Multi-robot task allocation for safe planning under dynamic uncertainties

Daniel Tihanyi Yimeng Lu Orcun Karaca Maryam Kamgarpour

March 3, 2021

Abstract

This paper considers the problem of multi-robot safe mission planning in uncertain dynamic environments. This problem arises in several applications including safety-critical exploration, surveillance, and emergency rescue missions. Computation of a multi-robot optimal control policy is challenging not only because of the complexity of incorporating dynamic uncertainties while planning, but also because of the exponential growth in problem size as a function of the number of robots. Leveraging recent works obtaining a tractable safety maximizing plan for a single robot, we propose a scalable two-stage framework to solve the problem at hand. Specifically, the problem is split into a low-level single-agent planning 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 handle the dynamic uncertainty. The task allocation, on the other hand, is solved using polynomial-time forward and reverse greedy heuristics. The safety objective of our multi-robot safe planning problem allows an implementation of the greedy heuristics through a distributed auction-based approach. Moreover, by leveraging the properties of the safety objective function, we ensure provable performance bounds on the safety of the approximate solutions proposed by these two heuristics. Our result is illustrated through case studies.

Keywords: multi-robot systems, motion and path planning, stochastic reachability, task allocation, greedy algorithms.

1 Introduction

Autonomous robots are increasingly used in safety-critical applications including surveillance [1–5] and emergency rescue missions [6]. 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. This setup is commonly used in situations where the workload can be distributed to the robots to reduce the execution time by working in parallel [7–9], and to increase robustness due to redundancy [10].

* *Corresponding author: D. Tihanyi.*

D. Tihanyi, Y. Lu, and O. Karaca are with D-ITET, ETH Zürich, Switzerland. emails: {tihanyid, luyi,okaraca}@ethz.ch M. Kamgarpour is with the Electrical and Computer Eng. at the University of British Columbia, Canada. email: maryamk@ece.ubc.ca.

In an emergency rescue scenario, the objective of visiting all the targets (e.g., to save survivors) can be fulfilled collectively by a team of robots, where each robot is assigned a safety-maximizing path visiting a subset of these targets. This can be formulated as a multi-robot optimal control synthesis problem for a Markov decision process. However, the challenge is twofold. The first is handling the dynamic uncertainty and the size of the hazard space in a tractable manner, whereas the second is the growth of the problem size with the number of robots. Past work provides methods based on dynamic programming to solve a single-robot path planning problem under dynamic uncertainty [11–15]. Leveraging an efficient implementation of this approach, this work provides a scalable framework offering safety guarantees by splitting the multi-robot planning problem under dynamic uncertainty into a single-robot path planning problem for each robot and a multi-robot task allocation problem.

Even when the problem is split into two stages, the number of all possible target-robot assignments to consider grows exponentially with the number of robots [16]. Moreover, finding the optimal task allocation is an instance of the well-studied set partitioning problem, which is NP-hard [17–20]. Existing literature on task allocation problems provides fully decentralized efficient methods, which can be highly suboptimal since they rely solely on local information [21]. Having a suboptimal assignment and not accounting for safety objectives would jeopardize the safety of the mission. Thus, it is desirable to formalize the safety objective in task allocation and derive guarantees on the performance of the task allocation algorithms. Auction-based approaches [22–25], on the other hand, consider the robots as bidders who iteratively submit bids for the most desirable tasks. Such iterative and distributed assignments of tasks are special instances of the efficient forward greedy heuristics, which are generally equipped with suboptimality guarantees [26].[1] In contrast with these auction-based approaches, our objective function is a safety measure defined over the completion of the tasks for each robot. This function originates from the underlying low-level stochastic optimal control problem [6]. The collective goal of multi-robot planning is to maximize the safety objective, and in our case, it has a multiplicative form, which allows a distributed implementation of the greedy heuristics. This formulation corresponds to a novel variant of the auction-based approaches, since it incorporates the multiplicative objective evaluation for allocating bids.

Our task allocation problem can be formalized as a matroid optimization with a safety objective function. Greedy algorithms are equipped with provable performance bounds when the objective functions satisfy (super-)submodularity assumptions [26–43]. However, we will show that our safety objective is nonsubmodular and nonsupermodular. The aforementioned studies on auction-based approaches either do not propose any optimality guarantee, or when they do, their problems lie in the class of set partitioning problems with fixed task-robot pair costs (that is, additive/modular objective functions) [44, 45] or submodular objective functions [24, 26]. In terms of safety-oriented task allocation, there are studies where the objective is either the conditional value-at-risk cost [46, 47], or the worst-case cost [48]. Objectives in these works are also additive. To the best of our knowledge, nonsubmodular and nonsupermodular objective functions have not been addressed in any of the existing auction-based task allocation studies or in the more general class of set partitioning problems. We show that the objective function to accomplish the mission safely under dynamic uncertainties is weakly sub and supermodular. Thus, we can leverage existing results from

[1] The forward greedy refers to the greedy heuristic that adds elements iteratively, whereas the reverse greedy refers to the one that removes.

[49] to obtain safety guarantees on our auction-based approaches. To the best of our knowledge, this is also the first work that demonstrates the benefits of an auction-based approach using the reverse greedy algorithm.

Our contributions are summarized as follows.

(i) We develop a scalable two-stage framework for an emergency rescue scenario by splitting the multi-robot controller synthesis problem into a safe planning problem (for each robot) and a multi-robot task allocation. To this end, we utilize an efficient implementation of a single-robot plan under dynamic uncertainties. This approach, based on Monte-Carlo sampling, serves as the low-level planner, and it allows the computation of the multiplicative safety objective for the high-level task allocation.

(ii) To allocate the targets in a tractable manner, we introduce two variants of greedy algorithms, the forward and the reverse. The multiplicative safety objective of our formulation allows a distributed auction-based implementation. Moreover, its weak submodularity and weak supermodularity properties enable us to invoke results from [28, 29, 49]. These results translate into performance guarantees on mission success probabilities of the forward and the reverse greedy solutions. The theoretical guarantee for the forward greedy algorithm is shown to deteriorate if the safety measure of not accomplishing any tasks is high.

(iii) We compare these two greedy algorithms not only in terms of their theoretical guarantees but also in terms of their computational and practical performance in numerical case studies. We show that both algorithms perform well in terms of computation time and optimality, compared to the optimal solution obtained via brute force. As suggested by their theoretical guarantees, the reverse greedy algorithm performs better than the forward greedy algorithm, however, this performance comes with increased computational complexity. Our case studies are based on the implementation of our two-stage framework in an example environment for an emergency rescue scenario.[2]

Organization. Problem formulation and statement are presented in Section 2. We then provide our two-stage framework by formulating both the single-robot safe planning and the multi-robot task allocation problems in Section 3. Section 4 proposes the two distributed greedy approaches for task allocation. Finally, we study a numerical case study comparing algorithms and their performance in Section 5.

2 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 hazard locations. We will model the problem in detail and formulate it as a stochastic reachability problem as follows.

[2] Our code and several case studies are publicly available at github.com/TihanyiD/multi_alloc

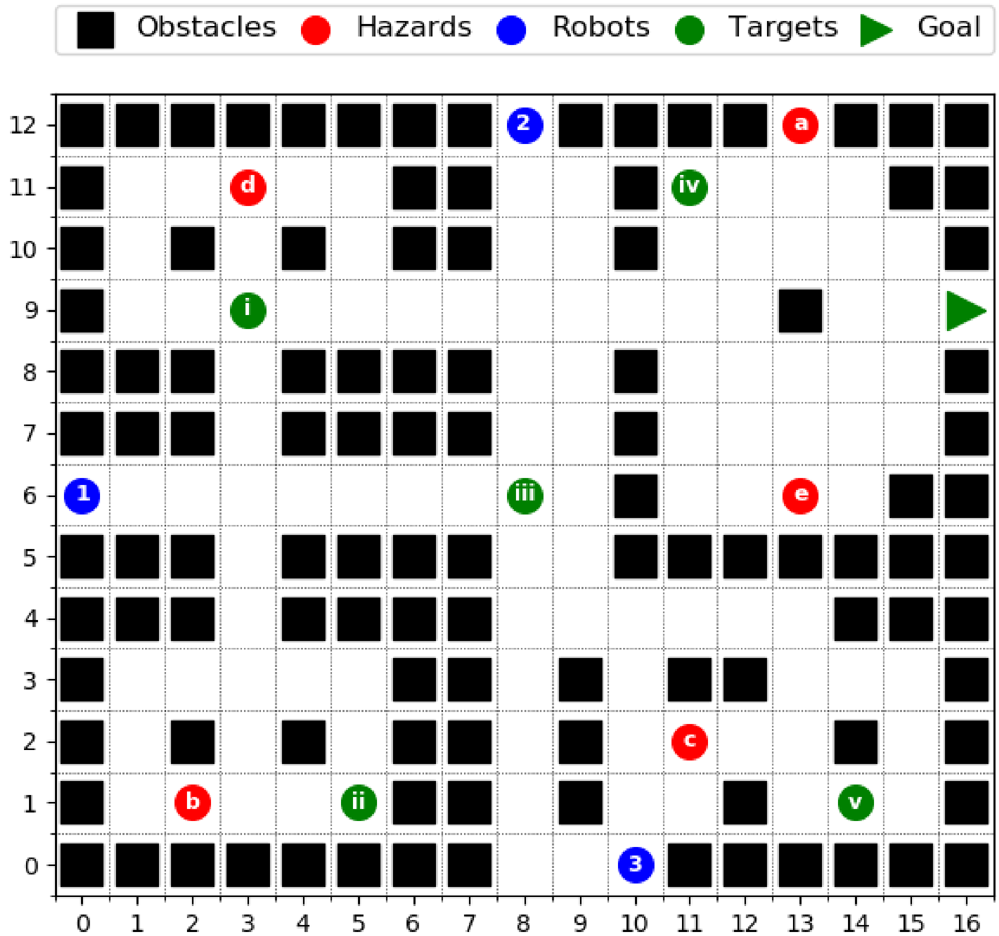

Figure 1: An example setup of our problem. The robots need to collectively visit all the targets and reach the goal while avoiding the evolving hazard.

2.1 Modeling the environment and the robots

Map model. The environment is modeled by a discretized map, as illustrated with an example in Figure 1. We let $M_{m\times n}$ be an $m \times n$-sized grid-shaped map (we normalize the grid size 1×1) and $O \subset M_{m\times n}$ be a set of obstacles. $X = M_{m\times n} \setminus O$ is the set of free cells. For the cell $x \in X$, let $N(x)$ be the set of neighboring cells of x. Such a grid-based map has been introduced by [50, 51], and is common in the robotics literature to model free space and obstacles [6, 11, 52, 53].

Robots and targets. We define the set of robots as $R = \{1, \dots, |R|\}$ and the set of targets as $T \subset X$, which represent their locations on the map.

Robot motion. The set of possible inputs correspond to the direction the robot can move: $U = \{\text{Stay}, \text{North}, \text{East}, \text{South}, \text{West}\}$. 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 $x \in X$, the set of $U(x) = \{u \in U \mid x + d_u \in X\} \subseteq U$, are the inputs available to the robot. The motion of the robot is defined by a stochastic transition kernel $x^{k+1} \sim \tau_X\left(\cdot \mid x^k, u^k\right)$, $k \in \{0, 1, \dots\}$ 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 $u^k \in U(x^k)$. The stochastic model accounts for potential uncertainty of the robot control. A deterministic robot dynamic can be considered by defining τ_X as

$$\tau_X\left(x^{k+1}\,|\,x^k, u^k\right) = \begin{cases} 1 & \text{if } x^{k+1} = x^k + d_{u^k}, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Hazard spread. Let $Y = 2^X$ be the hazard state space. Each element $y \in Y$ denotes a set of contaminated cells, which is a subset of X. The stochastic Markov process $y^{k+1} \sim \tau_Y\left(\cdot\,|\,y^k\right)$, $k \in \{0, 1, \dots\}$ 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 $k+1$. Detailed dynamics of the transition kernel depends on the modeling assumptions. For example, in the case of fire, an estimation of the probability of fire spread from a given grid to the neighboring grids can be used to derive τ_Y from [6, 54–56].

2.2 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). To this end, our objective is to determine each robot's control input at every time step to maximize the probability of completing the mission within a given 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 suitably defined Markov decision process [6, 11, 57–59]. The stochastic reachability problem can then be solved using the dynamic programming principle as shown in the above works.[3]

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 to compute an approximate solution, as detailed in the following section.

3 Two-stage multi-robot safe planning

We present a scalable framework to allocate targets to robots and control the robot fleet to maximize the probability of the mission's success. We split the problem into the following two stages: low-level path planning (optimizing control policies of each robot individually for assigned targets) and high-level task allocation.

3.1 Single-robot path planning

Here, we provide a tractable solution to the safe control problem for a single-robot system and a given set of targets. It serves as a building block of our multi-robot framework. To this end, we first define a stochastic Markov process to model the robot's motion, the hazard evolution, and the robot's progress towards completing its mission. We then utilize a dynamic programming algorithm to find the optimal control policy by maximizing the probability of success.

[3] The precise mathematical formulate requires introducing several new notations, and we postpone a single-robot formulation to the next section.

3.1.1 Definition of the stochastic process

We define the discrete-time stochastic process by its state space and transition kernels.

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 of the mission, the robot needs to keep track of the visited targets. We first define $T_r \subset T \subset X$ as the *target list*, the set of targets assigned to robot r. We then define the set $Q = 2^{T_r}$ and the *target execution state* $q^k \in Q$ at time step k, where $q^k \subseteq T_r$ for all k. The state q^k tracks the visited targets.

Contamination – As discussed in the previous section, the state of the hazard is $y^k \in Y$, the set of grid cells that are contaminated at time step k. Including the hazard state y^k in the state space results in a computationally intractable problem.[4] To address this issue, a *hazard state* denoted by s_H is introduced to capture the case where the robot enters a contaminated cell, namely, $x^k \in y^k$. A robot reaching the hazard state indicates an *unsuccessful mission* and the robot will remain in this state.

Combining the elements mentioned above, we define the state space $S = \{s_H\} \cup (Q \times X) \setminus \{(q,x) \mid x \in T_r \wedge x \notin q\}$. 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 $s_r^0 = (\emptyset, x_r^0)$, 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. Let $\tau_S^k : S \times S \times U \to [0,1]$ be the transition kernel at time step k. The quantity $\tau_S^k(s^{k+1} \mid s^k, u^k)$ represents the probability of a state transiting 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, the execution of the task and the hazard dynamics as follows.

Robot motion – Defined by $\tau_X(x^{k+1} \mid x^k, u^k)$ in Sec. 2.

Task execution state transition – The transition at time step k from $q^k \subseteq Q$ to $q^{k+1} \subseteq Q$ given that the robot is at position x^{k+1} at step $k+1$: This is described by the following time-homogeneous transition kernel

$$\tau_Q(q^{k+1} \mid q^k, x^{k+1}) = \begin{cases} 1 & \text{if } q^{k+1} = q^k \cup (x^{k+1} \cap T_r), \\ 0 & \text{otherwise,} \end{cases}$$

where $\tau_Q : Q \times Q \times X \to [0,1]$. The interpretation of this transition kernel is as follows. Whenever a target position $x^{k+1} \in T_r$ is visited, it is added to the list q^k to form q^{k+1}. For any non-target position $x^{k+1} \notin T_r$, we have $x^{k+1} \cap T_r = \emptyset$ and $q^{k+1} = q^k$. If a target position $x^{k+1} \in T_r$ is visited more than once, since $x^{k+1} \in q^k$, we have $q^{k+1} = q^k$.

Contamination probability – As discussed, since the hazard state y^k can potentially be any subset of X, the computation of its transition kernel τ_Y is intractable. Hence, the control policy of the robot cannot depend explicitly on the hazard state. However, the robot can account for the initial hazard state y^0 and its evolution dynamics τ_Y to evaluate the probability of a given

[4] Let us illustrate the cardinality of X, Q, and Y. The cardinality of X grows with the map's size, whereas the number of target execution states $|Q| = 2^{|T_r|}$ grows exponentially with the size of T_r. In practice, we have $|X| \gg |T_r|$. Thus, the hazard state space's size $|Y| = 2^{|X|}$ would be the main source of complexity.

trajectory being contaminated during the planning phase. To this end, we define the function $p_H^k : X \times X \to [0,1]$, such that the value of $p_H^k(x^{k+1}, x^k)$ equals to the probability of transitioning from an uncontaminated location $x^k \notin y^k$ at time k to a contaminated location $x^{k+1} \in y^{k+1}$ at time $k+1$. We use Monte-Carlo simulation to evaluate this function during planning, given y_0 and τ_Y.[5]

With the elements above, we can define the transition kernel τ_S^k as follows

$$\tau_S^k(s^{k+1}|s^k,u^k) = \begin{cases} 1 & \text{if } s^{k+1}=s^k \in \{s_G, s_H\}, \\ \sum\limits_{x^{k+1}\in X} p_H^k(x^{k+1},x^k)\cdot\tau_X(x^{k+1}|x^k,u^k) & \\ & \text{if } s^{k+1}=s_H \wedge s^k \notin \{s_G,s_H\}, \\ (1-p_H^k(x^{k+1},x^k))\cdot\tau_Q(q^{k+1}|q^k,x^{k+1})\cdot\tau_X(x^{k+1}|x^k,u^k) & \\ & \text{if } s^{k+1}\neq s_H \wedge s^k \notin \{s_G,s_H\}, \\ 0 & \text{otherwise.} \end{cases}$$

The transition kernel captures that the goal state s_G and the hazard state s_H are *absorbing*. Once they are reached, the system state does not change anymore due to either contamination or mission completion. In any other state $s^k \notin \{s_G, s_H\}$, the robot can move to a different state following the dynamics defined by the transition kernel above.

3.1.2 Controller synthesis via dynamic programming

Let $\pi = \{\mu^0, \ldots, \mu^{N-1}\}$ be a control policy, where $\mu^k : S \to U$ refers to the control law at time step k. With the state-space defined above, the objective of mission success defined in Section 2.2 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. We denote $f_r(T_r)$ as the probability of reaching the goal set of robot r under control policy π given the target list T_r as $f_r(\pi, T_r)$ and the probability of success under the optimal control policy $\pi_r(T_r)$. Our goal is to find

$$\pi_r(T_r) = \arg\max_{\pi} f_r(\pi, T_r). \tag{2}$$

Problem (2) can be solved using Algorithm 1 [57,60], similar to the dynamic programming approach to a stochastic control problem. The function value $V^k(s)$ (note that $V^k : S \to [0,1]$) captures the maximum probability of reaching the goal state starting from time step k at state $s \in S$, and is defined recursively in the algorithm. It can be shown that $f_r(T_r) = \max_\pi f_r(\pi, T_r) = V^0(s_r^0)$, whereas π_r can be computed as the input achieving the maximum in the algorithm above.

[5] The precise evaluation of the function p_H^k is computationally intractable due to the size of $Y = 2^X$. In [11, Alg. 1], a Monte-Carlo sampling based algorithm was proposed to provide a tractable approximation of p_H^k. The idea of the approximation is to forward propagate the hazard dynamics using the initial state y_0 and the transition kernel τ_Y.

Algorithm 1: Dynamic Programming Algorithm

Input: robot r, horizon $N \in \mathbb{N}_{>0}$, targets T_r

Output: policy $\pi_r(T_r)$, probability of success $f_r(T_r)$

begin

initialization: $V^N(s) = 1$ if $s = s_G$; 0 otherwise

for $k = N-1, \dots, 0$ **do**

$$V^{k-1}(s) = \max_{u \in U(x^{k-1})} \Big\{ \sum_{s' \in S} \tau_S^{k-1}(s' \mid s, u) \cdot V^k(s') \Big\}$$

end

end

3.2 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. We start by formulating the task allocation problem and then introduce the collective objective function.

Representation. We assume it is sufficient that a target is visited once. Thus, a valid 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 $\bigcup_{r \in R} T_r = T$. Each T_r represents the subset assigned to the robot r.

Objective function. The goal is to find the task allocation that maximizes the *probability of group success*, that is, the probability of all robots safely finishing their missions. However, maximizing the probability of group success is computationally challenging since we need to formulate the stochastic reachability of Section 3.1.2 over a product state space considering the robot fleet. Thus, we introduce the *multiplicative group success* as an approximation of the 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 3.1 for each $r \in R$. In many realistic examples, conditioning on the success of one robot improves the probability of others successfully completing their mission, because it confirms that the hazard propagation is limited in a certain area. Under this mild condition, it can be shown that the multiplicative group success is a lower bound to the probability of group success. For further discussions and proof, please see [61, §8.5].

Optimization problem. We formulate the task allocation problem for maximizing probability of safety as follows

$$\begin{aligned} F^* = \max_{\{T_r\}_{r \in R}} & \prod_{r \in R} f_r(T_r) \\ \text{s.t.} \quad & T_r \cap T_{r'} = \emptyset, \forall r \neq r', \ \bigcup_{r \in R} T_r = T. \end{aligned} \tag{3}$$

[6]According to the taxonomy in [20], this problem can be classified as an NP-hard multi-task robot, single-robot task, instantaneous assignment task allocation problem (MT-SR-IA). Multi-task robot – because one robot can visit multiple targets, single-robot task – since it is enough for targets to be visited by only a single robot, and instantaneous assignment – since tasks are allocated only once before the robots actually running.

The problem above is an instance of set partitioning and is NP-hard [18–20].[6] In the next section, we propose two variants of the greedy algorithms to enable efficient distributed implementations of task allocation for maximizing safe mission completion. In contrast with task allocation and set partitioning applications where the objective function is additive or sub/supermodular, our objective function is neither submodular nor supermodular. However, we show that our problem benefits from weak submodularity (via submodularity ratio) and weak supermodularity (via curvature). We discuss relevant performance guarantees and utilize them to obtain suboptimality bounds on the considered task allocation with a stochastic safety objective.

4 Greedy heuristics

We introduce two greedy heuristics, the forward and the reverse, in Sections 4.1 and 4.2, respectively. 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 separated among the robots and implemented in an auction-based fashion. At each iteration, each robot can submit a bid based on their individual objective f_r. The bids are then collected by a central unit such that a decision based on the group objective F can be made. The properties of the group objective F and theoretical performance guarantees of the algorithms are discussed in Section 4.3.

4.1 Forward greedy algorithm

The forward greedy algorithm (see Algorithm 2) is initialized with no tasks allocated to any of the robots. It then iteratively updates this allocation by choosing the task-robot pair with the best optimality gain until every task is allocated to one robot. The computation of the iteration steps is distributed among the robots.

Algorithm 2 proceeds as follows. First, define the following variables for each step k: $\{T_r^k\}_{r\in R}$ denotes the current task allocation, and $\{f_r^k\}_{r\in R}$ stores the evaluated function values for each robot r. Furthermore, J^k is the set of tasks yet to be allocated and R^k is the set of robots which needs to update their bids in the next step. Initially, no tasks are assigned, hence $T_r^0 = \emptyset$ for all $r \in R$ (see Line 2). In each step exactly one task is allocated, hence we need $|T|$ steps to complete the algorithm (Line 3). At each iteration k, all robots $r \in R$ submit a bid (see Line 4–8), which consists of the pair (t_r^k, δ_r^k). Each robot r chooses the task t_r^k from the list of unallocated tasks J^{k-1}, such that it obtains the best optimality gain δ_r^k 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 success 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_r^k, T_r^k for all $r \in R$ and R^k, J^k according to our choice from the task allocation.[7]

We will later see that the solution obtained by this algorithm will give rise to a performance

[7] Note that only the robots choosing the same task as 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). The variable R^k is initialized with $R^0 = R$, since in the first iteration all robots have to calculate their bids.

Algorithm 2: Forward Distributed Greedy Algorithm

Input: R, T, $\{f_r\}_{r\in R}$

Output: $\{T_r^{\text{fg}} = T_r^{|T|}\}_{r\in R}$

begin
  initialization: $T_r^0 = \emptyset$, $f_r^0 = f_r(\emptyset)$, $\forall r$, $J^0 = T$, $R^0 = R$
  for $k = 1, \ldots, |T|$ **do**
    for $r \in R^{k-1}$ **do**
      $t_r^k \leftarrow \underset{t\in J^{k-1}}{\arg\max}\, f_r\left(T_r^{k-1} \cup t\right) - f_r\left(T_r^{k-1}\right)$
      $\delta_r^k \leftarrow f_r\left(T_r^{k-1} \cup t_r^k\right) - f_r\left(T_r^{k-1}\right)$
    end
    $(t_r^k, \delta_r^k) \leftarrow (t_r^{k-1}, \delta_r^{k-1})\ \ \forall r \notin R^{k-1}$
    $r^k \leftarrow \arg\max_{r\in R} \delta_r^k \cdot \prod_{r'\in R\setminus\{r\}} f_{r'}^{k-1}$
    $f_r^k \leftarrow f_r^k - \delta_r^k$, if $r = r^k$; f_r^{k-1} otherwise
    $T_r^k \leftarrow T_r^k \cup t_r^k$, if $r = r^k$; T_r^{k-1}, otherwise
    $R^k \leftarrow \left\{r \,|\, t_r^k = t_{r^k}^k\right\}$
    $J^k \leftarrow J^{k-1} \setminus t_{r^k}^k$
  end
end

guarantee as a function of the value $F(\{\emptyset\}_{r\in R})$, the initial empty allocation. This could potentially take a large value, e.g., 1, which could deteriorate the performance guarantee. To address this issue, we introduce the reverse greedy algorithm next.

4.2 Reverse greedy algorithm

The reverse greedy algorithm (see Algorithm 3) is initialized with all tasks being allocated to every robot simultaneously. Due to this reason, its performance guarantee will instead be a function of the value $F\left(\{T\}_{r\in R}\right)$. 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.

Algorithm 3 proceeds as follows. First, define the following variables for each step k: $\{T_r^k\}_{r\in R}$ denotes the current task allocation, while $\{f_r^k\}_{r\in R}$ stores the evaluated function values for each robot r. Furthermore, J^k is the set of tasks not yet removed and R^k is the set of robots which need to update their bids in the next step. We initially assign all tasks to every robot, hence $T_r^0 = T$ for all $r \in R$ (see Line 2). In each step, exactly one task is removed from one of the robots. Hence, the algorithm needs $|T| \cdot (|R| - 1)$ steps to complete (Line 3). At each k, the robot $r \in R$ submits a bid (see Line 4–8), which consists of the pair (t_r^k, δ_r^k). Each robot r submitting a bid chooses the task t_r^k from the not yet removed tasks $J^{k-1} \cap T_r^{k-1}$ that corresponds to the largest optimality loss δ_r^k with respect to the individual objective function f_r. After collecting all the bids, we choose the

Algorithm 3: Reverse Distributed Greedy Algorithm

Input: R, T, $\{f_r\}_{r\in R}$

Output: $\{T_r^{\text{rg}} = T_r^{|T|\cdot(|R|-1)}\}_{r\in R}$

1 **begin**

2 initialization: $T_r^0 = T,\ f_r^0 = f_r(T),\ \forall r,\ J^0 = T,\ R^0 = R$

3 **for** $k = 1, \ldots, |T|\cdot(|R|-1)$ **do**

4 **for** $r \in R^{k-1}$ **do**

5 $t_r^k \leftarrow \underset{t\in J^{k-1}\cap T_r^{k-1}}{\arg\max}\ f_r(T_r^{k-1}\setminus t) - f_r(T_r^{k-1})$

6 $\delta_r^k \leftarrow f_r(T_r^{k-1}\setminus t_r^k) - f_r(T_r^{k-1})$

7 **end**

8 $(t_r^k, \delta_r^k) \leftarrow (t_r^{k-1}, \delta_r^{k-1})\ \ \forall r \notin R^{k-1}$

9 $r^k \leftarrow \arg\max_{r\in R} \delta_r^k \cdot \prod_{r'\in R\setminus\{r\}} f_{r'}^{k-1}$

10 $f_r^k \leftarrow f_r^k + \delta_r^k$, if $r = r^k$; f_r^{k-1}, otherwise

11 $T_r^k \leftarrow T_r^k \setminus t_r^k$, if $r = r^k$; T_r^{k-1}, otherwise

12 $R^k \leftarrow \begin{cases} \{r \mid t_r^k = t_{r^k}^k\}, \text{ if } \left|\{r \mid t_{r^k}^k \in T_r^k\}\right| = 1 \\ \{r^k\}, \text{ otherwise} \end{cases}$

13 $J^k \leftarrow \begin{cases} J^{k-1}\setminus t_{r^k}^k, \text{ if } \left|\{r \mid t_{r^k}^k \in T_r^k\}\right| = 1 \\ J^{k-1}, \text{ otherwise} \end{cases}$

14 **end**

15 **end**

robot r^k which generates the largest optimality loss with respect to the collective objective: the multiplicative group success F (Line 9). Due to our auction-based formulation in this line, we can choose the task-robot pair with the largest collective loss efficiently. Between Lines 10–13, we set the values of f_r^k, T_r^k for all $r \in R$ and R^k, J^k according to our choice of task allocation.[8]

4.3 Performance guarantees

To discuss the theoretical performance guarantees for Algorithms 2 and 3, we bring in the definitions for the curvature α and the submodularity ratio γ. Our multiplicative safety objective F gives rise to non-trivial values for these notions, which then enable the performance guarantees. Let W be the ground set of all task-robot pairs for the following definitions.

[8]Note that a task $t_{r^k}^k$ is removed from J^k if it is only allocated to single robot, hence $|\{r|t_{r^k}^k \in T_r^k\}| = 1$. Also note that only the robots choosing the same task as r^k have to update their bids in the next iteration and only when $t_{r^k}^k$ just got removed from J^k (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). The variable R^k is initialized with $R^0 = R$, since in the first iteration all robots have to calculate their bids.

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 $e \in W \setminus B$. Observe that F is supermodular if and only if $\alpha = 0$, and we also have $\alpha \in [0,1]$. Refer to [28] for derivations.

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 $e \in W \setminus B$. Observe that F is submodular if and only if $\gamma = 1$, and we also have $\gamma \in [0,1]$. Refer to [28] for derivations.

Calculating these values is computationally expensive. Moreover, unlike many past studies on these notions [28, 62, 63], the multiplicative group success we consider is not amenable to ex-ante bounds on these ratios. In practice, we can verify the objective function F to be strictly decreasing after each additional task assignment. It is known that strict monotonicity results in non-trivial values for submodularity ratio ($\gamma > 0$) and curvature ($\alpha < 1$) [49, Prop. 1]. Moreover, similar to [28], we compute ex-post bounds from the function evaluations obtained from the greedy algorithms.

Let F^* denote the optimal value of Problem (3). We state the guarantees as follows.

Theorem 1 *Let F^{fg} denote the objective of the forward greedy solution from Algorithm 2. We then have*

$$\frac{F^{fg} - F\left(\{\emptyset\}_{r \in R}\right)}{F^* - F\left(\{\emptyset\}_{r \in R}\right)} \leq \frac{1}{\gamma \cdot (1-\alpha)}.$$

Theorem 2 *Let F^{rg} denote the objective of the reverse greedy solution from Algorithm 3. We then have*

$$\frac{\gamma}{1+\gamma \cdot \alpha} \leq \frac{F^{rg} - F\left(\{T\}_{r \in R}\right)}{F^* - F\left(\{T\}_{r \in R}\right)}.$$

Proofs of Theorems 1 and 2: Using the fact that we are working over the base of a partition matroid, the proofs of Theorems 1 and 2 follow the steps of proofs of [49, Thm. 1] and [49, Thm. 2] closely, respectively. ■

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. Hence $F\left(\{\emptyset\}_{r \in R}\right) \approx 1$ takes a high value. For the reverse greedy, this is the full allocation resulting in a low probability of success $F\left(\{T\}_{r \in R}\right) \approx 0$, since the robots are overwhelmed with the tasks with the increased danger of being contaminated. Assuming

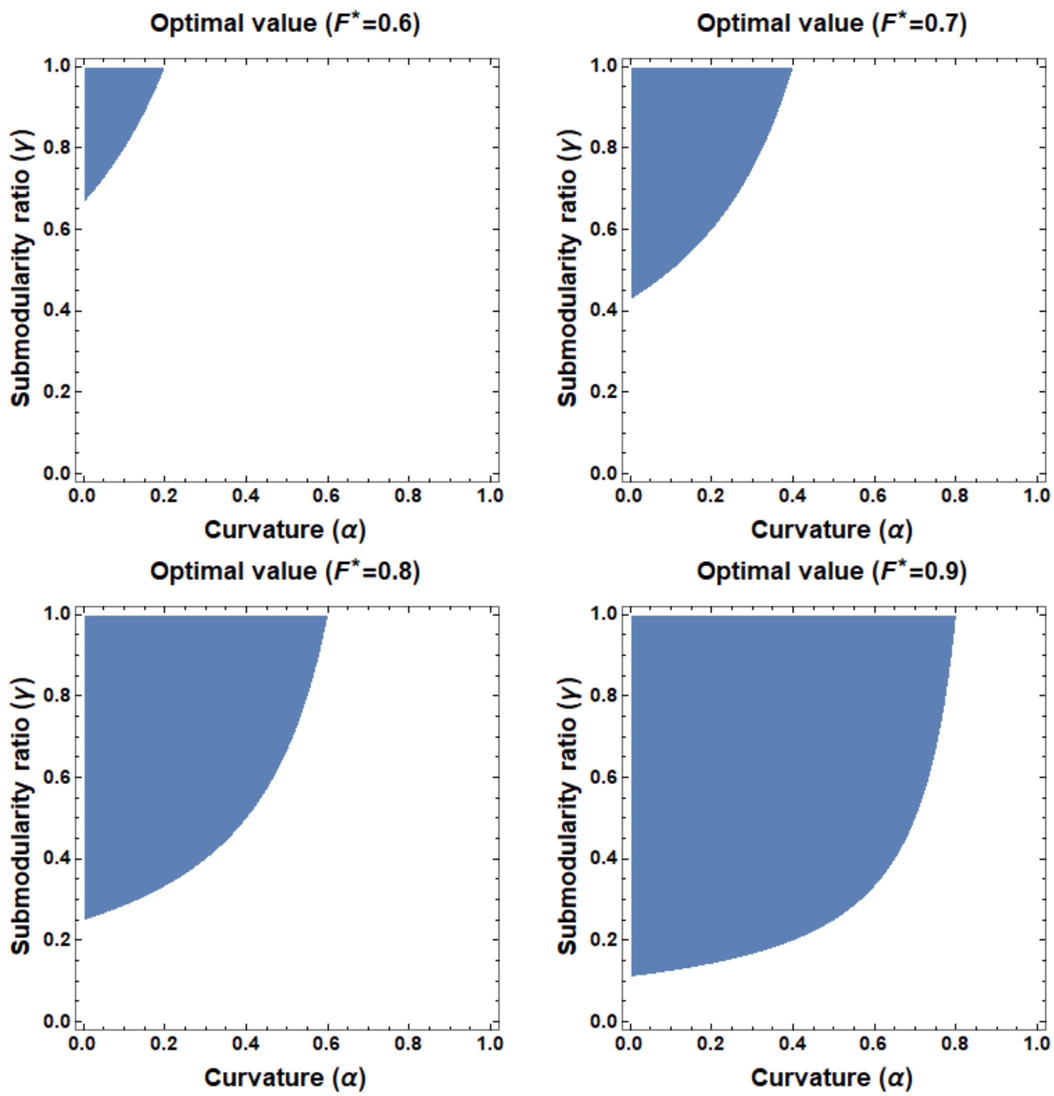

Figure 2: Comparison of guarantees. For each optimal F^* value, the shaded regions represent the (α, γ) pairs for which the forward greedy outperforms the reverse greedy.

these values, the guarantees are given by

$$\begin{aligned} &\frac{F^*}{\gamma \cdot (1-\alpha)} + \frac{\gamma \cdot (1-\alpha) - 1}{\gamma \cdot (1-\alpha)} = g^{\text{fg}}(\alpha, \gamma, F^*) \leq F^{\text{fg}}, \\ &F^* \cdot \frac{\gamma}{1 + \gamma \cdot \alpha} = g^{\text{rg}}(\alpha, \gamma, F^*) \leq F^{\text{rg}}. \end{aligned} \tag{4}$$

In (4), $g^{\text{fg}}(\alpha, \gamma, F^*)$ and $g^{\text{rg}}(\alpha, \gamma, F^*)$ provide lower bounds on the probabilities of success: F^{fg} and F^{rg}.

Figure 2 illustrates the area defined by $\{(\alpha, \gamma) \in [0,1] \times [0,1] \,|\, g^{\text{fg}}(\alpha, \gamma, F^*) \geq g^{\text{rg}}(\alpha, \gamma, F^*)\}$ for fixed values of F^*. Observe that if F^* is close to 1, using the forward greedy is a better choice for a wide range of values of α and γ. However, if $F^* \leq 0.5$, the reverse greedy provides a better guarantee for all (α, γ) pairs. As the value of F^* decreases from 1 to 0.5, the area where the forward

Table 1: Comparison of task allocation algorithms. *Task allocation* – Allocation computed by the given algorithm, *Computation time* – Total algorithm run time[9], *Success probability* – Corresponds to the given allocation.

Algorithm	Task allocation		Computation time	Success probability
	Robot	Tasks		
Forward Greedy	1	{i, ii, iii}	7 minutes 5 seconds	0.699
	2	{iv}		
	3	{v}		
	Robot	Tasks		
Reverse Greedy	1	{ii, iii}	29 minutes	0.717
	2	{i, iv}		
	3	{v}		
	Robot	Tasks		
Brute Force	1	{ii, iii}	4 hours 53 minutes 19 seconds	0.717
	2	{i, iv}		
	3	{v}		

greedy algorithm is more reliable shrinks. For this range of values, the performance guarantee of the forward greedy algorithm is better only when the function is close to being both supermodular and submodular. In fact, [29, Prop. 4 and 5] prove that there is no performance guarantee for the forward greedy algorithm unless both the submodularity ratio and the curvature are utilized. Hence, if one of the two properties does not hold, we expect the reverse greedy algorithm to perform better.

5 Numerical results

We present a case study for the two-stage multi-robot safe planning framework in Section 3. For additional case studies please refer to our report [61]. The code that generates the case studies 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 4. The example is tailored such that we can compute the optimal allocation via brute force for performance comparison. This brute force solution is obtained by enumerating all valid task allocations. For larger examples, the optimal allocation cannot be computed.

The environment is a 17-by-13 grid map with the initial state in Figure 3. The time horizon is $N = 75$ steps, which is long enough for each robot to visit all the targets in the grid. The robot dynamics are defined by (1). Five hazard sources are illustrated by their initial positions in Figure 3. The hazard dynamics are described in [61, §8.8] and are based on fire propagation models [54, 55]. To visualize the evolution of the hazard, the heat map in Figure 3 shows the probability of a grid cell being contaminated within 75 steps.

Performance results. Table 1 compares the solutions from three different task allocation

[9]Measured on a computer equipped with Core i7 (2.6GHz), 8GB RAM.

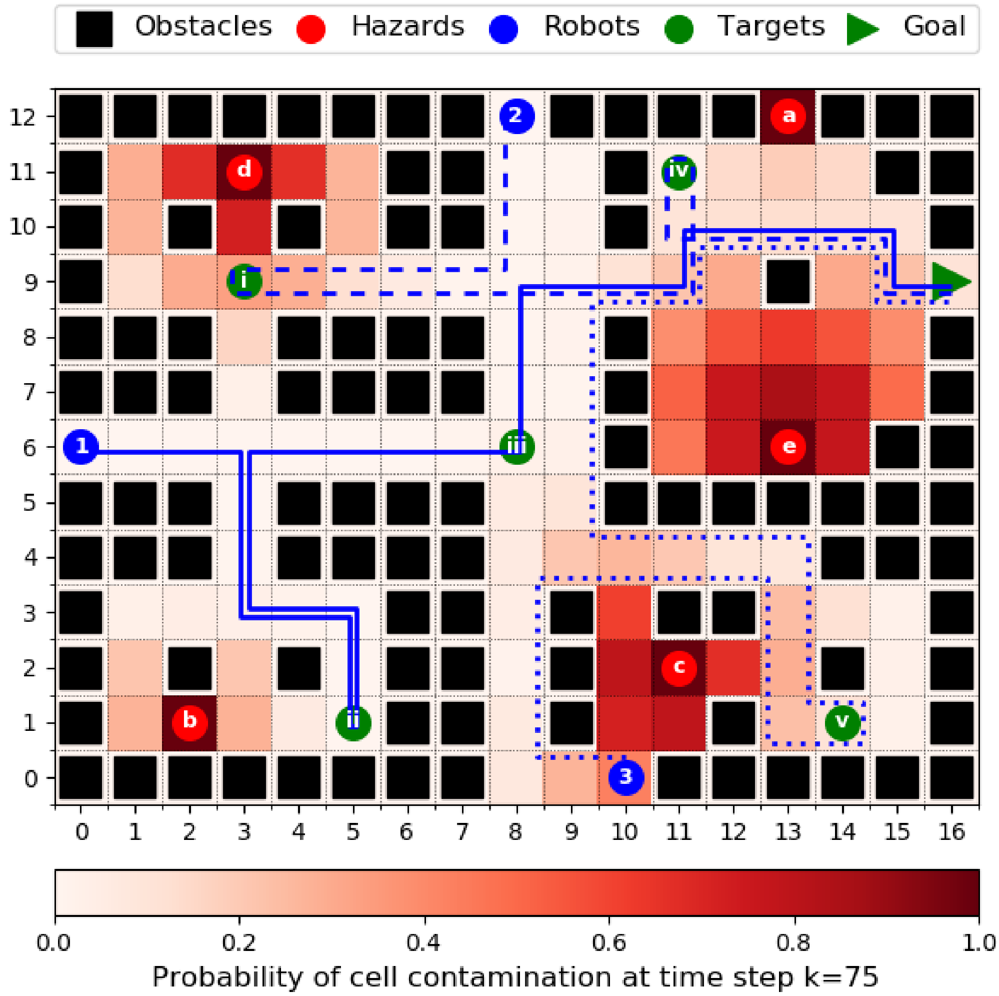

Figure 3: Example environment and generated robot paths for the brute force optimal task allocation.

methods. The optimal task allocation and the corresponding robot paths can be found in Figure 3. The forward and the reverse greedy algorithms are shown to provide tractable approximations of the optimal solution. The computation times illustrate the clear benefits of greedy heuristics. Both greedy algorithms are faster than the brute force approach by order of magnitude without significant optimality loss (in reverse greedy, there is no optimality 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 2, the forward greedy takes $|T|$ steps, whereas the reverse greedy (Algorithm 3) takes $|T| \cdot (|R|-1)$ steps. Second, the computation time for evaluating the solution of the single-robot safe planning problem (see Section 3.1) for a subset of tasks $T_r \subset T$ 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 greedy requires solving larger instances of the single-robot safe planning problem. Notice that planning for the shortest path might jeopardize safety. For example, for 'Robot 3', the shortest path between its initial cell and 'target v' would pass through 'hazard c'. Thus, a safe planning framework, such as the one proposed here, is crucial for accounting for dynamic uncertainties.

Properties of the safety objective and the guarantees. There is no computationally tractable approach to obtaining the exact curvature α and the exact submodularity ratio γ properties of F (see Definitions 1 and 2). However, we can confirm that these ratios are non-trivial, $\alpha < 1$ and $\gamma > 0$, because we verified that F is strictly decreasing in practice. Moreover, we obtain computationally efficient ex-post bounds called *greedy-approximate curvature* $\alpha^G \leq \alpha$ and *greedy-approximate submodularity ratio* $\gamma^G \geq \gamma$ using the function evaluations during the execution of the greedy algorithms [28]. For this particular example, we obtained the values $\alpha^G = 0.989$ and $\gamma^G = 0.525$. Hence, we can conclude that the objective function F is neither supermodular nor submodular. Although the guarantees of Theorems 1 and 2 do not necessarily hold for α^G and γ^G, evaluating (4) at α^G and γ^G suggests that the reverse greedy may essentially have a better performance guarantee than the forward greedy. This can be attributed to the fact that the function F is far away from being supermodular, $\alpha^G = 0.989$, and this value heavily deteriorates the bound in Theorem 1. This observation is confirmed by the actual performances of the algorithms in Table 1.

6 Conclusion

We proposed a two-stage framework to solve a multi-robot safe planning problem in a computationally tractable manner. An efficient implementation of a stochastic reachability for a Markov decision process addressing safe planning under dynamic uncertainties served as the low-level planner. The multiplicative safety objective allowed implementations of the forward and reverse greedy heuristics in a distributed manner to allocate the tasks among the robots efficiently. 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. As is suggested by the performance guarantee analyses and the properties of the multiplicative safety objective, the reverse greedy computed a slightly better solution than the forward greedy, but this benefit came with an increased computational burden. Future works include accounting for robot failure and incorporating hazard observation feedback.

References

[1] D. Di Paola, A. Milella, G. Cicirelli, and A. Distante, "An autonomous mobile robotic system for surveillance of indoor environments," *Int. J. of Adv. Rob. Syst.*, vol. 7, no. 1, p. 8, 2010.

[2] S. Jorgensen, R. H. Chen, M. B. Milam, and M. Pavone, "The matroid team surviving orienteers problem: Constrained routing of heterogeneous teams with risky traversal," in *IROS*. IEEE, 2017, pp. 5622–5629.

[3] T. Li, H.-S. Shin, and A. Tsourdos, "Threshold greedy based task allocation for multiple robot operations," *arXiv preprint arXiv:1909.01239*, 2019.

[4] X. Sun, C. G. Cassandras, and X. Meng, "Exploiting submodularity to quantify near-optimality in multi-agent coverage problems," *Aut.*, vol. 100, pp. 349–359, 2019.

[5] L. Zhou, V. Tzoumas, G. J. Pappas, and P. Tokekar, "Resilient active target tracking with multiple robots," *IEEE Rob. and Aut. Letters*, vol. 4, no. 1, pp. 129–136, 2018.

[6] T. A. Wood and M. Kamgarpour, "Automaton-based stochastic control for navigation of emergency rescuers in buildings," in *IEEE Conference on Control Applications.* IEEE, 2016, pp. 587–592.

[7] 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.

[8] P. Stone and M. Veloso, "Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork," *AI*, vol. 110, no. 2, pp. 241–273, 1999.

[9] H. Fazlollahtabar and M. Saidi-Mehrabad, "Methodologies to optimize automated guided vehicle scheduling and routing problems: a review study," *J. of Intel. & Rob. Syst.*, vol. 77, no. 3, pp. 525–545, 2015.

[10] 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.

[11] Y. Lu and M. Kamgarpour, "Safe mission planning under dynamical uncertainties," in *ICRA.* IEEE, 2020, pp. 2209–2215.

[12] T. A. Wood, S. Summers, and J. Lygeros, "A stochastic reachability approach to emergency building evacuation," in *52nd CDC.* IEEE, 2013, pp. 5722–5727.

[13] B. Zhou, W. Schwarting, D. Rus, and J. Alonso-Mora, "Joint multi-policy behavior estimation and receding-horizon trajectory planning for automated urban driving," in *ICRA.* IEEE, 2018, pp. 2388–2394.

[14] S. L. Smith, J. Tumova, C. Belta, 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.

[15] M. Ono, M. Pavone, Y. Kuwata, and J. Balaram, "Chance-constrained dynamic programming with application to risk-aware robotic space exploration," *Aut. Rob.*, vol. 39, no. 4, pp. 555–571, 2015.

[16] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The Int. J. of Rob. Res.*, vol. 32, no. 12, pp. 1495–1512, 2013.

[17] M. Andersson and T. Sandholm, "Contract type sequencing for reallocative negotiation," in *Proc. 20th IEEE Int. Conf. on Dist. Comp. Syst.* IEEE, 2000, pp. 154–160.

[18] H.-S. Shin and P. Segui-Gasco, "Uav swarms: Decision-making paradigms," *Encyclopedia of aerospace engineering*, pp. 1–13, 2010.

[19] L. A. Wolsey and G. L. Nemhauser, *Integer and combinatorial optimization.* John Wiley & Sons, 1988, vol. 55.

[20] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The Int. J. of Rob. Res.*, vol. 23, no. 9, pp. 939–954, 2004.

[21] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proc. of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

[22] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Auctioning over probabilistic options for temporal logic-based multi-robot cooperation under uncertainty," in *ICRA*. IEEE, 2018, pp. 7330–7337.

[23] F. Faruq, D. Parker, B. Laccrda, and N. Hawes, "Simultaneous task allocation and planning under uncertainty," in *IROS*. IEEE, 2018, pp. 3559–3564.

[24] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. J. Kleywegt, S. Koenig, C. A. Tovey, A. Meyerson, and S. Jain, "Auction-based multi-robot routing." in *Rob.: Science and Syst.*, vol. 5. Rome, Italy, 2005, pp. 343–350.

[25] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt, "Robot exploration with combinatorial auctions," in *IROS*, vol. 2. IEEE, 2003, pp. 1957–1962.

[26] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i," *Math. Prog.*, vol. 14, no. 1, pp. 265–294, 1978.

[27] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions—ii," in *Polyhedral Combinatorics.* Springer, 1978, pp. 73–87.

[28] 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.

[29] B. Guo, O. Karaca, T. H. Summers, and M. Kamgarpour, "Actuator placement under structural controllability using forward and reverse greedy algorithms," *IEEE Trans. on Aut. Contr.*, 2020.

[30] L. Chen, M. Feldman, and A. Karbasi, "Weakly submodular maximization beyond cardinality constraints: Does randomization help greedy?" in *35th Int. Conf. on Mach. Lrn.*, 2018, pp. 804–813.

[31] E. R. Elenberg, R. Khanna, A. G. Dimakis, S. Negahban, *et al.*, "Restricted strong convexity implies weak submodularity," *Annals of Statistics*, vol. 46, no. 6B, pp. 3539–3568, 2018.

[32] A. Clark, B. Alomair, L. Bushnell, and R. Poovendran, "Submodularity in input node selection for networked linear systems: Efficient algorithms for performance and controllability," *IEEE Contr. Syst. Magazine*, vol. 37, no. 6, pp. 52–74, 2017.

[33] 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.

[34] M. Sviridenko, J. Vondrák, and J. Ward, "Optimal approximation for submodular and supermodular optimization with bounded curvature," *Math. of OR*, vol. 42, no. 4, pp. 1197–1218, 2017.

[35] T. Zhang, "Adaptive forward-backward greedy algorithm for learning sparse representations," *IEEE Trans. on Inf. T.*, vol. 57, no. 7, pp. 4689–4708, 2011.

[36] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. on Inf. Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[37] M. Chrobak, C. Kenyon, and N. Young, "The reverse greedy algorithm for the metric k-median problem," *Inf. Processing Letters*, vol. 97, no. 2, pp. 68–72, 2006.

[38] V. Il'ev and S. Il'eva, "On minimizing supermodular functions on hereditary systems," in *Optimization Problems and Their Applications*, 2018, pp. 3–15.

[39] M. Conforti and G. Cornuéjols, "Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the rado-edmonds theorem," *Discrete App. Math.*, vol. 7, no. 3, pp. 251–274, 1984.

[40] V. Il'ev, "Hereditary systems and greedy-type algorithms," *Discrete Applied Mathematics*, vol. 132, no. 1-3, pp. 137–148, 2003.

[41] V. Il'ev and N. Linker, "Performance guarantees of a greedy algorithm for minimizing a supermodular set function on comatroid," *EJOR*, vol. 171, no. 2, pp. 648–660, 2006.

[42] V. Tzoumas, A. Jadbabaie, and G. J. Pappas, "Resilient non-submodular maximization over matroid constraints," *arXiv preprint arXiv:1804.01013*, 2018.

[43] O. Karaca, B. Guo, and M. Kamgarpour, "A comment on performance guarantees of a greedy algorithm for minimizing a supermodular set function on comatroid," *EJOR*, 2020.

[44] A. Atamtürk, G. L. Nemhauser, and M. W. Savelsbergh, "A combined lagrangian, linear programming, and implication heuristic for large-scale set partitioning problems," *J. of Heuristics*, vol. 1, no. 2, pp. 247–259, 1996.

[45] 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.

[46] 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.

[47] F. Yang and N. Chakraborty, "Algorithm for optimal chance constrained linear assignment," in *ICRA*. IEEE, 2017, pp. 801–808.

[48] 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.

[49] O. Karaca, D. Tihanyi, and M. Kamgarpour, "Performance guarantees of forward and reverse greedy algorithms for minimizing nonsupermodular nonsubmodular functions on a matroid," *arXiv submission*, 2021. [Online]. Available: https://www.research-collection.ethz.ch/handle/20.500.11850/471342

[50] A. Elfes, "Sonar-based real-world mapping and navigation," *IEEE J. on Rob. and Automation*, vol. 3, no. 3, pp. 249–265, 1987.

[51] H. P. Moravec, "Sensor fusion in certainty grids for mobile robots," in *Sensor devices and systems for robotics*. Springer, 1989, pp. 253–276.

[52] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *IROS*. IEEE, 2017, pp. 1366–1373.

[53] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Aut. Rob.*, vol. 34, no. 3, pp. 189–206, 2013.

[54] T. Beer and I. Enting, "Fire spread and percolation modelling," *Math. and Comp. Model.*, vol. 13, no. 11, pp. 77–96, 1990.

[55] C. Tymstra, R. Bryce, B. Wotton, S. Taylor, O. Armitage, *et al.*, "Development and structure of prometheus: the canadian wildland fire growth simulation model," *Nat. Res. Canada Rep. NOR-X-417*, 2010.

[56] H. Cheng and G. V. Hadjisophocleous, "Dynamic modeling of fire spread in building," *Fire Safety J.*, vol. 46, no. 4, pp. 211–224, 2011.

[57] 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.

[58] S. Summers and J. Lygeros, "Verification of discrete time stochastic hybrid systems: A stochastic reach-avoid decision problem," *Automatica*, vol. 46, no. 12, pp. 1951–1961, 2010.

[59] 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.

[60] S. Summers, M. Kamgarpour, J. Lygeros, and C. Tomlin, "A stochastic reach-avoid problem with random obstacles," in *Proc. of the 14th HSCC*. ACM, 2011, pp. 251–260.

[61] D. Tihanyi, "Multi-robot extension for safe planning under dynamic uncertainty," *Master*

thesis, ETH Zürich, 2021. [Online]. Available: https://www.research-collection.ethz.ch/handle/20.500.11850/471229

[62] T. Summers and M. Kamgarpour, "Performance guarantees for greedy maximization of non-submodular controllability metrics," in *18th ECC*, 2019, pp. 2796–2801.

[63] O. Karaca and M. Kamgarpour, "Exploiting weak supermodularity for coalition-proof mechanisms," in *CDC*. IEEE, 2018, pp. 1118–1123.