Multi-robot task allocation for safe planning against stochastic hazard dynamics

Daniel Tihanyi, Yimeng Lu, Orcun Karaca, and Maryam Kamgarpour

Abstract—We address multi-robot safe mission planning in uncertain dynamic environments. This problem arises in safety-critical exploration, surveillance, and emergency rescue missions. The multi-robot optimal control problem is challenging because of the dynamic uncertainties and the exponentially increasing problem size with the number of robots. Leveraging recent works obtaining a tractable safety maximizing plan for a single robot, we propose a scalable two-stage framework. Specifically, the problem is split into a low-level single-agent problem and a high-level task allocation problem. The low-level problem uses an efficient approximation of stochastic reachability for a Markov decision process to derive the optimal control policy under dynamic uncertainty. The task allocation is solved using forward and reverse greedy heuristics and in a distributed auction-based manner. Properties of our safety objective enable provable performance bounds on the safety of the approximate solutions of the two heuristics.

Index terms—stochastic reachability, optimal control, task allocation, greedy algorithms, multi-robot systems

I. INTRODUCTION

Autonomous robots are increasingly used in safety-critical applications including surveillance [1] and emergency rescue missions [2]. Safety against dynamic uncertainties, such as moving obstacles and evolving hazards is indispensable in such applications. A natural idea to improve safety is to use multiple robots to reduce the execution time by working in parallel or to increase robustness due to redundancy [3], [4].

In an emergency rescue scenario, the objective of visiting a set of target locations (e.g., to save survivors) can be fulfilled collectively by a team of robots, where each robot is assigned a safety-maximizing trajectory visiting a subset of these targets. This objective can be formulated as a multi-robot optimal control problem for a Markov decision process. The challenge in solving this problem is twofold. The first is computing a target-robot assignment maximizing the safety and success of all robots. The second is tractably solving for the trajectories of the robots. Past works provide methods based on dynamic programming to solve the latter through probabilistic safety and reachability [2], [5], [6]. Leveraging an efficient implementation from these, we provide a scalable two-stage framework for solving the multi-robot task/target allocation to maximize the safety of the mission.

The number of all possible task assignments to consider grows exponentially with the number of robots and targets, generalizing the NP-hard set partitioning problem [7]. Moreover, in order to not jeopardize safety of the mission, it is desirable to formalize the safety objective in task allocation and to derive guarantees on the performance/safety of the task allocation algorithms. To address the complexity, auction-based approaches [8] consider the robots as bidders who iteratively submit bids for the most desirable tasks. Such iterative and distributed assignments of tasks are instances of the forward greedy heuristics, which could also provide suboptimality guarantees [9]. In contrast with the existing auction-based approaches, our objective function originates from the underlying stochastic optimal control problem [10]. The collective goal of multi-robot planning is to maximize a safety objective, and in our case, we propose a multiplicative objective form that allows a distributed implementation of greedy heuristics. Hence, our proposed methods are variations on the auction-based approaches, in which the safety objective is incorporated into the bids and allocations.

Greedy algorithms are equipped with provable performance bounds when the objective functions satisfy sub- or supermodularity assumptions [9]. However, we will show that our safety objective is nonsubmodular and nonsupermodular. Past studies on auction-based approaches either do not mention any optimality guarantee, or when they do, their problem formulations do not capture nonsub- and nonsupermodular objective functions. In the studies of [11] and [12], a set partitioning problem is formulated. This problem is based on a bipartite graph requiring fixed target-robot pair costs for edges (that is, additive/modular objective functions). Our objective associates a different nonadditive cost to each possible configuration of tasks that could be allocated to robots, which makes these methods inapplicable. Other methods in the literature for task allocation problems assume submodular objective functions, e.g., [13]. In terms of safe task allocation, there are studies where the objective is either the conditional value-at-risk cost [14], or the worst-case cost [15]. Objectives in these works are also additive. To the best of our knowledge, general nonsub- and nonsupermodular objective functions have not been addressed in any of the existing task allocation studies or in related applications of set partitioning problems.

We show through numerical studies that our objective is weakly sub- and supermodular, notions characterized by submodularity ratio and curvature, respectively. Thus, we leverage recent theoretical results from [16]–[18] to obtain

1The forward greedy refers to the greedy heuristic that adds elements iteratively, whereas the reverse greedy refers to the one that removes.
safety guarantees on our auction-based algorithms. To the best of our knowledge, this work is the first to demonstrate the benefits of a novel auction-based task allocation using the reverse greedy algorithm both in theory and in numerics.

Our contributions are summarized as follows: (i) We develop a scalable two-stage framework for an emergency rescue scenario by splitting the multi-robot control synthesis problem into a safe planning problem (for each robot) and a task allocation. To this end, we utilize an efficient implementation of a single-robot plan under dynamic uncertainties. (ii) We introduce two greedy variants, the forward and the reverse. We show that the multiplicative safety formulation of the objective decouples the individual robots’ optimal control problems under a fixed task assignment. This enables a distributed auction-based implementation. Utilizing weak sub- and supermodularity properties we provide performance guarantees on the safety of the forward and the reverse solutions. (iii) We compare these two greedy algorithms in terms of their theoretical guarantees, and computational and practical performance in numerical case studies. Theoretical analyses suggest that reverse greedy can have a better guarantee than the forward greedy algorithm in a larger range of problem instances. However, this improved theoretical guarantee of the reverse greedy comes with an increased computational complexity. In terms of empirical performance, we observe that both algorithms perform similar and close to optimality.

Code is available at github.com/TianyiID/multi_alloc. Extended version and online appendix are available at [19].

II. PROBLEM FORMULATION AND STATEMENT

Consider a team of autonomous robots operating in an environment containing obstacles (e.g., walls), a set of targets (e.g., survivors), and a stochastically evolving hazard (e.g., fire or toxic contamination). The goal of the robots is to visit the targets and exit while avoiding unsafe locations.

A. Modeling the environment and the robots

Assumptions. The map of the environment including the obstacles, the location of the targets, and hazard sources are known a priori. The hazard spreads with a known stochastic model. The map consists of cells arranged in a discrete grid. We assume robots can avoid collisions locally, hence, a cell can be occupied by multiple robots simultaneously.

Map model. A map has been provided for our case studies, see Figure 2. We let \( M_{m \times n} \) be a set containing the coordinates of an \( m \times n \)-sized grid and \( O \subseteq M_{m \times n} \) be a set of obstacles, \( X = M_{m \times n} \setminus O \) is the set of free cells. For a cell \( x \in X \), let \( N(x) \) be the set of neighboring cells.

Robots and targets. Define the set of robots as \( R = \{1, \ldots, |R|\} \) and the target set as \( T \subset X \).

Robot motion. Possible inputs correspond to the direction the robot can move: \( U = \{\text{Stay}, \text{North, East, South, West}\} \). Thus, \( d_{\text{stay}} = (0, 0) \), \( d_{\text{north}} = (0, 1) \), \( d_{\text{east}} = (1, 0) \), \( d_{\text{south}} = (0, -1) \), \( d_{\text{west}} = (-1, 0) \). In each position \( x \in X \), \( U(x) = \{u \in U \mid x + d_u \in X\} \subseteq U \), are the inputs available to the robot. The motion of the robot is defined by a stochastic transition kernel \( x_{k+1} \sim \tau_X(x, u_k) \), \( k \in \{0, 1, \ldots\} \) with initial position \( x^0 \in X \), where \( \tau_X : X \times X \times U \to [0, 1] \) denotes the probability of transiting from \( x_k \in X \) at time step \( k \) to \( x_{k+1} \in N(x) \) at \( k + 1 \) under control input \( u_k \in U(x_k) \). Our case studies utilize deterministic inputs: \( \tau_X(x_{k+1} \mid x_k, u_k) = \delta(x_{k+1}, x_k + d_u) \), with \( \delta(i, j) \) being the Kronecker-delta.

Hazard spread. Let \( Y = 2^X \) be the hazard state space. Each element \( y \in Y \) with \( y \subset X \) denotes a set of contaminated cells. The stochastic Markov process \( y_{k+1} \sim \tau_Y(y, k) \), \( k \in \{0, 1, \ldots\} \) defines the hazard evolution dynamics with transition kernel \( \tau_Y : Y \times Y \to [0, 1] \) from state \( y_k \in Y \) at time step \( k \) to \( y_{k+1} \in Y \) at time step \( k + 1 \). In case of fire, an estimation of the probability of fire spread from a given cell to the neighboring cells can be used to derive \( \tau_Y \), see [20]. Our modeling choice for numerical studies is provided in [19, App. A].

B. Problem statement for multi-robot multi-task safe control

The mission of the robot fleet is to visit every target and exit the hazard site safely (e.g., a rescue scenario). Implementing safety as a hard constraint by considering the worst-case hazard spread generally yields infeasibility. To this end, our objective is instead to determine each robot’s control input at every time step to maximize the probability of completing the mission within a given large enough time horizon \( N \in \mathbb{N}_{>0} \), while avoiding the stochastically evolving hazard. By considering the robot fleet as one system, whose state-space is the product space of each robot’s state-space \( X^{\{R\}} \), this objective can be formalized as a stochastic reachability problem for a Markov decision process [21], [22]. The stochastic reachability problem can then be solved using the dynamic programming principle. Given that maximizing the mission success probability relies on dynamic programming on the robots’ product space, the computation of a multi-robot optimal control policy is intractable. This motivates a two-stage approach.

III. TWO-STAGE MULTI-ROBOT SAFE PLANNING

We present a scalable framework to control the robot fleet to maximize safety. We split the problem into: low-level path planning (optimizing control policies of each robot for assigned targets) and high-level task allocation.

A. Single-robot path planning

Leveraging the works of [2], [5], [10], we provide a tractable solution to the low-level safe planning problem for a single-robot system and a given set of targets. This serves as a building block for high-level task allocation.

1) Definition of the stochastic process

The discrete-time stochastic process is defined by its state space and transition kernels as follows.

State space. Robot state – At each time step \( k \) we keep track of the 2D position of the robot \( x_k \in X \) on the map.

Task execution – During the execution, the robot needs to keep track of the visited targets. Define \( T_r \subset T \subset X \) as the target list, the set of targets assigned to robot \( r \). Define the set \( Q = 2^{\tau_r} \) and the target execution state \( q^k \in Q \) at time step \( k \). The state \( q^k \) tracks the visited targets.
**Hazard state** — Including the hazard state $y_k$ in the state space results in computational intractability. Thus, introduce a single hazard state denoted by $s_H$ to capture when the robot enters a hazardous cell, namely, $x^k \in y_k$. A robot reaching the hazard state indicates an unsuccessful mission.

Combining the elements mentioned above, define the state space $S = \{s_H\} \cup (Q \times X \setminus \{(q,x) | x \in T_r \land x \notin q\})$. We specify the goal location as $x_G \in X$. This refers to the exit of the map. We also define the goal state denoted by $s_G = (T_r, x_G)$. The state $s_G$ indicates a successful mission, where every target is visited and the robot reaches the safe goal location. Finally, we define the initial state for robot $r$ as $s_r^0 = (0, x_r^0)$, where no targets are visited and the robot is at its initial position $x_r^0$. The state $s_r^0$ is known to the robot.

**Transition probabilities of the stochastic process.** Let $\tau^k_S : S \times S \times U \rightarrow [0, 1]$ be the transition kernel at time step $k$. The quantity $\tau^k_S(s^k, u^k)$ represents the probability of a state transitioning from $s^k$ to $s^{k+1}$ given the control input $u^k$ at time $k$. This kernel can be computed using the robot dynamics ($T_X$), the execution of the tasks, and the uncertain hazard dynamics ($T_Y$). We provide it in [19, App. B].

2) Controller synthesis via dynamic programming

Let $\pi = \{\mu^0, \ldots, \mu^{N-1}\}$ be a control policy, where $\mu^k : S \rightarrow U$ refers to the control law at time step $k$. With the state-space defined above, the objective of mission success defined in Section II-B can be cast as a stochastic reachability problem: finding a control policy so that the probability of reaching the goal state $s_G$ within a given time horizon $N \in \mathbb{N}_{>0}$ is maximized. Parameter $N$ is chosen to be sufficiently long to ensure the robot can indeed reach the targets.

For the remainder, $f_r(T_r)$ denotes the probability of reaching the goal state of robot $r$ under the optimal control policy $\pi^*(T_r)$ for a given target list $T_r$, whereas $f_r(\pi, T_r)$ denotes the probability of success under the control policy $\pi$. Moreover, we will use the terms success and safety probabilities interchangeably, and both will imply the execution of the assigned tasks by the robot (or the robot fleet). Given these definitions, our goal is to solve

$$\pi^*_r(T_r) = \arg \max_{\pi} f_r(\pi, T_r).$$  \hspace{1cm} (1)

Problem (1) of a single robot can be solved in a tractable way, similar to the dynamic programming approach to a stochastic control problem, see the algorithm [19, Alg. 1] in our extended version. Note that it can easily be verified that $f_r(\cdot)$ is nonincreasing as a function of set of tasks.

B. Multi-robot task allocation

The high-level stage of our framework assigns the targets to the robots to maximize a collective objective. This part builds on the solutions obtained by the low-level stage described above. A feasible task allocation assigns each target to exactly one robot: partitioning the set $T$ into $\{T_r\}_{r \in R}$. To be more specific, $T_r \subset T$ for all $r \in R$, $T_r \cap T_{r'} = \emptyset$ for any pair $r, r' \in R$ where $r \neq r'$ and $\bigcup_{r \in R} T_r = T$.

**Objective function.** The goal is to find a feasible task allocation that maximizes the probability of group safety, that is, the probability of all robots safely finishing their missions (assigned tasks) simultaneously. Denote this probability by $f_R(\{T_r, \pi^*_r\}_{r \in R})$ for a fixed set of assigned tasks and control policies, and its optimizer by $\{T^*_r, \pi^*_r\}_{r \in R}$. However, maximizing the probability of group safety is computationally challenging since it requires formulating the stochastic reachability of Section III-A.2 over a product state space $X^{[R]}$ considering the whole robot fleet in a centralized manner. Thus, we introduce the multiplicative group safety as an approximation of this objective function. It is defined as $F(\{T_r\}_{r \in R}) = \prod_{r \in R} f_r(T_r)$, where the values of $f_r(T_r)$ are obtained by solving the single-robot path planning problem introduced in Section III-A.

Note that the robots use the same map under the same hazard state evolution, thus, the safety of individual robots are not independent events. This implies that the probability of group safety can differ from the multiplicative group safety. To support this objective function choice, we now show that the multiplicative group safety is a lower bound to the probability of group safety under a mild assumption. Specifically, the assumption requires that conditioning on the success/safety of other robots does not decrease the probability of a robot successfully completing its own set of tasks.

**Assumption 1:** Consider a fixed task allocation and a control policy for each robot. Let $E_r$ denote the random variable with support $\{0, 1\}$ indicating the safety of robot $r$ by taking the value $1$ with probability $P(E_r = 1)$. We assume $P(E_r = 1) \prod_{r' \in R \setminus \{r\}} E_{r'} = 1 \geq P(E_r = 1)$ for any $r \in R$ and $R' \subset R$.

In realistic examples, the assumption above holds since an observation regarding the safety of other robots confirms that the hazard did not propagate in certain regions. In a more general setting, our condition in Assumption 1 is also implied whenever knowing that a cell is safe at any time point does not decrease the safety probability of any other cell at any future time. This is also a reasonable assumption and can be verified numerically when considering fire models as in [20].

**Proposition 1:** Under Assumption 1, multiplicative group safety is a lower bound to the probability of group safety: $f_R(\{T^*_r, \pi^*_r\}_{r \in R}) \geq f_R(\{T_r, \pi^*_r\}_{r \in R}) \geq F(\{T_r\}_{r \in R})$.

Proof: Observe that $P(\prod_{r \in R} E_r = 1) = \prod_{r \in R} P(E_r = 1) = \prod_{r \in R} P(E_r = 1)$ via chain rule. Clearly, the condition in the proposition above yields $\prod_{r \in R} P(E_r = 1) = \prod_{r \in R} F(\{T_{r'}\}_{r' \in R}) = \prod_{r \in R} P(E_r = 1)$. Thus, we obtain $P(\prod_{r \in R} E_r = 1) \geq \prod_{r \in R} P(E_r = 1)$ under a change of notation, this is equivalent to $f_R(\{T_r, \pi^*_r\}_{r \in R}) \geq \prod_{r \in R} f_r(T_r) = F(\{T_r\}_{r \in R})$. The remaining inequality in the proposition statement, $f_R(\{T^*_r, \pi^*_r\}_{r \in R}) \geq f_R(\{T_r, \pi^*_r\}_{r \in R})$, follows from the optimality of task assignments and policies listed in $\{T^*_r, \pi^*_r\}_{r \in R}$. This concludes the proof.

The result above implies that the value of the introduced multiplicative objective function at a given policy implies a lower bound on the probability of group safety of the policy. For the remainder, we consider the multiplicative group safety to be our success/safety measure.

**Optimization problem.** The task allocation problem for
Algorithm 1: Forward Distributed Greedy Algorithm

Input: $R$, $T$, $\{f_r\}_{r \in R}$
Output: $\{T^{fg}_r = T_r^{(k)}\}_{r \in R}$

1 begin
2 \begin{algorithmic}
3 \end{algorithmic}

maximizing safety can thus be formulated as

\[ F^* = \max_{\{T_r\}_{r \in R}} \prod_{r \in R} f_r(T_r) \quad (2) \]

s.t. \[ T_r \cap T_{r'} = \emptyset, \forall r \neq r', \cup_{r \in R} T_r = T. \]

The problem above generalizes set partitioning, which is NP-hard [7]. We highlight that none of the past works we reviewed in the introduction had a safety objective as in (2).

IV. GREEDY HEURISTICS

To solve (2) in tractable way, we introduce two greedy heuristics, the forward and the reverse. From the practical standpoint, these algorithms iteratively update the task allocation by adding or removing one task-robot pair at each step. Moreover, thanks to its multiplicative form, our objective can be implemented in an auction-based fashion. At each iteration, each robot chooses a task bids from the previous iteration (see Line 8). The rest of the robots simply submit their bids from the previous iteration (see Line 8).

A. Forward greedy algorithm

The forward greedy algorithm (see Algorithm 1) is initialized with all robots submit a bid (see Line 4–8), which consists of the pair $(t^k_r, f^k_r)$. Each robot $r$ chooses the task $t^k_r$ from the list of unallocated tasks $j^{k-1}$, such that it obtains the best optimality gain $\delta^k_r$ with respect to the individual objective function $f_r$. After collecting all the bids, we choose the robot $r^k$ which generates the best optimality gain with respect to the collective objective: the multiplicative group safety $F$ (Line 9). Due to our auction-based formulation in this line, we can choose the task-robot pair with the best collective gain efficiently. Between Lines 10–13, we simply set the values of $f^k_r, T^k_r$ for all $r \in R$ and $R^k, J^k$ according to our choice from the task allocation.\footnote{Note that only the robots choosing the same task as $r^k$ (i.e., $r$ such that $t^k_r = t^k_{r^k}$) have to update their bids in the next iteration (see Line 12 at step $k$ and Line 4 at step $k+1$). The rest of the robots simply submit their bids from the previous iteration (see Line 8).}

We will later see that the solution obtained by this algorithm will give rise to a performance guarantee as a function of $F(\{T_r\}_{r \in R})$, the safety of the initial allocation without any tasks assigned but with only the goal of reaching the exit. This could potentially take a large value as no task is needed, e.g., 1, which could deteriorate the performance guarantee.

B. Reverse greedy algorithm

The reverse greedy algorithm is initialized with all tasks being allocated to every robot simultaneously. Due to this reason, its performance guarantee will instead be a function of $F(\{T_r\}_{r \in R})$. This algorithm iteratively updates its provisional allocation by removing tasks from the robots. In each step, the task-robot pair causing the largest optimality loss is removed. It converges when every task is allocated to exactly one robot. As is the case for the forward greedy, computation of the iteration steps is distributed among the robots. Since the reverse greedy implementation uses similar principles to those found in the forward, its detailed description is relegated to [19, App. C].

C. Performance guarantees

To discuss the theoretical guarantees for Algorithms 1 and [19, App. C: Alg. 3], we bring in the definitions for the curvature $\alpha$ and the submodularity ratio $\gamma$.

Definition 1: Curvature of a nonincreasing $F$ is the smallest $\alpha \in \mathbb{R}_+$ such that $(1 - \alpha) \cdot F(B \cup \{e\}) - F(B) \geq F(A \cup \{e\}) - F(A)$, for all $A \subseteq B \subseteq W$, for all $e \in W \setminus B$. $F$ is supermodular iff $\alpha = 0$, and we have $\alpha \in [0, 1]$.

Definition 2: Submodularity ratio of a nonincreasing $F$ is the largest $\gamma \in \mathbb{R}_+$ such that $\gamma \cdot [F(A \cup \{e\}) - F(A)] \geq F(B \cup \{e\}) - F(B)$ for all $A \subseteq B \subseteq W$, for all $e \in W \setminus B$. $F$ is submodular iff $\gamma = 1$, and we have $\gamma \in [0, 1]$.

We will see in the numerics that our objective function does not exhibit submodularity or supermodularity. Instead, we bring in the submodularity ratio and curvature properties above describing how far a nonsubmodular or nonsupermodular set function is from being submodular or supermodular, respectively. Calculating these values is computationally expensive. In our numerical case studies, we will verify our objective function $F$ to be strictly decreasing after each additional task assignment. Strict monotonicity is
Fig. 1: Comparison of guarantees. For each optimal $F^*$ value, the shaded regions represent the $(\alpha, \gamma)$ pairs for which the forward greedy outperforms the reverse greedy.

a sufficient condition for non-trivial values for submodularity ratio ($\gamma > 0$) and curvature ($\alpha < 1$) [17]. Moreover, we will compute ex-post bounds from the greedy function evaluations, verifying also numerically that the objective is indeed nonsubmodular and nonsupernormal.

Let $F^*$ denote the optimal value of (2). We can now invoke two performance guarantees from our recent work.

**Theorem 1:** [18, Thm. 1] Let $F_{fg}$ denote the objective of the forward greedy solution from Algorithm 1. We then have

$$\frac{F_{fg} - F(\emptyset_{T \in R})}{F^* - F(\emptyset_{T \in R})} \leq \frac{1}{\gamma \cdot (1 - \alpha)}.$$ 

**Theorem 2:** [18, Thm. 2] Let $F_{rg}$ denote the objective of the reverse greedy solution from [19, App. C]. We then have

$$\frac{\gamma}{1 + \gamma \cdot \alpha} \leq \frac{F_{rg} - F(\emptyset_{T \in R})}{F^* - F(\emptyset_{T \in R})}.$$ 

**Comparison of the two performance guarantees.** Both guarantees involve the objective evaluated at their initial step as a reference. For the forward greedy, this is the empty allocation, which entails that the robots are safe with high probability. In other words, safety objective $F(\emptyset_{T \in R}) \approx 1$ generally takes a high value. For the reverse greedy, this is the fully redundant allocation resulting in a low probability of success $F(\emptyset_{T \in R}) \approx 0$, since the robots are overwhelmed with the tasks and with the increased danger of being contaminated. Assuming these values, the guarantees are

$$\frac{F^*}{\gamma \cdot (1 - \alpha)} + \frac{\gamma \cdot (1 - \alpha) - 1}{\gamma \cdot (1 - \alpha)} = g_{fg}(\alpha, \gamma, F^*) \leq F_{fg},$$

$$\frac{F^*}{1 + \gamma \cdot \alpha} = g_{rg}(\alpha, \gamma, F^*) \leq F_{rg}.$$ 

In (3), $g_{fg}(\alpha, \gamma, F^*)$ and $g_{rg}(\alpha, \gamma, F^*)$ provide lower bounds on the probabilities of success: $F_{fg}$ and $F_{rg}$. We can already see that the reverse greedy guarantee directly relates to the optimal value $F^*$, whereas as the forward greedy guarantee can even take negative values due to additional terms.

Figure 1 illustrates the area defined by $(\alpha, \gamma) \in [0,1] \times [0,1]$ such that $g_{fg}(\alpha, \gamma, F^*) \geq g_{bg}(\alpha, \gamma, F^*)$ for fixed values of $F^*$.

Fig. 2: The example environment. The robots collectively visit all the targets and reach the goal while avoiding the evolving hazard. Generated paths are shown for brute force optimal task allocation.

| Algorithm         | Computation time | Success probability |
|-------------------|------------------|---------------------|
| Forward Greedy    | 7 minutes        | 0.699               |
| Reverse Greedy    | 29 minutes       | 0.717               |
| Brute Force       | 4 hours 53 minutes | 0.717               |

Observe that if $F^*$ is close to 1, using the forward greedy could be a better choice. However, if $F^* \leq 0.5$, the reverse greedy provides a better guarantee for all possible $(\alpha, \gamma)$ pairs. As the value of $F^*$ decreases from 1 to 0.5, the area where the forward greedy algorithm is more reliable shrinks. For these ranges of values, the performance guarantee of the forward greedy is better only when the function is close to being both supermodular and submodular. In fact, [17, Prop. 4 and 5] prove that there is no performance guarantee for the forward greedy unless both the submodularity ratio and the curvature are utilized. Unless $F^*$ is expected to be sufficiently large, theory suggests implementing the reverse greedy for a larger range of problem instances.

**V. Numerical results**

We present a case study for the two-stage multi-robot safe planning framework in Section III. The code is available at github.com/TihanyiD/multi_alloc. For the high-level task allocation stage, we implement the forward and the reverse greedy from Section IV. The example is tailored such that we can compute the optimal allocation via brute force for performance comparison. For larger examples, it would not be possible to compute it. The environment is a 17-by-13 grid map with the initial state in Figure 2. The time horizon is $N = 75$ steps, which is long enough for each robot to visit all the targets in the grid. Five hazard sources are illustrated by their initial positions. The hazard dynamics are described in [19, App. A]. To visualize the evolution of the hazard, the heat map in Figure 2 shows the probability of a grid cell being hazardous within $N = 75$ steps.

3Measured on a computer equipped with Core i7 (2.6GHz), 8GB RAM.
Performance results. Table I compares the solutions from three task allocation methods. The optimal task allocation and the corresponding robot paths can be found in Figure 2. The forward and the reverse greedy algorithms provide tractable approximations. The computation times illustrate the clear benefits of greedy. Both greedy algorithms are faster than the brute force by order of magnitude without significant optimality loss (in reverse greedy, there is no loss at all).

The reverse greedy algorithm takes significantly more time than the forward greedy. We explain this by the following points. First, in Algorithm 1, the forward greedy takes $|T|$ steps, whereas the reverse greedy ([19, App. C]) takes $|T| \cdot (|R| - 1)$ steps. Second, the computation time for evaluating the solution of the single-robot safe planning problem (see Section III-A) for a subset of tasks $T \subset T'$ depends greatly on the number of targets $|T_r|$ allocated to a robot. The target execution state $|Q| = 2^{|T_r|}$ grows exponentially with $|T_r|$, which has a major effect on computational complexity. The forward greedy explores the cases where a smaller number of tasks are allocated to the robots, as it is initialized with no tasks assigned to any robot. In contrast, the reverse greedy is initialized with all tasks being allocated to every robot and removes tasks gradually. Hence, the reverse requires solving larger instances of the single-robot problem.

Properties of the safety objective and the guarantees. There is no computationally tractable approach to obtaining the exact curvature $\alpha$ and the exact submodularity ratio $\gamma$ of $F$. However, we can confirm that these ratios are non-trivial, $\alpha < 1$ and $\gamma > 0$, because we numerically verified that $F$ is strictly decreasing. Moreover, we obtained ex-post bounds called greedy-approximate curvature $\alpha^G \leq \alpha$ and greedy-approximate submodularity ratio $\gamma^G \geq \gamma$, see their definitions in [23]. We calculated these bounds using only the function evaluations during the execution of the greedy algorithms instead of taking all possible allocations into account. In literature, they are commonly used as computationally efficient alternatives. For this particular example, we obtained the values $\alpha^G = 0.989$ and $\gamma^G = 0.525$. Since $\alpha^G > 0$ and $\gamma^G < 1$, we can verify that the objective function is indeed nonsupermodular and nonsubmodular, respectively. Although the guarantees of Theorems 1 and 2 do not necessarily hold for $\alpha^G$ and $\gamma^G$, evaluating (3) at $\alpha^G$ and $\gamma^G$ suggests that the reverse greedy may essentially have a better performance guarantee than the forward greedy for this particular problem instance. This can be attributed to the fact that the function $F$ is far away from being supermodular, $\alpha^G = 0.989$, and this deteriorates the bound in Theorem 1. This is confirmed by the empirical performances in Table 1.

Additional results. We provide two additional studies in [19, App. D and App. E], including a large set of randomized examples for different and larger number of robots and tasks.

VI. CONCLUSION

We proposed a two-stage framework to solve a multi-robot safe planning problem in a tractable manner. An efficient implementation of a stochastic reachability addressing safe planning under dynamic uncertainties served as the low-level planner. The multiplicative safety objective allowed implementations of the forward and reverse greedy in a distributed manner to allocate the tasks. Through case studies, we compared our solutions with the computationally intractable optimal solution. We illustrated that our algorithms perform well both in terms of computation time and optimality. The reverse greedy can have a better performance guarantee than the forward greedy, but this benefit came with an increased computational burden.

REFERENCES

[1] D. Di Paola, A. Milella, G. Cicirelli, and A. D’Anza, “An autonomous mobile robotic system for surveillance of indoor environments,” Int. J. of Adv. Rob. Syst., vol. 7, no. 1, p. 8, 2010.
[2] T. A. Wood, S. Summers, and J. Lygeros, “A stochastic reachability approach to emergency building evacuation,” in 52nd CDC. IEEE, 2013, pp. 5722–5727.
[3] Z. Yan, N. Jouandeau, and A. A. Cherif, “A survey and analysis of multi-robot coordination,” Int. J. of Adv. Rob. Syst., vol. 10, no. 12, p. 399, 2013.
[4] A. Khamis, A. Hussein, and A. Elmogy, “Multi-robot task allocation: A review of the state-of-the-art,” Coop. Rob. and Sens. Netw., pp. 31–51, 2015.
[5] Y. Lu and M. Kamgarpour, “Safe mission planning under dynamical uncertainties,” in ICRA. IEEE, 2020.
[6] S. C. Smith, J. Turnova, C. Bela, and D. Rus, “Optimal path planning for surveillance with temporal-logic constraints,” The Int. J. of Rob. Res., vol. 30, no. 14, pp. 1695–1708, 2011.
[7] L. A. Wolsey and G. L. Nemhauser, Integer and combinatorial optimization. John Wiley & Sons, 1988, vol. 55.
[8] M. G. Lagoudakis, E. Markakis, D. Kenpe, P. Keskinocak, A. J. Kleywegt, S. Koening, C. A. Tovey, A. Meyerison, and S. Jain, “Auction-based multi-robot routing,” in Rob.: Science and Syst., vol. 5. Rome, Italy, 2005, pp. 343–350.
[9] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, “An analysis of approximations for maximizing submodular set functions—i,” Math. Program., vol. 14, no. 1, pp. 265–294, 1978.
[10] T. A. Wood and M. Kamgarpour, “Automaton-based stochastic control for navigation of emergency rescuers in buildings,” in 2016 IEEE Conf. on Control Applications (CCA). IEEE, 2016, pp. 587–592.
[11] M. Khoo, T. A. Wood, C. Manzie, and I. Shames, “A distributed augmenting path approach for the bottleneck assignment problem,” arXiv preprint arXiv:2011.09606, 2020.
[12] S. Chopra, G. Notarstefano, M. Rice, and M. Egerstedt, “A distributed version of the hungarian method for multi-robot assignment,” IEEE Transactions on Robotics, vol. 33, no. 4, pp. 932–947, 2017.
[13] T. Li, H.-S. Shin, and A. Tsourdos, “Threshold greedy based task allocation for multiple robot operations,” arXiv preprint arXiv:1909.01239, 2019.
[14] C. Nam and D. A. Shell, “Analyzing the sensitivity of the optimal assignment in probabilistic multi-robot task allocation,” IEEE Rob. and Aut. Letters, vol. 2, no. 1, pp. 193–200, 2016.
[15] S. S. Ponda, L. B. Johnson, and J. P. How, “Distributed chance-constrained task allocation for autonomous multi-agent teams,” in ACC. IEEE, 2012, pp. 4528–4533.
[16] B. Guo, O. Karaca, T. Summers, and M. Kamgarpour, “Actuator placement for optimizing network performance under controllability constraints,” in 58th CDC. IEEE, 2019, pp. 7140–7147.
[17] ———, “Actuator placement under structural controllability using forward and reverse greedy algorithms,” IEEE Trans. on Aut. Contr., vol. 66, no. 12, pp. 5845–5860, 2020.
[18] O. Karaca, D. Tihanyi, and M. Kamgarpour, “Performance guarantees of forward and reverse greedy algorithms for minimizing nonsupermodular nonsubmodular functions on a matroid,” Oper. Res. Letters, vol. 49, no. 6, pp. 855–861, 2021.
[19] D. Tihanyi, Y. Lu, O. Karaca, and M. Kamgarpour, “Multi-robot task allocation for safe planning against stochastic hazard dynamics,” arXiv:2103.01840, online appendix, 2022.
[20] T. Beer and I. Einzing. “Fire spread and percolation modelling,” Math. and Comp. Model., vol. 13, no. 11, pp. 77–96, 1990.
[21] A. Abate, M. Prandini, J. Lygeros, and S. Sastry, “Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems,” Automatica, vol. 44, no. 11, pp. 2724–2734, 2008.
[22] S. Summers, M. Kamgarpour, C. Tomlin, and J. Lygeros, “Stochastic system controller synthesis for reachability specifications encoded by random sets,” Automatica, vol. 49, no. 9, pp. 2906–2910, 2013.
[23] A. A. Bian, J. M. Buhmann, A. Krause, and S. Tschiatschek, “Guarantees for greedy maximization of non-submodular functions with applications,” in ICML. PMLR, 2017, pp. 498–507.