Resilient Active Target Tracking with Multiple Robots

Lifeng Zhou, Vasileios Tzoumas, George J. Pappas, Pratap Tokekar

Abstract—The problem of target tracking with multiple robots consists of actively planning the motion of the robots to track the targets. A major challenge for practical deployments is to make the robots resilient to failures. In particular, robots may be attacked in adversarial scenarios, or their sensors may fail or get occluded. In this paper, we introduce planning algorithms for multi-target tracking that are resilient to such failures. In general, resilient target tracking is computationally hard. Contrary to the case where there are no failures, no scalable approximation algorithms are known for resilient target tracking when the targets are indistinguishable, or unknown in number, or with unknown motion model. In this paper we provide the first such algorithm, that also has the following properties: First, it achieves maximal resiliency, since the algorithm is valid for any number of failures. Second, it is scalable, as our algorithm terminates with the same running time as state-of-the-art algorithms for (non-resilient) target tracking. Third, it provides provable approximation bounds on the tracking performance, since our algorithm guarantees a solution that is guaranteed to be close to the optimal. We quantify our algorithm’s approximation performance using a novel notion of curvature for monotone set functions subject to matroid constraints. Finally, we demonstrate the efficacy of our algorithm through MATLAB and Gazebo simulations, and a sensitivity analysis; we focus on scenarios that involve a known number of distinguishable targets.

I. INTRODUCTION

Tasks such as surveillance, exploration, and security often require the capability to detect, localize, and track targets within a prescribed area. For example, consider the tasks:

• (Surveillance) Detect and localize invasive fish in an ecosystem; [1]
• (Area monitoring) Detect and localize trapped people in a burning building; [2]
• (Patrolling) Detect and localize adversarial agents that move in an urban environment. [3]

These tasks can greatly benefit by the use of robots that act as mobile sensors. Indeed, advancements in robotic mobility, sensing, and communication envision the deployment of collaborative robots to support target tracking [4]. The problem of planning the (joint) motion of robots for target tracking is known as multi-robot active target tracking in the literature [5]. This is a challenging problem due to the fact that the targets may be mobile and whose motion model may only be partially known. The targets may even be moving adversarially. There may be a large number of targets (more than the number of robots), even unknown in number, and may be indistinguishable from each other. Nevertheless, a number of algorithms have been designed that ensure near-optimal tracking for all of the aforementioned scenarios [5]–[12].

In this paper, we focus on scenarios where the robots operate in failure-prone or adversarial environments. In such scenarios, the robots may be subject to attacks leading to robot failures [13], the robots’ fields-of-view may become obstructed due to environmental hazards [14], or their sensors can fail completely [15] (see also Fig. 1). We seek planning and coordination algorithms that are resilient to such failures and/or adversarial attacks.

In this paper, we introduce a problem of resilient target tracking that guards against worst-case robot failures even when the targets are indistinguishable, unknown in number, or even with unknown motion model. Resilient target tracking is a computationally challenging problem since it needs to account for all possible robot and/or sensor failures, a problem of combinatorial complexity—even in the presence of no failures, the problem is NP-hard [11]. This computational challenge motivates the main goal in this paper: to provide a scalable and provably close-to-optimal approximation algorithm. To this end, we capitalize on recent algorithmic results on resilient optimization subject to matroids [16], and present an approximation algorithm for resilient target tracking.
Contributions. In this paper, we make the contributions:

- **(Problem)** We formalize the problem of resilient active target tracking against worst-case failures even in the presence of targets that are (possibly) indistinguishable, unknown in number, or even of partially unknown motion model. This is the first work to formalize this problem.
- **(Solution)** We develop the first algorithm for the problem, and prove it has the following properties:
  - **maximal resiliency:** the algorithm is valid for any number of robot and/or sensor failures;
  - **minimal running time:** the algorithm terminates with the same running time as state-of-the-art algorithms for non-resilient target tracking;
  - **provable approximation performance:** the algorithm ensures a close-to-optimal solution for any target tracking objective function that is monotone and submodular (submodularity is a diminishing returns property [17]). Examples of such functions are the expected number of detected targets at a prescribed time and the mutual information between the predicted targets’ location and the robots’ sensor measurements [12].

- **(Empirical Evaluation)** We demonstrate with MATLAB and Gazebo simulations both the necessity for resilient target tracking against robot failures, and the efficacy and robustness of our approach. To this end, we focus on scenarios that involve distinguishable targets (known in number), and also conduct sensitivity analysis against non-worst-case attacks (random and greedy attacks).

Overall, in this paper we go beyond non-resilient target tracking [5]–[12] by proposing resilient target tracking; and beyond resilient tracking with distinguishable and known targets [18] by proposing resilient tracking with targets that are (possibly) indistinguishable, and/or unknown.

Organization of rest of the paper. Section II formulates resilient target tracking (Problem 1). Section III presents the first scalable algorithm for Problem 1. Section IV presents the main result in this paper: the scalability and performance guarantees of the proposed algorithm. Section V presents MATLAB and Gazebo simulations. Section VI concludes the paper.

Notation. Calligraphic fonts denote sets (e.g., $\mathcal{A}$). Given a set $\mathcal{A}$, $2^\mathcal{A}$ denotes the power set of $\mathcal{A}$; $|\mathcal{A}|$ denotes $\mathcal{A}$’s cardinality; given another set $\mathcal{B}$, the set $\mathcal{A} \setminus \mathcal{B}$ denotes the set of elements in $\mathcal{A}$ that are not in $\mathcal{B}$. Given a set $\mathcal{V}$, a set function $f : 2^\mathcal{V} \mapsto \mathbb{R}$, and an element $x \in \mathcal{V}$, $f(x)$ is a shorthand that denotes $f(\{x\})$.

II. PROBLEM FORMULATION

We formalize the problem of resilient multi-target tracking. In particular, the problem consists of planning the motion of the robots to optimally track targets despite robotic/sensor failures. The optimality of tracking is captured by an objective function such as the expected number of detected targets or the reduction in the uncertainty of the targets’ positions.

A. Framework

**Attacks:** We assume that the maximum number of robotic/sensor failures are known, and denote it by $\alpha$. At any time at most $\alpha$ robots/sensor may fail. In addition, without loss of generality, the set of robots that fail may vary over time. A robot that fails at time $t$ may be active at another time $t'$.

The rest of our problem formulation, e.g., assumptions about the targets, robots, and the objective function, follows the standard in the target tracking literature; see [12] and the references therein. Specifically:

- **Targets:** Targets exists in an area of interest (environment). The targets can be ground or aerial vehicles, and can be mobile or immobile. They can be distinguishable [11] or indistinguishable [12]. Their number can be known [11] or unknown [12], fixed [11] or time-varying [12]. The target motion model can be known (e.g., a single integrator with known maximal speed [19]) or partially known; in the latter case, data-driven learning techniques may be employed [12].

- **Robots/sensors:** We consider a team of mobile robots, and denote it by $\mathcal{R}$. The team is tasked to track targets in an area of interest. The robots can be ground or aerial vehicles (e.g., quad-rotors). We assume that the robots can communicate with each other at all times.

The robots carry onboard sensors (e.g., cameras or lidars), which enable the team’s tracking capability. In particular, each robot $r \in \mathcal{R}$, at every time $t$, receives measurements from targets detected in the field-of-view of its sensors. Additional measurements may be obtained from off-board sensors in the environment. Given the measurements and a target model, each robot employs a detector and trajectory estimator. If the targets are distinguishable and their number is known, a Kalman or particle filter can be employed [20], whereas, if the targets are indistinguishable and their number is unknown, a Random Finite Sets (RFS) filter can be employed [12]. In both cases, the robots have only an estimate of the targets’ true positions. The estimate is represented by a set of possible target locations in the environment. Given a target model, the robots propagate this set to obtain a predicted target position, by employing one of the above techniques.

The robots can also move in the environment, and detect that way multiple targets per motion step (Fig. 2). The robot trajectory generation framework is as follows: We assume that the robots have perfect localization (e.g., using GPS). Time is divided into rounds of finite duration denoted by $T$ (without loss of generality, we assume it fixed). At the beginning of each round, each robot generates a set of candidate trajectories, one of which will be followed in the current round. The trajectories can be generated by employing, for example, motion-space discretization [12] or spatial-sampling methods [21]. We denote the set of valid trajectories, for a round that starts at a time $t$, and for a robot $r \in \mathcal{R}$, by $\mathcal{T}_{r,t}$. We denote by $\mathcal{T}_{\mathcal{R},t}$ the set of all robots’ valid trajectories, i.e., $\mathcal{T}_{\mathcal{R},t} \triangleq \cup_{r \in \mathcal{R}} \mathcal{T}_{r,t}$.

1Henceforth, we refer to robotic and sensor failures interchangeably.

2The uncertainty in the robot’s position can be incorporated in the uncertainty in the targets’ estimates [11].
problem of traditional target tracking with mobile robots, by protecting (in a receding horizon fashion) the robots’ motion plan against failures.

III. ALGORITHM FOR PROBLEM 1

We present the first scalable algorithm for Problem 1, by capitalizing on the algorithmic results in [16]. The pseudocode of the algorithm is described in Algorithm 1.

A. Intuition behind Algorithm 1

Problem 1 selects trajectories for all robots, denoted by the set $S$ in eq. (1), so to maximize the value of the objective function $f$ despite that $S$ can incur a removal $A$ of $\alpha$ elements.
due to robotic failures. In this context, Algorithm 1 aims to maximize $f$ by constructing $S$ as the union of two sets, the $S_1$ and $S_2$ (line 16), whose role we describe below.

Set $S_1$ approximates a worst-case set removal from $S$: Algorithm 1 aims with the trajectory set $S_1$ to capture a worst-case removal of $\alpha$ trajectories among the trajectories Algorithm 1 will select in $S$. Equivalently, $S_1$ is aimed to act as a “bait” to an attacker that selects to remove the best $\alpha$ trajectories from $S$ (best with respect to the trajectories’ contribution towards maximizing the function $f$). However, the problem of selecting the best trajectories in $T_R$ per Problem 1 is combinatorial and, in general, intractable [23]. For this reason, Algorithm 1 aims to approximate the best set of $\alpha$ trajectories, by letting $S_1$ be the trajectories with the largest contributions to the value of $f$ (lines 3). In addition, since $S$ needs to satisfy the constraint that each robot $r \in R$ can be assigned one trajectory, Algorithm 1 constructs $S_1$ so that not only $|S_1| \leq \alpha$ but also $|S_1 \cap T_r| \leq 1$ for all $r \in R$ (lines 4-6).

Set $S_2$ is such that the set $S_1 \cup S_2$ approximates solution to Problem 1: Assuming $S_1$ is the set to be removed from Algorithm 1’s selection $S$, Algorithm 1 needs to select a trajectory set $S_2$ to complete the construction of $S$. In particular, for $S = S_1 \cup S_2$ to be a solution to Problem 1, Algorithm 1 needs to select $S_2$ as a best set of trajectories from $T_R \setminus S_1$ subject to the natural constraint that one trajectory is assigned to each robot (lines 11-13). Nevertheless, the problem of selecting a best set of elements subject to such a constraint is combinatorial and, in general, intractable [23]. Hence, Algorithm 1 aims to approximate such a best set, using the greedy procedure in lines 9-15.

Overall, Algorithm 1 constructs the sets $S_1$ and $S_2$ to approximate with their union $S$ an optimal solution to Problem 1.

We next describe the steps in Algorithm 1 in more detail.

B. Description of steps in Algorithm 1

Algorithm 1 executes four steps:

a) Initialization (line 1): Algorithm 1 defines 4 sets, the $S_1$, $M_1$, $S_2$, and $M_2$, and initializes each of them with the empty set (line 1). The purpose of $S_1$ and $S_2$ is to construct the set $S$. Specifically, the union of $S_1$ and $S_2$ constructs $S$ by the end of Algorithm 1 (line 16). The purpose of $M_1$ and of $M_2$ is to support the construction of $S_1$ and $S_2$. During the construction of $S_1$, Algorithm 1 stores in $M_1$ the trajectories in $T_R$ that have either been included already or cannot be included in $S_1$ (line 7); that way, Algorithm 1 keeps track of which trajectories remain to be checked if they could be added in $S_1$ (line 5). During the construction of $S_2$, Algorithm 1 stores in $M_2$ the trajectories of $T_R \setminus S_1$ that have either been included already or cannot be included in $S_2$ (line 14); that way, Algorithm 1 keeps track of which trajectories remain to be checked if they could be added in $S_2$ (line 12).

b) Construction of set $S_1$ (lines 2-8): Algorithm 1 constructs $S_1$ sequentially by adding one trajectory at a time from $T_R$ to $S_1$. Specifically, $S_1$, being the “bait” set, is constructed such that it satisfies both the trajectory assignment constraint (one trajectory per robot) and the failures cardinality constraint (line 4). Also, $S_1$ is constructed such that each trajectory $s \in T_R$ added in $S_1$ achieves the highest value of $f(s)$ among all the trajectories in $T_R$ that have not been yet added in $S_1$ and can be added in $S_1$ (line 5).

c) Construction of set $S_2$ (lines 9-15): Algorithm 1 constructs the set $S_2$ sequentially, by picking greedily trajectories from the set $T_R \setminus S_1$ such that $S_1 \cup S_2$ satisfies the trajectory assignment constraint in Problem 1 (one trajectory per robot). Specifically, the greedy procedure in Algorithm 1’s “while loop” (lines 9-15) selects a trajectory $y \in T_R \setminus (S_1 \cup M_2)$ to add in $S_2$ only if $y$ maximizes the value of $f(S_2 \cup \{y\}) - f(S_2)$, where the set $M_2$ stores the trajectories that either have already been added to $S_2$ or have been considered to be added to $S_2$ but they were not since the resultant set $S_1 \cup S_2$ would not satisfy the trajectory assignment constraint.

d) Construction of set $S$ (line 16): Algorithm 1 constructs the set $S$ as the union of the previously constructed sets $S_1$ and $S_2$ (lines 16).

In sum, Algorithm 1 proposes a trajectory assignment $S$ as solution to Problem 1. In particular, Algorithm 1 constructs $S$ to withstand any compromising robotic/sensor failure.

IV. PERFORMANCE ANALYSIS OF ALGORITHM 1

We quantify the performance of Algorithm 1, by bounding its running time, and its approximation performance. To this end, we use the following notion of curvature for set functions.

A. Constrained curvature of monotone submodular functions

Definition 1 (Constrained curvature). Consider a set $T$, and a non-decreasing submodular set function $f : 2^T \rightarrow \mathbb{R}$ such that (without loss of generality) for any element $s \in T$, it is $f(s) \neq 0$. Moreover, consider a collection of subsets of $T$, denoted by $\mathcal{I}$; e.g., $\mathcal{I}$ represents admissible sets where $f$ can be evaluated at. Then, the constrained curvature of $f$ over $\mathcal{I}$ is:

$$\nu_f(\mathcal{I}) \triangleq 1 - \min_{S \in \mathcal{I}} \min_{s \in S} \frac{f(S) - f(S \setminus \{s\})}{f(s)}.$$

The curvature $\nu_f$ measures how far $f$ is from being additive. In particular, Definition 1 implies $0 \leq \nu_f \leq 1$: If $\nu_f = 0$, then for all sets $S \in \mathcal{I}$ it holds $f(S) = \sum_{s \in S} f(s)$. In contrast, if $\nu_f = 1$, then there exist a set $S \in \mathcal{I}$ and an element $s \in T$ such that $f(S) = f(S \setminus \{s\})$; that is, in the presence of $S \setminus \{s\}$, the element $s$ loses all its contribution to the value of $f(S)$. Notably, Definition 1 adapts the notion of curvature discussed in [24] to the case where the set $S$ is constrained in an $\mathcal{I}$, instead of $S$ being able to be any subset of $T$.

For example, in reference to the target tracking framework of Section II, consider the expected number of detected targets as a function of the robot trajectories. Then, this function has curvature zero if each robot detects different targets from the rest of the robots. In contrast, it has curvature one if, for example, at least two robots by following their trajectories receive the exact same measurements.
B. Performance Analysis for Algorithm 1

**Theorem 1 (Performance of Algorithm 1).** Consider an instance of Problem 1, the notation therein, the notation in Algorithm 1, and the definitions:

- let the number $f^*$ be the (optimal) value to Problem 1;
- given a set $S$ as solution to Problem 1, let $A^*(S)$ be a worst-case set removal from $S$, per Problem 1, that is: $A^*(S) \in \arg \min_{A \subseteq S, |A| \leq \alpha} f(S \setminus A)$. Evidently, a removal from $S$ corresponds to a set of robot/sensor failures;
- define $h(|R|, \alpha) \triangleq \max \{1/(1 + \alpha), 1/(|R| - \alpha)\}$.

Finally, without loss of generality, consider that $f(0) = 0$.

The performance of Algorithm 1 is bounded as follows:

1) (Approximation performance) Algorithm 1 returns a trajectory set $S$ such that each robot is assigned a single trajectory, and:

$$\frac{f(S \setminus A^*(S))}{f^*} \geq \frac{\max \{1 - \nu_f(I), h(|R|, \alpha)\}}{2}, \quad (3)$$

where $I$ is the collection of valid trajectory assignments to robots per Problem 1 (one trajectory per robot), that is: $I \triangleq \{S : S \subseteq T_R, |S \cap T_r| = 1 \text{ for all } r \in R\}$.

2) (Running time) Algorithm 1 constructs the trajectory set $S$ as a solution to Problem 1 with $O(|T_R|^2)$ evaluations of the function $f$ [11], [12], and Algorithm 1 also terminates with the same time.

**V. Numerical Evaluation**

We present MATLAB and Gazebo evaluations of our algorithm (Algorithm 1) that demonstrate both the necessity for resilient target tracking and the benefits of our approach. In particular, both evaluations demonstrate: (i) the near-optimal performance of Algorithm 1, since the algorithm performs close to the brute-force algorithm (which is viable only in small-scale scenarios) and is superior to the greedy and random heuristics; and (ii) the superior robustness of Algorithm 1 to scenarios where non-worst-case or even no attacks occur. Our MATLAB and Gazebo implementations are available online.\(^5\)

**Compared algorithms.** We compare Algorithm 1 with three other algorithms. The algorithms differ in how they select the robot trajectories. The first algorithm is an optimal, brute-force algorithm, and it attains the optimal value for Problem 1. Evidently, the brute-force approach is viable only when the number of available robots is small. We refer to this

\(^5\)https://github.com/raaslab/resilient_target_tracking.git
algorithm by “brute-force.” The second algorithm is a greedy algorithm that ignores the possibility of robotic/sensor attacks, and picks greedily the robot trajectories per the algorithm proposed in [17]; we refer to this algorithm by “greedy.” The third algorithm is a random algorithm that picks randomly (uniformly) the robot trajectories; we refer to this algorithm by “random.” Finally, we refer to Algorithm 1 by “resilient.”

A. MATLAB evaluation over one step with static targets

We study the effect of the number of targets and of the attack strategy by running the algorithms over random instances of Problem 1 for a single round (one-step time horizon).

Simulation setup. We consider 6 robots and a number of targets \( m \) that varies from 30 to 60. We set the number of attacks \( \alpha \) equal to 3 and 4. A top view of the robots and targets is shown in Fig. 4. We assume that each robot \( r_i \in \mathcal{R} \) flies on a fixed plane and has 4 trajectories: forward, backward, left, and right, denoted by \( \tau^j_i \) for \( j = 1, 2, 3, 4 \), respectively. Each robot \( r_i \) has a square field-of-view, centered at the planar position of robot \( r_i \), and is illustrated in Fig. 4 by the darker blue square of dimension \( l_o \times l_o \). Once each robot selects a trajectory, it flies a distance \( l_f \) along that trajectory. Thus, each trajectory \( \tau^j_i \) has a rectangular tracking region with length \( l_t = l_f + l_o \) and width \( l_o \); we set \( l_t = 10 \) and \( l_o = 3 \) for all robots. For each number of targets \( m = 30, 31, \ldots, 60 \), the planar positions of the robots and targets are randomly generated in the 2D space \([0,10] \times [0,10] \in \mathbb{R}^2\) across 30 trials. We consider that the robots have already available an estimate of the targets position. For each trial, all algorithms are executed with the same initialization, i.e., the same positions of targets and robots. All algorithms are executed for one round.

The algorithms’ performance is captured as the number of tracked targets given the selected robot trajectories. We examine the performance across 3 settings that differ on how an attacker would select to attack \( \alpha \) robots so to minimize the performance of the remaining robots: we first consider an attacker that uses a brute-force algorithm to find an optimal robot attack; this scenario is in agreement with the definition of Problem 1, where the attacks are indeed worst-case attacks. Then, we consider an attacker that uses the greedy algorithm in [17] to approximate an optimal robot attack; and finally, we consider an attacker that chooses randomly a robot attack (uniformly across all robots). We examine the last two cases (Fig. 5(c) and Fig. 5(d)) as part of a sensitivity analysis of Algorithm 1’s performance against non-worst-case attacks.

Results. The comparison results are reported Fig. 5. The following observations from Fig. 5 are due:

a) Close-to-optimality of Algorithm 1: Algorithm 1 is designed to guarantee superior performance in the presence of worst-case attacks; indeed, per Fig. 5(a) (and per Fig. 5(b)), Algorithm 1 —colored blue in Fig. 5— has on average superior performance to the greedy and random heuristics. In particular, Algorithm 1’s performance is close to the optimal achieved by the brute force algorithm (red in Fig. 5).

b) Robustness of Algorithm 1’s performance to non-worst-case attacks: Although Algorithm 1 is designed to guarantee superior performance for worst-case attacks, in practice, the attack of robots may not necessarily be the worst-case one. For example, from the perspective of an attacker, finding the optimal robot attack is also an NP-hard problem, since it constitutes a cardinality constrained submodular minimization problem [25]. It is therefore relevant to ask whether Algorithm 1, being an approximation algorithm, will indeed have a better target tracking performance when the attacks are non-worst-case. By comparing Fig. 5(a) with both Fig. 5(c) and Fig. 5(d), we observe that for each given number of targets (horizontal axes in each plot in Fig. 5) the performance of Algorithm 1 increases for non-worst-case attacks. For example, for 30 targets, when the attack is worst-case (Fig. 5(a)) Algorithm 1 achieves 14 tracked targets, whereas: when the attack is greedy the performance increases to 17 (Fig. 5(c)). When the attacks are random, the performance increases to 18 (Fig. 5(d)). Overall, since Algorithm 1 is designed to protect against worst-case attacks, it also protects at least equally well against non-worst-case attacks (as it would be expected). Importantly, that way an attacker is forced by Algorithm 1
to deploy a worst-case attack, which is exactly the scenario that Algorithm 1 guarantees protection from. That conclusion makes also irrelevant the observation that, for example, in the case of random attacks (Fig. 5(d)) both Algorithm 1 and the greedy heuristic perform similarly. Notably, in the case of greedy attacks (Fig. 5(c)), Algorithm 1 is again superior to both the greedy and the random heuristics. In general, in the above simulation setup, Algorithm 1 achieves a superior and close-to-optimal performance, and remains superior even against non-worst-case attacks.

B. Gazebo evaluation over multiple steps with mobile targets

We study the effect of the number of targets and of the attack strategy by running the algorithms across multiple rounds (multi-step time horizon). That way, we take into account the kinematics and dynamics of the robots, as well as the fact that the kinematics and dynamics of the robot, the actual trajectories of the targets, and the sensing noise may force the robots to track fewer targets than expected.

Simulation setup. We consider a scenario of 4 aerial robots tasked to track 30 ground mobile targets (Fig. 6(a)). We set the number of attacks $\alpha$ equal to 2. We also visualize the robots, their field-of-view, and the targets using the Rviz environment (Fig. 6(b)): in particular, we visualize the robots as spherical markers, their field-of-views as colored areas with the same color as their corresponding robot, and the targets as white cylindrical markers. Similarly to the MATLAB simulation setup above, each robot has 4 trajectories (forward, backward, left, and right), and flies on a different fixed plane (to avoid collision with other robots). Moreover, we set the tracking length $l_t = 6$ and width $l_o = 3$ for all robots. We assume each target has the single integrator motion model

$$p_j^t(k + 1) = p_j^t(k) + v_j^t(k),$$

where $p_j^t$ and $v_j^t$ denote the position and the velocity of target $j = 1, \ldots, 30$, respectively. The robots obtain noisy position measurements of all targets. They use a Kalman filter for updating the estimated position of the target at the next round. The targets’ velocity is initialized to zero and is updated by using two consecutive measurements and the time interval of these two measurements, as follows:

$$v_j^t(k') = (\tilde{p}_j^t(k') - \tilde{p}_j^t(k))/(k' - k),$$

where $\tilde{p}_j^t(k')$ and $\tilde{p}_j^t(k)$ are two consecutive position measurements of target $j$ at round $k'$ and round $k$ with $k' > k$.

For each algorithm (see “Compared algorithms” at the beginning of the section), at each round each robot selects one of its 4 trajectories. Then, the robots fly a $l_f = 3$ distance along their selected trajectory. When an attack happens, we assume that the attacked robot’s camera is turned-off; nevertheless, we assume that it can be active again at the next round, so that at each round the worst-case set of $\alpha$ robots is considered failed. We repeat this process for 50 rounds.

At each round, we capture the performance of each algorithm with the expected number of targets tracked. We first compare the algorithms with respect to the average and the standard deviation of the expected number of targets tracked. Moreover, we compare the sensitivity of the algo-
rithms' performance against the case where no attacks are present: specifically, we compare the average and the standard deviation of their attack rate per round, which is defined—for an algorithm that selects a set of trajectories \( S \)—by

\[
f(S) = \frac{f(S) - f(S \setminus A^*(S))}{f(S)},
\]

where \( f(S) \) is the expected number of targets tracked in the presence of no attacks, and \( f(S \setminus A^*(S)) \) is the expected number of targets tracked in the presence of an optimal attack \( A^*(S) \). All in all, the above definition of attack rate captures how much worse is the performance of an algorithm in the presence of attacks than in the absence of attacks. A video for this implementation is available online.6

**Results.** The comparison results are reported Fig. 7. The following observations from Fig. 7 are due:

a) **Close-to-optimality of Algorithm 1:** Fig. 7(a) suggests that Algorithm 1 has on average superior performance than the greedy and the random. In particular, Algorithm 1’s performance is close to the optimal, as it is achieved by a brute force algorithm.

b) **Robustness of Algorithm 1’s performance to no-attacks:** Per Fig. 7(b), Algorithm 1 exhibits superior attack rate than the greedy and random heuristics, and as a result, when for example the scenario “at most \( \alpha \) attacks per round” and the scenario “no attacks per round” happen with equal probability, Algorithm 1 still guarantees superior average performance.

All in all, in the above simulation setup, Algorithm 1 achieves a superior and close-to-optimal performance, and remains superior even when no-attacks happen.

VI. CONCLUDING REMARKS & FUTURE WORK

We take the first steps to protect critical target tracking tasks from robot failures (Problem 1). In particular, we provide the first algorithm for Problem 1, and proved its guaranteed performance against any number of failures, and even for targets that are indistinguishable and/or unknown. We demonstrate the need for resilient target tracking and the robustness of our algorithm with MATLAB and Gazebo evaluations.

This work opens a number of avenues for future research, both theoretical and experimental. Future theoretical work in theory includes the decentralized design of the robots’ motion plan. Moreover, online extensions of Algorithm 1, that guarantee near-optimality across multiple rounds, are natural next steps. Future experimental work includes real-world testing of our resilient target tracking framework in the context of practical applications of surveillance and patrolling.

**References**

[1] P. Tokekar, E. Branson, J. Vander Hook, and V. Isler, “Tracking aquatic invaders: Autonomous robots for monitoring invasive fish,” *IEEE Robotics & Automation Magazine*, vol. 20, no. 3, pp. 33–41, 2013.

[2] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, “Cooperative air and ground surveillance,” *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 16–25, 2006.

[3] U. Zengin and A. Dogan, “Real-time target tracking for autonomous uavs in adversarial environments: A gradient search algorithm,” *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 294–307, 2007.

[4] V. Kumar and N. Michael, “Opportunities and challenges with autonomous micro aerial vehicles,” *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1279–1291, 2012.

[5] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, “Information acquisition with sensing robots,” in *IEEE International Conference on Robotics and Automation*, 2014, pp. 6447–6454.

[6] C. Robin and S. LaCroix, “Multi-robot target detection and tracking: taxonomy and survey,” *Autonomous Robots*, vol. 40, no. 4, pp. 729–760, 2016.

[7] J. R. Spletzer and C. J. Taylor, “Dynamic sensor planning and control for optimally tracking targets,” *The International Journal of Robotics Research*, vol. 22, no. 1, pp. 7–20, 2003.

[8] E. W. Frew, *Observer trajectory generation for target-motion estimation using monocular vision*. Stanford University, 2003.

[9] M. Schwager, B. J. Julian, M. Angermann, and D. Rus, “Eyes in the sky: Decentralized control for the deployment of robotic camera networks,” *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1541–1561, 2011.

[10] A. Pierson, Z. Wang, and M. Schwager, “Intercepting rogue robots: An algorithm for capturing multiple evaders with multiple pursuers,” *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 530–537, 2017.

[11] P. Tokekar, V. Isler, and A. Franchi, “Multi-target visual tracking with aerial robots,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 3067–3072.

[12] P. Dames, P. Tokekar, and V. Kumar, “Detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots,” *The International Journal of Robotics Research*, vol. 36, no. 13–14, pp. 1540–1553, 2017.

[13] E. Sless, N. Agmon, and S. Kraus, “Multi-robot adversarial patrolling: Facing coordinated attacks,” in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 1093–1100.

[14] H. González-Banos, C.-Y. Lee, and J.-C. Latombe, “Real-time combinatorial tracking of a target moving unpredictably among obstacles,” in *Robotics and Automation, 2002. Proceedings. ICRA’02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 1683–1690.

[15] S. I. Roumeliotis, G. S. Sukhatme, and G. A. Bekey, “Sensor fault detection and identification in a mobile robot,” in *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, vol. 3. IEEE, 1998, pp. 1383–1388.

[16] V. Touzamas, A. Jadabaha, and G. J. Pappas, “Resilient Non-Submodular Maximization over Matroid Constraints,” arXiv e-prints: 1804.01013, 2018.

[17] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, “An analysis of approximations for maximizing submodular set functions—I,” *Polyhedral combinatorics*, pp. 73–87, 1978.

[18] B. Schlotfeldt, V. Touzamas, D. Thakur, and G. J. Pappas, “Resilient active information gathering with mobile robots,” *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, to appear.

[19] S. Martinez, J. Cortes, and F. Bullo, “Motion coordination with distributed information,” *IEEE Contr. Syst.*, vol. 27, no. 4, pp. 75–88, 2007.

[20] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[21] C. J. Green and A. Kelly, “Toward optimal sampling in the space of paths,” in *Robotics Research*, 2010, pp. 281–292.

[22] R. B. Myerson, *Game theory*. Harvard University Press, 2013.

[23] D. S. Hochbaum and A. Pathria, “Analysis of the greedy approach in problems of maximum k-coverage,” *Naval Research Logistics*, vol. 45, no. 6, pp. 615–627, 1998.

[24] M. Conforti and G. Cornuéjols, “Submodular set functions, matroids and the greedy algorithm,” *Discrete Applied Mathematics*, vol. 7, no. 3, pp. 251–274, 1984.

[25] R. Iyer, S. Jegelka, and J. Bilmes, “Fast semidifferential-based submodular function optimization,” in *International Conference on Machine Learning*, 2013, pp. 855–863.

[26] G. Nemhauser, L. Wolsey, and M. Fisher, “An analysis of approximations for maximizing submodular set functions—II,” *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.
APPENDIX A
MONOTONICITY AND SUBMODULARITY

We define monotonicity and submodularity, and discuss them within the target tracking framework of Section II.

**Definition 2 (Monotonicity [26]).** Consider a finite ground set $\mathcal{T}$. Then, a set function $f : 2^\mathcal{T} \mapsto \mathbb{R}$ is non-decreasing if and only if for any sets $S \subseteq S' \subseteq \mathcal{T}$, it holds $f(S) \leq f(S')$.

For example, the expected number of detected targets is a non-decreasing function in the choice of robot trajectories: as the number of robots participating in the target tracking increases, the expected number of detected targets also increases.

**Definition 3 (Submodularity [26, Proposition 2.1]).** Consider any finite set $\mathcal{T}$. Then, the set function $f : 2^\mathcal{T} \mapsto \mathbb{R}$ is submodular if and only if for any sets $S \subseteq S' \subseteq \mathcal{T}$, and any element $s \in \mathcal{T}$, it holds $f(S \cup \{s\}) - f(S) \geq f(S' \cup \{s\}) - f(S')$.

Definition 3 implies that a set function $f$ is submodular if and only if it satisfies the following diminishing returns property: for any set $S \subseteq \mathcal{T}$, and any element $s \in \mathcal{T}$, the marginal gain $f(S \cup \{s\}) - g(S)$ is non-increasing.

For example, the expected number of detected targets is submodular in the choice of robot trajectories [12, Lemma 2]: in the presence of more robots, the addition of a robot has a smaller effect on increasing the number of detected targets.