2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Inverse Submodular Maximization with Application to Human-in-the-Loop Multi-Robot Multi-Objective Coverage Control

Guangyao Shi, Gaurav S. Sukhatme

***Abstract*— We consider a new type of inverse combinatorial optimization, Inverse Submodular Maximization (ISM), for human-in-the-loop multi-robot coordination. Forward combinatorial optimization - solving a combinatorial problem given the reward (cost)-related parameters - is widely used in multi-robot coordination. In the standard pipeline, the reward (cost)-related parameters are designed offline by domain experts. These parameters are utilized for coordinating robots online. What if non-expert human supervisors desire to change these parameters during task execution to adapt to some new requirements? We are interested in the case where human supervisors can suggest what actions to take, and the robots need to change these internal parameters accordingly. We study such problems from the perspective of inverse combinatorial optimization, i.e., the process of finding parameters given solutions to the problem. Specifically, we propose a new formulation for ISM, in which we aim to find a new set of parameters that minimally deviate from the current parameters while causing a greedy algorithm to output actions which are the same as those desired by the human supervisors. We show that such problems can be formulated as a Mixed Integer Quadratic Program (MIQP) which is intractable for existing solvers when the problem size is large. We propose a new Branch & Bound algorithm to solve such problems. In numerical simulations, we demonstrate how to use ISM in multi-robot multi-objective coverage control, and we show that the proposed algorithm provides significant advantages in running time and peak memory usage compared to directly using an existing solver.**

I. Introduction

Multi-robot teams are widely used in information gathering, environment monitoring, exploration, and target tracking [1], [2]. Many such multi-robot decision-making problems can be cast as combinatorial optimization problems and are usually NP-hard [3]. In combinatorial problems for multi-robot decision-making, many objectives (e.g., mutual information [4], [5], area coverage [6], target tracking [7]–[9], detection probability [10] etc.) have a diminishing returns property, i.e., submodularity. Intuitively, submodularity formalizes the notion that adding a robot to a larger multi-robot team will yield a smaller marginal gain in the objective than adding the same robot to a smaller team. We are interested in considering maximizing such submodular objectives, which is NP-hard but can be solved with an $(1-\frac{1}{e})$-approximation by a greedy algorithm [11].

Classically, experts carefully design the optimization objective offline. For example, as shown in Fig. 1, in a multi-robot event detection application, experts will use the event data observed in the past and their preferences about events (e.g., how important/urgent each event is) to design an objective. The design contains several parameters related to the prior knowledge of the events and user preferences. Subsequently, robots are deployed for event detection by solving the optimization problem online. However, when robot teams are deployed in the field a human supervisor may receive additional information from external sources or extract additional information based on their experience and the current observations and suggest possibly better and different actions to take compared those obtained from solving the original optimization problem. Such suggestions reflect either an oversight in the design phase or the arrival of new information/instructions. In such a case, it is undesirable to stop the team and redesign the decision-making process. Moreover, the human supervisor in the loop may not be an optimization expert, and they may not know how to recast the optimization problem to accommodate their new insights. By contrast, we want the robot team to have the capability to modify the decision-making problem in a *minimal* way to accommodate new supervisory suggestions. The reason for the minimal change is that the decision-making problem itself is already the embodiment of much expertise and historical data. The technical path that we explore here to equip robots with such capabilities is to add a new adaptation component to decision-making. The core of this

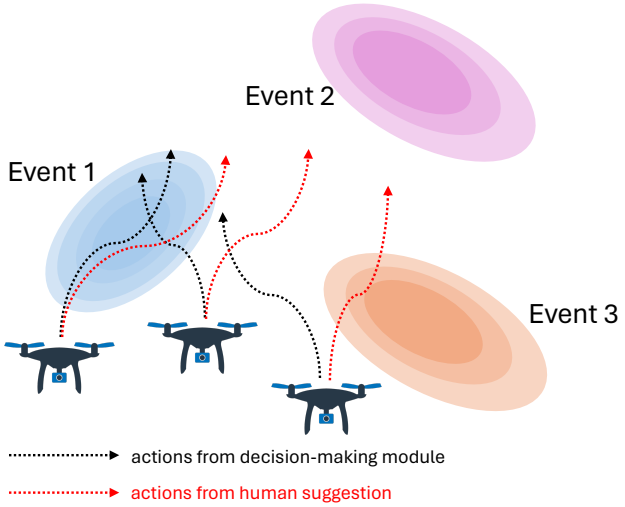


Fig. 1. A motivating example of inverse submodular maximization. A team of robots is deployed to detect multiple events, each of which is associated with a priority, and the task is cast as a submodular maximization problem. Black dotted lines are actions derived from task optimization and red dotted lines are actions suggested by humans. The team needs to minimally adjust the submodular objective to account for human suggestions.

University of Southern California, Los Angeles, LA 90007 USA shig, gaurav@usc.edu. GSS holds concurrent appointments as a Professor at USC and as an Amazon Scholar. This paper describes work performed at USC and is not associated with Amazon.

new component is the Inverse Submodular Maximization (ISM) problem.

Specifically, for a multi-robot coordination problem with a parameterized submodular objective, e.g, [12], [13], if the parameters are known, we can use a greedy approximation algorithm and its variants to solve the problem and obtain a (near)-optimal solution. We call such a process Forward Submodular Maximization (FSM). By contrast, if we have a solution obtained using an approximation algorithm, we want to find the corresponding parameters in the objective such that when we use those parameters to solve the FSM problem, the resulting solutions match the known solutions. We call this process Inverse Submodular Maximization. ISM problems fall into the more general class of inverse optimization problems, which are widely used to decode human decision-making processes [14]–[16] for robotic applications.

Here, we study such ISM problems for multi-robot coordination, which are formulated as FSM problems, with human suggestions. This is the first ISM formulation for a multi-robot coordination problem. We note that in the optimization literature, the ISM problem is not well explored and there is no existing published canonical formulation of ISM. In addition to the ISM framing, we introduce a new algorithm under the Branch and Bound (BB) paradigm to solve the ISM problem.

The main contributions of this paper are:

- We formulate a new type of inverse optimization problem, ISM, and propose a novel algorithm for its solution.
- We show how to use ISM in multi-robot coordination to account for human suggestions.
- On the application side, we propose a multi-robot multi-objective coverage objective for event detection and prove its submodularity.

II. Related Work

Inverse Optimization Inverse optimization is gaining increasing attention over the past years and is studied for a diverse of applications, including vehicle routing [17], [18], power system [19], [20], and robotics [21]–[23]. Based on the nature of the forward optimization problem, we can roughly classify inverse problems into two categories: continuous case and discrete case. Most existing work related to robotics fits into the continuous category. Researchers use Inverse Optimal Control (IOC) or Inverse Reinforcement Learning (IRL) to refer to inverse optimization. IOC/IRL is usually studied to learn the human behaviors for one robot, for example, human walking for humanoid locomotion and human grasping skills for robot arm manipulation. Extending frameworks of IOC/IRL from one robot to multiple robots is not a trivial task and research along this line is still less mature [24]–[26]. By contrast, the discrete case is less considered in the robotics literature. The discrete case corresponds to combinatorial optimization, which is widely used in multi-robot coordination, in the forward problem. In the optimization community, such a discrete inverse problem is referred to as Inverse Combinatorial Optimization (ICO) [27], among which a subbranch called the Inverse Integer Optimization (IIO) is widely studied [28]. However, IIO can not deal with a submodular objective. The ISM formulations proposed in this paper can be viewed as another subbranch of ICO and this is the first paper to study such problems in the context of multi-robot coordination.

Human Preference Learning ISM is also related to the research on human preference learning. In these works [29]–[31], humans are usually given two solutions iteratively and need to compare these two solutions for each iteration. Most research focuses on how to efficiently generate queries for humans to learn the preference parameters. By contrast, in ISM, humans are present with a ground set and they will select a subset that they think the *approximation algorithm* should output. The research focus is how to minimally modify the parameters in the objective to match the human suggestions. There is no iterative process.

III. Preliminaries

We use calligraphic fonts to denote sets (e.g. $\mathcal{A}$). Given a set $\mathcal{A}$, $2^{\mathcal{A}}$ denotes the power set of $\mathcal{A}$ and $|\mathcal{A}|$ denotes the cardinality of $\mathcal{A}$. Given another set $\mathcal{B}$, the set $\mathcal{A} \setminus \mathcal{B}$ denotes the set of elements in $\mathcal{A}$ but not in $\mathcal{B}$. We will use $\Delta f(s \mid \mathcal{S})$ to denote the marginal gain of adding an element s to a set $\mathcal{S}$, i.e., $\Delta f(s \mid \mathcal{S}) = f(\{s\} \cup \mathcal{S}) - f(\mathcal{S})$. We will denote $\|\cdot\|$ as the ℓ^2-norm. A set is an ordered set if it can preserve the order in which we insert the elements. For an ordered set $\mathcal{S}$, we use $\mathcal{S}[i]$ and $\mathcal{S}[1:i]$ to refer to its i-th element and its first i elements, respectively. For an unordered set, we call each permutation of the set as an *ordering* of the set.

Definition 1 (Submodularity)**.** For a finite ground set $\mathcal{V}$, a function $f : 2^{\mathcal{V}} \mapsto \mathbb{R}$ is submodular if the following condition is satisfied: for every $\mathcal{A}, \mathcal{B} \subseteq \mathcal{V}$ with $\mathcal{A} \subseteq \mathcal{B}$ and every $v \in \mathcal{V} \setminus \mathcal{B}$ we have that

$$f(\mathcal{A} \cup \{v\}) - f(\mathcal{A}) \geq f(\mathcal{B} \cup \{v\}) - f(\mathcal{B}). \tag{1}$$

Similarly, if the LHS of Eq. (1) is always no greater than the RHS of Eq. (1), the function f is supermodular.

A. Forward Submodular Maximization

We are interested in maximizing a monotone submodular function. Mathematically, we want to solve the following optimization problem:

$$\max_{\mathcal{S} \subseteq \mathcal{T}} f(\mathcal{S}, \boldsymbol{\theta}), \tag{2}$$

where $\boldsymbol{\theta}$ is the parameter vector of the objective; $\mathcal{T}$ is the ground set.

As an initial trial to solve the inverse version of submodular maximization, we will confine our discussion to the case where $f(\mathcal{S}, \boldsymbol{\theta})$ is a linear combination of submodular basis functions [10], [32], i.e.,

$$f(\mathcal{S}, \boldsymbol{\theta}) = \boldsymbol{\theta}^T g(\mathcal{S}), \tag{3}$$

where $g(\mathcal{S})$ is a monotone submodular function. We leave the more general case for future work.

The forward problem in Eq. (2) can be solved nearly optimally using the greedy algorithm [11] as shown in Algorithm 1.

Algorithm 1: Greedy Algorithm

Input :
- A monotone submodular function f
- A matroid $\mathcal{M} = (\mathcal{T},\ \mathcal{I})$

Output: A subset $\mathcal{S} \in \mathcal{I}$ of the ground set $\mathcal{T}$

$\mathcal{S} \leftarrow \emptyset$
while $|\mathcal{S}| < N$ **do**
  # find the element with the largest marginal gain
  $s = \text{argmax}_{s \in \mathcal{T}, \{s\} \cup \mathcal{S} \in \mathcal{I}}\ \Delta f(s \mid \mathcal{S})$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{s\}$
end
return $\mathcal{S}$

IV. Problem Formulation

Problem 1 (Inverse Submodular Maximization (ISM))**.** *Given a suggestion set $\hat{\mathcal{S}} \subseteq \mathcal{T}$ from human, find a new parameter vector $\hat{\boldsymbol{\theta}}$ such that the distance between the original parameter vector $\boldsymbol{\theta}$ and the new parameter vector $\hat{\boldsymbol{\theta}}$ is minimized and the human suggestion $\hat{\mathcal{S}}$ becomes the solution for the new problem using $\hat{\boldsymbol{\theta}}$. Mathematically, the inverse problem can be formulated as:*

$$\min_{\hat{\boldsymbol{\theta}}} \ \|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\| \tag{4}$$

$$\text{s.t.} \quad \mathcal{S}(\hat{\boldsymbol{\theta}}) = \underset{\mathcal{S}}{\text{argmax\_greedy}}\ f(\mathcal{S}, \hat{\boldsymbol{\theta}}), \tag{5}$$

$$\hat{\mathcal{S}} = \mathcal{S}(\hat{\boldsymbol{\theta}}), \tag{6}$$

where $\mathcal{S}(\hat{\boldsymbol{\theta}})$ is the solution returned by the greedy algorithm and the constraint (6) *enforces the solution to be equal to the human suggestion.*

ISM formulations presented in this section can be used as a complementary part of the existing multi-robot coordination framework based on submodular maximization to accommodate human suggestions.

A. Case Study: Multi-Robot Coverage Control

We present a case study on multi-robot multi-objective coverage control that motivates our research. The goal is to coordinate the motion of robots to detect multiple events as shown in Fig. 1. Such a problem can be formulated as a maximization problem with a submodular objective, built on the probability of detecting stochastic events. Such a submodular objective is usually designed offline based on human preferences and prior knowledge of the events. However, the preferences for detecting different events may change when the human supervisor receives the new information online or some external instructions (e.g., detect event i and j as soon as possible). In such cases, the human supervisors may give a suggestion on which actions the robot team should take. Once receiving such suggestions, the robot team will solve the ISM problem to minimally change the objective to adapt to the new suggestion. It should be noted that such suggestions are given only occasionally, not at every planning cycle. When there are no suggestions from humans, the robot team will be coordinated by solving the forward optimization problems.

Specifically, there is a team of robots $\mathcal{R} = \{1, \ldots, n\}$ in the task environment. Each robot has the sensing capability to detect events and we want to coordinate the team to collectively detect independent events $\mathcal{E} = \{1, \ldots, m\}$ in the environment. Each robot has a set of available actions, each of which spans a planning horizon of length H. We consider the action selection problem for the team.

Environment and Event Models The task environment $\Omega \subset \mathbb{R}^2$ is a polygon embedded in the 2-dimensional space. Each event $j \in \mathcal{E}$ is associated with a *event density* function $\phi_j(x) : \Omega \to \mathbb{R}_+$, which describes the frequency or density of the occurrences of a particular stochastic event $j \in \mathcal{E}$ at the location x. Depending on applications, $\phi_j(x)$ can be the frequency that a target shows at x, or it can be the probability that some physical quantities, e.g., temperature, humidity, exceed some threshold [33]. Generally, $\phi_j(x)$ needs to satisfy two conditions: $\phi_j(x) \geq 0$ and $\int_\Omega \phi_j(x)dx < \infty$. In this paper, we are interested in a particular case where ϕ_j is a probability distribution over Ω and $\int_\Omega \phi_j(x)dx = 1$. When we select actions for robots, we need to consider all the events in $\mathcal{E}$. To achieve this, we give each event j an importance factor $\theta_j \in \mathbb{R}_+$, which are designed offline, and use a weighted sum approach to balance multiple events. We assume that within a planning horizon H the density function $\phi_j(x)$ is time-invariant.

Robots and Sensing The position of the robot $i \in \mathcal{R}$ is denoted as $p_i \in \Omega$. Each robot carries onboard sensors and has a sensing radius δ. Therefore, the sensing region of the robot i is $\Omega_i = \{x \in \Omega \mid \|x - p_i\| \leq \delta\}$. For a planning horizon H, an *action* a, is defined as a sequence of positions that the robot will reach in order, i.e., $a = [p_i(t = k), \ldots, p_i(t = k+H)]$. We assume that a lower-level planner can generate a set of feasible actions for robots. The sensing model of robot i is given as $\Pr(x, p_i)$, the probability that the robot i can detect the event occurrences at $x \in \Omega_i$. If a point is not in the sensing region, the probability is zero. If it is within the sensing region, we assume that $\Pr(x, p_i)$ is a function of the distance between x and p_i, i.e., $\|x - p_i\|$, and is monotonically decreasing and differentiable. Overall, the sensing model of the robot that we use here is

$$\Pr(x, p_i) = \begin{cases} \exp(-\lambda_i \|x - p_i\|) & \text{if } x \in \Omega_i(p_i) \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where λ_i is a decay factor for sensing.

Combining all the sensing results from all robots, we can compute the joint detection probability that an event at $x \in \Omega$ is detected by the team at time step $t = k$ as

$$\Pr(x, \boldsymbol{p}(k)) = 1 - \prod_{i=1}^{n} (1 - \Pr(x, p_i(k))), \tag{8}$$

where $\boldsymbol{p}(k) = [p_1(k), \ldots, p_n(k)]^T$ denotes the positions vector of all robots at time k.

Objective Function For time step k, we define the event

coverage objective of the event j, as defined in [34], as

$$h_j(\boldsymbol{p}(k)) = \int_{\Omega} \phi_j(x)\text{Pr}(x, \boldsymbol{p}(k))dx. \tag{9}$$

When ϕ_j is a probability distribution over Ω, Eq. (9) can be viewed as the probability of detecting the event j if the robot team senses at $\boldsymbol{p}(k)$. Therefore, $1 - h_j(\boldsymbol{p}(k))$ is the probability of not detection the event j at time k. We assume the sensing at each time step is independent. The probability of not detection the event j for all H steps is

$$\prod_{t=k}^{t=k+H} 1 - h_j(\boldsymbol{p}(t)). \tag{10}$$

As a result, the detection probability of an event j in the whole planning horizon is $1 - \prod_{t=k}^{t=k+H} 1 - h_j(\boldsymbol{p}(t))$. There are m such events and each is associated with parameter θ_j describing its priority. Overall, the coverage objective for the team and all events over the whole planning horizon can be expressed as:

$$f(\mathcal{A}, \boldsymbol{\theta}) = \sum_{j=1}^{n} \theta_j (1 - \prod_{t=k}^{t=k+H} 1 - h_j(\boldsymbol{p}(t))), \tag{11}$$

where $\mathcal{A} = \{a_1, \ldots, a_n\}$ is the selected action set for the team. $a_i = [p_i(t = k), \ldots, p_i(t = k + H)]$, consisting of a sequence of positions, is the action for the robot i. $\mathcal{A}$ can be viewed as a $n \times H$ matrix and $\boldsymbol{p}(t)$ is a column. $\theta_j \in \mathbb{R}_+$ is the importance factor for the event j.

Theorem 1. *The objective defined in Eq.* (11) *is monotone submodular.*

The proof is given in the Appendix.

V. Algorithm for ISM

Let us first consider the case where the human suggestion $\hat{\mathcal{S}}$ is an ordered set. Based on Algorithm 1, at each step, the element with the largest marginal gain will be selected. If $\hat{\mathcal{S}}$ is an ordered set and is the output of the Algorithm 1, for each prefix $\hat{\mathcal{S}}[1:i]$, it should satisfy the following inequality based on line 3 in Algorithm 1:

$$\begin{aligned} f(\hat{\mathcal{S}}[1:i], \hat{\boldsymbol{\theta}}) - f(\hat{\mathcal{S}}[1:i-1], \hat{\boldsymbol{\theta}}) \geq \\ f(\hat{\mathcal{S}}[1:i-1] \cup \{s\}, \hat{\boldsymbol{\theta}}) - f(\hat{\mathcal{S}}[1:i-1], \hat{\boldsymbol{\theta}}), \\ \forall s \in \{s \in \mathcal{T} \setminus \hat{\mathcal{S}}[1:i] \mid \{s\} \cup \hat{\mathcal{S}}[1:i-1] \in \mathcal{I}\}. \end{aligned} \tag{12}$$

The left side of the inequality is the marginal gain of adding an element $\hat{\mathcal{S}}[i]$ to the ordered set $\hat{\mathcal{S}}[1:i-1]$. The right side of the equality of the marginal gain of adding other feasible elements s. Intuitively, each $\hat{\mathcal{S}}[i] \in \hat{\mathcal{S}}$ should be the one with the largest marginal gain in the i-th selection step.

For the case that f is a linear function w.r.t. $\hat{\boldsymbol{\theta}}$ as shown in Eq. (3), each inequality (12) is a linear inequality:

$$\begin{aligned} \hat{\boldsymbol{\theta}}^T g(\hat{\mathcal{S}}[1:i]) - \hat{\boldsymbol{\theta}}^T g(\hat{\mathcal{S}}[1:i-1]) \geq \\ \hat{\boldsymbol{\theta}}^T g(\hat{\mathcal{S}}[1:i-1] \cup \{s\}) - \hat{\boldsymbol{\theta}}^T g(\hat{\mathcal{S}}[1:i-1]). \end{aligned} \tag{13}$$

There will be $O(|\hat{\mathcal{S}}| \cdot |\mathcal{T}|)$ such constraints. Combining all the linear inequalities, the Ordered-set variant of Problem 1 boils down to a convex optimization problem as follows.

Problem 2 (Ordered-ISM (O-ISM))**.**

$$\min_{\hat{\boldsymbol{\theta}}} \ \|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\| \tag{14}$$

$$\text{s.t.} \quad \hat{\boldsymbol{\theta}}^T b_j \leq 0, \forall j \tag{15}$$

where b_j denotes a coefficient vector corresponding to Eq. (13)*; j is the index for all linear inequalities.*

It should be noted that we can define an O-ISM problem for any ordered set other than $\hat{\mathcal{S}}$. In the rest of this paper, we will treat O-ISM as a function: for an ordered set $\mathcal{A}$ which is assumed to be returned from the greedy algorithm for some parameter $\hat{\boldsymbol{\theta}}$, O-ISM($\mathcal{A}$) denotes the problem instance as defined in Problem 2 for the set $\mathcal{A}$. We will use this in the Algorithm 2.

For the case where the human suggestion $\hat{\mathcal{S}}$ is not an ordered set, each possible ordering corresponds to a set of constraints like Eq. (15). Such a case is more practical in applications: human operators know the solutions based on their expertise and observation but they do not have the concept of ordering of a solution set. All these sets of constraints can be connected using OR logic similar to disjunctive inequalities. We can use the Big-M reformulation technique to formulate the problem as a Mixed Integer Quadratic Programming (MIQP) problem.

Problem 3 (Unordered-ISM (U-ISM))**.**

$$\min_{\hat{\boldsymbol{\theta}}} \ \|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\| \tag{16}$$

$$\text{s.t.} \quad \hat{\boldsymbol{\theta}}^T b_j^k \leq M(1 - y_k), \ \forall j, \ \forall k \tag{17}$$

$$\sum_k y_k = 1, \quad y_k \in \{0, 1\}, \tag{18}$$

where k is the index for the possible ordering of $\hat{\mathcal{S}}$; j is the index for all linear inequalities corresponding to an ordering; y_k is a binary variable to indicate which ordering constraint is active; M is a large enough positive number.

In Problem 3, we add a binary variable y_k to indicate whether an ordering is active. If $y_k = 1$, then the inequalities for k-th ordering is active in Eq. (17), i.e., the RHS of Eq. (17) is zero. If $y_k = 0$, the RHS of Eq. (17) is a large number M, the inequalities are trivially satisfied. We enforce that there should be only one active ordering in Eq. (18).

It should be noted that there are exponentially many constraints and integer variables in Eq. (17). The brute-force way to solve the problem is to explicitly list all the constraints and use an existing solver, e.g., Gurobi, to find the solution. However, it is both time-consuming and memory-consuming to explicitly list all the constraints. Besides, we will lose structure information about these binary variables, i.e., they correspond to the orderings of a set. Therefore, we develop a branch and bound algorithm to solve this type of problem without explicitly listing all the constraints. The main idea is that instead of treating each possible ordering of $\hat{\mathcal{S}}$ as one candidate and searching for the best one, we incrementally add elements to form a sequence of relaxed problems. By keeping track of the solutions from these

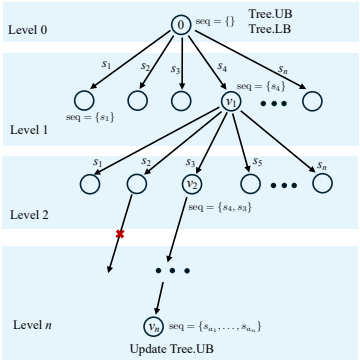


Fig. 2. One iteration of the proposed BB-ISM algorithm.

relaxed problems, we can gradually find the upper and lower bounds of the objective of the original problem, and prune the suboptimal solution. The incremental search is conducted by growing a search tree in the depth-first-search fashion. An illustrative example is shown in Fig. 2. At the root node, we start with an empty sequence. For the next level (level 1) of the tree, we can add any elements in $\hat{\mathcal{S}}$ to the sequence to form a new node. For each such node, we will solve a problem O-ISM(*seq*) using the *seq* property of the node. When the *seq* includes only some of elements in $\hat{\mathcal{S}}$, the objective value returned by solving O-ISM(*seq*) can be viewed as the lower bound of all cases where the orderings of $\hat{\mathcal{S}}$ start with *seq* since the ordering with all elements implies more constraints.

Then, we will choose the node (node v_1 in Fig. 2) with the smallest objective value (returned by solving O-ISM(*seq*)) to further expand on that node. Here, we use the objective value of O-ISM(*seq*) as a heuristic to guide search and use a greedy strategy to expand to the next level. The expansion is similar to that from level 0 to level 1. There is only one element s_4 in the *seq* of the node v_1. So we can add any elements in $\hat{\mathcal{S}} \backslash \{s_4\}$ to the sequence to form a new child node on level 2 and solve a corresponding problem O-ISM(*seq*). Such a process will repeat until the *seq* of the node includes all elements in $\hat{\mathcal{S}}$, which means that an ordering of $\hat{\mathcal{S}}$ is found. We will use the objective value, returned by solving O-ISM(*seq*), of this node to update the upper bound of the problem (Tree.UB in the root node) since this represents the solution for a particular ordering of $\hat{\mathcal{S}}$ and the optimal solution corresponding to optimal orderings should achieve a lower objective value. After that, we will continue to grow the tree by going back to the previous level of the node and expanding similar to Depth First Search (DFS). When we grow the search tree, if a particular O-ISM(*seq*) problem is infeasible, which implies all the orderings with a prefix *seq* are infeasible, or the objective value returned from O-ISM(*seq*) is higher than the upper bound of the tree identified so far, which implies all the orderings with a prefix *seq* will not result in a better solution identified so far, we will prune that branch as shown in Fig. 2 (the red cross prunes the branch). Such pruning operation will accelerate the search.

The details are shown in the Algorithm 2. In lines 1-3, we initialize an empty tree with a root node whose *seq* property is an empty ordered set. The upper bound property is initialized to a larger number. This value will be updated as we grow the search tree. In lines 4-5, we initialize a stack for a DFS-style search and push the root node into the stack. In the while loop, we first pop the top element from the stack (line 7) and initialize a priority queue (line 8). Then we will check each element that is in $\hat{\mathcal{S}}$ but not in u.seq (line 9) whether it is feasible to solve a relaxed problem O-ISM if we append this element to the existing sequence (line 10). If it is not feasible, it means that all sequences with such a prefix are not possible. We can prune this branch by ignoring it. If it is feasible, we will insert this element s and the corresponding $\hat{\boldsymbol{\theta}}$ using the returned objective value as the priority value. After this, we will start to update the search tree (lines 15-23). We expand branches to the search tree in the increasing order of the objective value (line 16). For each candidate, we

Algorithm 2: BB for ISM (BB-ISM)

Input : Problem instance with human suggestion $\hat{\mathcal{S}}$
Output: $\hat{\boldsymbol{\theta}}$

1 Tree ← empty tree # Initialize a search tree
2 Tree.UB ← a large number
3 Tree.add_node(node_ID = 0, sequence = {})
  # Initialize an empty stack for Depth First Search
4 Stack ← empty stack
5 Stack.push(Tree.get_node(node_ID = 0))
6 **while** *Stack is not empty* **do**
7 u ← Stack.pop()
8 PQ ← priority_queue()
9 **for** $s \in \hat{\mathcal{S}} \setminus u.\text{seq}$ **do**
10 feasible, obj, $\hat{\boldsymbol{\theta}}$ ← O-ISM($u.\text{seq} + \{\text{s}\}$)
11 **if** *feasible* **then**
12 PQ.insert(s, $\hat{\boldsymbol{\theta}}$, priority_value=obj)
13 **end**
14 **end**
15 **while** *PQ is not empty* **do**
16 s, $\boldsymbol{\theta}$, obj ← PQ.pop()
17 **if** *obj* $<$ *Tree.UB* **then**
18 Tree.update_UB(u.seq+$\{s\}$, obj)
19 new_node ← Tree.add_branch(u, s, obj, $\boldsymbol{\theta}$)
20 Stack.push(new_node)
21 **end**
22 **end**
23 **end**
24 **return** $tree$
25 **Function** `update_UB` ($Tree$, *sequence, obj*) **:**
26 **if** *length of sequence* = *length of* $\hat{\mathcal{S}}$ **then**
27 **if** *obj* $<$ $Tree$*.UB* **then**
28 $Tree$.UB ← obj
29 $Tree$.UB_seq ← $sequence$
30 **end**
31 **end**
32 **end**

first check whether we should prune it or not (line 17) by comparing the objective value with the upper bound of the tree. If the objective value is greater than the upper bound, we should prune branches with such sequences as the prefix. This is because the relaxed problem O-ISM returns the lower bound of the objective value for all the ordering of $\hat{\mathcal{S}}$ with a prefix u.seq. If we use the whole ordering, which means that more constraints are added, the objective will be greater than this objective value. The upper bound of the tree represents the best solution found so far. If other orderings generate solutions worse than the identified ones, we should prune all of them. By contrast, if we should not prune this branch (line 17), we should update the upper bound of the search tree and add the new branch to the tree (line 19). To keep a DFS search style, we need to push the new nodes to the stack (line 20). After considering all elements in the priority queue, we will continue with the while loop: pop the element on the top of the stack (line 7) and repeat such a process.

Theorem 2. *Given a feasible problem instance as described in Problem* (3)*, Algorithm 2 returns the optimal solution in finite iterations of the outer while loop (lines 6-24).*

Like all the algorithms that are developed within the BB paradigm, the BB-ISM algorithm is in nature enumerating all the possible combinations by incrementally adding elements. The efficiency relies on the pruning steps to remove unnecessary expansion of the search tree.

Remark 1. It should be noted that the problem as described in Problem (3) is an NP-hard problem in general (mixed integer quadratic programming). In the worst case, the proposed algorithm will have to enumerate all the possible orderings of the human suggestion to find the optimal solution or find that the problem is infeasible. We will experimentally compare BB-ISM with the brute-force approach using Gurobi in Section VI.

VI. Experiments

We validate the proposed ISM formulation and evaluate the BB-ISM algorithm in a case study on the multi-robot multi-objective coverage control as described in Sec. IV-A. We will first present a qualitative example to show how the ISM and human suggestion affect the behaviors of robots. Then, we will evaluate the proposed algorithm in terms of its optimality, running time, and peak memory usage.

A. A Qualitative Example

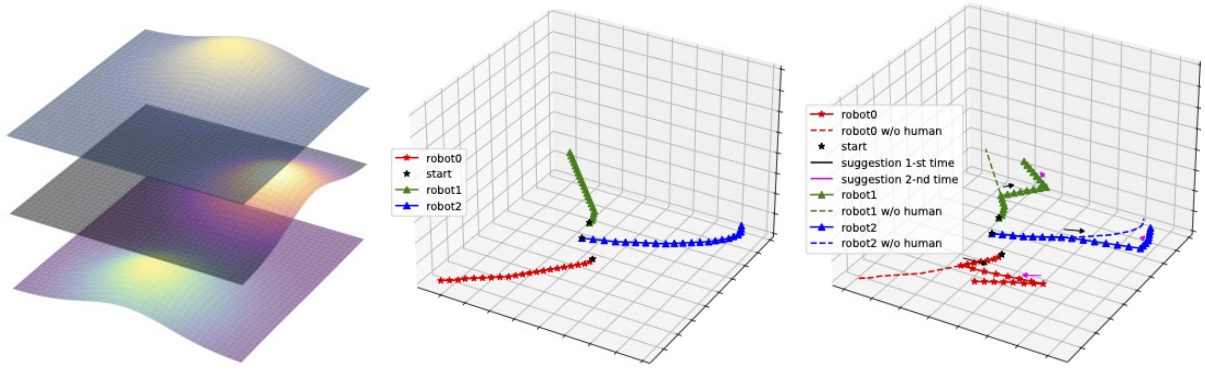
Three stochastic events may happen in the environment Ω. Each event j is associated with a Gaussian event density ϕ_j as shown in Fig. 3a. For each robot, it can move along the radial direction of a circle centered at its current position. Each radial direction denotes an action and each robot has 20 actions. In the offline design phase, the three events are considered to be equally important, i.e., they have the same importance factor in (11). As a result, when we deploy robots for coverage control, they will have trajectories as shown in Fig. 3b, i.e., they will be guided toward the regions with high event density and each will move towards one specific region with high event density. However, if the human thinks some area is more important and suggests the robot choose different actions, the robots will solve ISM to update the importance factors based on such suggestion and change their behavior correspondingly. Such an example is shown in Fig. 3c. When the robots move based on the offline designed objective, the human operator at some time gives suggestions denoted as black arrows as shown in Fig. 3c. After solving ISM, the robots will increase the important factor of the event which has a higher density in the suggested direction. Subsequently, the robots will make decisions using updated importance factors, and the resulting trajectories will change. Suppose after some time, the human operator thinks all events are equally important, and suggests all robots go toward three regions with high event density separately as shown in the magenta arrows in Fig. 3c. The importance factor will be updated again and the trajectories of robots will change subsequently.

B. Algorithm Validation

Optimality We compare two types of baselines with the proposed algorithm. The first is to directly solve Problem 3 using Gurobi. We refer to such a baseline as IP. The second type is to randomly sample a few suggestion orderings and solve Problem 2 for each. Then, select the best solution among all the samples. We denote such a baseline as RS-z, where z refers to the number of samples. We generate test instances in the following way. First, we generate a collection of submodular objectives, each of which is associated with a particular $\boldsymbol{\theta}$. For each $\boldsymbol{\theta}$, we randomly generate some feasible suggestions and solve the ISM for each to obtain a $\hat{\boldsymbol{\theta}}$. To compare results across all $\boldsymbol{\theta}$, we use the normalized deviation $\frac{\|\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}\|}{\|\boldsymbol{\theta}\|}$ as the criterion to compare different approaches. The desired result should make this metric as small as possible. The result is shown in Fig. 4. First, we can observe that across all the test groups, i.e., different dimensions of $\boldsymbol{\theta}$, the proposed algorithm, denoted as BB, achieves the same normalized deviation, which suggests the proposed algorithm returns the optimal solution as the IP does. For RS baselines, they all have higher normalized deviation due to sub-optimality and such optimality can be improved as the number of samples increases.

Running Time We compare the running time of the proposed algorithm (BB) with the integer programming using Gurobi (IP). We compare three groups for different dimensions of $\boldsymbol{\theta}$. The results are shown in Fig. 5. Since the running time of IP will increase too fast after 7 robots, we only plot the results with less than 8 robots for IP. As shown in Fig. 5a, 5b, and 5c, the proposed algorithm (BB) is much more scalable w.r.t. the number of robots and the running time increase relatively slowly in the test cases. Besides, when the dimension of $\boldsymbol{\theta}$ increases, the running time increase is not very significant compared to that of IP.

Peak Memory Usage The results of peak memory usage are shown in Fig. 5d. We can observe the BB approach requires much less memory compared to that of IP as the number of robots increases.



(a) (b) (c)

Fig. 3. A qualitative example to illustrate how ISM can be used in the human-in-the-loop multi-robot multi-objective coverage control. (a) Three event density functions. (b) Robot trajectories without human suggestions. (c) Robot trajectories with human suggestions.

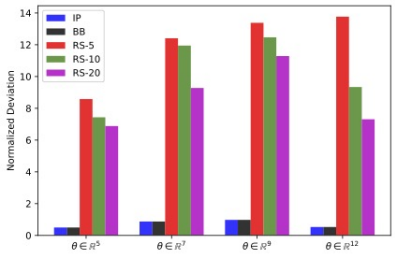
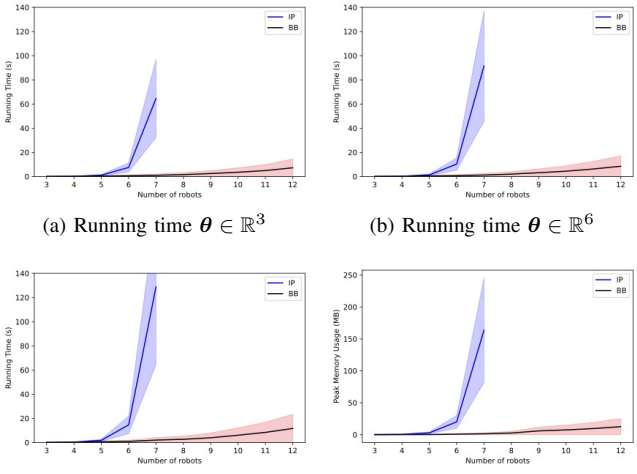


Fig. 4. Optimality comparisons with baselines.



(a) Running time $\boldsymbol{\theta} \in \mathbb{R}^3$ (b) Running time $\boldsymbol{\theta} \in \mathbb{R}^6$

(c) Running time $\boldsymbol{\theta} \in \mathbb{R}^9$ (d) Peak memory $\boldsymbol{\theta} \in \mathbb{R}^9$

Fig. 5. Running time and peak memory usage comparisons with baselines.

VII. Conclusion

We consider a new type of inverse combinatorial optimization problem - inverse submodular optimization (ISM)- for human-in-the-loop multi-robot coordination. We show that such problems can be formulated as MIQP and introduce a new branch and bound algorithm that can optimally solve ISM. We validate our formulation and the proposed algorithm in multi-robot multi-objective coverage control. In numerical simulations, we demonstrate the advantages of the proposed algorithm w.r.t. the running time and peak memory usage compared to an existing solver.

VIII. Appendix

Lemma 1. *If h_1 and h_2 are supermodular, both non-decreasing (or non-increasing) and non-negative set functions defined on a finite set $\mathcal{V}$, then $h_1 h_2$ is still a supermodular non-decreasing (or non-increasing) and non-negative function.*

Proof. We prove this lemma using the definition of the supermodular function. Let us first consider the case where both functions are non-decreasing. Given two sets $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{A} \subseteq \mathcal{B}$, by definition, if we add one extra element a to each set, we have

$$h_1(\mathcal{A} \cup \{a\})h_2(\mathcal{A} \cup \{a\}) - h_1(\mathcal{A})h_2(\mathcal{A}) \tag{19}$$

$$= \begin{aligned}&(h_1(\mathcal{A} \cup \{a\}) - h_1(\mathcal{A}))h_2(\mathcal{A} \cup \{a\}) + \\ &h_1(\mathcal{A})(h_2(\mathcal{A} \cup \{a\}) - h_2(\mathcal{A})).\end{aligned} \tag{20}$$

By supermodularity, we have

$$h_1(\mathcal{A} \cup \{a\}) - h_1(\mathcal{A}) \leq h_1(\mathcal{B} \cup \{a\}) - h_1(\mathcal{B}) \tag{21}$$

$$h_2(\mathcal{A} \cup \{a\}) - h_2(\mathcal{A}) \leq h_2(\mathcal{B} \cup \{a\}) - h_2(\mathcal{B}). \tag{22}$$

By non-decreasing monotonicity, we have

$$h_1(\mathcal{A} \cup \{a\}) \leq h_1(\mathcal{B} \cup \{a\}) \tag{23}$$

$$h_2(\mathcal{A} \cup \{a\}) \leq h_2(\mathcal{B} \cup \{a\}). \tag{24}$$

Together with non-negativity, we have

$$\begin{aligned}&h_1(\mathcal{A} \cup \{a\}) - h_1(\mathcal{A}))h_2(\mathcal{A} \cup \{a\}) + \\ &h_1(\mathcal{A})(h_2(\mathcal{A} \cup \{a\}) - h_2(\mathcal{A})\end{aligned} \tag{25}$$

$$\leq \begin{aligned}&h_1(\mathcal{B} \cup \{a\}) - h_1(\mathcal{B}))h_2(\mathcal{B} \cup \{a\}) + \\ &h_1(\mathcal{B})(h_2(\mathcal{B} \cup \{a\}) - h_2(\mathcal{B}))\end{aligned} \tag{26}$$

$$= h_1(\mathcal{B} \cup \{a\})h_2(\mathcal{B} \cup \{a\}) - h_1(\mathcal{B})h_2(\mathcal{B}). \tag{27}$$

By definition, $h_1 h_2$ is a supermodular function.

For the case where both functions are non-increasing, Eq. (21) and Eq. (22) still hold. The difference is that both sides of the inequalities are non-positive. In Eq. (23) and Eq. (24),

the inequalities should be flipped. However, since both sides of Eq. (21) and Eq. (22) are non-positive, we can still get the same result as those in Eq. (25) and Eq. (26).

■

A. Proof of Theorem 1

Theorem 3 (Theorem 1 in [34])**.** *If we treat the $h_j(\boldsymbol{p}(k))$ in Eq.* (9) *as a set function with input set as $\boldsymbol{p}(k)$, $h_j(\boldsymbol{p}(k))$ is monotone submodular.*

It should be noted that $\boldsymbol{p}(k)$ can be viewed as a column of the selected set $\mathcal{A}$ (different actions correspond to different rows). As a result, $h_j(\boldsymbol{p}(t))$ is submodular w.r.t. $\mathcal{A}$. Next, we will prove Theorem 1 using Theorem 1 and Lemma 1.

Proof. By Theorem 1, $h_j(\boldsymbol{p}(t))$ is a monotone non-decreasing non-negative submodular function w.r.t. $\mathcal{A}$. Therefore, $1-h_j(\boldsymbol{p}(t))$ is a monotone non-increasing non-negative supermodular function w.r.t. $\mathcal{A}$. By Lemma 1, $\prod_{t=k}^{t=k+H} 1-h(\boldsymbol{p}(t))$ is also a monotone non-increasing non-negative supermodular function. Then, $(1-\prod_{t=k}^{t=k+H} 1-h(\boldsymbol{p}(t)))$ is a monotone non-decreasing non-negative submodular function. Since the weighted sum of submodular functions using positive weight results in a submodular function, $f(\mathcal{A}, \boldsymbol{\theta})$ is a monotone submodular function. ■

REFERENCES

[1] G. Shi, L. Zhou, and P. Tokekar, "Robust multiple-path orienteering problem: Securing against adversarial attacks," *IEEE Transactions on Robotics*, 2023.

[2] X. Xu, G. Shi, P. Tokekar, and Y. Diaz-Mercado, "Interactive multi-robot aerial cinematography through hemispherical manifold coverage," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 11 528–11 534.

[3] L. E. Parker, "Decision making as optimization in multi-robot teams," in *International Conference on Distributed Computing and Internet Technology*. Springer, 2012, pp. 35–49.

[4] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies." *Journal of Machine Learning Research*, vol. 9, no. 2, 2008.

[5] G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1271–1287, 2014.

[6] S. Poduri and G. S. Sukhatme, "Constrained coverage for mobile sensor networks," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 1. IEEE, 2004, pp. 165–171.

[7] Z. Xu, H. Zhou, and V. Tzoumas, "Online submodular coordination with bounded tracking regret: Theory, algorithm, and applications to multi-robot coordination," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2261–2268, 2023.

[8] Z. Xu, X. Lin, and V. Tzoumas, "Bandit submodular maximization for multi-robot coordination in unpredictable and partially observable environments," *Robotics: Science and Systems*, 2023.

[9] G. Shi, I. E. Rabban, L. Zhou, and P. Tokekar, "Communication-aware multi-robot coordination with submodular maximization," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8955–8961.

[10] K.-S. Tseng and B. Mettler, "Near-optimal probabilistic search via submodularity and sparse regression," *Autonomous Robots*, vol. 41, no. 1, pp. 205–229, 2017.

[11] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—I," *Mathematical programming*, vol. 14, pp. 265–294, 1978.

[12] B. Wilder, B. Dilkina, and M. Tambe, "Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 1658–1665.

[13] G. Shi and P. Tokekar, "Decision-oriented learning with differentiable submodular maximization for vehicle routing problem," *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[14] A. V. Terekhov, Y. B. Pesin, X. Niu, M. L. Latash, and V. M. Zatsiorsky, "An analytical approach to the problem of inverse optimization with additive objective functions: an application to human prehension," *Journal of mathematical biology*, vol. 61, pp. 423–453, 2010.

[15] D. Tsirakos, V. Baltzopoulos, and R. Bartlett, "Inverse optimization: functional and physiological considerations related to the force-sharing problem," *Critical Reviews™ in Biomedical Engineering*, vol. 25, no. 4-5, 1997.

[16] N. Sandholtz, Y. Miyamoto, L. Bornn, and M. A. Smith, "Inverse bayesian optimization: Learning human acquisition functions in an exploration vs exploitation search task," *Bayesian Analysis*, vol. 18, no. 1, pp. 1–24, 2023.

[17] P. Z. Scroccaro, P. van Beek, P. M. Esfahani, and B. Atasoy, "Inverse optimization for routing problems," *arXiv preprint arXiv:2307.07357*, 2023.

[18] L. Chen, Y. Chen, and A. Langevin, "An inverse optimization approach for a capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 295, no. 3, pp. 1087–1098, 2021.

[19] Z. Tan, Z. Yan, Q. Xia, and Y. Wang, "Data-driven inverse optimization for modeling intertemporally responsive loads," *IEEE Transactions on Smart Grid*, 2023.

[20] B. Hu, K. Xie, and H.-M. Tai, "Inverse problem of power system reliability evaluation: Analytical model and solution method," *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 6569–6578, 2018.

[21] K. Mombaur, A. Truong, and J.-P. Laumond, "From human to humanoid locomotion—an inverse optimal control approach," *Autonomous robots*, vol. 28, pp. 369–383, 2010.

[22] W. Jin, D. Kulić, S. Mou, and S. Hirche, "Inverse optimal control from incomplete trajectory observations," *The International Journal of Robotics Research*, vol. 40, no. 6-7, pp. 848–865, 2021.

[23] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International conference on machine learning*. PMLR, 2016, pp. 49–58.

[24] A. Šošić, W. R. KhudaBukhsh, A. M. Zoubir, and H. Koeppl, "Inverse reinforcement learning in swarm systems," *arXiv preprint arXiv:1602.05450*, 2016.

[25] L. Yu, J. Song, and S. Ermon, "Multi-agent adversarial inverse reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7194–7201.

[26] X. Wang and D. Klabjan, "Competitive multi-agent inverse reinforcement learning with sub-optimal demonstrations," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5143–5151.

[27] C. Heuberger, "Inverse combinatorial optimization: A survey on problems, methods, and results," *Journal of combinatorial optimization*, vol. 8, pp. 329–361, 2004.

[28] A. J. Schaefer, "Inverse integer programming," *Optimization Letters*, vol. 3, pp. 483–489, 2009.

[29] N. Wilde, A. Blidaru, S. L. Smith, and D. Kulić, "Improving user specifications for robot behavior through active preference learning: Framework and evaluation," *The International Journal of Robotics Research*, vol. 39, no. 6, pp. 651–667, 2020.

[30] E. Biyik and M. Palan, "Asking easy questions: A user-friendly approach to active reward learning," in *Proceedings of the 3rd Conference on Robot Learning*, 2019.

[31] N. Wilde, D. Kulić, and S. L. Smith, "Active preference learning using maximum regret," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 952–10 959.

[32] N. Wilde, A. Sadeghi, and S. L. Smith, "Learning submodular objectives for team environmental monitoring," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 960–967, 2021.

[33] W. Li and C. G. Cassandras, "Distributed cooperative coverage control of sensor networks," in *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 2005, pp. 2542–2547.

[34] X. Sun, C. G. Cassandras, and X. Meng, "A submodularity-based approach for multi-agent optimal coverage problems," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 4082–4087.