \cdot c_{ia}, \ldots, c_{n_{\xi} \cdot a}, 1\] \( d_{ia}^{[\ell]} = \sum_{\xi=1}^{n_{\xi}} \pi_{ia}(\xi)[p] \cdot \gamma_{ia}. \) \tag{25}

The Lagrangian multipliers \( \pi_{ia}(\xi)[p] \) are obtained by solving the second-stage dual problem (26), and the solutions are in (27)

\[
\max \left( - \sum_{k \in V} c_{ka} \cdot y_{ki} + \gamma_{ia} - \lambda_{ia} \right) \cdot \pi_{ia}(\xi) \tag{26}
\]

s.t. 0 ≤ \( \pi_{ia}(\xi) \) ≤ 1

\[
\pi_{ia}(\xi)[p] = \begin{cases} 0, & - \sum_{k \in V} c_{ka} \cdot y_{ki} + \gamma_{ia} - \lambda_{ia} < 0 \\ 1, & \text{otherwise} \end{cases} \tag{27}
\]

IV. DECOMPOSE AGENT FLOWS INTO PATHS

Solving the teaming problem in Section III using the algorithm in Section III-E provides the optimal solution to \( x_{kij}, \forall k \in V, \forall i, j \in N \), i.e., the flows of different agent species. This section discusses an optimal algorithm to extract a routing plan for each individual agent from the agent flows.

The flow decomposition problem contains two steps: rounding and splitting. First, we round the agent network flow obtained from the MIP risk minimization model in the previous section. Second, we split the network flow into a set of agent routes to obtain a plan for each individual agent.

An example is shown in Fig. 5. The flow network of species 1 is extracted and rounded off in Fig. 5(b). Then, in Fig. 5(c), three individual agent paths with flows of 1 on the path edges are selected such that the sum of the flows for the three paths equals the rounded integer network flow in Fig. 5(b). Note that in order to maintain the task requirements, we have to round up a fractional flow rather than round it to the nearest integer.
Then, the function $f(\cdot)$ should satisfy the flow constraint
\[
\sum_{e \in E^m_{in}} f(e) = \sum_{e \in E^m_{out}} f(e) \quad \forall m \in M.
\]

In this section, the flow function $f_k(i, j) = x_{kij}$ indicates the number of agent species $k \in V$ on edge $(i, j) \in E$.

See Fig. 5(b) as an example. Suppose that the flow output from the MIP model in Section III for agent species $k \in V$ is $x_{kij}$ on edge $(i, j)$. Let the integer flow after the rounding process be $x'_{kij}$. Then, the linear program in below will return an integer flow network with minimum energy cost
\[
\begin{align*}
\min \quad & \sum_{i,j \in N} b_{kij} \cdot x'_{kij} \\
\text{s.t.} \quad & x'_{kij} \geq \lceil x_{kij} \rceil \quad \forall x_{kij} > 0 \\
& x'_{kij} = 0 \quad \forall x_{kij} = 0 \\
& \sum_{i \in S \cup M} x_{kim} = \sum_{j \in U \cup M} x_{kmj} \quad \forall m \in M.
\end{align*}
\]

The objective function (28) penalizes the energy cost. Equation (29) ensures that the rounding is happening upward, where $\lceil x_{kij} \rceil$ denotes the smallest integer larger than $x_{kij}$. The network flow constraint (30) should be maintained. $S$, $U$, and $M$ are the set of start, terminal, and task nodes, respectively.

Note that there is no explicit integer constraint to ensure that $x'_{kij}$ is an integer. However, if this linear program is solved using Simplex-based algorithms, the solutions to $x'_{kij}$ are guaranteed to be integers. The proof is given in Appendix B. Because of this, the minimum energy rounding problem could be solved in polynomial time through a linear program [instead of an integer linear program (ILP)].

B. Minimum Max-Energy Flow Split

After rounding, we obtain the integer flows $x'_{kij}$ in Fig. 5(b); the next step is to split this integer flow into individual agent paths in Fig. 5(c). By summing the outgoing flow at the start node, we are able to get the needed number of agents from species $k \in V$, denoted as $n_k$. Then, we compose $n_k'$ graphs as in Fig. 6 for the $n_k'$ agents.

We can formalize an ILP as follows: find the routes for all the agents, such that the maximum individual energy cost is minimized and the unit agent flow on the paths sums to the original network flow
\[
\begin{align*}
\min \quad & \max \sum_{i,j \in N} b_{kij} \cdot x^l_{kij} \\
\text{s.t.} \quad & \sum_{i=1}^{n_k} x^l_{kij} = x'_{kij} \quad \forall i, j \in N \\
& \sum_{i \in N} x^l_{kim} = \sum_{j \in N} x^l_{kmj} \quad \forall m \in M, \ \forall l = 1, \ldots, n_k' \\
& \sum_{i \in M} x^l_{kst} = 1 \quad \forall l = 1, \ldots, n_k' \\
& x^l_{kij} \in \{0, 1\} \quad \forall i, j \in N, \ \forall l = 1, \ldots, n_k',
\end{align*}
\]

\[
\begin{align*}
\sum_{i \in N} x^l_{kim} = \sum_{j \in N} x^l_{kmj} \quad \forall m \in M, \ \forall l = 1, \ldots, n_k' \\
\sum_{i \in M} x^l_{kst} = 1 \quad \forall l = 1, \ldots, n_k' \\
x^l_{kij} \in \{0, 1\} \quad \forall i, j \in N, \ \forall l = 1, \ldots, n_k'.
\end{align*}
\]
The objective (31) penalizes the maximum energy cost of an individual agent. Equation (32) ensures that the resulting agent flows in sum up to the rounded flow. Equation (33) is the network constraint. Equation (34) requires that in each subgraph, there is only one agent.

V. EXPERIMENTS AND RESULTS

In this section, we first initialize randomized flow networks to test the decomposition of the agent flows and then use two practical applications to evaluate the robustness, generalizability, and scalability of the proposed risk minimization model. The models and algorithms are implemented using the GUROBI solver. All the computations were done on a laptop with Intel i7-7660U (2.50 GHz).

A. Decompose Agent Flows Into Paths

In this section, we use randomized test cases of different sizes to evaluate the computational cost of the flow decomposition process. The evaluation metric is the time to solve both the rounding and splitting problems to zero optimality gaps.

1) Setup and Test Cases: An agent flow network with random connections is initialized, and the flow and unit cost of each edge is sampled from uniform random distributions. The hyperparameters are the maximum flow on an edge and the node number of the network. The optimization models in Section IV are then applied to the initialized random flow network to solve the minimum energy cost rounding and minimum max-energy flow split problems.

2) Result and Discussion: Different sizes of flow decomposition problems are solved, and the computational costs of the two steps are listed in Table III. The size of the rounding and split problems are proportional to edge number and edge number × sum integer flow, respectively. For the cases shown in Table III, the rounding steps can be completed within several milliseconds. The rounding process scales well because the linear program can be solved in polynomial time. The split step involves solving an ILP, which does not scale well generally with the problem size. The largest test cases shown in the table involve 70 tasks and 241 agents on average, much larger than the typical teaming problem sizes that the risk minimization model will be applied to. Therefore, the flow decomposition part will not be the bottleneck of the overall teaming planner.

Though there is no explicit integer constraint in the rounding model in Section IV-A, we provided a proof in Appendix B that showed that the solutions would be integers, and the results support it. As an example, the rounding solution for a case from the first row of the table is shown in Fig. 7, where we can see that

![Fig. 7](image-url)

Fig. 7. Flow rounding results (five task nodes, 15 edges). The two numbers on an edge indicate the agent flows before and after the rounding process. This example corresponds to one instance in the first row of Table III.

![Table III](image-url)

**TABLE III**

| Task | Edge | Flow | Int flow | Round time | Split time |
|------|------|------|----------|------------|------------|
| 5    | 11±1 | 110.6±33.7 | 114±34 | 0.001±0.0005 | 0.09±0.04  |
| 10   | 28±3 | 213.7±30.5 | 221±31 | 0.001±0.0002 | 1.23±0.39  |
| 35   | 107±4 | 135.7±13.2 | 172±13 | 0.001±0.0001 | 19.33±9.50 |
| 70   | 210±7 | 163.6±7.5 | 241±8  | 0.001±0.0009 | 197.0±32.9 |

*"Task" is the number of tasks, which equals the number of nodes in the flow network. 'Flow' is the sum of agent flows from the start to the terminal node and ‘int flow’ is the value after rounding (i.e., the number of agents used). 'Round time' and 'split time' are the times used to solve the rounding and splitting problems, with units in seconds. For each row, we randomly initialize 10 similar sized test cases to evaluate the algorithm and show the mean and standard deviation.
the agent flow on each edge is rounded up to an integer, while the overall network flow constraint is maintained. For instance, 13.41 on edge (S, 1) is rounded to 15 instead of 14 in order to maintain the flow constraint.

B. Capture the Flag

In this section, we apply the risk minimization model in Section III to a team of agents in a capture the flag game setting and compare the team performance against the baseline (STRATA [39]) in a simulation environment shown in Fig. 8. The goal of this simulation is to evaluate and demonstrate the performance of the task assignment component of our framework. The number of wins is used as the metric for task performance.

1) Game Setup, Baseline, and Metrics: The blue and green teams contain 12 heterogeneous agents initialized at random locations within their own sides. The overall goal is to win the game by capturing the flag from the other team. The 12 agents are from four species (three individuals for each species). These species have different speed, viewing distance, health, and ammunition capabilities (see Table IV). Among these capabilities, the first two are modeled as noncumulative, while the last two are cumulative.

Each agent is designed to play one role out of the three tasks listed in Table V and will disappear when its health reaches zero. The adversarial elements of the game are not explicitly modeled, and the focus is to assign tasks such that the requirements are fulfilled. Therefore, the agents follow predefined behaviors once their tasks are determined.

2) Results: Based on all the settings described in the above section, we developed a simulation environment in C++. A
screenshot is shown in Fig. 8. The three task assignment models, random, baseline, and CTAS, are linked with the simulation environment. During the simulation, the mean time for the baseline and CTAS to optimize and output an assignment is 0.43 and 0.04 s, respectively.

The relative performances of the models are shown in Fig. 9. As can be seen from Fig. 9, CTAS results in a higher win rate as compared to the baseline or random selection methods. The average number of agents assigned to attack, grouped by species, is shown in Fig. 10. Both the baseline and CTAS prefer using species 4 for the attacking task with the difference being that the baseline still uses species 1 and 2 for attack occasionally.

Though the baseline claims to be able to consider noncumulative capabilities through thresholding, it lacks an explicit mechanism to prevent incompetent agents from joining a task. For instance, though species 1 and 2 are not competent for the attack task due to their low speed, they are still allowed to conduct the task and contribute to other required capabilities such as health. However, this is not a good choice as there exist other agents that satisfy both the speed and health requirements.

Another possible factor contributing to CTAS’s improved performance is that the CTAS algorithm directly minimizes the CVaR metric, which ensures enough task-required capabilities, whereas the baseline focuses on matching the expected requirements and penalizing variance of the assigned capability distributions.

In conclusion, through the comparison, our framework demonstrates superior task assignment performance against the baseline algorithm, and the task assignment patterns in Fig. 10 support this performance result.

C. Robotic Services During a Pandemic

In this section, we demonstrate the generality and the task assignment and scheduling components of our framework by considering a more complex scenario inspired by the COVID-19 pandemic. We use our previous work in [14] as a baseline. The energy cost and probability of success will be used to evaluate the optimality of the generated plans. The optimality gap given limited time for the optimization will be used to evaluate the scalability of the framework.

1) Experiment Setup and Model Description: Consider pandemic robotic services in a city environment consisting of multiple delivery, disinfection, test, and treatment subtasks that require one or multiple agents to complete. The task-required capabilities and agent capabilities are modeled as random distributions. We include eight task types and seven agent species in this example, some of the agents needing a human operator.

Applying the CTAS model to describe the problem of the pandemic robotic services, we first define nine capability types according to the chosen tasks and agents in Table VII.

The seven agent species and their capabilities are defined in Table VIII, according to real-world references [56]. The columns are different capabilities. The agent capabilities are Gaussian distributed random variables in this example. The values in the table are expectations of the random distributions, and the standard deviations are 10% of the expectations.

The eight task types are described in Table IX. Their required capabilities are listed in Table X. The columns are different capabilities. A value in the table is the expected requirement for that specific capability, and the standard deviation is 10% of
the expectation. $\gamma$ is a scaling coefficient. For example, $\gamma$ in (row 1, column 3) means task type 1 requires the team’s capability $\alpha_3 \geq \gamma$. When $\gamma$ is set larger, more agents get involved in the optimization and the problem space is larger. We do not scale noncumulative capabilities. For one task, the requirements on different capability types are imposed with “and” logic. These tasks are distributed in a city. An example of 16 tasks (two tasks from each type) is shown in Fig. 17(a). The agents start from the base, which, in practice, can be a hospital.

We use the M3500 dataset [57] as the city’s road map in this illustration. We assume that we have a viral exposure level (a real number) at some locations and can infer a cost map of the virality level using Gaussian process [14], [58] based on the assumption that the neighborhood of a high-cost location is also high-cost. For example, in the city map in Fig. 17(a), we randomly choose six contaminated and six proven-safe regions as samples and learn a viral exposure map. The blue and red regions have low and high cost, respectively, while the white regions have less information, high ambivalence, and are assigned a medium cost. Based on this, we compute a viral-exposure-based travel cost ($A^*$ path [59]) between the tasks and regard it as the energy cost for the edges in the graph in Fig. 4.

We choose the agents and tasks from their types in Tables VIII and X and compose 32 mission cases where the agent numbers, task numbers, and the coefficient $\gamma$ are chosen from $\{21, 70, 140\}$, $\{16, 24, 32, 40\}$, and $\{1, 3, 5, 10\}$, respectively. For each mission case, we generate six instances with randomly selected tasks locations, which result in different graphical models in Fig. 4. For the test cases, tasks $i$ and $i + 8$ have the same requirements, but are at different locations.

Based on these cases, we evaluate the mission performance and computational cost of the three models in Table XI.

1) CTAS: The risk minimization model.
2) CTAS-D: The risk part, $h_i$, is removed from the objective function (14).
3) CTAS-O: Solve the same problem as CTAS-D without the flow decomposition. The size of the math problem is larger and harder to solve. This is our baseline from [14].

2) Computational Cost and Discussion: A 120-s time limitation is added to the solvers of the models for all the test cases. For each mission case, we run the six test instances, and if the solver can find a feasible solution within the time limit, we consider it successful. The success rate of each model is shown in Fig. 11. We show the detailed average performances for the mission sizes with success rates larger than 50% in Figs. 12–15. The optimality gaps after the rounding process in Section IV-A are given in Fig. 12(a). The blue cells mean the solver’s success rate for the specific mission size is less than 50% for the time limit picked. According to the two figures, CTAS-D and CTAS can solve much larger teaming problems than CTAS-O. The largest problem that CTAS-D solves in this example involves 140 agents and 40 tasks.

The increases in the optimality gaps due to the rounding process are shown in Fig. 12(b). For most of the cases, the rounding process introduces an increase in the optimality gap within 1%. The leftmost cases have larger increases because the agent numbers are small, and rounding has a larger overall influence. Since CTAS-O involves no rounding process, the increased gap is 0 for all test cases.
Fig. 12. Optimality gap of the three models applied to the 32 test cases, defined as (objective value − lower bound)/lower bound. A smaller optimality gap indicates a better solution. The grayscale colors correspond to the value specified in the color bars on the right. The blue cells mean the solver’s success rate for the specific mission case is < 50% for the time limit picked (120 s).

(a) Optimality gap before the flow rounding process. The two white cells in the CTAS group are outliers whose values are around 1.0. (b) Increased optimality gap due to the rounding process.

Fig. 13. Mean probability of success for the tasks using the three models.

Fig. 14. Comparing the results of CTAS to CTAS-D. The values (colors) in the grids are (a) increased energy (relative) and (b) increased mean probability (relative) by adding the risk as an objective. That is, the values are (CTAS − CTAS-D)/CTAS-D.

Fig. 15. Relative approximation gap of the nonlinear CVaR through sampling and linear programming.

In these test cases, the agent capabilities are Gaussian distributed. Therefore, given the estimation of the means and variances, we can calculate the probability of success and its mean as follows:

\[ P(\text{task } i \text{ succeeds}) = \prod_{\text{capability } a} P(\text{a satisfied}) \quad \forall i \in M \]

\[ \text{mean } P(\text{success}) = \left( \prod_{i \in M} P(\text{task } i \text{ succeeds}) \right) ^{1/n_m} . \]

The mean probabilities of success for the three models are shown in Fig. 13. For all the cases, CTAS-O and CTAS-D models roughly result in a probability of 0.25. This is because each task requires two capabilities on average. When the risk is not considered, these two models tend only to match the expected requirement, and the probability of matching a single capability is 0.5. Clearly, the risk minimization model increases the probability of success for the tasks.

In Fig. 14, we compare the result of CTAS to its deterministic version CTAS-D. With the chosen \( \beta, C_e, \) and \( C_h, \) for most of the cases, a \( \sim 20\% \) increase in energy cost introduces a \( \sim 35\% \) increase in the mean probability of success.

The tradeoff between energy cost and robustness in the objective function and can be tuned smoothly. The tradeoff gained by changing \( C_e, C_h, \) and \( \beta \) is shown in Fig. 16, using the smallest test case as an example. The penalty coefficient has a major impact on the tradeoff, while \( \beta \) has a local impact. According to our investigation, choosing \( \beta = 0.8 - 0.97 \) would mostly cover the Pareto set. This shows the advantage of using the CVaR as a risk metric: better Pareto optima are gained than optimizing the expectation (\( \beta \to 0 \)).

For the SAA of the CVaR, we use 500 samples to approximate the Gaussian distributions. Given the solution, we compare the nonlinear objectives to their sample approximations and find that the relative approximation gaps are smaller than 1% for most of the cases, as shown in Fig. 15. This shows that the approximation quality is good using 500 samples.

\( \beta: \) hyperparameter in the CVaR function \( \eta_\beta(\cdot) \) in Fig. 3 and objective function (16) and (17).

\( C_e \) and \( C_h: \) the coefficients on energy cost and task completion in (14).
3) Mission Performance and Discussion: This section uses a test case with 16 tasks, 21 agents, and $\gamma = 1$ as an example to compare the solutions generated by the three models. In Table XII, we list the performance metrics of five tasks whose success rate is increased by the CTAS model. All three models are solved to optimal within the time limit. The models presented in this article end up with much fewer variables for the same problem. By comparing CTAS with CTAS-D, we see that the risk minimization model reduces the CVaR and increases the probability of success for five tasks out of 16, with a small increase (5.3%) in the overall energy cost.

The teams for the last eight tasks are shown in Table XIII. In the table, $v_i$ stands for one agent from species $i$. As expected, the CTAS model puts more agents in the team to reduce the CVaR. This task assignment results from simultaneously considering the energy cost. As an example, CTAS puts more agents in the team of tasks 9–12, but not tasks 1–4, even if they are of the same task types. Because energy and time costs are jointly considered, and tasks 1–4 are far from the depot, adding more agents to ensure redundancy and robustness could potentially result in much higher costs. The risk minimization model also generates the routes and a consistent schedule. For example, one agent from species 1, 4, and 5 visit task 13 at the same time as a team. The routes of species 1, 4, and 5 are shown in Fig. 17. These routes minimize overall travel distances and avoid the high-cost red regions to lower energy costs.

![Fig. 17](image)

(a) Task distribution: the 16 tasks are distributed in a city [57] where the unit travel cost depends on viral exposure. Blue and red stand for low and high energy costs, respectively. (b)–(d) Planned routes from the CTAS model for species 1, 4, and 5. Different colored lines represent distinct agent individuals from the same species.
In summary, according to the teams in Table XIII and routes in Fig. 17, the CTAS framework generates a consistent schedule for coordination, outputs routes that minimize energy costs, and assigns tasks to agents such that redundancy is preserved at low costs to ensure a higher probability of task completion. The computational evaluation in Section V-C2 (particularly, Fig. 12) shows that the frameworks CTAS and CTAS-D scale to 140 agents and 40 tasks with low optimality gaps. The scalability in agent number is better than the task number. Furthermore, CTAS-D still shows no optimality gap degeneration with the largest test case we tested.

VI. CONCLUSION

This article addressed a complex task allocation problem in the category of CD-[ST-MR-TA]. We proposed a mixed-integer programming model that simultaneously optimizes the task decomposition, assignment, and scheduling. The uncertainty within the team’s capability was considered through risk minimization, and a robust metric, CVaR, was minimized to ensure the robustness to task completion (enough capability were assigned). The framework contributed to a domain-independent representation for complex tasks and heterogeneous agent capabilities that can generalize to multiagent applications where the major goals are satisfying task-required capabilities. A two-step solution method was described, and the whole framework was evaluated in two different practical test cases. Results showed that the framework scales up to 140 agents and 40 tasks for the cases tested and solves the problems with low optimality gaps. Given the selected hyperparameters, the resulting assignments and schedules provided a reasonable tradeoff between energy, time, and the probability of success. The task assignment performance (apart from the scheduling) was also demonstrated through a comparison with the STRATA framework in the capture the flag case.

Future work will consider an extension of this work to a distributed framework, which could further improve the scalability of the system and the robustness to communication constraints and the loss of agents. In addition, a probabilistic learning method that automatically estimates the parameters in the representation of task requirements and agent capabilities from current and previous task executions is an interesting future work. Such a learning method would enable the possibility of closing the loop of the task assignment and scheduling, and iteratively improving the performance. For modeling choices, we will consider imposing a necessary and sufficient energy constraint in the model in Section III while preserving the scalability of the current framework.

Appendix A

NONTRIVIALNESS OF THE FLOW DECOMPOSITION

A rounding or splitting without solving an optimization can break flow constraints or result in suboptimal solutions.

The rounding process has to maintain the flow constraints: for all nodes, the incoming and outgoing flow must equal. A naive rounding might break the flow constraint. Take node 3 in Fig. 18(a) as an example: naively rounding up the flows on an edge will result in two incoming agents, but three outgoing agents.

The rounding selection process is not unique, even when meeting the flow constraints. For instance, Fig. 18(b) and (c) are both valid ways to round the flow in Fig. 18(a). However, the energy cost of the flow in Fig. 18(c) is higher. Therefore, the optimal rounding process will need to maintain the flow constraint and introduce the lowest additional energy cost.

After rounding the flow to integers, there are multiple choices to assign individual agent paths. We call this “splitting the flow into paths”. For example, the flow in Fig. 18(b) can be split into the three agent routes either in Fig. 19(a) or (b). Though the total energy costs are equivalent, the energy costs of the three paths in the two choices are \{20, 20, 20\} and \{20, 24, 16\}, respectively. The choice affects the behavior of an individual agent. Here, we prefer the former split, because the energy costs of individual agents are more evenly distributed and the maximum energy cost of an individual is smaller. We define the optimal flow split as the set of paths that minimizes the maximum individual energy cost.
the flow constraint (30) is converted into $A_1 x \leq b_1$, and the bound constraint (29) is converted into $A_2 x \leq b_2$.

According to Definition 1, $A_1$ is an incidence matrix of a directed graph. $b_1$ is a zero vector. According to the format of constraints (29), $A_2$ is an identity matrix, and $b_2$ is an integer-valued vector. Therefore, $b$ is integer valued.

According to Theorem 1, $A_1$ is a TU matrix.

According to Theorem 3, $A_1^T$ is a TU matrix.

Because $A_1^T$ is an identity matrix, according to Theorem 2, $[A_1^T | A_2^T]$ is a TU matrix.

According to Fig. 20 and Theorem 3, $A = [A_1^T | A_2^T]^T$ is a TU matrix.

Because matrix $A$ is a TU matrix and $b$ is integer valued, according to Theorem 4, all the vertices of the polytope defined by $Ax \leq b$ are integer valued.

If Simplex-based algorithms are applied to the optimization problem, the algorithms search through vertices of the polytope to find the optimal point. Therefore, the optimal point must be a vertex of the polytope. Since we have proved that all the vertices are integral, the solution to $x$ will be integral.

\[ \text{Proof: The solutions of the rounding problem from Simplex-based algorithms will be integer valued.} \]

Now, we write the constraints of the rounding problem, i.e., (29) and (30), in the form of $Ax \leq b$. See Fig. 20; suppose that

\[ \text{REFERENCES} \]

[1] P. Tokekar, J. V. Hook, D. Mulla, and V. Isler, “Sensor planning for a symbiotic UAV and UGV system for precision agriculture,” IEEE Trans. Robot., vol. 32, no. 6, pp. 1498–1511, Dec. 2016.

[2] P. R. Wurman, R. D’Andrea, and M. Mountz, “Coordinating hundreds of cooperative, autonomous vehicles in warehouses,” AI Mag., vol. 29, no. 1, p. 9, 2008.

[3] J. Werfel, K. Petersen, and R. Nagpal, “Designing collective behavior in a termite-inspired robot construction team,” Science, vol. 343, no. 6172, pp. 754–758, 2014.

[4] D. Shishika, J. Paulos, and V. Kumar, “Cooperative team strategies for multi-player perimeter-defense games,” IEEE Robot. Autom. Lett., vol. 5, no. 2, pp. 2738–2745, Apr. 2020.

[5] X. Cai, B. Schlotfeldt, K. Khosoussi, N. Atanasov, G. J. Pappas, and J. P. How, “Non-monotone energy-aware information gathering for heterogeneous robot teams,” in Proc. IEEE Int. Conf. Robot. Autom., 2021, pp. 8859–8865.

[6] M. Quann, L. Ojeda, W. Smith, D. Rizzo, M. Castanier, and K. Barton, “An energy-efficient method for multi-robot reconnaissance in an unknown environment,” in Proc. Amer. Control Conf., 2017, pp. 2279–2284.

[7] M. Quann, L. Ojeda, W. Smith, D. Rizzo, M. Castanier, and K. Barton, “Power prediction for heterogeneous ground robots through spatial mapping and sharing of terrain data,” IEEE Robot. Autom. Lett., vol. 5, no. 2, pp. 1579–1586, Apr. 2020.

[8] K. Yu, J. M. O’Kane, and P. Tokekar, “Coverage of an environment using energy-constrained unmanned aerial vehicles,” in Proc. IEEE Int. Conf. Robot. Autom., 2019, pp. 3259–3265.

[9] Y. Sung, A. K. Budhiraja, R. K. Williams, and P. Tokekar, “Distributed assignment with limited communication for multi-robot multi-target tracking,” Auton. Robot., vol. 44, no. 1, pp. 57–73, 2020.

[10] Y. Tan and Z.-y. Zheng, “Research advance in swarm robotics,” Defence Technol., vol. 9, no. 1, pp. 18–39, 2013.

[11] K. Sundar, S. Venkatachalam, and S. G. Manyam, “Path planning for multiple heterogeneous unmanned vehicles with uncertain service times,” in Proc. Int. Conf. Unmanned Aircr. Syst., 2017, pp. 480–487.

[12] G. A. Korsah, A. Stentz, and M. B. Dias, “A comprehensive taxonomy for multi-robot task allocation,” Int. J. Robot. Res., vol. 32, no. 12, pp. 1495–1512, 2013.

[13] L. Zhou and P. Tokekar, “Multi-robot coordination and planning in uncertain and adversarial environments,” Curr. Robot. Rep., vol. 2, no. 2, pp. 147–157, 2021.

[14] B. Fu, W. Smith, D. Rizzo, M. Castanier, and K. Barton, “Heterogeneous vehicle routing and teaming with Gaussian distributed energy uncertainty,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2020, pp. 4315–4322.
[15] R. M. Karp, “Reducibility among combinatorial problems,” in Complexity of Computer Computations. New York, NY, USA: Springer, 1972, pp. 85–103.

[16] B. P. Gerkey and M. J. Matarić, “A formal analysis and taxonomy of task allocation in multi-robot systems,” Int. J. Robot. Res., vol. 23, no. 9, pp. 939–954, 2004.

[17] A. Khamis, A. Hussein, and A. Elmogy, “Multi-robot task allocation: A review of the state-of-the-art,” in Cooperative Robots and Sensor Networks. New York, NY, USA: Springer, 2015, pp. 31–51.

[18] E. Nunes, M. Manner, H. Mitiche, and M. Gini, “A taxonomy for task allocation problems with temporal and ordering constraints,” Robot. Auton. Syst., vol. 90, pp. 55–70, 2017.

[19] Y. Rizk, M. Awad, and E. W. Tunstel, “Cooperative heterogeneous multi-robot systems: A survey,” ACM Comput. Surv., vol. 52, no. 2, pp. 1–31, 2019.

[20] A. Prorok, M. Malenciac, L. Carlone, G. S. Sukhatme, B. M. Sadler, and V. Kumar, “Beyond robustness: A taxonomy of approaches towards resilient multi-robot systems,” 2021, arXiv:2109.12343.

[21] G. A. Korsah, B. Kannan, B. Browning, A. Stentz, and M. B. Dias, “xBots: An approach to generating and executing optimal multi-robot plans with cross-scale dependencies,” in Proc. IEEE Int. Conf. Robot. Autom., 2012, pp. 115–122.

[22] C. Özgüven, Y. Yazuv, and L. Özbuga, “Mixed integer goal programming models for the flexible job-shop scheduling problems with separable and non-separable sequence dependent setup times,” Appl. Math. Model., vol. 36, no. 2, pp. 846–858, 2012.

[23] W.-Y. Ku and J. C. Beck, “Mixed integer programming models for job shop scheduling: A computational analysis,” Comput. Oper. Res., vol. 73, pp. 165–173, 2016.

[24] S. Liembetcharat and M. Veloso, “Modeling mutual capabilities in heterogeneous teams for role assignment,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2011, pp. 3638–3644.

[25] S. Liembetcharat and M. Veloso, “Weighted synergy graphs for role assignment in ad hoc heterogeneous robot teams,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2012, pp. 5247–5252.

[26] E. G. Jones, M. B. Dias, and A. Stentz, “Time-extended multirobot coordination for domains with intra-path constraints,” Auton. Robot., vol. 30, no. 1, pp. 41–56, 2011.

[27] M. Pujol-Gonzalez, J. Cerquides, P. Meseguer, and J. A. Rodriguez-Aguilar, “Efficient inter-team task allocation in RoboCup rescue,” Proc. Int. Conf. Auton. Agents Multiagent Syst., 2015, pp. 413–421.

[28] D. Song, K. Huebner, V. Kyrki, and D. Kragic, “Learning task constraints for robot grasping using graphical models,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2010, pp. 1579–1585.

[29] C. H. Ek, D. Song, K. Huebner, and D. Kragic, “Task modeling in imitation learning using latent variable models,” in Proc. IEEE-RAS Int. Conf. Humanoid Robots, 2010, pp. 548–553.

[30] R. Zlot and A. Stentz, “Market-based multirobot coordination using task abstraction,” in Field and Service Robotics. New York, NY, USA: Springer, 2003, pp. 167–177.

[31] R. Zlot and A. Stentz, “Complex task allocation for multiple robots,” in Proc. IEEE Int. Conf. Robot. Autom., 2005, pp. 1515–1522.

[32] B. Hayes and B. Scassellati, “Autonomously constructing hierarchical task networks for planning and human-robot collaboration,” in Proc. IEEE Int. Conf. Robot. Autom., 2016, pp. 5469–5476.

[33] S. D. Klee, G. Gemignani, D. Nardi, and M. Veloso, “Graph-based task libraries for robots: Generalization and autonomy completion,” in Congress of the Italian Association for Artificial Intelligence. New York, NY, USA: Springer, 2015, pp. 397–409.

[34] S. Ekvall and D. Kragic, “Robot learning from demonstration: A task-level planning approach,” Int. J. Adv. Robot. Syst., vol. 5, no. 3, p. 33, 2008.

[35] T. Barfoot et al., “Making sense of the robotized pandemic response: A comparison of global and Canadian robot deployments and success factors,” 2020, arXiv:2009.08577.

[36] M. Rudolph, S. Chernova, and H. Ravichandar, “Desperate times call for desperate measures: Towards risk-adaptive task allocation,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2021, pp. 2592–2597.

[37] F. Faruq, D. Parker, B. Lacerda, and N. Hawes, “Simultaneous task allocation and planning under uncertainty,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2018, pp. 3559–3564.

[38] P. J. Bernhard, S. Polloko, and A. Knoll, “Addressing inherent uncertainty: Risk-averse mission planning for automated driving using distribution-reinforcement learning,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2019, pp. 2148–2155.

[39] M. Rudolph, S. Chernova, and H. Ravichandar, “Desperate times call for desperate measures: Towards risk-adaptive task allocation,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2021, pp. 2592–2597.

[40] I. Heller and C. Tompkins, “An extension of a theorem of Dantzig’s,” in Linear Inequalities Related Syst., vol. 38, pp. 247–254, 1956.
Bo Fu received the M.S. degree in mechanical engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2019. He is working toward the Ph.D. degree with the Department of Robotics, University of Michigan, Ann Arbor, MI, USA.

His current research interests include robust and resilient decision making for heterogeneous robot teams under uncertainty using optimization and learning-based approaches.

William Smith received the Ph.D. degree in mechanical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2014.

He is currently the Autonomous Mobility Technical Specialist within Ground Vehicle Robotics, U.S. Army Ground Vehicle Systems Center, Warren, MI, USA, where he focuses on improving autonomous ground vehicle software through better infrastructure tools, such as simulation, data management, and DevOps. His research interests include efficient high-fidelity ground–vehicle interaction and vehicle team coordination.

Matthew Castanier received the Ph.D. degree in mechanical engineering from the University of Michigan, Ann Arbor, MI, USA, in 1995.

He is currently a Research Mechanical Engineer with the U.S. Army DEVCOM Ground Vehicle Systems Center (GVSC), Warren, MI. From 1996 to 2008, he was a Member of the Research Faculty with the University of Michigan. In 2008, he joined GVSC, where he was promoted to his current position in 2011. He has authored or coauthored more than 50 journal articles and more than 100 conference papers. His research interests include powertrain modeling and simulation, multidisciplinary design optimization, and trade space exploration.

Denise M. Rizzo received the Ph.D. degree in mechanical engineering from Michigan Technological University, Houghton, MI, USA, in 2014.

She is currently the Deputy Chief Scientist with the Office of the Chief Scientist, U.S. Army Ground Vehicle Systems Center (GVSC), Warren, MI. From 2000 to 2008, she was a Control Research and Development Engineer with the Powertrain Group, Chrysler LLC, Detroit, MI. In November 2008, she joined GVSC and was promoted to her current position in 2020. She has authored or coauthored 40 articles in archival journals, 49 papers in refereed conference proceedings, and four technical government reports. She holds two patents. Her research interests include modeling, simulation, and control of energy intelligent autonomous ground vehicles.

Dr. Ghaffari is the recipient of the 2021 Amazon Research Awards.

Maani Ghaffari received the Ph.D. degree in robotics from the Centre for Autonomous Systems, University of Technology Sydney, Ultimo, NSW, Australia, in 2017.

He is currently an Assistant Professor with the Department of Naval Architecture and Marine Engineering and the Department of Robotics, University of Michigan, Ann Arbor, MI, USA, where he has recently established the Computational Autonomy and Robotics Laboratory. His research interests include theory and applications of robotics and autonomous systems.

Kira Barton received the B.Sc. degree from the University of Colorado at Boulder, Boulder, CO, USA, in 2001, and the M.Sc. and Ph.D. degrees from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 2006 and 2010, respectively, all in mechanical engineering.

She is currently an Associate Professor with the Department of Robotics and the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI, USA. She is also the Associate Director of the Automotive Research Center, a University-based U.S. Army Center of Excellence for modeling and simulation of military and civilian ground systems. From 2017 to 2020, she was a Miller Faculty Scholar with the University of Michigan. Her research interests include advancements in modeling, sensing, and control for applications in smart manufacturing and robotics, with a specialization in learning and multiagent systems.