2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Multi-Robot Coordination and Cooperation with Task Precedence Relationships

Walker Gosrich[1], Siddharth Mayya[2], Saaketh Narayan[1], Matthew Malencia[1], Saurav Agarwal[1], and Vijay Kumar[1]

***Abstract*— We propose a new formulation for the multi-robot task planning and allocation problem that incorporates (a) precedence relationships between tasks; (b) coordination for tasks allowing multiple robots to achieve increased efficiency; and (c) cooperation through the formation of robot coalitions for tasks that cannot be performed by individual robots alone. In our formulation, the tasks and the relationships between the tasks are specified by a task graph. We define a set of reward functions over the task graph's nodes and edges. These functions model the effect of robot coalition size on task performance while incorporating the influence of one task's performance on a dependent task. Solving this problem optimally is NP-hard. However, using the task graph formulation allows us to leverage min-cost network flow approaches to obtain approximate solutions efficiently. Additionally, we explore a mixed integer programming approach, which gives optimal solutions for small instances of the problem but is computationally expensive. We also develop a greedy heuristic algorithm as a baseline. Our modeling and solution approaches result in task plans that leverage task precedence relationships and robot coordination and cooperation to achieve high mission performance, even in large missions with many agents.**

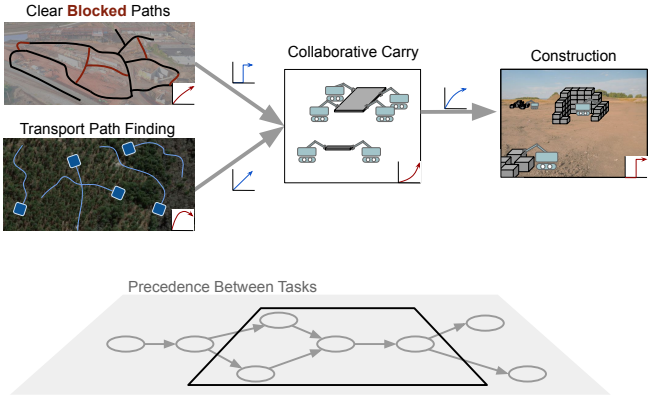

Fig. 1. The task graph (*bottom*) shows precedence relationships between tasks in a mission. Highlighted is a subset of the tasks, showing coalition functions (*red*) and precedence relationships (*blue*). For example, before transporting construction materials, robots may need to find or clear paths through a debris-filled construction site. The quality of these paths directly impact the team's ability to transport material. Additionally, the performance of transporting material, e.g., whether any materials are damaged, impacts how quickly and how well the team performs a following construction task.

I. Introduction

Multi-robot systems offer versatility to accomplish large missions by forming teams to efficiently complete tasks. A fundamental challenge in multi-robot missions is to allocate tasks to robots, referred to as the multi-robot task allocation (MRTA) problem. This paper considers a generalization of the MRTA problem—tasks have precedence relationships among them, i.e., the quality of a task's execution may depend on how well a related task is executed. For example, in an autonomous construction mission, an area must be cleared and leveled, and materials gathered before construction takes place, as illustrated in Figure 1. In such a scenario, the quality with which prior tasks are completed impacts the ability of the robot team to complete future tasks, e.g., a well-leveled foundation will result in a better overall performance of the construction mission. Thus, it is imperative to model relationships among tasks, especially the *influence* of a preceding task on subsequent tasks.

The other important generalization of MRTA we consider is that a coalition of multiple robots can work on a task together. On some tasks, such as multi-robot coverage, robots may *coordinate* to achieve improved results. Other tasks, such as collaborative carry, require multiple robots to *cooperate* to perform the task when a single robot cannot (e.g., Figure 1). The standard MRTA problem considers only the coordination of robots [1]. In the simplest case of single robot teams, this problem can be posed as an assignment problem [2]. However, some tasks require multi-agent teams to be effectively executed. Additionally, multi-agent teams can bring greater efficacy to tasks such as sensor coverage, exploration, and resource gathering when these tasks are constrained by a time budget. Such scenarios raise two challenges: (1) the estimation of task utility by modeling the relationship between robot coalitions and tasks, and (2) the assignment of robot coalitions to tasks under temporal and resource constraints. A desirable characteristic of a task planner is to be able to reason about resource allocation, e.g., if task relationships indicate that there is a mission-critical task, a larger robot coalition may be allocated to that task, even at the expense of other tasks' performance. Hence, there is a need for a formulation that models the relationship of robot coalitions to tasks and unifies coordination and cooperation of robots for MRTA.

This paper presents a novel formulation that unifies precedence relationships among tasks, coordination and cooperation among robots through coalitions, and allocation of coalitions to tasks. The formulation is applicable to complex scenarios relevant to autonomous construction, precision

We gratefully acknowledge the support of ARL DCIST CRA W911NF-17-2-0181. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship.

[1]W. Gosrich, S. Narayan, M. Malencia, S. Agarwal, and V. Kumar are with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA, USA {gosrich, saaketh, malencia, sauravag, kumar}@seas.upenn.edu

[2]S. Mayya is with Amazon Robotics, North Reading, MA, USA (mayya.siddharth@gmail.com). This work is not related to Amazon.

agriculture, and industrial robotic applications. The contributions are:

1) A modular and expressive mission model defined over a *task graph* and a reward function (Section III). The task graph captures the precedence relationships among tasks. The reward function quantifies the utilities of the tasks and is characterized by *influence functions*—modeling of relationships among tasks, *aggregation functions*—effect of all preceding tasks on a dependent task, and *coalition functions*—modeling of task execution efficacies with robot coalition size.
2) A formulation of the problem as a min-cost network flow by leveraging the task graph (Section IV). This enables the allocation of robots to sequences of connected tasks in the task graph, and scales well with the number of tasks and robots in the mission.
3) We benchmark our flow-based solution approach with a mixed integer approach and a greedy heuristic algorithm (Section VI). The comparison is performed on over 300 randomly generated missions, including missions drawn from an autonomous construction simulator. Our flow-based approaches significantly outperform the mixed integer benchmark in large problem sizes, and perform similarly in the smaller problem instances where the mixed integer approach can compute a non-trivial solution. Our flow-based non-linear programming (NLP) approach significantly outperforms the greedy approach across the problem domain.

II. Previous Work

This work focuses on multi-task missions composed of multi-robot tasks with inter-task precedence relationships. In applications where these mission characteristics are common, such as autonomous construction and assembly [3], agriculture [4], and other complex multi-robot missions, efficient allocation of robot coalitions that enable *coordination* and *cooperation* is essential. In prior work, each of these characteristics has been considered extensively but rarely together in a cohesive model.

Many approaches allocate robots among interrelated tasks without considering multi-robot coalitions. Some approaches formulate the problem purely as a constraint satisfaction problem (CSP) [5], [6], while others hybridize Mixed Integer Linear Programming (MILP) with CSP in order to reason about task utility [7]. In operations research [8], the task allocation problem with inter-task relationships is most similar to the Vehicle Routing Problem with Time Windows [9], with additional ordering constraints [10], though these formulations do not consider multi-agent tasks or expressive inter-task relationships.

In the computer science literature, Multi-Agent Planning (MAP) with *joint actions* expresses the tight coupling between tasks that we represent with precedence relationships. Several approaches ([11], [12]) consider this combination of challenges but consider coalitions in a limited fashion. Some approaches ([13], [14], [15]) consider soft precedence constraints encoded via time-varying task rewards or imposing costs. Other methods [14], [16] model inter-task relationships among single-agent tasks in detail on a heterogeneous graph, with nodes representing tasks, agents, and locations.

Other approaches focus on *cooperation*, modeling the relationship between the coalition assigned to a task and task performance. Some works model reward as a function of the number of homogeneous robots assigned to complete the task [17], [18]. Others consider task success based on required attributes aggregated among a team of robots with heterogeneous capabilities [19]. Some such methods perform short horizon task assignment [20], which allows robust response to dynamic missions but does not permit reasoning about interrelated tasks over a time horizon. Others do not consider task reward explicitly [21].

Some approaches, like ours, consider the challenging intersection of these problem characteristics, requiring a system to reason about intersecting robot schedules and fluidly form coalitions among different tasks over an extended time horizon. The approaches that consider both task order and coalition typically use binary task rewards [22], and some additionally cannot scale to problems larger than a few robots and tasks [23], [24]. In this work, we present an expressive model for task reward that expands these models, and we present solution methods that scale to large problem sizes.

III. Modeling Coalition Performance and Task Inter-dependencies

In this section, we introduce *task graphs*, which encode the structure of task relationships, and we define a *task reward model* over the task graph. Taken together, these models can represent complex missions for homogeneous robot teams.

Let $\mathcal{T} := \{\mathcal{T}_1, \ldots, \mathcal{T}_M\}$ represent the set of M available tasks in the environment, and $\mathcal{M} := \{1, \ldots, M\}$ represent the index set of tasks. Each task $\mathcal{T}_j$ has a constant duration d_j and yields a reward r_j upon completion. The reward is a function of the assigned robot coalition and the rewards of related preceding tasks, as defined later in the section. Note that a task does not start until the complete coalition assigned to the task has formed.

Definition 1 (Task Graph): Given a set of tasks $\mathcal{T}$, the task graph is a directed acyclic graph $\mathcal{G}_T = (\overline{\mathcal{M}}, \mathcal{E})$, where $\overline{\mathcal{M}} = \{0, 1, \ldots, M\}$ is the vertex set corresponding to tasks and $\mathcal{E}$ is the edge set corresponding to task relationships. The vertex 0 represents a virtual "source" node, and all other vertices correspond to actual tasks. We add a directed edge (i, j) to the set $\mathcal{E}$ if there exists a precedence relationship between tasks $\mathcal{T}_i$ and $\mathcal{T}_j$ such that the reward r_i associated with task $\mathcal{T}_i$ impacts the reward r_j from task $\mathcal{T}_j$. We add an edge from the source node $\mathcal{T}_0$ to all tasks without preceding neighbors, resulting in a directed acyclic graph.

Furthermore, let $\mathcal{R}$ represent the index set of N robots available to execute the tasks. Let $x_k^r \in \{0, 1\}$ indicate whether a robot $r \in \mathcal{R}$ executes task $k \in \overline{\mathcal{M}}$ during the mission. We define a robot coalition $\mathcal{C}_j$ as the set of robots allocated to task $\mathcal{T}_j$. Furthermore, let $C_j = |\mathcal{C}_j|$ represent the size of the coalition, where $|\cdot|$ is the set cardinality operator.

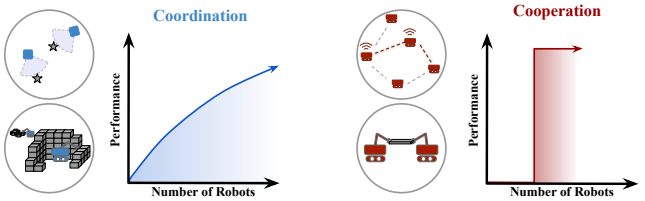

Fig. 2. Figure adapted from [25]. Two types of coalition function ρ. *Left* tasks such as coverage control and construction have a subadditive coalition function, as larger coalitions improve performance. *Right* network connectivity tasks and cooperative transport are represented by a "step" coalition function. When a critical mass of robots is collected in the coalition, the task can successfully be performed.

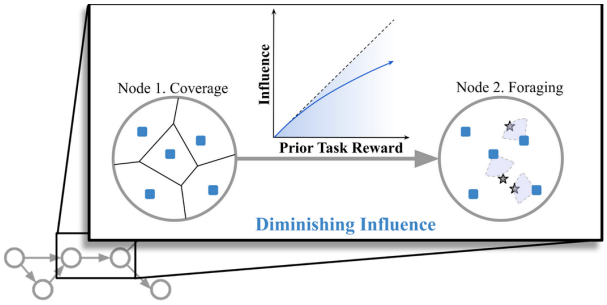

Fig. 3. The influence function, illustrated here, models the relationship between the reward at the preceding (or *influencing*) task and the reward at the current task. A better result on a coverage task would result in higher performance in the foraging task (exploiting the information accrued during coverage). This is modeled by a subadditive influence function.

A. Task Reward Model

We present a model for the reward r_j, associated with the task $\mathcal{T}_j \in \mathcal{T}$. The task reward model comprises three functions: (1) a *coalition function* ρ_j, an (2) *aggregation function* α_j, and (3) *influence functions* δ_{ij}.

Definition 2 (Task Coalition Function): Given a robot coalition $\mathcal{C}_j$ assigned to task $\mathcal{T}_j$, the task coalition function $\rho_j(C_j) : \mathbb{R} \mapsto \mathbb{R}$ returns a scalar that represents the effectiveness of the robot coalition at accomplishing the given task.

The coalition function provides an expressive model for the relationship between the robot coalitions assigned to a task and the resulting task reward. While prior works [26], [27] implicitly assume that the coalition functions are linear, this paper uses task coalition functions to model both linear and non-linear performance characteristics. Examples of these more complex coalition-reward relationships are seen in Figure 2 with common robotic tasks such as collaborative tasks with subadditive reward (left) and cooperative tasks that can only be achieved by a threshold number of robots (right).

We now model the impact of the precedence relationships on the reward r_j for each task $\mathcal{T}_j$.

Definition 3 (Task Influence Function): The influence function $\delta_{ij}(r_i) : \mathbb{R} \mapsto \mathbb{R}$, $(i,j) \in \mathcal{E}$ returns a scalar that quantifies the extent to which the performance of the robots on task $\mathcal{T}_i$ hinders or facilitates the execution of task $\mathcal{T}_j$. In other words, it models the influence of task reward r_i on r_j.

The task influence function provides a generalization of precedence constraints commonly studied in task allocation models. Typical approaches model task precedence as a binary construct: a dependent task may be executed only if a preceding task is completed with performance above a certain threshold. In contrast, representing δ_{ij} as a function enables the modeling of complex relationships between tasks often found in real-world scenarios. For instance, as shown in Figure 3, the performance of an exploration or a coverage task could be sub-linearly related to the performance of a subsequent transportation or foraging task. Similarly, the performance of a foundation leveling task could influence the performance of future construction tasks on top of that foundation. Furthermore, our formulation can also represent a classical "binary" precedence constraint by choosing δ_{ij} as a step function.

When a task $\mathcal{T}_j$ has multiple incident edges, the outputs of these multiple influence functions must be aggregated over the set of incoming precedence neighbors $\mathcal{N}_j^{\text{in}} = \{\mathcal{T}_i \mid (i,j) \in \mathcal{E}\}$.

Definition 4 (Task Influence Aggregation Function): The task influence aggregation function $\alpha_j : \mathbb{R}^{|\mathcal{N}_j^{\text{in}}|} \mapsto \mathbb{R}$ takes as an input the set of influencing rewards $\{r_i \mid i \in \mathcal{N}_j^{\text{in}}\}$ and outputs a value representing the total influence of preceding tasks on the reward r_j associated with task $\mathcal{T}_j$.

For instance, the task influence aggregation function can be set to $\sum_{i \in \mathcal{N}_j^{\text{in}}} \delta_{ij}(r_i)$ to represent the case where completing any preceding task can result in reward on task $\mathcal{T}_j$, or to $\prod_{i \in \mathcal{N}_j^{\text{in}}} \delta_{ij}(r_i)$ to represent the case when all preceding tasks must be sufficiently completed to achieve favorable performance on task $\mathcal{T}_j$.

B. Composing Final Task Rewards

The reward r_j accrued by each task $\mathcal{T}_j$ represents the quality of the outcome of the task.

$$r_j = \rho_j(C_j) \ddagger_j \alpha_j \left(\{\delta_{ij}(r_i) \mid i \in \mathcal{N}_j^{\text{in}}\}\right) \tag{1}$$

In the context of task $\mathcal{T}_j$, the symbol $\ddagger_j$ represents a function that combines outputs of the coalition function ρ_j and aggregated influence function α_j. For example, summing ρ_j and α_j can represent a "soft" constraint, where the reward is non-zero even if precedence-related tasks are incomplete. Alternatively, the product of ρ_j and α_j could be interpreted as a "hard" constraint, where a zero aggregated influence value results in a task reward of zero.

Taken together, the task graph $\mathcal{G}_T$, the set of coalition, aggregation, and $\ddagger$ functions on its vertices, $\{\rho_j, \alpha_j, \ddagger_j\}, \forall j \in \mathcal{M}$, and the set of influence functions defined over its edges $\{\delta_{jk}\}, \forall (j,k) \in \mathcal{E}$ completely specify the mission.

C. Problem Statement, Characteristics, and Assumptions

The primary objective of the MRTA problem is to generate a set of feasible robot-task assignments $x_k^r, \forall r \in \mathcal{R}, k \in \mathcal{M}$, forming a set of coalitions $\mathcal{C}$ that maximizes the rewards obtained over all tasks:

$$\max_{\mathcal{C}_j, \forall j \in \mathcal{M}} \sum_{j=1}^{M} r_j \tag{2}$$

while satisfying the following constraints: *(i)* the last task is completed by the robots before the makespan constraint time

T_f, *(ii)* robots may only work on one task at a time, *(iii)* the robot-task assignments respect the precedence relationships defined over the graph edges $(i, j) \in \mathcal{E}$, *(iv)* the task coalitions do not exceed the total number of robots N. We assume that all reward functions are non-negative. See Sections IV and V for specific instantiations of these constraints.

This MRTA problem is NP-hard, which can be shown by reducing an instance of the NP-hard 0-1 knapsack problem [28] to an instance of this problem by considering the *single-robot* case where each item in the knapsack problem corresponds to a task in the MRTA problem, item weights are task durations, item values are task rewards, and the weight capacity of the knapsack is the time budget.

IV. Flow-Based Solution to the Task Graph

In this section, we represent the MRTA problem described in Section III-C over the directed acyclic task graph, as a min-cost network flow problem. By reinterpreting the robot coalitions assigned to tasks as flows between edges in the task graph, we compute a set of reward-maximizing flows $\{f_{ij}\}$ along the edges $(i, j) \in \mathcal{E}$.

Definition 5 (Robot Flows): Let $f_{ij} \in [0, 1]$ represent the population fraction of robots flowing along edge (i, j) [Equation (3)]. The robots represented by flow f_{ij} are assigned to complete task $\mathcal{T}_i$, followed by task $\mathcal{T}_j$. According to this definition, the total available flow at the source node 0 is equal to 1 (4), where $\mathcal{N}_i^{out} = \{\mathcal{T}_j \mid (i, j) \in \mathcal{E}\}$.

$$\sum_{i \in \mathcal{N}_j^{\text{in}}} f_{ij} = \frac{C_j}{N} \quad (3) \qquad \sum_{k \in \mathcal{N}_0^{\text{out}}} f_{0k} \leq 1 \quad (4)$$

The number of robots flowing out of a node cannot exceed the number of robots that flow into that node. This physical limitation is represented by the following conservation of flow constraint:

$$\sum_{k \in \mathcal{N}_j^{\text{out}}} f_{jk} \leq \sum_{i \in \mathcal{N}_j^{\text{in}}} f_{ij} \quad \forall j \in \mathcal{M} \quad (5)$$

The representation of task assignments with population fractions results in solution methods that are *agnostic to the number of agents in the mission*: larger total coalition sizes do not increase computational complexity.

No explicit precedence constraints are necessary in this formulation because precedence relationships are encoded in the graph topology and reward functions. Note that the inequality in (5) allows robots to leave the coalition if necessary. Furthermore, we do not assume any capacity limits on the flow along the task graph edges.

1) Graph Pruning to Satisfy the Makespan Constraint: As described in Section III-C, the makespan of the mission is constrained, and this must be reflected in the network flow formulation. Towards this end, we prune nodes from the task graph if robots cannot complete them within the makespan budget. We generate a *makespan graph* $\mathcal{G}_M = (\overline{\mathcal{M}}, \mathcal{E})$ with identical topology to the task graph $\mathcal{G}_T$. Each path in the graph has a duration equal to the sum of all node durations in the path. Each node j is labeled with a *maximum* (worst-case) finish time F_j corresponding to the longest duration path that terminates at that node. This conservative definition is necessary because we assume that robot coalitions must be fully formed to begin a task. We prune from the graph any node j for which $F_j > T_f$. Therefore, any flow solution over the pruned graph will respect the makespan constraint.

2) Rounding of Population Fraction: Additionally, the flow-based solution deals in the continuous measure of population fraction, whereas real-world task allocation requires discrete quantities of robots in order to compute a task schedule for each robot. In order to translate flow solutions into realistic allocations of robots to tasks, we apply a rounding algorithm. Iterating over the nodes in the task graph in a topologically sorted manner, we compute a minimum error rounding of the flows along the outgoing edges that respects the flow conservation constraints (5). This results in a vector of integer flows that can be converted to individual robot schedules.

A. Non-Linear Programming (NLP) Flow Solution

Our primary approach to solving the task allocation problem via network flow is through nonlinear programming, using an off-the-shelf solver [29]. The objective is to maximize the task reward sum (2) on the pruned graph via a set of flows $\{f_{ij} | (i, j) \in \mathcal{E}\}$ respecting the constraints (4) and (5).

B. Greedy One-Step Lookahead Flow Solution

Because we have no solver that guarantees a globally optimal solution to the task graph problem, we develop a greedy one-step lookahead algorithm as a basis of comparison for the NLP solution. The greedy algorithm starts from node 0 and proceeds sequentially through the graph nodes in topological order. For each node $\mathcal{T}_j$, the greedy algorithm allocates all available flow among the outgoing edges of the node according to which allocation maximizes the rewards of the outgoing neighborhood $\sum_{k \in \mathcal{N}_j^{out}} r_k$. This local maximum is computed by first taking the best of 50 random samples of the outgoing flow space. We then perform gradient ascent on the reward sum $\sum_{k \in \mathcal{N}_j^{out}} r_k$ to come to the final values for the outgoing edges of $\mathcal{T}_j$. This one-step lookahead is myopic but computationally simple. The gradient ascent step requires task coalition functions and aggregation functions to be differentiable.

V. Mixed Integer Non-Linear Programming (MINLP) Solution to the Task Graph

This section describes a mixed integer non-linear programming (MINLP) approach, which leverages the mission model from Section III but differs from the flow-based approach in two crucial ways: (1) plans are constructed *robot-wise*, generating an ordered schedule of tasks for each robot; (2) the robots can switch branches of the task graph. The objective is to maximize the total reward, as described in Equations (1) and (2), with the coalition size of robots assigned to a task $\mathcal{T}_k,\ k \in \mathcal{M}$ given by $C_k = \sum_{r \in \mathcal{R}} x_k^r$.

We let non-negative continuous variables S_k and F_k denote the start and the finish times for the task $\mathcal{T}_k$. Additionally, let the binary variable x_k be non-zero when the task $\mathcal{T}_k$ is executed by at least one robot, as ensured by constraints (6)

below. The precedence relationships are dictated by the task graph $\mathcal{G}_T = (\overline{\mathcal{M}}, \mathcal{E})$. The following constraints (7) and (8), with the help of auxiliary binary variables w_{ij}, ensure that for an edge $(i,j) \in \mathcal{E}$, the preceding task $\mathcal{T}_i$ is executed before the dependent task $\mathcal{T}_j$, if and only if both $\mathcal{T}_i$ and $\mathcal{T}_j$ are executed. Furthermore, for each executed task, we need to ensure that the duration constraints (9) are enforced.

$$x_k \geq x_k^a, \forall a \in \mathcal{R} \text{ and } x_k \leq \sum_{r \in \mathcal{R}} x_k^r, \quad \forall k \in \mathcal{M} \tag{6}$$

$$w_{ij}(S_j - F_i) \geq 0,\ w_{ij} \geq x_i + x_j - 1, \tag{7}$$

$$w_{ij} \leq x_i,\ w_{ij} \leq x_j, \text{ and } w_{ij} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{E} \tag{8}$$

$$x_k\,(F_k - S_k) \geq x_k d_k, \quad \forall k \in \mathcal{M} \tag{9}$$

The rest of the constraints follow from the formulation by Nunes et al. [8]. Binary variables z_k^r denote if k is the last task executed by the robot r, and binary variables $o_{kk'}^r$ denote if robot r executes a task $k' \in \mathcal{M}$ immediately after task k.

$$\sum_{k \in \overline{\mathcal{M}}} o_{kk'}^r = x_{k'}^r, \qquad \forall r \in \mathcal{R},\ \forall k' \in \mathcal{M}$$

$$\sum_{k' \in \mathcal{M}} o_{kk'}^r + z_k^r = x_k^r, \qquad \forall r \in \mathcal{R},\ \forall k \in \overline{\mathcal{M}}$$

$$\sum_{k \in \overline{\mathcal{M}}} x_k^r\, z_k^r = 1,\ x_0^r = 1 \qquad \forall r \in \mathcal{R}$$

$$o_{kk'}^r\,(S_{k'} - F_k) \geq 0, \qquad \forall k \in \overline{\mathcal{M}},\ \forall k' \in \mathcal{M}$$

$$F_k \leq T_f, \qquad \forall k \in \mathcal{M}$$

Given the computational complexity of solving a mixed-integer problem with non-linear constraints, this approach is more applicable to instances with sparser precedence relationships and fewer robots. Nevertheless, as solving the MILNP provides an optimal solution when solved to convergence, it serves as a benchmark for heuristic solvers. The MINLP formulation admits a trivial solution corresponding to none of the tasks being executed. Thus, when solved using methods such as branch-and-bound, it always maintains a set of feasible solutions, and the solver can be terminated *anytime* to obtain a non-optimal feasible solution.

VI. Experimental Evaluation

We evaluate the three algorithms on two simulated sets of scenarios. The first generates random task graphs given a target number of tasks. The second is a block dry-stacking scenario that represents the mission structures present in autonomous construction settings. All experiments were run on a desktop computer with an 8-core Intel Core i7-9700 CPU. Our implementation uses the PySCIPOpt optimization suite [30], [31] as the solver for the MINLP approach and SciPy optimize [29] for the flow-based NLP approach.

A. Random Graph Generator Platform and Experiments

We developed a random graph generator that generates graphs with random topologies and reward functions, allowing us to explore the mission specification space broadly. We specify a target number of tasks, and randomly generate

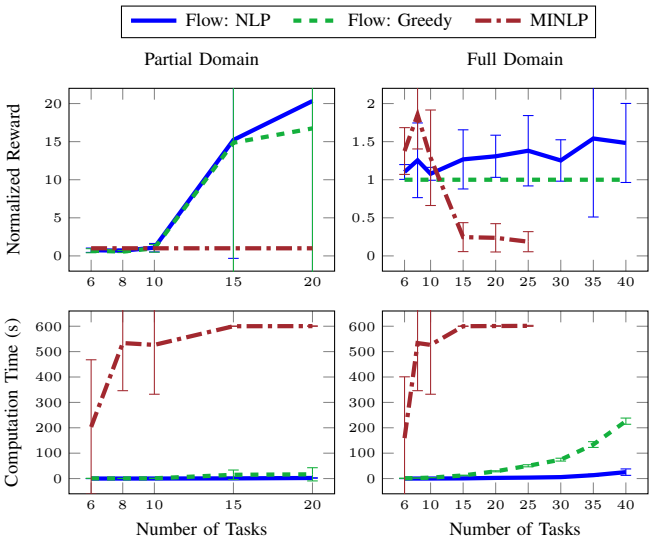

Fig. 4. Experiment 1 tests planner performance as mission size (number of tasks) is increased—we measure solution reward (top) and computation time (bottom). The left plots depict the portion of the domain where MINLP computes a non-trivial solution, normalized by MINLP reward. The right plots depict the full domain, normalized by the greedy approach's reward. As mission size grows, the flow-based approaches outperform the MINLP approach until the MINLP fails to produce non-trivial solutions.

coalition and influence functions. We sample from polynomial, sub-linear, and sigmoid functions (a smoothed approximation of a step function) with randomized parameters. We sample randomly between using a sum and product as ‡, which combines the coalition and influence function outputs. For these experiments, we use the sum operation as our influence aggregation function α_j for each task $\mathcal{T}_j$.

1) Experiment 1: Number of Tasks: In this experiment, we evaluate the impact of the number of tasks in the mission on the relative performance of the three solution methods. For each number of tasks, we conduct ten trials consisting of unique randomly generated task graphs. For this experiment, we use 4 agents, a makespan constraint $T_f = 0.6 \sum_{j \in \mathcal{M}} d_j$, and limit computation time to 10 minutes per trial.

In Figure 4 (left), we examine the portion of the domain in which the MINLP solution consistently computes a non-trivial feasible solution: ≤ 20 tasks. We graph the mean of rewards normalized by the MINLP reward for each trial, e.g., if the reward from the flow NLP solution is 20% higher than the reward from the MINLP solution, we report 1.2.

In Figure 4 (right), we show the full domain we tested, normalize the results by the *greedy* flow solution, and graph the MINLP results only when feasible and non-trivial. On average over all trials in the full domain, the flow-based NLP approach outperforms the greedy approach by 30% and takes on average 10% the time of the greedy approach to compute a solution.

2) Experiment 2: Number of Agents: Experiment 2 evaluates the relative performance of the solution methods as the number of agents varies. We test with total agent populations of {2,4,6,10,15,20}, and conduct 10 trials with unique randomly generated task graphs for each value. For this experiment, we use 10 tasks, a makespan constraint $T_f = 0.6 \sum_{j \in \mathcal{M}} d_j$, and limit computation time to 10 minutes per

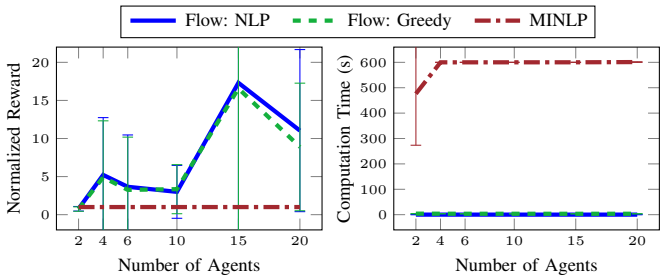

Fig. 5. Experiment 2 demonstrates the impact of the number of agents in the coalition on the performance of the three solution methods. We plot reward (left) and computation time (right). The rewards are normalized by the MINLP solution. As the number of agents increases, the MINLP problem size grows, whereas the flow-based solutions only get more accurate, due to decreased rounding error.

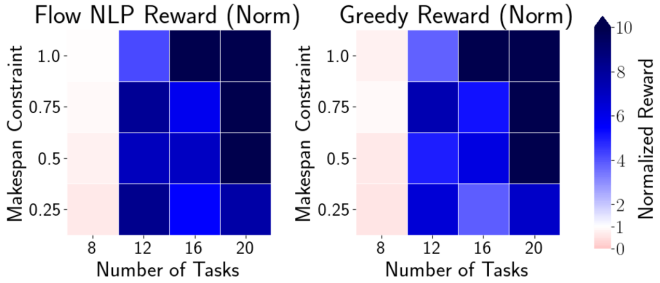

Fig. 6. The reward accrued by the NLP flow approach (left) and the greedy flow approach (right) for varying makespan constraint and number of tasks. The rewards are normalized by the MINLP reward. The flow-based approaches perform worse than the MINLP on small problems with a tight makespan constraint, but outperform on problems larger than 8 tasks. The MINLP approach fails to compute a non-trivial solution on larger problems than those shown here.

trial. In Figure 5, we plot the rewards normalized by the MINLP reward, and the computation time for each method.

3) Experiment 3: Makespan and Number of Tasks: In Experiment 3, we examine the impact of both the makespan constraint and the number of tasks on the relative performance of the solution methods. In this experiment, we examine *only the portion of the domain in which the MINLP approach computes non-trivial solutions.* We formulate the makespan constraint for randomly generated graphs as a proportion of the total duration of all tasks, $\mu \sum_{j \in \mathcal{M}} d_j$. We evaluate $\mu = \{0.25, 0.5, 0.75, 1.0\}$ and $M = \{8, 12, 16, 20\}$ tasks and conduct 10 trials at each pair of values. For this experiment, we use 4 agents and limit computation time to 10 minutes per trial. We show the NLP and greedy flow solution results normalized by the MINLP results in Figure 6.

B. Autonomous Construction Platform and Experiments

In order to evaluate our system on task graphs derived from realistic mission structures, we created a dry-stacking problem generator that is applicable to autonomous construction scenarios. Given a tower base width and a number of layers, we create a layered task graph representing the tower, where each block placement task is a node and block heights and widths are randomized. Blocks in adjacent layers have precedence relationships. Each block task's coalition function is a sigmoid function structured such that heavier blocks require larger coalitions to place.

TABLE I
AUTONOMOUS CONSTRUCTION EXPERIMENT RESULTS

No. Tasks	Norm. NLP	Norm. Greedy
8	0.944	0.656
12	2.299	1.644
16	1.874	1.311
20	1.843	1.117

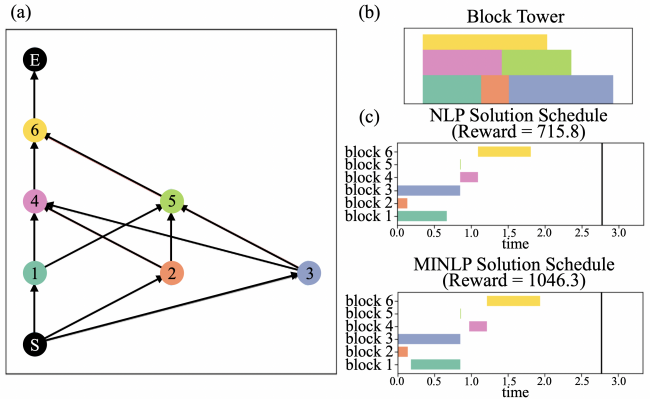

Fig. 7. (a) Generated autonomous construction graph. Start and end nodes are labeled "S" and "E". (b) Corresponding block tower. (c) Construction schedules for NLP (flow) and MINLP solutions. The makespan constraint is set to 2.79, shown as the vertical black line. In this example, the MINLP approach is able to allocate a group of agents to perform task 2 followed by task 1, while the flow-based approaches must divide agents among the edges (0,1), (0,2), and (0,3).

In Table I, we show the results of an experiment conducted with the autonomous construction testbed. We generate block towers with $\{8, 12, 16, 20\}$ blocks, with 5 random trials for each value. We test all three solution methods, and report the rewards of the NLP flow and greedy flow solutions, normalized by the MINLP reward on each trial. Figure 7 illustrates the block tower, corresponding task graph, and the solution schedules of the NLP flow and MINLP solutions for one autonomous construction example.

As this experiment demonstrates, the MINLP approach solves small problems optimally but performs poorly on large problems. The flow-based NLP approach provides high-quality solutions albeit non-optimal but boasts orders of magnitude faster computation. This enables solving much larger problems and makes it well-suited to online applications, which we will explore in future work.

VII. Conclusion

In this paper, we presented a framework that models coordination and cooperation in missions consisting of inter-dependent multi-robot tasks. Our task graph formulation, coupled with expressive task precedence relationships and reward-based coalition models, enabled the automatic generation of task plans in such complex missions. We presented flow-based solution approaches that use a coalition fraction to scale to problems with any number of agents. These approaches perform faster and better than the mixed integer approach on large problem sizes by leveraging the fundamental graph structure of the task allocation problem.

REFERENCES

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

[2] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. SIAM, 2012.

[3] R. A. Knepper, T. Layton, J. Romanishin, and D. Rus, "Ikeabot: An autonomous multi-robot coordinated furniture assembly system," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 855–862.

[4] W. Mao, Z. Liu, H. Liu, F. Yang, and M. Wang, "Research progress on synergistic technologies of agricultural multi-robots," *Applied Sciences*, vol. 11, no. 4, p. 1448, 2021.

[5] Y. Deng, Y. Hua, N. Napp, and K. Petersen, "A compiler for scalable construction by the termes robot collective," *Robotics and Autonomous Systems*, vol. 121, p. 103240, 2019.

[6] M. Dogar, A. Spielberg, S. Baker, and D. Rus, "Multi-robot grasp planning for sequential assembly operations," *Autonomous Robots*, vol. 43, no. 3, pp. 649–664, 2019.

[7] M. C. Gombolay, R. J. Wilcox, and J. A. Shah, "Fast scheduling of robot teams performing tasks with temporospatial constraints," *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 220–239, 2018.

[8] E. Nunes, M. Manner, H. Mitiche, and M. Gini, "A taxonomy for task allocation problems with temporal and ordering constraints," *Robotics and Autonomous Systems*, vol. 90, pp. 55–70, 2017, special Issue on New Research Frontiers for Intelligent Autonomous Systems.

[9] A. W. Kolen, A. Rinnooy Kan, and H. W. Trienekens, "Vehicle routing with time windows," *Operations Research*, vol. 35, no. 2, pp. 266–273, 1987.

[10] D. Bredström and M. Rönnqvist, "Combined vehicle routing and scheduling with temporal precedence and synchronization constraints," *European Journal of Operational Research*, vol. 191, no. 1, pp. 19–31, 2008.

[11] R. Brafman and U. Zoran, "Distributed heuristic forward search for multi-agent systems," in *ICAPS DMAP workshop*, 2014, pp. 1–6.

[12] S. Shekhar, R. I. Brafman, and G. Shani, "Signaling in contingent multi-agent planning," in *Workshop on Epistemic Planning (EpiP)*, 2020.

[13] V. Tereshchuk, N. Bykov, S. Pedigo, S. Devasia, and A. G. Banerjee, "A scheduling method for multi-robot assembly of aircraft structures with soft task precedence constraints," *Robotics and Computer-Integrated Manufacturing*, vol. 71, p. 102154, 2021.

[14] A. J. Smith, G. Best, J. Yu, and G. A. Hollinger, "Real-time distributed non-myopic task selection for heterogeneous robotic teams," *Autonomous Robots*, vol. 43, no. 3, pp. 789–811, 2019.

[15] S. Ponda, J. Redding, H.-L. Choi, J. P. How, M. Vavrina, and J. Vian, "Decentralized planning for complex missions with dynamic communication constraints," in *Proceedings of the 2010 American Control Conference*. IEEE, 2010, pp. 3998–4003.

[16] Z. Wang, C. Liu, and M. Gombolay, "Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints," *Autonomous Robots*, vol. 46, no. 1, pp. 249–268, 2022.

[17] G. A. Korsah, M. B. Dias, and A. Stentz, "A comprehensive taxonomy for multi-robot task allocation background," *The International Journal of Robotics Research*, pp. 1–29, 2013.

[18] N. Seenu, K. C. RM, M. Ramya, and M. N. Janardhanan, "Review on state-of-the-art dynamic task allocation strategies for multiple-robot systems," *Industrial Robot: The International Journal of Robotics Research and Application*, vol. 47, no. 6, pp. 929–942, 2020.

[19] A. Prorok, M. A. Hsieh, and V. Kumar, "The impact of diversity on optimal control policies for heterogeneous robot swarms," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 346–358, 2017.

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

[21] A. Messing, G. Neville, S. Chernova, S. Hutchinson, and H. Ravichandar, "GRSTAPS: Graphically recursive simultaneous task allocation, planning, and scheduling," *The International Journal of Robotics Research*, vol. 41, no. 2, pp. 232–256, 2022.

[22] L. Capezzuto, D. Tarapore, and S. Ramchurn, "Anytime and efficient coalition formation with spatial and temporal constraints," in *Multi-Agent Systems and Agreement Technologies*, N. Bassiliades, G. Chalkiadakis, and D. de Jonge, Eds. Cham: Springer International Publishing, 2020, pp. 589–606.

[23] S. D. Ramchurn, M. Polukarov, A. Farinelli, C. Truong, and N. R. Jennings, "Coalition formation with spatial and temporal constraints," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 3*, ser. AAMAS '10. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 1181–1188.

[24] 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-schedule dependencies," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 115–122.

[25] A. Prorok, M. Malencia, L. Carlone, G. S. Sukhatme, B. M. Sadler, and V. Kumar, "Beyond robustness: A taxonomy of approaches towards resilient multi-robot systems," *arXiv preprint arXiv:2109.12343*, 2021.

[26] A. Dutta and A. Asaithambi, "One-to-many bipartite matching based coalition formation for multi-robot task allocation," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2181–2187.

[27] F. Zitouni, R. Maamri, and S. Harous, "FA–QABC–MRTA: a solution for solving the multi-robot task allocation problem," *Intelligent Service Robotics*, vol. 12, no. 4, pp. 407–418, 2019.

[28] G. Gens and E. Levner, "Complexity of approximation algorithms for combinatorial problems: a survey," *ACM SIGACT News*, vol. 12, no. 3, pp. 52–65, 1980.

[29] Scipy, "Docs.scipy.org." 2022, date accessed: 2022-05-10. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html#scipy-optimize-minimize

[30] S. J. Maher, M. Miltenberger, J. P. Pedroso, D. Rehfeldt, R. Schwarz, and F. Serrano, "Pyscipopt: Mathematical programming in python with the scip optimization suite," in *Mathematical Software - ICMS 2016*, vol. 9725, 2016, pp. 301 – 307.

[31] K. Bestuzheva, M. Besançon, W.-K. Chen, A. Chmiela, T. Donkiewicz, J. van Doornmalen, L. Eifler, O. Gaul, G. Gamrath, A. Gleixner, L. Gottwald, C. Graczyk, K. Halbig, A. Hoen, C. Hojny, R. van der Hulst, T. Koch, M. Lübbecke, S. J. Maher, F. Matter, E. Mühmer, B. Müller, M. E. Pfetsch, D. Rehfeldt, S. Schlein, F. Schlösser, F. Serrano, Y. Shinano, B. Sofranac, M. Turner, S. Vigerske, F. Wegscheider, P. Wellner, D. Weninger, and J. Witzig, "The SCIP Optimization Suite 8.0," Optimization Online, Technical Report, December 2021. [Online]. Available: http://www.optimization-online.org/DB_HTML/2021/12/8728.html