Inverse Risk-sensitive Multi-Robot Task Allocation

Guangyao Shi and Gaurav S. Sukhatme

University of Southern California, Los Angeles CA 90089, USA
shig, gaurav@usc.edu

Abstract. We consider a new variant of the multi-robot task allocation problem - Inverse Risk-sensitive Multi-Robot Task Allocation (IR-MRTA). "Forward" MRTA - the process of deciding which robot should perform a task given the reward (cost)-related parameters, is widely studied in the multi-robot literature. In this setting, the reward (cost)-related parameters are assumed to be already known: parameters are first fixed offline by domain experts, followed by coordinating robots online. What if these parameters need to be changed during runtime by non-expert human supervisors? This may happen for example, when the human supervisor's perception of the allocation risk (e.g., the probability of robot failure) changes. In such cases, the robots need to change the parameters of the originally posed task allocation problem based on evolving human preferences. We study such problems through the lens of *inverse task allocation*, i.e., the process of finding parameters given solutions to the problem. Specifically, we propose a new formulation IR-MRTA, whose goal is to find a new set of parameters of the human behavioral risk model that minimally deviates from the current MRTA parameters while causing a greedy task allocation algorithm to allocate robot resources in line with the updated human supervisory input. We show that even in the simple case this is a non-convex optimization problem. We propose a Branch & Bound algorithm (BB-IR-MRTA) to solve such problems. We demonstrate BB-IR-MRTA on numerical simulations of multi-robot target capture and show that it provides significant advantages in running time and peak memory usage compared to a brute-force baseline.

Keywords: Multi-Agent Systems and Distributed Robotics · Human-Robot Interaction.

1 Introduction

Multi-robot task Allocation (MRTA) is a fundamental problem in multi-robot systems [4, 9, 10]. It determines which robots execute particular tasks to collaboratively achieve a system-level goal [10]. It can generally be cast as a combinatorial optimization problem with constraints and is usually NP-hard.

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.

In the classic MRTA pipeline, we first design reward/cost functions for allocations using domain knowledge. This phase is typically offline. Then, we deploy the robots for tasks by solving a combinatorial optimization problem. A missing piece in such a pipeline is how to incorporate feedback from human supervisors into the task allocation process when we deploy the robots in the field, and human supervisors stay in the loop to watch over the team. Considering human feedback in multi-robot task allocation is crucial especially when there is uncertainty. First, humans can provide valuable insights and contextual knowledge that may not be fully captured by pre-programmed algorithms, especially in complex or unforeseen situations. Second, involving humans in the decision-making process helps to align robot operations with human preferences (and potentially ethical standards). In the risk-sensitive multi-robot task allocation problem, human supervisors may evaluate the risk (e.g., the probability of robot failure) associated with a task allocation schema in a way that is different from what statistical models may assess due to either humans having better situation understanding or their personal preference. As a result, humans may suggest different (possibly better) allocations of tasks compared to those obtained from solving pre-designed optimization problems. In such cases, it is undesirable to stop the team and redesign the optimization problem. Moreover, the human supervisors in the loop may not be optimization experts, and they may not know how to modify the problems to distill their new insights. Instead, we want the robot team to have the capability to adjust the parameters of the problem in a *minimal* way to accommodate human suggestions. The reason for the minimal change is that the current parameters of the problem are already the embodiment of much expertise and historical data, and we should not give them away in favor of a few new suggestions. We consider such problems from the perspective of the inverse problem of risk-sensitive task allocation.

Specifically, for a multi-robot task allocation problem with parameterized risk constraints/objectives [1, 12, 29], if the parameters are known, we can use an approximation algorithm to solve the problem. We call such a process Forward Task Allocation (FTA). By contrast, if we have a solution obtained using approximation algorithms, we want to find the corresponding parameters related to risk such that when we use those parameters to solve FTA problems the resulting solutions match the known solutions. We call such a process Inverse Risk-sensitive Multi-Robot Task Allocation (IR-MRTA). Such problems fall into the more general class of inverse optimization problems, which are widely used to decode human decision-making processes [18, 22, 23] for robotic applications. By formulating and solving IR-MRTA we can incorporate human suggestions into the task allocation process. To our knowledge, this is the first paper that introduces such a risk-sensitive inverse problem formulation to MRTA. Further, we introduce a Branch and Bound (BB) algorithm to solve IR-MRTA.

The main contributions of this paper are:

- We formulate a new variant of MRTA named IR-MRTA and propose a novel algorithm to solve IR-MRTA.

– We show how to use IR-MRTA in multi-robot task allocation to account for human suggestions.
– The proposed formulation and algorithm are validated extensively through simulation.

2 Related Work

Multi-Robot Task Allocation MRTA problems have been widely studied in the context of multi-robot systems [7]. Most existing research focuses on how to develop efficient (scalable to a large number of robots) and effective (optimal or close to optimal) strategies, which can be centralized or decentralized, to solve deterministic allocation problems. Uncertainty in MRTA is also studied using either risk-neutral approaches [5, 15] or risk-averse approaches [12, 16, 29]. The differences between our work and existing work lie in two aspects. First, most existing work considers the forward problem, i.e., given the specified parameters in the objective and the constraints, find a solution. By contrast, our focus is the inverse problem, i.e., given a solution, find parameters in the constraints. Second, our formulation of MRTA incorporates a parameterized human behavioral model, which describes how humans perceive uncertainty.

Inverse Optimization Inverse Optimization (IO) is widely used to learn human behavior, for example, using human walking gaits for humanoid locomotion and using human grasping skills for robot arm manipulation. In the related literature, IO is usually referred to as Inverse Optimal Control (IOC) or Inverse Reinforcement Learning (IRL). Existing frameworks for IOC/IRL are usually designed for one robot and their multi-robot counterparts are still in their infancy [21, 24, 30]. The main difference between our work and the existing IOC/IRL is that our forward problem is combinatorial with discrete decision variables rather than continuous optimization. Such combinatorial cases in inverse optimization are less well studied. Since combinatorial optimization is widely used for multi-robot coordination, e.g., task allocation and vehicle routing and scheduling, the research for such Inverse Combinatorial Optimization (ICO) [8] will provide more tools for multi-robot systems. In ICO, Inverse Integer Optimization (IIO) is widely studied [19]. However, IIO can not deal with the inverse problem studied in this paper as shown in Sec. 4 since it involves a nonlinear and non-convex constraint.

Human Multi-Robot Interaction Our research is closely related to two directions. One is human-in-the-loop multi-robot coordination [6, 11, 25, 28]. Our work is different from these works in two aspects: first, we consider the problem for a novel task allocation problem; second, we develop a new paradigm by using ICO to incorporate human suggestions. Another closely related direction is human preference learning [3, 26, 27], in which humans are typically presented with two solutions iteratively, requiring them to compare these options during each iteration. The majority of research in this area concentrates on developing efficient methods for generating queries to facilitate the learning of human

preference parameters. By contrast, our ICO does not involve an iterative process. Humans suggest an allocation that they think the *approximation algorithm* should output. We develop a framework that minimally modifies the parameters of the problem to match the human suggestions.

3 Preliminaries

We begin with the notations and conventions used in this paper and introduce some background before formally defining the inverse problem. We use calligraphic fonts to denote sets (e.g. $\mathcal{A}$). We denote $\|\cdot\|$ as the ℓ^2-norm. A set is ordered if it preserves the order in which elements are inserted. 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 an *ordering* of the set.

3.1 Forward Problem: Risk-sensitive Multi-Robot Task Allocation

We first introduce a risk-sensitive multi-robot task allocation formulation [1]. In this formulation, each allocation is associated with a reward and a probability of staying intact after executing the allocated task. The objective is to maximize the sum of rewards and guarantee the likelihood of all the allocated robots being intact being greater than a threshold. Then, we will introduce the behavioral model of humans (i.e., how humans perceive uncertainty) into the formulation, leading to the behavioral risk-sensitive task allocation.

Risk-sensitive task allocation is formulated as

$$\max_{\boldsymbol{x}} \quad \sum_{i,j} r_{i,j} x_{i,j} \tag{1a}$$

$$\text{s.t.} \quad \sum_{i}^{n_r} x_{i,j} \leq 1, \forall j, \quad \sum_{j}^{n_t} x_{i,j} \leq 1, \forall i \tag{1b}$$

$$\prod_{i,j} ((1 - x_{i,j}) + x_{i,j} p_{i,j}) \geq \delta \tag{1c}$$

$$\sum_{i,j} x_{i,j} \leq n_r \tag{1d}$$

$$x_{i,j} \in \{0, 1\}, \tag{1e}$$

where r_{ij} denotes the reward of allocating robot i to task j; the constraint (1b) enforces that one robot can be allocated to up to one task and one task can be allocated at most one robot; in the constraint (1c), if the robot i is allocated to the task j (i.e., $x_{ij} = 1$), $((1 - x_{i,j}) + x_{i,j} p_{i,j})$ is equal to p_{ij}, otherwise is 1. The product is the likelihood of all the allocated robots being intact and δ is a threshold; Eq. (1d) is a constraint on the number of available robots; Eq. (1e) specifies that x_{ij} is a binary variable.

Human Perceived Uncertainty According to research in behavioral economics, humans usually perceive uncertainty in an irrational fashion, especially in evaluating outcomes [2, 14]. We introduce such a concept into the risk-sensitive task allocation to account for the observation that if we need to solve a sequence of task allocation problems to coordinate robots, humans may change their views on risks based on the results of robots after executing the allocated tasks. Humans may become more conservative or more aggressive. Specifically, we consider the probability weight function $w : [0, 1] \to [0, 1]$ which maps a probability to another and can be used to model human irrationality on uncertainty [2, 14]. In this paper, we use the popular Prelec's model [14] for its simplicity

$$w(p) = e^{-\beta(-\log p)^{\alpha}}, \alpha > 0, \beta > 0, w(0) = 0. \tag{2}$$

More details on the intuition of such a model are given in Sec. 6.1.

Using Prelec's model, the constraint in (1c) becomes

$$\prod_{i,j}((1 - x_{i,j}) + x_{i,j}w(p_{i,j})) \geq \delta \tag{3}$$

By taking logarithmic operation on both sides, Eq. (3) can be equivalently represented as

$$\sum \beta(-\log p_{i,j})^{\alpha} x_{i,j} \leq \ -\log \delta. \tag{4}$$

Replacing Eq. (1c) with Eq. (4), we obtain the forward task allocation problem with the behavioral risk constraint. Such a formulation can be viewed as a standard task allocation with one extra knapsack constraint (Eq. (4)), rendering the problem NP-hard [13, 17]. As an initial trial to solve the inverse version of such problems, we will confine our discussion to the case where a simple greedy algorithm is used to solve the problem as shown in the Algorithm 1. At each step, we will select one allocation based on a score which is the allocation reward (r_{ij}) divided by its associated cost ($\beta(-\log p_{i,j})^{\alpha}$).

4 Problem Formulation

Problem 1 (Inverse Risk-sensitive Multi-Robot Task Allocation (IR-MRTA)).

Given a suggested feasible allocation result $\hat{\boldsymbol{x}} \in \{0, 1\}^{n_r \times n_t}$ from human, find new parameters $\hat{\alpha}, \hat{\beta}, \hat{\delta}$ for the human behavior model such that the weighted sum of the distances between the original parameters (α, β, δ) and the new parameters is minimized, and the suggested allocation $\hat{\boldsymbol{x}}$ becomes the output of the greedy algorithm using $(\hat{\alpha}, \hat{\beta}, \hat{\delta})$. Mathematically, such an inverse task allocation problem can be formulated as follows:

$$\min_{\hat{\alpha},\hat{\beta},\hat{\delta}} \ w_{\alpha}\|\hat{\alpha} - \alpha\| + w_{\beta}\|\hat{\beta} - \beta\| + w_{\delta}\|\hat{\delta} - \delta\| \tag{5a}$$

$$\text{s.t.} \quad \boldsymbol{x}^*(\hat{\alpha}, \hat{\beta}, \hat{\delta}) = \text{argmax\_greedy}_{\boldsymbol{x}} \, \boldsymbol{f}(\boldsymbol{x} \mid \hat{\alpha}, \hat{\beta}, \hat{\delta}) \tag{5b}$$

$$\hat{\boldsymbol{x}} = \boldsymbol{x}^*(\hat{\alpha}, \hat{\beta}, \hat{\delta}) \tag{5c}$$

$$\hat{\alpha} \in [\hat{\alpha}_{min}, \hat{\alpha}_{max}], \hat{\beta} \in [\hat{\beta}_{min}, \hat{\beta}_{max}], \hat{\delta} \in [\hat{\delta}_{min}, \hat{\delta}_{max}], \tag{5d}$$

Algorithm 1: Greedy Algorithm

Input : A TA-BRC instance
Output: A task allocation plan

$\mathcal{S} \leftarrow \emptyset$
while $|\mathcal{S}| < n_r$ **do**
  for unallocated robots, find the allocation with the largest cost-scaled reward $(i^*, j^*) = \text{argmax}_{i \in \{1,\dots,n_r\} \setminus \mathcal{S}^i, j \in \{1,\dots,n_t\} \setminus \mathcal{S}^j} \frac{r_{i,j}}{\beta(-\log p_{i,j})^\alpha}$
  if $\sum_{(i,j) \in \mathcal{S} \cup \{(i^*,j^*)\}} \beta(-\log p_{i,j})^\alpha \leq -\log \delta$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{(i^*, j^*)\}$
  else
    break
  end
end
return $\mathcal{S}$

where $f(\boldsymbol{x} \mid \hat{\alpha}, \hat{\beta}, \hat{\delta})$ is the objective in Eq. (1a) and $\hat{\alpha}, \hat{\beta}, \hat{\delta}$ is incorporated to highlight that the optimization will rely on these parameters; $\boldsymbol{x}^*(\hat{\alpha}, \hat{\beta}, \hat{\delta})$ in Eq. (5b) is the solution returned by the greedy algorithm (Algorithm 1) using parameters $\hat{\alpha}, \hat{\beta}, \hat{\delta}$ and Eq. (5c) enforces the solution to be equal to the human suggested allocation; w_α, w_β and w_δ are three hyperparameters.

4.1 Motivating Case Study: Multi-Robot Target Capture

There are n_r robots in the task area $\mathcal{E}$. Robots are represented as discs with different sizes. The larger the disc, the more energy it will consume to finish the same task. A team of n_t non-strategic targets with $n_t \leq n_r$ will occasionally enter the $\mathcal{E}$. The targets are also represented as discs with different sizes. These targets are of value to us, and we want to capture them by robots. We consider the case where we must allocate one robot to each target for successful capture. When the robot captures a target, the target can cause damage to the robot with a probability that depends on the relative size of each other. When the robot is larger compared to the target, the likelihood of being damaged is small. by contrast, when the robot is relatively small compared to the target, the damage probability will be higher. After going through the damage, the robot is still functional but needs inspection and repair. The high-level goal here is to allocate robots to capture more targets to maximize the reward and try to avoid damage.

Specifically, the reward of allocating robot i to target j is defined as

$$r_{ij} = r_j - \xi_c \hat{c}_{ij} - \xi_d (1 - p_{ij}) d_i, \tag{6}$$

where r_j is the reward term for capturing the target j; ξ_c is a scaling factor for the motion cost; $\hat{c}_{ij}$ is the estimated motion cost (e.g., energy consumption). Suppose the target is non-strategic and moves with constant velocities, the motion cost will be proportional to the travel distance to intercept the target; ξ_d

is a scaling factor for repair cost after damage; p_{ij} is the probability of without being damaged; and d_i is the repair cost for robot i; r_{ij} is designed to be positive.

p_{ij} should capture that the relative size between the robot and the target will affect the probability of being damaged. We use the following function in this paper

$$p_{ij} = \frac{1}{1 + e^{k(s_j - s_i)}}, \tag{7}$$

where $k > 0$ is a parameter; s_i and s_j are sizes for robot i and target j respectively.

After a few rounds of capture, if no robots get damaged, humans may think the Eq. (7) underestimates the probability of being intact and suggest a more aggressive allocation (e.g., replace a larger robot with a smaller robot to reduce motion cost). We need to solve the inverse problem to determine the parameters that can lead to such suggested decisions.

5 Algorithm

Let us first consider the case where the human suggestion is ordered, i.e., the suggested allocation is given sequentially. Let $\mathcal{S} = \{(i_1, j_1), (i_2, j_2), \ldots, (i_m, j_m)\}$ ($m \leq n_r$) be the ordered set of allocations corresponding to the human suggestions. We will denote $\mathcal{S}^i$ and $\mathcal{S}^j$ as the ordered sets $\{i_1, i_2, \ldots, i_m\}$ and $\{j_1, j_2, \ldots, j_m\}$ respectively. Recall that when we use the greedy algorithm (Algorithm 1) to determine allocations, we will find the allocation with maximum score each step and make the new allocation will satisfy the knapsack constraint. Suppose that the human suggests that $\mathcal{S}$ should be the output of using the greedy algorithm, such an ordered set is equivalent to a set of inequalities as follows.

$$\frac{r_{i_1,j_1}}{\hat{\beta}(-\log p_{i_1,j_1})^{\hat{\alpha}}} \geq \frac{r_{i,j}}{\hat{\beta}(-\log p_{i,j})^{\hat{\alpha}}}, \forall i, j \tag{8a}$$

$$\hat{\beta}(-\log p_{i_1,j_1})^{\hat{\alpha}} \leq -\log \hat{\delta} \tag{8b}$$

$$\cdots$$

$$\frac{r_{i_m,j_m}}{\hat{\beta}(-\log p_{i_m,j_m})^{\hat{\alpha}}} \geq \frac{r_{i,j}}{\hat{\beta}(-\log p_{i,j})^{\hat{\alpha}}}, \forall i(j) \notin \mathcal{S}^{i(j)}[1 : m-1] \tag{8c}$$

$$\sum_{k=1}^{m} \hat{\beta}(-\log p_{i_k,j_k})^{\hat{\alpha}} \leq -\log \hat{\delta} \tag{8d}$$

These inequalities correspond to the greedy selection process as described in Algorithm 1 and can be simplified as follows. First, we can observe that all the inequalities corresponding to the knapsack constraint in Eq. (4) can be summarized as Eq. (8d) since other inequalities are included in this inequality. Second, we can transform the inequalities corresponding to the score comparison and greedy selection into linear constraints by taking log operations on both sides of the inequalities. Therefore, the inequalities above can be simplified as Eq. (9a) to Eq. (9b) and Eq. (8d).

$$\log \frac{r_{i_1,j_1}}{r_{i,j}} \geq \hat{\alpha} \cdot (\log(\log \frac{1}{p_{i_1,j_1}}) - \log(\log \frac{1}{p_{i,j}})) \tag{9a}$$

$$\cdots$$

$$\log \frac{r_{i_m,j_m}}{r_{i,j}} \geq \hat{\alpha} \cdot (\log(\log \frac{1}{p_{i_1,j_1}}) - \log(\log \frac{1}{p_{i,j}})), \forall i(j) \notin \mathcal{S}^{i(j)}[1:m-1] \tag{9b}$$

In sum, the ordered set case of IR-MRTA can be summarized as the following problem.

Problem 2 (Ordered-IR-MRTA (O-IR-MRTA)).

$$\min_{\hat{\alpha},\hat{\beta},\hat{\delta}} \ w_\alpha \|\hat{\alpha} - \alpha\| + w_\beta \|\hat{\beta} - \beta\| + w_\delta \|\hat{\delta} - \delta\| \tag{10a}$$

$$\text{s.t.} \quad \hat{\alpha} \in \mathcal{A}, \hat{\alpha} \in [\hat{\alpha}_{min}, \hat{\alpha}_{max}], \hat{\beta} \in [\hat{\beta}_{min}, \hat{\beta}_{max}], \hat{\delta} \in [\hat{\delta}_{min}, \hat{\delta}_{max}] \tag{10b}$$

$$\sum_{k=1}^{m} \hat{\beta}(-\log p_{i_k,j_k})^{\hat{\alpha}} \leq -\log \hat{\delta}, \tag{10c}$$

where $\mathcal{A}$ is a convex set corresponding linear constraints described from Eq. (9a) to Eq. (9b).

It should be noted that Eq. (10c) is not a convex constraint. In the following, we will show how to solve such a non-convex optimization problem using the branch and bound paradigm. The key idea is to partition the feasible set into convex sets and find the lower/upper bound for each, which is then used to decide the expansion of the search tree. The algorithm will refine the partition until the search depth reaches the termination condition.

Specifically, we will partition $[\hat{\beta}_{min}, \hat{\beta}_{max}]$ and $[\hat{\delta}_{min}, \hat{\delta}_{max}]$ into smaller convex subsets $\mathcal{B} = [\underline{\beta}, \bar{\beta}]$ and $\mathcal{D} = [\underline{\delta}, \bar{\delta}]$. We aim to solve the following problem.

$$\min_{\hat{\alpha},\hat{\beta},\hat{\delta}} \ w_\alpha \|\hat{\alpha} - \alpha\| + w_\beta \|\hat{\beta} - \beta\| + w_\delta \|\hat{\delta} - \delta\| \tag{11a}$$

$$\text{s.t.} \quad \hat{\alpha} \in \mathcal{A}, \quad \hat{\beta} \in \mathcal{B}, \quad \hat{\delta} \in \mathcal{D} \tag{11b}$$

$$\sum_{k=1}^{m} (-\log p_{i_k,j_k})^{\hat{\alpha}} \leq -\frac{1}{\hat{\beta}} \log \hat{\delta} \tag{11c}$$

It should be noted that Eq. (11c) is a transformation of Eq. (10c) and $-\frac{1}{\hat{\beta}} \log \hat{\delta}$ is a monotone decreasing function w.r.t. both $\hat{\beta}$ and $\hat{\delta}$, i.e., $-\frac{1}{\hat{\beta}} \log \hat{\delta} \leq -\frac{1}{\bar{\beta}} \log \underline{\delta}$. Using this observation, we can get the convex relaxation to compute the lower bound of the optimal objective value by replacing expression $-\frac{1}{\hat{\beta}} \log \hat{\delta}$ in Eq. (11c) with a constant $-\frac{1}{\bar{\beta}} \log \underline{\delta}$.

In such convex relaxation, the constraint is imposed only on $\hat{\alpha}$. If it is not feasible, it suggests that the original problem is not feasible. If it is feasible, we

Algorithm 2: B&B for O-IR-MRTA (BB-O-IR-MRTA)

```
   Input  : An O-IR-MRTA instance
   Output: An approximate global solution α*, β*, δ*
 1 Tree ← empty tree
 2 Tree.add_node(node_ID=0, LB=−1, UB=∞)
 3 Tree.sol ← null # best solution found
 4 Tree.obj_ub ← ∞, Tree.obj_lb ← 0
 5 Q ← queue
 6 Q.insert(node_ID=0)
 7 while Q is not empty or Tree.depth reaches n_d do
 8 |   v_p ← Q.pop()
 9 |   V_c ← generate_child_nodes(v_p)
10 |   for each child node v_c ∈ V_c do
11 |   |   LB_feasibility, LB ← lower_bound(v_c)
12 |   |   UB_feasibility, UB ← upper_bound(v_c)
13 |   |   if not LB_feasibility then
14 |   |   |   continue
15 |   |   else
16 |   |   |   if LB > Tree.obj_ub then
17 |   |   |   |   continue
18 |   |   |   end
19 |   |   |   if UB < Tree.obj_ub then
20 |   |   |   |   Tree.sol ← solution at v_c
21 |   |   |   |   Tree.obj_ub ← UB at v_c
22 |   |   |   end
23 |   |   |   Q.insert(v_c)
24 |   |   |   Tree.add_node(v_c)
25 |   |   end
26 |   end
27 end
28 return Tree.sol
```

can find a feasible solution by imposing two extra constraints (a) $\hat{\beta} = \underline{\beta}$ (b) $\hat{\delta} = \underline{\delta}$ in addition to Eq. (11c).

Details are given in Algorithm 2. In lines 1-6, we initialize the search process. Tree.sol is used to track the best solution found so far and Tree.obj_ub is the corresponding upper bound. In the while loop, we first extract a node from the queue (line 8). Then, we will generate some child nodes by dividing the sets of β and δ (line 10). We choose the following splitting strategies to create child nodes. As shown in Fig. 1a, at the root node, we divide the set by using the original parameter (β and δ) as a separator and we will generate four child nodes. For the rest, we generate four child nodes by evenly dividing each interval into two parts (from node 2 to nodes 5 and 8). After this step, for each generated child node, we will find its lower bound and upper bound (lines 11-12) considering the relaxation described above. If it is infeasible to find its lower bound (lines 13-15), it suggests that the problem is infeasible for this node and we can prune

this branch away (red cross in Fig. 1a). Otherwise, we check whether the lower bound is greater than the best upper bound identified so far (lines 16-18). If it is, it suggests that we do not need to further branch on this node since it will not result in a solution better than the one identified so far. If the upper bound is less than the global upper bound which suggests that we have found a better solution, we will update the global solution and the upper bound (lines 19-22). This node will be inserted into the queue for further branching (lines 23-24). After updating all the child nodes, we will check whether the termination condition is reached to decide whether we should further branch the search tree (line 7). Upon termination, the solution will satisfy the property outlined in Theorem 1.

Theorem 1. *Given a feasible problem instance as described in Problem 2, Algorithm 2 returns a solution, sol, satisfying*

$$g(sol) - g^* \leq \epsilon = w_\beta \max(\frac{\beta - \hat{\beta}_{min}}{2^{n_d-1}}, \frac{\hat{\beta}_{max} - \beta}{2^{n_d-1}}) + w_\delta \max(\frac{\delta - \hat{\delta}_{min}}{2^{n_d-1}}, \frac{\hat{\delta}_{max} - \delta}{2^{n_d-1}}),$$

where $g(\cdot)$ is the objective in Eq. (5a) *and g^* is the optimal objective to Problem 2; $w_\beta, w_\delta, \beta, \delta, \hat{\beta}_{min}, \hat{\beta}_{max}, \hat{\delta}_{min}, \hat{\delta}_{max}$ are defined in the Problem 2; n_d is the search depth.*

The proof of Theorem 1 and the analysis of how the optimality gap depends on the depth of the search tree are given in the Appendix. We provide a high-level overview of the proof here. In Algorithm 2, we grow the search tree in a breadth-first search fashion: we will search all the possible combinations in a certain depth and then move to the next depth. At each depth level, the leaf nodes contain all the possible combinations of subsets of $\hat{\beta}$ and $\hat{\delta}$. Suppose that the best solution identified belongs to a node v_u, i.e., $v_u.UB = g(sol)$ and $v_u.UB \geq v.UB$ for all leaf nodes. Suppose that the optimal solution will be in a node v_{OPT} which is not known to us. But what we know is that $v_{OPT}.LB \leq g^* \leq v_{OPT}.UB$. The proof is on how to relate the gap $g(sol) - g^*$ to the set sizes of each leaf node.

Next, we will consider the general case where the human suggestion is not ordered. 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 the ordering of a solution set. The naive way to deal with such general cases is to enumerate all the possible orderings of the human-suggested allocation and then solve O-IR-MRTA for each. However, such an approach is not scalable w.r.t. the size of the set of the human suggestion. Next, we will introduce another BB algorithm, which will use Algorithm 2 as a subroutine, to solve the general case of IR-MRTA. A similar algorithmic paradigm has been proposed in our previous work [20] for inverse submodular maximization. The key difference is that we use a novel subroutine (Algorithm 2), which will only return an approximate optimal solution rather than the optimal one compared to [20]. Correspondingly, we change the pruning condition (line 17) in Algorithm 3 and will result in a weaker result (Theorem 2) compared to that in [20].

Algorithm 3: B&B for IR-MRTA (BB-IR-MRTA)

Input : human suggestion $\hat{\mathcal{S}}$, tolerance ϵ
Output: $\hat{\boldsymbol{\theta}} = [\hat{\alpha}, \hat{\beta}, \hat{\delta}]$

Tree ← empty tree # Initialize a search tree
Tree.UB ← a large number
Tree.add_node(node_ID = 0, sequence = {})
# Initialize an empty stack for Depth First Search
Stack ← empty stack
Stack.push(Tree.get_node(node_ID = 0))
while *Stack is not empty* **do**
  u ← Stack.pop()
  PQ ← priority_queue()
  for $s \in \hat{\mathcal{S}} \setminus u$.seq **do**
    feasible, obj, $\hat{\boldsymbol{\theta}}$ ← O-IR-MRTA(u.seq + {s})
    if *feasible* **then**
      PQ.insert(s, $\hat{\boldsymbol{\theta}}$, priority_value=obj)
    end
  end
  while *PQ is not empty* **do**
    s, $\hat{\boldsymbol{\theta}}$, obj ← PQ.pop()
    if *obj* < *Tree.UB*+ϵ **then**
      Tree.update_UB(u.seq+{s}, obj)
      new_node ← Tree.add_branch(u, s, obj, $\hat{\boldsymbol{\theta}}$)
      Stack.push(new_node)
    end
  end
end
return *tree*
Function `update_UB`(*Tree*, *sequence*, *obj*)**:**
  if *length of sequence* = *length of* $\hat{\mathcal{S}}$ **then**
    if *obj* < *Tree.UB* **then**
      Tree.UB ← obj
      Tree.UB_seq ← *sequence*
    end
  end
end

The main idea is that instead of directly optimizing over the best ordering of human suggestion $\hat{\mathcal{S}}$, we incrementally add elements to form a sequence of relaxed problems. Solving these relaxed problems can help us 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. 1b. 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

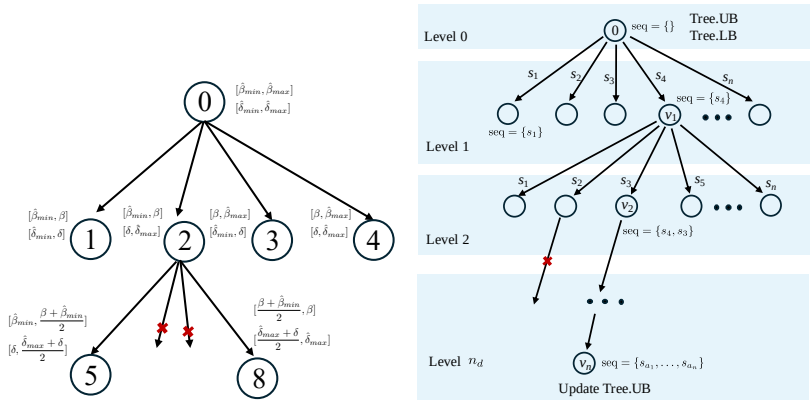

Fig. 1. (a) One search iteration of the proposed subroutine algorithm BB-O-IR-MRTA. Red crosses mean denote that the expanding branches are pruned (b) One iteration of the proposed BB-IR-MRTA algorithm.

node. For each such node, we will solve a problem O-IR-MRTA(*seq*) using the *seq* property of the node. When the *seq* includes only part of elements in $\hat{\mathcal{S}}$, the objective value returned by solving O-IR-MRTA(*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.

Next, we select the node (node v_1 in Fig. 1b) with the lowest objective value, determined by solving O-IR-MRTA(*seq*), for further expansion. This objective value serves as a heuristic for guiding the search, and we employ a greedy strategy to advance to the subsequent level. The expansion process is analogous to the transition from level 0 to level 1. At node v_1, the *seq* contains only one element, s_4. Thus, we can add any element from $\hat{\mathcal{S}} \setminus \{s_4\}$ to form a new child node at level 2 and solve the associated O-IR-MRTA(*seq*) problem. This process continues until the node's *seq* encompasses all elements in $\hat{\mathcal{S}}$, signifying that a complete ordering of $\hat{\mathcal{S}}$ has been identified. The objective value returned from solving O-IR-MRTA(*seq*) for this node will be used to update the problem's upper bound (Tree.UB in the root node), as it represents the solution for a specific ordering of $\hat{\mathcal{S}}$. We then proceed to expand the tree by backtracking to the previous level and continuing similarly to Depth First Search (DFS). During the tree expansion, if a particular O-IR-MRTA(*seq*) problem is found to be infeasible—indicating that all orderings with the prefix *seq* are infeasible—or if the objective value from O-IR-MRTA(*seq*) exceeds the Tree.UB+ϵ, where ϵ refers to the gap in Theorem 1, suggesting that all orderings with the prefix *seq* will not result in an improved solution, we will prune that branch as illustrated in Fig. 1b (the red cross marks the pruned branch). This pruning process accelerates the search.

The detailed procedure is outlined in Algorithm 3. Initially, in lines 1-5, we set up the search tree with a root node whose *seq* attribute is an empty ordered set and a stack for a DFS-style search. Within the while loop, we first pop the top element from the stack (line 7) and initialize a priority queue (line 8). Next, we iterate over each element in $\hat{\mathcal{S}}$ that is not in $u.seq$ (line 9) to check if appending this element to the existing sequence makes it feasible to solve a relaxed O-IR-MRTA problem (line 10). If infeasible, we can prune this branch, as all sequences with such a prefix are impossible. If feasible, we insert the element s and the corresponding $\hat{\boldsymbol{\theta}}$ using the returned objective value as the priority value. Subsequently, we update the search tree (lines 15-22), expanding branches in increasing order of the objective value (line 16). For each candidate, we determine whether to prune it (line 17) by comparing the objective value (obj) with the tree's upper bound plus a tolerance ($Tree$.UB+ϵ). If obj exceeds this value, we prune branches with such sequences, since any ordering of $\hat{\mathcal{S}}$ with a prefix $u.seq$ will yield an objective greater than $Tree$.UB+ϵ. Given that we employ an approximation algorithm (Algorithm 2) ensuring an objective within ϵ of the optimal solution, it follows that the optimal solution for any ordering of $\hat{\mathcal{S}}$ with prefix $u.seq$ will be greater than $Tree$.UB. Hence, we prune all such branches. Conversely, if the branch should not be pruned (line 17), we update the search tree's upper bound and add the new branch to the tree (line 19). To maintain the DFS search style, we push the new nodes onto the stack (line 20). After processing all elements in the priority queue, we continue the while loop by popping the top element from the stack (line 7) and repeating the process.

Theorem 2. *Given a feasible problem instance as described in Problem 1, Algorithm 3 returns a solution, sol, in a finite number of iterations of the outer while loop (lines 6-23) satisfying*

$$g(sol) - g^* \leq \epsilon,$$

where $g(\cdot)$ is the objective in Eq. (5a) *and g^* is the optimal solution to Problem 1; ϵ is the gap described in Theorem 1.*

Like all the algorithms that are developed within the BB paradigm, the BB-IR-MRTA 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. It should be noted that the problem as described in Problem 1 is an NP-hard problem in general. In the worst case, the proposed algorithms (Algorithm 2 and 3) will have to enumerate all the possible cases to find the solution or find that the problem is infeasible. We will experimentally evaluate the proposed algorithms in Section 6.

6 Experiments

We validate the proposed IR-MRTA formulation and evaluate the BB-IR-MRTA algorithm in a case study on multi-robot target capture as described in Sec. 4.1.

We will first present a qualitative example to show how different human suggestions correspond to different parameters in the behavior models (probability weighting function). Then, we will evaluate the proposed algorithm w.r.t. its optimality, running time, and peak memory usage.

6.1 A Qualitative Example

An illustrative example is shown in Fig. 2. There are ten robots of different sizes (magenta circles) and four targets (blue circles). Let us first look at Fig. 2b. Suppose the decision-making algorithm first uses a rational model to find task allocations, i.e., $w(p) = p$ (dotted black line in Fig. 2a) and $\alpha = 1, \beta = 1, \delta = 0.8$. The resulting allocations are black edges in Fig. 2b. In such a case, if the human suggests the allocations should be those blue edges in Fig. 2b, it implies that the human perceives the uncertainty in a non-rational way. By solving the IR-MRTA problem, we can find that the human-suggested solution corresponds to parameters $\alpha = 0.49, \beta = 0.36, \delta = 0.75$. The corresponding curve (blue) is shown in Fig. 2a. From this curve, we can find that the majority of the curve is above the dotted line, i.e., $w(p) > p$, which implies that the human tends to overestimate the success probability in that part. By contrast, if the human suggests the allocations should be those orange edges in Fig. 2c, we can find the suggestion of the human corresponds to parameters $\alpha = 0.75, \beta = 1.0, \delta = 0.8$. From the associated curve (orange) in Fig. 2c, we can see that in the part $p > 0.4$, the curve is below the dotted line, i.e., $w(p) < p$, which implies the human underestimates the success probability. Reward and probability parameters used in this example are given in the Appendix.

6.2 Quantitative Results

Optimality We compare the proposed algorithm with the brute-force approach, i.e., directly discretize parameters and enumerate all the possible combinations. As stated in the Theorem 1, the performance of the proposed algorithm depends on the depth of the search tree. We denote the BB-IR-MRTA with a depth x as depth-x in the experiments. We randomly generate a collection of forward problem instances, each of which is associated with a particular $[\alpha, \beta, \delta]$. In experiment, we set $\hat{\alpha}_{\min} = 0.01, \hat{\alpha}_{\max} = 20$, $\hat{\beta}_{\min} = 0.01, \hat{\beta}_{\max} = 20$, $\hat{\delta}_{\min} = 0.1, \hat{\beta}_{\max} = 0.9$ and we set the discrete step size for brute force baseline as $0.1, 0.1, 0.05$ respectively. We set the number of robots and targets to eight for each instance. For each instance, we randomly generate several feasible pseudo-human suggestions (i.e., there is a solution to the inverse problem) and solve the IR-MRTA to obtain $[\hat{\alpha}, \hat{\beta}, \hat{\delta}]$. Let g^* be the optimal objective value obtained using the brute-force approach and $g(sol)$ be the one obtained using BB-IR-MRTA. To statistically compare results across all the instances, we use the normalized objective $\frac{g(sol)}{g^*}$ as the criterion in experiments. The algorithm should make this criterion close to 1. As shown in Fig. 3a, as the search depth increases the normalized objective gradually approaches one, which suggests the

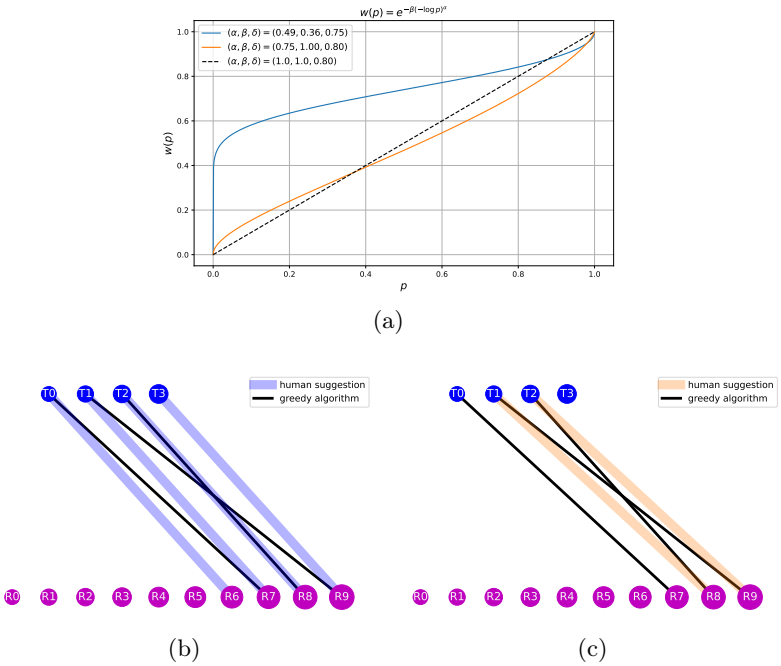

Fig. 2. A qualitative example illustrates how to use human suggestion to identify human behavior parameters using IR-MRTA. (a) Prelec's weighting functions are identified through human suggestion. The dotted line is the case $w(p) = p$. The blue line corresponds to the suggestion in Fig. 2b and the orange one corresponds to that in Fig. 2c. (b) Human suggests an aggressive allocation. The robots are in magenta and the targets are in blue. (c) Human suggests a conservative allocation.

solution obtained using BB-IR-MRTA is close to the optimal solution (after increasing the depth to eight).

Running Time and Peak Memory Usage We study the running time of the proposed algorithm in two directions. One is to study how the running time changes when the search depth increases. The other is to study how the running time changes as the size of the human suggestion (relates to the size of robots and targets) increases which will affect the size of the search tree in BB-IR-MRTA. In both cases, we set the number of robots to be equal to that of targets, and in Fig. 3b the number is set to be eight. Fig. 3b and Fig. 3c, we can observe that as the search depth (the number of robots) increases the running time will also increase but not exponentially. Such a mild increase in running time may be attributed to the pruning steps in the algorithm and the algorithm does not need to enumerate all the possible cases. By contrast, if we use brute-force approaches to solve the problem, the running time can vary between 5 minutes to 10 minutes which depends on the discretizing resolution. The results of peak memory usage

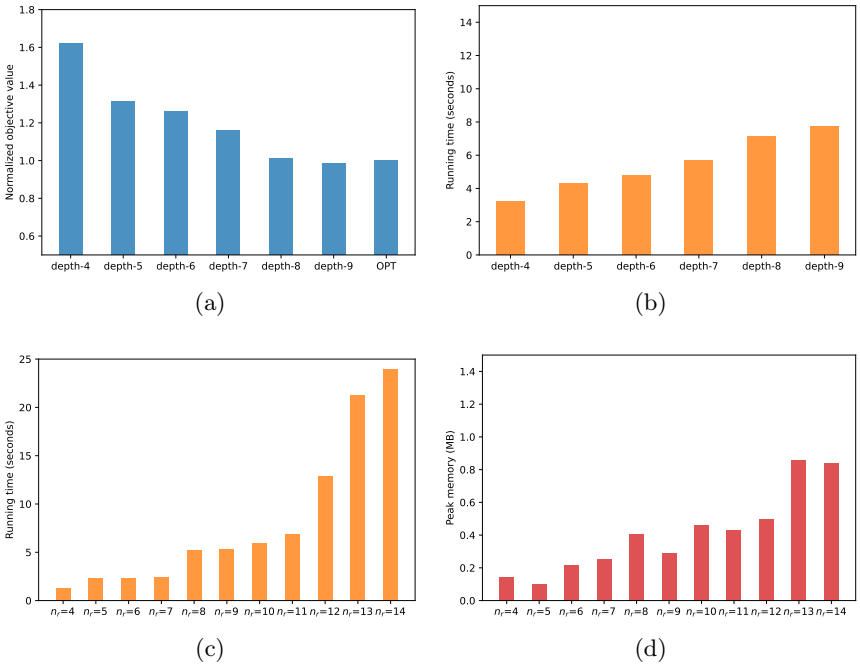

Fig. 3. (a) Optimality. Depth-x denotes the search depth of the subroutine BB-O-IR-MRTA. (b) Running time w.r.t. the search depth. (c) Running time w.r.t. the number of robots. (d) Peak memory usage.

are shown in Fig. 3d. We can observe that the memory usage will increase mildly as the search depth increases and the overall memory usage is pretty low (≤ 1 MB). These results show the possibility of deploying the proposed algorithm in embedded hardware.

7 Conclusion

We introduce a new type of multi-robot task allocation problem named IR-MRTA to accommodate human suggestions in a risk-sensitive multi-robot task allocation setup. We introduce a new branch and bound algorithm to solve IR-MRTA. We validate our formulation and algorithms through numerical simulations. In future work, we plan to extend our results in several directions. First, if a human gives multiple suggestions at the same time, how to deal with such suggestions? Second, in this paper, we assume that it is always feasible to solve the inverse problem using human suggestion. In the future, we plan to deal with the case where this is infeasible. Also of interest is the case where the forward algorithm is not greedy. How to design algorithms to solve inverse problems in this setting will be explored in the future research.

Bibliography

[1] Asghar, A.B., Shi, G., Karapetyan, N., Humann, J., Reddinger, J.P., Dotterweich, J., Tokekar, P.: Risk-aware recharging rendezvous for a collaborative team of uavs and ugvs. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 5544–5550. IEEE (2023)

[2] Bach, D.R., Dolan, R.J.: Knowing how much you don't know: a neural organization of uncertainty estimates. Nature reviews neuroscience **13**(8), 572–586 (2012)

[3] Biyik, E., Palan, M.: Asking easy questions: A user-friendly approach to active reward learning. In: Proceedings of the 3rd Conference on Robot Learning (2019)

[4] Chakraa, H., Guérin, F., Leclercq, E., Lefebvre, D.: Optimization techniques for multi-robot task allocation problems: Review on the state-of-the-art. Robotics and Autonomous Systems p. 104492 (2023)

[5] Choudhury, S., Gupta, J.K., Kochenderfer, M.J., Sadigh, D., Bohg, J.: Dynamic multi-robot task allocation under uncertainty and temporal constraints. Autonomous Robots **46**(1), 231–247 (2022)

[6] Diaz-Mercado, Y., Lee, S.G., Egerstedt, M.: Distributed dynamic density coverage for human-swarm interactions. In: 2015 American control conference (ACC). pp. 353–358. IEEE (2015)

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

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

[9] Khamis, A., Hussein, A., Elmogy, A.: Multi-robot task allocation: A review of the state-of-the-art. Cooperative robots and sensor networks 2015 pp. 31–51 (2015)

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

[11] Lin, C.W., Khong, M.H., Liu, Y.C.: Experiments on human-in-the-loop coordination for multirobot system with task abstraction. IEEE Transactions on Automation Science and Engineering **12**(3), 981–989 (2015)

[12] Nam, C., Shell, D.A.: Analyzing the sensitivity of the optimal assignment in probabilistic multi-robot task allocation. IEEE Robotics and Automation Letters **2**(1), 193–200 (2016)

[13] Özbakir, L., Baykasoğlu, A., Tapkan, P.: Bees algorithm for generalized assignment problem. Applied Mathematics and Computation **215**(11), 3782–3795 (2010)

[14] Prelec, D.: The probability weighting function. Econometrica pp. 497–527 (1998)

[15] Prorok, A.: Redundant robot assignment on graphs with uncertain edge costs. In: Distributed Autonomous Robotic Systems: The 14th International Symposium. pp. 313–327. Springer (2019)
[16] Rudolph, M., Chernova, S., Ravichandar, H.: Desperate times call for desperate measures: Towards risk-adaptive task allocation. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 2592–2597. IEEE (2021)
[17] Salkin, H.M., De Kluyver, C.A.: The knapsack problem: a survey. Naval Research Logistics Quarterly **22**(1), 127–144 (1975)
[18] Sandholtz, N., Miyamoto, Y., Bornn, L., Smith, M.A.: Inverse bayesian optimization: Learning human acquisition functions in an exploration vs exploitation search task. Bayesian Analysis **18**(1), 1–24 (2023)
[19] Schaefer, A.J.: Inverse integer programming. Optimization Letters **3**, 483–489 (2009)
[20] Shi, G., Sukhatme, G.S.: Inverse submodular maximization with application to human-in-the-loop multi-robot multi-objective coverage control. arXiv preprint arXiv:2403.10991 (2024)
[21] Šošić, A., KhudaBukhsh, W.R., Zoubir, A.M., Koeppl, H.: Inverse reinforcement learning in swarm systems. arXiv preprint arXiv:1602.05450 (2016)
[22] Terekhov, A.V., Pesin, Y.B., Niu, X., Latash, M.L., Zatsiorsky, V.M.: An analytical approach to the problem of inverse optimization with additive objective functions: an application to human prehension. Journal of mathematical biology **61**, 423–453 (2010)
[23] Tsirakos, D., Baltzopoulos, V., Bartlett, R.: Inverse optimization: functional and physiological considerations related to the force-sharing problem. Critical Reviews™ in Biomedical Engineering **25**(4-5) (1997)
[24] Wang, X., Klabjan, D.: Competitive multi-agent inverse reinforcement learning with sub-optimal demonstrations. In: International Conference on Machine Learning. pp. 5143–5151. PMLR (2018)
[25] Wang, Y., Humphrey, L.R., Liao, Z., Zheng, H.: Trust-based multi-robot symbolic motion planning with a human-in-the-loop. ACM Transactions on Interactive Intelligent Systems (TiiS) **8**(4), 1–33 (2018)
[26] Wilde, N., Blidaru, A., Smith, S.L., Kulić, D.: Improving user specifications for robot behavior through active preference learning: Framework and evaluation. The International Journal of Robotics Research **39**(6), 651–667 (2020)
[27] Wilde, N., Kulić, D., Smith, S.L.: Active preference learning using maximum regret. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 10952–10959. IEEE (2020)
[28] Xu, Z., Lin, X., Tzoumas, V.: Leveraging untrustworthy commands for multi-robot coordination in unpredictable environments: A bandit submodular maximization approach. arXiv preprint arXiv:2309.16161 (2023)
[29] Yang, F., Chakraborty, N.: Algorithm for optimal chance constrained linear assignment. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). pp. 801–808. IEEE (2017)

[30] Yu, L., Song, J., Ermon, S.: Multi-agent adversarial inverse reinforcement learning. In: International Conference on Machine Learning. pp. 7194–7201. PMLR (2019)

8 Appendix

8.1 Proof of Theorem 1

Proof. Suppose the algorithm terminates when the depth of the search tree is $n_d \geq 2$. In Algorithm 2, when we generate child nodes for root nodes (tree depth is 1), we split the set for $\hat{\beta}$ as two sets, i.e., $[\hat{\beta}_{min}, \beta]$ and $[\beta, \hat{\beta}_{max}]$. Similarly, the feasible set for $\hat{\delta}$ is split as two sets, i.e., $[\hat{\delta}_{min}, \delta]$ and $[\delta, \hat{\delta}_{max}]$. After this step (tree depth ≥ 2), when we generate child nodes for the node v_p, we will split the sets stored in v_p evenly for four child nodes, i.e., from $[\underline{\beta}, \bar{\beta}]$ and $[\underline{\delta}, \bar{\delta}]$ to $[\underline{\beta}, \frac{\underline{\beta}+\bar{\beta}}{2}], [\frac{\underline{\beta}+\bar{\beta}}{2}, \bar{\beta}]$ and $[\underline{\delta}, \frac{\underline{\delta}+\bar{\delta}}{2}], [\frac{\underline{\delta}+\bar{\delta}}{2}, \bar{\delta}]$. As a result, when the search process reaches the depth n_d, each leaf node with sets $[\underline{\beta}, \bar{\beta}]$ and $[\underline{\delta}, \bar{\delta}]$ should satisfy either $\underline{\beta} \geq \beta$ ($\underline{\delta} \geq \delta$) or $\bar{\beta} \leq \beta$ ($\bar{\delta} \leq \delta$). The size of the set satisfies either $\bar{\beta} - \underline{\beta} = \frac{\beta - \hat{\beta}_{min}}{2^{n_d - 1}}$ ($\bar{\delta} - \underline{\delta} = \frac{\delta - \hat{\delta}_{min}}{2^{n_d - 1}}$) or $\bar{\beta} - \underline{\beta} = \frac{\hat{\beta}_{max} - \beta}{2^{n_d - 1}}$($\bar{\delta} - \underline{\delta} = \frac{\hat{\delta}_{max} - \delta}{2^{n_d - 1}}$).

Since the leaf nodes include all the feasible combinations of subsets of $\hat{\beta}$ and $\hat{\delta}$, the global minimum, $\hat{\beta}^*$ and $\hat{\delta}^*$, must fall into one of them. Let leaf node v_u be the node with the lowest upper bound and v_{OPT} be the node that will generate the global minimum. It should be noted that v_{OPT} is not known to us and it can be any leaf node. Instead, we know two facts. One is that the upper bound of the objective identified in v_{OPT} is higher than that in v_u, i.e., $v_u.UB \leq v_{OPT}.UB$. Another is the optimal objective g^* should be between $v_{OPT}.LB$ and $v_{OPT}.UB$, i.e., $v_{OPT}.LB \leq g^* \leq v_{OPT}.UB$. Therefore, we have

$$v_u.UB - g^* \leq v_u.UB - v_{OPT}.LB \leq v_{OPT}.UB - v_{OPT}.LB. \tag{12}$$

Namely, the gap between our best objective identified so far and the optimal objective value is no greater than the gap between the lower bound and upper bound of the leaf node v_{OPT}. In the following, we will show how to find the upper bound for $v_{OPT}.UB - v_{OPT}.LB$.

For any leaf node v with $[\underline{\beta}, \bar{\beta}]$ and $[\underline{\delta}, \bar{\delta}]$, there are four cases.

Case1 : $\underline{\beta} \geq \beta$ and $\underline{\delta} \geq \delta$ In this case, by solving the upper and lower bound formulation, we can find that $v.UB - v.LB = 0$

Case2 : $\underline{\beta} \geq \beta$ and $\bar{\delta} \leq \delta$. In this case, the upper and lower bounds can be computed as

$$v.UB = w_\alpha \|(\alpha^* - \alpha)\| + w_\beta \|(\underline{\beta} - \beta)\| + w_\delta \|(\underline{\delta} - \delta)\|$$
$$v.LB = w_\alpha \|(\alpha^* - \alpha)\| + w_\beta \|(\underline{\beta} - \beta)\| + w_\delta \|(\bar{\delta} - \delta)\|.$$

Namely,

$$v.UB - v.LB = w_\delta (\|(\underline{\delta} - \delta)\| - \|(\bar{\delta} - \delta)\|) \leq w_\delta \|\bar{\delta} - \underline{\delta}\|$$

Case3 :$\bar{\beta} \leq \beta$ and $\underline{\delta} \geq \delta$. Similar to case 2, we have $v.UB - v.LB \leq w_\beta \|\bar{\beta} - \underline{\beta}\|$.

Case4 : $\bar{\beta} \leq \beta$ and $\bar{\delta} \leq \delta$ Similar to case 2, we have $v.UB - v.LB \leq w_\beta \|\bar{\beta} - \underline{\beta}\| + w_\delta \|\bar{\delta} - \underline{\delta}\|$.

Based on the above discussion, the Eq. (12) can be extended as

$$\begin{aligned} v_u.UB - g^* &\leq v_{OPT}.UB - v_{OPT}.LB \\ &\leq w_\beta \cdot \|\bar{\beta} - \underline{\beta}\| + w_\delta \|\bar{\delta} - \underline{\delta}\| \\ &\leq w_\beta \max\left(\frac{\beta - \hat{\beta}_{min}}{2^{n_d-1}}, \frac{\hat{\beta}_{max} - \beta}{2^{n_d-1}}\right) + w_\delta \max\left(\frac{\delta - \hat{\delta}_{min}}{2^{n_d-1}}, \frac{\hat{\delta}_{max} - \delta}{2^{n_d-1}}\right) = \epsilon. \end{aligned} \tag{13}$$

8.2 Details of Qualitative Example

The probability and reward matrices for the qualitative example are given below. In the objective, we set $w_\alpha = 1, w_\beta = 1, w_\delta = 20$.

$$p = \begin{bmatrix} 0.85 & 0.74 & 0.59 & 0.41 \\ 0.89 & 0.80 & 0.67 & 0.50 \\ 0.92 & 0.85 & 0.74 & 0.59 \\ 0.94 & 0.89 & 0.80 & 0.67 \\ 0.96 & 0.92 & 0.85 & 0.74 \\ 0.97 & 0.94 & 0.89 & 0.80 \\ 0.98 & 0.96 & 0.92 & 0.85 \\ 0.99 & 0.97 & 0.94 & 0.89 \\ 0.99 & 0.98 & 0.96 & 0.92 \\ 0.99 & 0.99 & 0.97 & 0.94 \end{bmatrix}, r = \begin{bmatrix} 67.00 & 207.59 & 347.96 & 488.23 \\ 67.15 & 207.84 & 348.29 & 488.55 \\ 67.26 & 208.05 & 348.60 & 488.91 \\ 67.34 & 208.21 & 348.87 & 489.27 \\ 67.40 & 208.34 & 349.10 & 489.60 \\ 67.43 & 208.43 & 349.28 & 489.90 \\ 67.45 & 208.49 & 349.42 & 490.15 \\ 67.45 & 208.53 & 349.51 & 490.34 \\ 67.45 & 208.55 & 349.58 & 490.49 \\ 67.43 & 208.55 & 349.62 & 490.60 \end{bmatrix}$$