Distributed Attack-Robust Submodular Maximization for Multirobot Planning

Lifeng Zhou, *Member, IEEE*, Vasileios Tzoumas, *Member, IEEE*, George J. Pappas, *Fellow, IEEE*, and Pratap Tokekar, *Member, IEEE*

***Abstract*—In this article, we design algorithms to protect swarm-robotics applications against sensor denial-of-service attacks on robots. We focus on applications requiring the robots to jointly select actions, e.g., which trajectory to follow, among a set of available actions. Such applications are central in large-scale robotic applications, such as multirobot motion planning for target tracking. But the current attack-robust algorithms are centralized. In this article, we propose a general-purpose distributed algorithm toward robust optimization at scale, with local communications only. We name it *distributed robust maximization* (DRM). DRM proposes a divide-and-conquer approach that distributively partitions the problem among cliques of robots. Then, the cliques optimize in parallel, independently of each other. We prove DRM achieves a close-to-optimal performance. We demonstrate DRM's performance in Gazebo and MATLAB simulations, in scenarios of *active target tracking with swarms of robots*. In the simulations, DRM achieves computational speed-ups, being 1 to 2 orders faster than the centralized algorithms. *Yet*, it nearly matches the tracking performance of the centralized counterparts. Since, DRM overestimates the number of attacks in each clique, in this article, we also introduce an *improved distributed robust maximization* (IDRM) algorithm. IDRM infers the number of attacks in each clique less conservatively than DRM by leveraging three-hop neighboring communications. We verify IDRM improves DRM's performance in simulations.**

***Index Terms*—Adversarial attacks, approximation algorithm, distributed optimization, multirobot planning, robust optimization, submodular optimization, target tracking.**

I. Introduction

SAFE-CRITICAL tasks of surveillance and exploration often require mobile agility, and fast capability to detect, localize, and monitor. For example, consider the following tasks.

1) *Adversarial Target Tracking*: Track adversarial targets that move across an urban environment, aiming to escape [1].
2) *Search and Rescue*: Explore a burning building to localize any people trapped inside [2].

Such scenarios can greatly benefit from teams of mobile robots that are agile, act as sensors, and plan their actions rapidly. For this reason, researchers are 1) pushing the frontier on robotic miniaturization and perception [1]–[7], to enable mobile agility and autonomous sensing, and 2) developing distributed coordination algorithms [8]–[12], to enable multirobot planning, i.e., the joint optimization of robots' actions.

Particularly, distributed planning algorithms are important when one wishes to deploy large-scale teams of robots; e.g., at the swarm level of tens or hundreds of robots. One reason is that distributed algorithms scale better for larger numbers of robots than their centralized counterparts [8]. In addition, in large-scale teams, it is infeasible for all robots to communicate with each other; typically, each robot can communicate only with the robots within a certain communication range.

However, the safety of the abovementioned critical scenarios can be at peril. Robots operating in adversarial scenarios may get cyber-attacked or simply face failures, resulting in a temporary withdrawal of robots from the task (e.g., because of temporary deactivation of their sensors, blockage of their field of view, among others; cf., Fig. 1). We refer to such attacks as *denial-of-service* (DoS). Hence, in such adversarial scenarios, distributed attack-robust planning algorithms become necessary.[1]

In this article, we formalize a general framework for *distributed attack-robust multirobot planning* for tasks that require the maximization of submodular functions, such as active target tracking [13] and informative path planning [14] with multiple-robots.[2] We focus on *worst-case* attacks that can result in up to α robot withdrawals from the task at each step.

Attack-robust multirobot planning is computationally hard and requires accounting for all possible α withdrawals, a problem of combinatorial complexity. Importantly, even in the presence of no withdrawals, the problem of multirobot planning is NP-hard [16], [17]. All in all, the necessity for distributed attack-robust algorithms, and the inherent computational

Manuscript received 24 September 2021; revised 20 February 2022; accepted 15 March 2022. Date of publication 26 April 2022; date of current version 4 October 2022. This work was supported in part by the ARL CRA DCIST, the National Science Foundation under Grant 1943368, and in part by the Office of Naval Research under Grant N000141812829. This paper was recommended for publication by Associate Editor P. Robuffo Giordano and Editor F. Amigoni upon evaluation of the reviewers' comments. *(Corresponding author: Lifeng Zhou.)*

Lifeng Zhou was with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061 USA. He is now with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: lfzhou@seas.upenn.edu).

Vasileios Tzoumas is with the Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: vtzoumas@umich.edu).

George J. Pappas is with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: pappasg@seas.upenn.edu).

Pratap Tokekar is with the Department of Computer Science, University of Maryland, College Park, MD 20742 USA (e-mail: tokekar@umd.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TRO.2022.3161765.

Digital Object Identifier 10.1109/TRO.2022.3161765

[1] We henceforth consider the terms *attack* and *failure* equivalent, both resulting in robot withdrawals from the task at hand.

[2] Submodularity is a diminishing returns property [15], capturing the intuition that the more robots participate in a task, the less the gain (return) one gets by adding an extra robot toward the task.

1552-3098 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:54:40 UTC from IEEE Xplore. Restrictions apply.

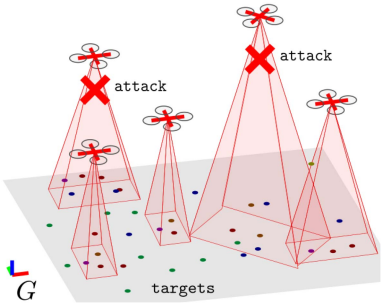


Fig. 1. Targets' attacks can block robots' field-of-view: In target tracking with aerial robots, the robots are mounted with down-facing cameras to track mobile targets (depicted as dots).

hardness motivates our goal in this article: to provide a distributed, provably close-to-optimal approximation algorithm. To this end, we capitalize on recent algorithmic results on centralized attack-robust multirobot planning [18]–[21] and present a distributed attack-robust algorithm.

Related Work: Researchers have developed several distributed, but attack-free, planning algorithms, such as [8]–[12]. For example, Atanasov *et al.* [8] developed a decentralized algorithm, building on the local greedy algorithm proposed in [22, Sec. IV], which guarantees a $1/2$ suboptimality bound for submodular objective functions. In [8], the robots form a string communication network, and sequentially choose an action, given the actions of the robots that have been chosen so far. Corah and Michael [11] proposed a speed-up of [8]'s approach, by enabling the greedy sequential optimization to be executed over directed acyclic graphs, instead of string ones. In scenarios where the robots cannot observe all the chosen actions so far, distributed, but still attack-free, algorithms for submodular maximization are developed in [23] and [24]. Other distributed, attack-free algorithms are developed in the machine learning literature on submodular maximization, toward sparse selection (e.g., for data selection or sensor placement) [25], instead of planning.

Researchers have also developed robust planning algorithms [18]–[21], [26]–[28]. Among these, Saulnier *et al.*, Saldana *et al.*, Mitra *et al.*, [26]–[28] focus on deceptive attacks, instead of DoS attacks. In more detail, [26] and [27] provide distributed resilient formation control algorithms to deal with malicious robots in the team that share deceptive information. Similarly, [28] provides a distributed resilient state estimation algorithm against robots executing Byzantine attacks by sending differing state estimates or not transmitting any estimates. In contrast to [26]–[28], the articles [20] and [21] consider DoS attacks in multirobot motion planning but in centralized settings: the work in [20] provides centralized attack-robust algorithms for active information gathering, and [21] for target tracking. The article [29] extended [20] to a model predictive control framework. The algorithms are based on the centralized algorithms proposed in [18] and [19]. Additional attack-robust algorithms are proposed in [30] and [31], which, however, are valid for only a limited number of attacks. For a thorough literature review on attack-robust combinatorial algorithms, we refer the reader to [32].

Contributions: Toward enabling attack-robust planning in multirobot scenarios requiring local communication among robots, we make the following contributions.

1) We present the algorithm *distributed robust maximization* (DRM): DRM distributively partitions the problem among cliques of robots, where all robots are within communication range. Then, naturally, the cliques optimize in parallel, using [21, Algorithm 1]. We prove (Section IV) the following.
   a) *System-Wide Attack-Robustness:* DRM is valid for any number α of worst-case attacks.
   b) *Computational Speed-Up:* DRM is faster up to a factor $1/K^2$ than its centralized counterparts in [21], where K is the number of cliques chosen by DRM. K depends on the inter-robot communication range, denoted henceforth by r_c, which is given as input to DRM (and, as a result, by changing r_c one controls K).
   c) *Near-to-Centralized Approximation Performance:* Even though DRM is a distributed algorithm, and faster than algorithm its centralized counterpart [21, Algorithm 1], DRM achieves a near-to-centralized performance, having a suboptimality bound equal to [21, Algorithm 1]'s.

2) We design an *improved distributed robust maximization* (IDRM) algorithm to relax the conservativeness of DRM in inferring the number of attacks against each clique (see Section V). Specifically, DRM assumes that the number of attacks against each clique is equal to α, the *total* number of attacks against all robots. Instead, in IDRM, robots communicate with their three-hop neighbors in the communication network of all robots to infer the number of attacks.[3] IDRM has the same approximation performance as DRM. In our numerical evaluations, it maintains a comparable running time to DRM (see Section V-B).

Notably, the proposed algorithms in this article allow for the communication graph to be disconnected.

Numerical Evaluations: First, we present Gazebo and MATLAB evaluations of DRM, in scenarios of *active target tracking with swarms of robots* (see Section VI-A). All simulation results demonstrate DRM's computational benefits: DRM runs 1 to 2 orders faster than its centralized counterpart in [21], achieving running times 0.5–15 ms for up to 100 robots. And, yet, DRM exhibits negligible deterioration in performance (in terms of number of targets covered). Second, we compare the performance of DRM and IDRM through MATLAB simulations (see Section VI-B). We show that IDRM performs better than DRM in practice (higher target coverage), while maintaining a comparable running time.

Comparison With the Preliminary Results: Preliminary results were presented in [33] and [34]. In this article, we present for the first time the algorithm *IDRM* (see Section V). Moreover, the corresponding MATLAB simulations are new (see Section VI-B). Also, we present a formal analysis of DRM's computation and approximation performance (see Appendices A and B). Further, we analyze the approximation of a myopic algorithm and compare its practical performance with DRM and IDRM through MATLAB simulations (see Remark 2 and Appendix C).

[3] A robot's three-hop neighbors are its neighbors, its neighbors' neighbors, and its neighbors' neighbors' neighbors.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:54:40 UTC from IEEE Xplore. Restrictions apply.

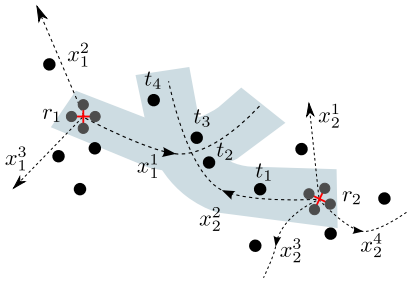


Fig. 2. Robots choose trajectories from a set of motion primitives: At each time step, each robot has a set of motion primitives to choose as its trajectory (each covering different targets, depicted as dots). For example, robot one has three motion primitives, $\{x_1^1, x_1^2, x_1^3\}$, and robot two has four motion primitives, $\{x_2^1, x_2^2, x_2^3, x_2^4\}$, where x_1^1 covers two targets, $\{t_2, t_3\}$, and x_2^2 covers four targets, $\{t_1, t_2, t_3, t_4\}$. In combination, however, the two motion primitives totally cover four targets, $\{t_1, t_2, t_3, t_4\}$.

II. Problem Formulation

We formalize the problem of *distributed attack-robust submodular maximization for multirobot planning*. At each time-step, the problem asks for assigning actions to the robots, to maximize an objective function despite attacks. For example, in the active target tracking with aerial robots (see Fig. 1), the robots' possible actions are their motion primitives; the objective function is the number of covered targets; and the attacks are field-of-view blocking attacks.

We next introduce our framework in more detail.[4]

1) Robots: We consider a multirobot team $\mathcal{R}$. At a given time-step, p_i is robot i's position in the environment ($i \in \mathcal{R}$). We define $\mathcal{P} \triangleq \{p_1, \ldots, p_{|\mathcal{R}|}\}$ with $|\mathcal{R}| = N$.

2) Communication Graph: Each robot communicates only with those robots within a prescribed *communication range*. Without loss of generality, we assume all robots to have the same communication range r_c. That way, an (undirected) *communication graph* $G = \{\mathcal{R}, \mathcal{E}\}$ is induced, with nodes the robots $\mathcal{R}$, and edges $\mathcal{E}$ such that $(i, j) \in \mathcal{E}$ if and only if $\|p_i - p_j\|_2 \leq r_c$. The *neighbors* of robot i are all robots within the range r_c, and are denoted by $\mathcal{N}_i$.

3) Action Set: Each robot i has an available set of *actions* to choose from; we denote it by $\mathcal{X}_i$. The robot can choose at most one action at each time, due to operational constraints; e.g., in motion planning, $\mathcal{X}_i$ denotes robot i's motion primitives, and the robot can choose only one motion primitive at a time to be its trajectory. For example, in Fig. 2, we have two robots, where $\mathcal{X}_1 = \{x_1^1, x_1^2, x_1^3\}$ (and robot one chooses x_1^1 as its trajectory) and $\mathcal{X}_2 = \{x_2^1, x_2^2, x_2^3, x_2^4\}$ (and robot two chooses x_2^2 as its trajectory). We let $\mathcal{X} \triangleq \bigcup_{i \in \mathcal{R}} \mathcal{X}_i$. Also, $\mathcal{S} \subseteq \mathcal{X}$ denotes a valid assignment of actions to all robots. For instance, in Fig. 2, $\mathcal{S} = \{x_1^1, x_2^2\}$.

4) Objective Function: The quality of each $\mathcal{S}$ is quantified by a nondecreasing and submodular function $f : 2^{\mathcal{X}} \to \mathbb{R}$. For example, this is the case in active target tracking with mobile robots, when f is the number of covered targets [17]. As shown in Fig. 2, the number of targets covered by the chosen actions, $\mathcal{S} = \{x_1^1, x_2^2\}$, is $f(\mathcal{S}) = 4$.

5) Attacks: At each time step, we assume the robots encounter worst-case sensor DoS attacks. Each attack results in a robot's sensing removal (at least temporarily, i.e., for the current time step). In this article, we study the case where the maximum number of attacks at each time step is known, per Problem 1 mentioned below. We denote the maximum number of attacks by α. We also assume that if a robot encounters a sensor DoS attack, it can still act as an information relay node to communicate with other robots.

Problem 1 (Distributed attack-robust submodular maximization for multirobot planning): At each time step, the robots, by exchanging information only over the communication graph G, assign an action to each robot $i \in \mathcal{R}$ to maximize f against α worst-case attacks/failures

$$\max_{\mathcal{S} \subseteq \mathcal{X}} \min_{\mathcal{A} \subseteq \mathcal{S}} f(\mathcal{S} \setminus \mathcal{A}) \quad \text{s.t.} \quad |\mathcal{S} \cap \mathcal{X}_i| = 1, \text{ for all } i \in \mathcal{R}, |\mathcal{A}| \leq \alpha \tag{1}$$

where $\mathcal{A}$ corresponds to the actions of the attacked robots, and α is assumed known.[5]

Problem 1 is equivalent to a two-stage perfect information sequential game [35, Ch. 4] between the robots and an attacker. Particularly, the robots first select $\mathcal{S}$, and then, the attacker, *after observing* $\mathcal{S}$, selects the worst-case $\mathcal{A}$ (which is unknown to the robots). The "min" operator means that the attacker aims to minimize the robots' objective function f by removing up to α robots' actions $\mathcal{A}$, no matter what actions $\mathcal{S}$ the robots take. This is the worst-case attack on the robots' actions $\mathcal{S}$. The "min" operation is from the attackers' point of view, and the robots do not know which robots (or their actions $\mathcal{A}$) will be attacked. Under this assumption, the robots aim to maximize the objective function f by assuming the attacker executes up to α worst-case attacks.

Notably, a robot does not have to understand if it encounters a DoS attack or not, and it does not need to communicate such information to its neighbors either. That is because the robots' planning happens before the attacks take place. So the robots do not even have a way of knowing who gets a DoS attack, and that is why they need to consider the worst-case scenario. The DoS attacks are not permanent, and a rational attacker will choose the worst-case subset of sensors to attack at each time step. In other words, Problem 1 asks for planning algorithms to prepare the robot team to be robust to withstand the worst-case attacks *without* knowing the attack details (e.g., which robots' sensors get attacked).

III. Distributed Algorithm: DRM

We present DRM, a distributed algorithm for Problem 1 for the case where α is known (see Algorithm 1). DRM executes sequentially two main steps: *distributed clique partition* (DRM's

[4]**Notations.** Calligraphic fonts denote sets (e.g., $\mathcal{A}$). $2^{\mathcal{A}}$ denotes $\mathcal{A}$'s power set and $|\mathcal{A}|$ its cardinality. $\mathcal{A} \setminus \mathcal{B}$ are the elements in $\mathcal{A}$ not in $\mathcal{B}$.

[5]The constraint in (1) ensures that each robot chooses one action per time step (e.g., one trajectory among a set of primitive trajectories).

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:54:40 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: Distributed Robust Maximization (DRM).

Input: Robots' available actions $\mathcal{X}_i, i \in \mathcal{R}$; monotone and submodular function f; attack number α.
Output: Robots' actions $\mathcal{S}$.
1: Partition G to cliques $\mathcal{C}_1, \ldots, \mathcal{C}_K$ by calling DCP$(\mathcal{P}, r_c)$;
2: $\mathcal{S}_k \leftarrow \emptyset$ for all $k = \{1, \ldots, K\}$;
3: **for each clique** $\mathcal{C}_k$ **in parallel, do**
4: **if** $\alpha < |\mathcal{C}_k|$ **then**
5: $\mathcal{S}_k =$ central-robust $(\bigcup_{i \in \mathcal{C}_k} \mathcal{X}_i, f, \alpha)$;
6: **else**
7: $\mathcal{S}_k =$ central-robust $(\bigcup_{i \in \mathcal{C}_k} \mathcal{X}_i, f, |\mathcal{C}_k|)$;
8: **return** $\mathcal{S} = \bigcup_{k=1}^{K} \mathcal{S}_k$.

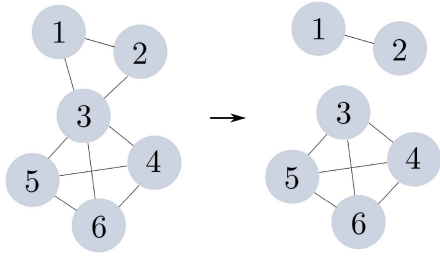


Fig. 3. DCP partitions a graph of six robots into two separate cliques. Particularly, after clique computation, robots 1–6 obtain cliques {1, 2, 3}, {1, 2, 3}, {3, 4, 5, 6}, {3, 4, 5, 6}, {3, 4, 5, 6}, and {3, 4, 5, 6}. Then, in the third communication round, robot 3 shares its cliques with its neighbors (i.e., tells its neighbors {1, 2} that it joins clique {3, 4, 5, 6}), and robots 1 and 2 reset their cliques as {1, 2} and {1, 2}.

line 1), and *per clique attack-robust optimization* (DRM's lines 2–8). During the first step, the robots communicate with their neighbors to partition G into cliques of *maximal* size (using Algorithm 2, named DCP in DRM's line 1).[6] During the second step, each clique computes an attack-robust action assignment (in parallel with the rest), using the centralized algorithm in [21]—henceforth, we refer to the algorithm in [21] as central-robust. Central-robust takes similar inputs to DRM: a set of actions, a function, and a number of attacks.

We describe DRM's two steps in more detail later; and quantify its running time and performance in Section IV.

A. Distributed Clique Partition (Step-1 of DRM)

We present the first step of DRM, *distributed clique partition*, which is inspired by [36, Algorithm 2] (DRM's line 1 that calls DCP, whose pseudocode is presented in Algorithm 2). The problem of distributed clique partition is inapproximable in polynomial time, since finding even a single clique of a desired size is inapproximable [37], even by centralized algorithms.

DCP requires three rounds of communications among neighboring robots to form separate cliques. In the first round, each robot i communicates to find its neighbors (see Algorithm 2, line 3). In the second round, it shares its augmented neighbor set $\mathcal{N}_i^+$ (containing its neighbors and itself) with its neighbors, and receives its neighbors' sets $\{\mathcal{N}_j^+\}$ (see Algorithm 2, line 4). Then, robot i intersects its augmented neighbor set $\mathcal{N}_i^+$ with each of its neighbors' augmented neighbor sets $\mathcal{N}_j^+$, and sets the largest intersection as its clique (see Algorithm 2, line 5).

The aforementioned clique computation of DCP differs from [36, Algorithm 2] in that in [36, Algorithm 2] each robot i computes its clique as the intersection of $\mathcal{N}_i^+$ and $\mathcal{N}_j^+$ where j is the neighbor with the largest degree in $\mathcal{N}_i$, whereas in DCP's line 5, each robot i computes its clique as the intersection of $\mathcal{N}_i^+$and $\mathcal{N}_j^+$, where j instead is the neighbor with the largest neighborhood overlap with i. That way, DCP is more likely to obtain larger cliques for each robot. Also, the cliques returned by [36, Algorithm 2] can overlap with each other. In order to form separate cliques, DCP executes the third round of communication to share the computed cliques among neighbors (see Algorithm 2, line 6). Specifically, each robot i tells its neighbors which clique it will join. If the clique of some neighboring robot j contains robot i but robot i chooses to join a different clique (by Algorithm 2, line 5), its neighboring robot j will update its clique by removing robot i from it. In this way, each robot i will eventually belong to a single clique, and thus nonoverlapping cliques can be generated. An illustrative example of DCP is shown in Fig. 3.

Remark 1: We partition the communication graph G into cliques instead of connected subgroups because we want robots to have a shorter communication time interval before making decisions. Robots within a clique have point-to-point communication, while robots in a connected subgroup need multihop communication to propagate information. Typically, multihop communication takes time when the diameter of the communication graph is large. More formally, point-to-point communication has $O(1)$ communication round, while multihop communication has $O(N)$ communication rounds in the worst case (e.g., a line graph). Therefore, we choose the clique partition model that enables robots to have a shorter communication time before making decisions.

Reducing the time of information sharing becomes essential when robots need to make decisions quickly. For example, in a multitarget tracking scenario, robots need to first share information (e.g., subsets of targets covered) and then decide their actions (e.g., movements). Since targets are mobile and may move fast, the robots need to choose actions as quickly as they can. Otherwise, the information shared can be outdated and thus misleading for decision making. Therefore, we use cliques with one-hop communication only to shorten the time of information sharing among robots.

Moreover, multihop communication may reduce communication bandwidth and increase energy consumption. For example, when robots communicate over a single channel, only one robot can be active to broadcast information during a certain time interval. Therefore, communicating messages over multiple hops result in a smaller communication bandwidth and a higher communication cost [38]. But investigating the number of communication hops to tradeoff energy consumption/bandwidth with performance could be an interesting future direction.

[6] A clique is a set of robots that can all communicate with each other.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:54:40 UTC from IEEE Xplore. Restrictions apply.

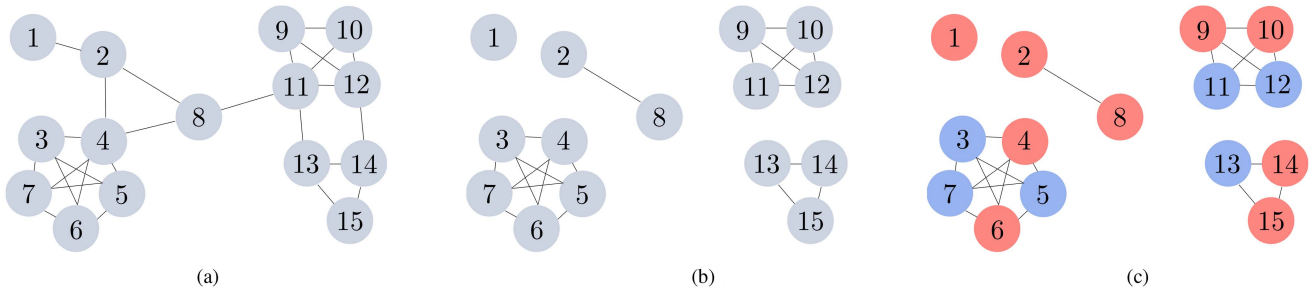


Fig. 4. Qualitative description of DRM's steps over the communication graph G in subfigure (a), composed of 15 robots. The number of conjectured attacks is considered to be $\alpha = 2$. In the first step, we assume DCP (DRM's line 1) partitions G into five cliques, as shown in subfigure (b). In the second step, all five cliques perform central-robust in parallel. Particularly, the cliques $\{(1), (2, 8)\}$, since α is larger than or equal to their size, consider that all of their robots will be attacked, and as a result they select all of their robots as baits [depicted with red in subfigure (c)], per central-robust. In contrast, the remaining three cliques, since α is smaller than their size, they select α of their robots as baits. The remaining robots [depicted with blue in subfigure (c)] of each clique choose their actions greedily, independently of the other cliques, and assuming that the red robots in their clique do not exist. (a) A communication graph G of 15 robots. (b) DCP partitions G into five cliques. (c) Each clique runs central-robust.

B. *Per Clique Attack-Robust Optimization (Step-2 of* DRM*)*

We now present DRM's second step: *per clique attack-robust optimization* (DRM's lines 2–8). Since the step calls central-robust as subroutine, we recall here its steps from [21]: central-robust takes as input the available actions of a set of robots $\mathcal{R}' \subseteq \mathcal{R}$ (i.e., the $\bigcup_{i\in\mathcal{R}'} \mathcal{X}_i$), a monotone submodular f, and a number of attacks $\alpha' \leq \alpha$, and constructs an action assignment $\mathcal{S}'$ by following a two-step process. First, it tries to approximate the conjectured worst-case attack to $\mathcal{S}'$, and, to this end, builds a "bait" set as part of $\mathcal{S}'$. Particularly, the bait set is aimed to attract all attacks at $\mathcal{S}'$, and for this reason, it has cardinality α' (the same as the number of conjectured attacks). In more detail, central-robust includes an action $x \in \bigcup_{i\in\mathcal{R}'} \mathcal{X}_i$ in the bait set (at most one action per robot, per Problem 1) only if $f(\{x\}) \geq f(\{x'\})$ for any other $x' \in \bigcup_{i\in\mathcal{R}'} \mathcal{X}_i$. That is, the bait set is composed of the "best" α' single actions. In the second step, central-robust 1) assumes the robots in the bait set are removed from $\mathcal{R}'$, and then 2) greedily assigns actions to the rest of the robots using the centralized greedy algorithm in [22, Sec. II] which ensures a near-optimal assignment (at least 1/2 close to the optimal). Specifically, for the remaining $|\mathcal{R}'| - \alpha'$ robots, denoted by $\mathcal{R}'_a$, the centralized greedy algorithm assigns actions $\mathcal{S}'_g$ for them as follows.

1: $\mathcal{S}'_g \leftarrow \emptyset$;
2: **while** $\mathcal{R}'_a \neq \emptyset$ **do**
3: $\quad (x'_g, i'_g) \in \underset{x\in\bigcup_{i\in\mathcal{R}'_a}\mathcal{X}_i}{\operatorname{argmax}} f(\mathcal{S}'_g \cup \{x\}) - f(\mathcal{S}'_g)$;
4: $\quad \mathcal{S}'_g \leftarrow \mathcal{S}'_g \cup \{x'_g\}$; $\mathcal{R}'_a \leftarrow \mathcal{R}'_a \setminus \{i'_g\}$.

Notably, it chooses an action x'_g and the corresponding robot i'_g with the maximal marginal contribution to the function value at each round (line 3).

The bait set is one way of approximating α' worst-case attacks to the action assignment $\mathcal{S}'$. By executing the worst-case attacks, the attacker may or may not attack the bait set. Selecting the bait action set for the robots can add robustness against the worst-case attacks. That is, no matter the attacker attacks the bait set or not, central-robust, composed of bait action assignment and greedy action assignment, gives suboptimality guarantees against the worst-cast attacks [18], [19].

Algorithm 2: Distributed Clique Partition (DCP).

Input: Robots' positions $\mathcal{P}$; communication range r_c.
Output: Clique partition of graph G.
1: Given $\mathcal{P}$ and r_c, find communication graph G;
2: **for** each robot i **do**
3: Find robot i's neighbor set $\mathcal{N}_i$ within r_c; {1st round communication}
4: Share $\mathcal{N}_i^+ := \{i, \mathcal{N}_i\}$ with robot i's neighbors, and receives all $\mathcal{N}_j^+$ from its neighbors, $j \in \mathcal{N}_i$; {2nd round communication}
5: Intersects $\mathcal{N}_i^+$ with every $\mathcal{N}_j^+$, and set the largest intersection as robot i's clique, i.e., $\mathcal{C}^i = \operatorname{argmax}_{\mathcal{N}_i^+ \bigcap \mathcal{N}_j^+} |\mathcal{N}_i^+ \bigcap \mathcal{N}_j^+|$, $j \in \mathcal{N}_i$;
6: Share $\mathcal{C}^i$ with robot i's neighbors; {3 rd round communication}
7: **return** Generated cliques.

In this context, DRM's second step is as follows: assuming the clique partition step returns K cliques (DRM's line 1), now each clique in parallel with the others computes an attack-robust assignment for its robots using central-robust (DRM's lines 3–8). To this end, the cliques need to assess how many of the α attacks each will incur. If there is no prior on the attack generation mechanism, then we consider a worst-case scenario where each clique incurs all the α attacks. Otherwise, we consider there is a prior on the attack mechanism such that each clique k infers it will incur $\alpha_k \leq \alpha$ attacks. Without the loss of generality, in DRM's pseudocode in Algorithm 1 we present the former scenario, where $\alpha_k = \alpha$ across all cliques; notwithstanding, our theoretical results on DRM's performance (see Section IV) hold for any α_k such that $\sum_{k=1}^{K} \alpha_k \geq \alpha$. Overall, DRM's lines 3–8 are as follows (cf., example in Fig. 4).

1) DRM*'s lines 4 and 5 ($\alpha < |\mathcal{C}_k|$):* If α is less than the clique's size (DRM's line 4), then the clique's robots choose actions by executing central-robust on the clique assuming α attacks (DRM's line 5).

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:54:40 UTC from IEEE Xplore. Restrictions apply.

2) DRM*'s lines 6 and 7 ($\alpha \geq |\mathcal{C}_k|$)*: But if α is larger than the clique's size (DRM's line 6), then the clique's robots choose actions by executing `central-robust` on the clique assuming $|\mathcal{C}_k|$ attacks (DRM's line 5); i.e., assuming that all clique's robots will be attacked.

3) DRM *'s line 8*: All in all, now all robots have assigned actions, and $\mathcal{S}$ is the union of all assigned actions across all cliques (notably, the robots of each clique k know only $\mathcal{S}_k$, where $\mathcal{S}_k$ is per the notation in DRM).

Finally, DRM is valid for any number of attacks.

IV. Performance Analysis of DRM

We now quantify DRM's performance, by bounding its computational and approximation performance. To this end, we use the following notion of curvature for set functions.

A. Curvature

Definition 1 (Curvature [39]): Consider nondecreasing submodular $f : 2^{\mathcal{X}} \mapsto \mathbb{R}$ such that $f(x) \neq 0$, for any $x \in \mathcal{X} \setminus \{\emptyset\}$ (without the loss of generality). Also, denote by $\mathcal{I}$ the collection of admissible sets where f can be evaluated at. Then, f's *curvature* is defined as

$$\nu_f \triangleq 1 - \min_{\mathcal{S}\in\mathcal{I}} \min_{x\in\mathcal{S}} \frac{f(\mathcal{S}) - f(\mathcal{S}\setminus\{x\})}{f(x)}. \tag{2}$$

The curvature, ν_f, measures how far f is from being additive. Particularly, Definition 1 implies $0 \leq \nu_f \leq 1$, and if $\nu_f = 0$, then $f(\mathcal{S}) = \sum_{x\in\mathcal{S}} f(\{x\})$ for all $\mathcal{S} \in \mathcal{I}$ (f is additive). On the other hand, if $\nu_f = 1$, then there exist $\mathcal{S} \in \mathcal{I}$ and $x \in \mathcal{X}$ such that $f(\mathcal{S}) = f(\mathcal{S} \setminus \{x\})$ (x has no contribution in the presence of $\mathcal{S} \setminus \{x\}$).

For example, in active target tracking, f is the expected number of covered targets (as a function of the robot trajectories). Then, f has curvature 0 if each robot covers different targets from the rest of the robots. In contrast, it has curvature one if, e.g., two robots cover the exact same targets.

B. Running Time and Approximation Performance

We present DRM's running time and suboptimality bounds. To this end, we use the following notation:

1) $t^c_{\texttt{DCP}}$ and $t^s_{\texttt{DCP}}$ denote the communication and computation time of the robot that spends the highest time on three-round communications (see Algorithm 2, lines 3, 4, 6) and neighbor set intersection (see Algorithm 2, line 5) in DCP;
2) $\mathcal{M}$ is a clique of G, which spends the highest time executing `central-robust`;
3) $t^c_{\texttt{CRO}}$ denotes the communication time of the robot that spends the highest time exchanging information collected (e.g., the subsets of targets covered) in $\mathcal{M}$;
4) t^f denotes the time of one evaluation of the objective function f (e.g., computing the number of targets covered by robots' actions);.
5) $\mathcal{X}_{\mathcal{M}}$ is the set of possible actions of all robots in $\mathcal{M}$; that is, $\mathcal{X}_{\mathcal{M}} \triangleq \cup_{i\in\mathcal{M}}\mathcal{X}_i$;
6) $f^\star$ is the optimal value of Problem 1;
7) $\mathcal{A}^\star(\mathcal{S})$ is a worst-case removal from $\mathcal{S}$ (a removal from $\mathcal{S}$ corresponds to a set of sensor attacks); that is, $\mathcal{A}^\star(\mathcal{S}) \in \arg\min_{\mathcal{A}\subseteq\mathcal{S},|\mathcal{A}|\leq\alpha} f(\mathcal{S}\setminus\mathcal{A})$.

Theorem 1 (Computational Performance): DRM runs in $O(1)(t^c_{\texttt{DCP}} + t^s_{\texttt{DCP}}) + O(1)t^c_{\texttt{CRO}} + O(|\mathcal{X}_{\mathcal{M}}|^2)t^f$ time. In addition, by DRM, each robot $i \in \mathcal{R}$ has four-round communications, including three rounds in DCP and one round in per clique `central-robust`, and exchanges $3|\mathcal{N}_i| + |\mathcal{C}_k| - 1$ messages with $i \in \mathcal{C}_k$. Moreover, DRM performs $O(|\mathcal{N}_i|)$ operations for set intersections of each robot i in DCP and $O(|\mathcal{X}_{\mathcal{C}_k}|^2)$ evaluations of objective function f in `central-robust` for each clique $\mathcal{C}_k, k \in \{1,\ldots,K\}$.

$O(1)(t^c_{\texttt{DCP}} + t^s_{\texttt{DCP}})$ corresponds to DRM's clique partition step (DRM's line 1), and $O(1)t^c_{\texttt{CRO}} + O(|\mathcal{X}_{\mathcal{M}}|^2)t^f$ corresponds to DRM's attack-robust step (DRM's lines 2–8). Particularly, DRM's communication time includes the time of three-round communications in the clique partition step and the time of one-round communication of information collected in the attack-robust step; that is $O(1)t^c_{\texttt{DCP}} + O(1)t^c_{\texttt{CRO}}$. DRM's computation time contains the time of neighbor set intersection in the clique partition step and the time of objective function evaluations in the attack-robust step; that is $t^s_{\texttt{DCP}} + O(|\mathcal{X}_{\mathcal{M}}|^2)t^f$. Notably, as the number of robots and/or the number of actions increase, DRM's running time will be dominated by the time for function evaluations, i.e., $O(|\mathcal{X}_{\mathcal{M}}|^2)t^f$.

In contrast, to evaluate the objective functions, `central-robust` [21, Algorithm 1] runs in $O(|\mathcal{X}|^2)t^f$ time. Thus, when $\mathcal{X}_{\mathcal{M}} \subset \mathcal{X}$ (which happens when G is partitioned into at least two cliques), DRM offers a computational speed-up. The reasons are two: *parallelization of action assignment*, and *smaller clique size*. Particularly, DRM splits the action assignment among multiple cliques, instead of performing the assignment in a centralized way, where all robots form *one large* clique (the $\mathcal{R}$). That way, DRM enables each clique to work in *parallel*, reducing the overall running time to that of clique $\mathcal{M}$ (Theorem 1). Besides parallelization, the smaller clique size also contributes to the computational reduction. To illustrate this, assume G is partitioned to K cliques of *equal* size, and all robots have the same number of actions ($|\mathcal{X}_i| = |\mathcal{X}_j|$ for all $i, j \in \mathcal{R}$). Then, $O(|\mathcal{X}_{\mathcal{M}}|^2) = O(|\mathcal{X}|^2)/K^2$, that is, DRM's function evaluation time is reduced by the factor K^2.

Theorem 2 (Approximation performance of DRM*):* DRM returns a feasible action-set $\mathcal{S}$ such that

$$\frac{f(\mathcal{S}\setminus\mathcal{A}^\star(\mathcal{S}))}{f^\star} \geq \frac{1-\nu_f}{2}. \tag{3}$$

The proof of the theorem follows from [19, Th. 1].

From (3), we conclude that even though DRM is a distributed, faster algorithm than its centralized counterpart, it still achieves a near-to-centralized performance. Generally, Theorem 2 implies DRM guarantees a close-to-optimal value for any submodular f with curvature $\nu_f < 1$.

Remark 2: A myopic algorithm that selects actions for each robot independently of all other robots (in contrast to DRM, whose subroutine `central-robust` accounts for the

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:54:40 UTC from IEEE Xplore. Restrictions apply.

other robots' actions during the greedy action assignment), guarantees the approximation bound $1-\nu_f$ (see Algorithm 4 in Appendix C). However, being exclusively myopic, Algorithm 4 has worse practical performance than DRM.

In Appendix C, we also show Algorithm 4 is equivalent to central-robust (cf., Section III-B) when applied to $\mathcal{R}$, under the assumption that *all* robots in $\mathcal{R}$ are attacked. This further reveals the practical inefficiency of Algorithm 4.

Remark 3: Notably, DCP always finds a set of cliques. That is, it always terminates. However, in some degenerate cases, it may end up with a set of singletons, i.e., cliques that have one robot only. This issue can be mitigated by allowing for longer computation times. For example, in DCP's line 5, there may exist multiple largest intersection sets $\mathcal{C}^i$ for each robot i. To break ties, we randomly select one, which may result in more singletons in the end. One natural way to find the largest intersection set that leads to the smallest number of singletons is to evaluate all the largest intersection sets $\mathcal{C}^i$. Clearly, the evaluation takes more time. Specifically, the number of largest intersection sets for each robot i is the number of its neighbors $|\mathcal{N}_i|$ in the worst case. Then, evaluating all the largest intersection sets takes $O(N)$ time, while random selection takes $O(1)$ time. However, with longer computation time, DCP is more likely to generate fewer and larger cliques.

The number of cliques plays a tradeoff between the running time and practical performance of DRM. Specifically, for a given number of robots, DRM runs faster with more and smaller cliques since all cliques operate in parallel. While, at the same time, DRM's practical performance becomes poorer. This is because, with more and smaller cliques, more attacks will be inferred by DRM since each clique $\mathcal{C}_k$ infers it will incur $\alpha_k=\min\{\alpha,|\mathcal{C}_k|\}$ attacks. Therefore, DRM is more conservative and closer to the myopic algorithm introduced in Remark 2. On the contrary, with fewer and larger cliques, DRM gives a better practical performance but runs slower.

V. Improved DRM

In DRM, each clique $\mathcal{C}_k$ assumes that the number of attacks α_k against the clique is either equal to the total number of attacks α, or equal to its size $|\mathcal{C}_k|$ (when $\alpha\geq|\mathcal{C}_k|$). Even though this strategy guarantees a close-to-optimal approximation performance (cf., Section IV), it is conservative, since the total number of attacks that all cliques infer ($\sum_k\alpha_k$) can be much larger than the real number of attacks α for the team. In this section, we design a strategy to amend this conservativeness. Particularly, we present an IDRM algorithm, and analyze its performance in terms of approximation performance and running time.

A. Improved Inference of Each Clique's Attack Number

The number of attacks in each clique α_k can be inferred by leveraging neighboring communications. First, note that robots can communicate with all their neighbors within communication range even though these neighbors are in different cliques. For example, in Fig. 4, robot 2 can still communicate with robots 1 and 4 even though they are partitioned into different cliques. However, in DRM, this available communication is ignored.

Algorithm 3: Algorithm to Approximate the Number of Attacks α_k Against a Clique k, Given a Known Number of Attacks α Against all Robots in $\mathcal{R}$.

Input: Robots' available actions $\mathcal{X}_i$, $i\in\mathcal{R}$; monotone and submodular function f; attack number α.
Output: Number of attacks $\hat{\alpha}_k$ for clique $\mathcal{C}_k$, $k\in\{1,\ldots,K\}$.

1: **if** $|\mathcal{C}_k|>\alpha$ **then**
2: $\quad\hat{\alpha}_k=\alpha$;
3: **else**
4: $\quad\hat{\alpha}_k=|\mathcal{C}_k|$;
5: Each clique selects the top $\hat{\alpha}_k$ robots as $\mathcal{C}_k^{\hat{\alpha}_k}$;
6: **for** each robot $i\in\mathcal{C}_k^{\hat{\alpha}_k}$ **do**
7: $\quad$**if** robot i is not one of the top α robots in its three-hop neighbors **then**
8: $\quad\quad\hat{\alpha}_k=\hat{\alpha}_k-1$;
9: **return** $\hat{\alpha}_k$.

Second, besides the three-hop communications required for the execution of distributed clique partition (DCP; cf., Algorithm 2), the robots can also share their actions' function values (e.g., the number of targets covered) with their three-hop neighbors. Evidently, while DRM assumes robots to share actions' function values with their one-hop neighbors within the same clique, sharing the actions' function values among three-hop neighbors can give a better inference of α_k, and, consequently, better performance.

Recall that DRM sets α_k equal to α (or to $|\mathcal{C}_k|$, when $\alpha\geq|\mathcal{C}_k|$) for each clique $\mathcal{C}_k$. Therefore, DRM selects a *bait set of* α_k robots in each clique (cf., Section III-B). Evidently, some of these "bait" robots may not be among the α *"bait" robots* that central-robust would have selected if it would have been applied directly to $\mathcal{R}$ (assuming centralized communication among all robots).[7] This can be checked by communicating actions' function values among three-hop neighbors. Particularly, if some robot is not one of the top α robots among its three-hop neighbors, it is impossible that it is in the top α robots among the entire team. Thus, this robot can be marked as "unselected" and α_k can be reduced. Based on this rule, we describe our α_k inferring strategy in detail in Algorithm 3.

We use Fig. 4 to illustrate how Algorithm 3 works with an example. When $\alpha=2$, clique $\mathcal{C}_2:=\{2,8\}$ first infers $\hat{\alpha}_1=2$, and robots 2 and 8 are selected. Then, robot 2 communicates with its one-hop, two-hop, and three-hop neighbors ($\{1,4,8\}$, $\{3,5,6,7,11\}$, $\{9,10,12,13\}$). If robot 2 is not one of the top 2 robots among its three-hop neighbors, robot 2 will be marked as "unselected" and $\hat{\alpha}_1$ will be reduced by 1 ($\hat{\alpha}_1=1$). Similarly, if robot 8 is not one of the top 2 robots among its three-hop neighbors, $\hat{\alpha}_1$ will be further reduced by 1 ($\hat{\alpha}_1=0$). This way, instead of picking out nine "bait" robots from five cliques [see Fig. 4(c)], fewer robots will be selected. All in all, by using

[7]We refer to any robot in the selected bait set of a clique k as a *top* α_k *robot* in the clique; similarly, when we consider the set of all robots $\mathcal{R}$, the *top* α *robots* are the robots in the bait set selected by a central-robust applied to $\mathcal{R}$ (when the number of attacks against $\mathcal{R}$ is α).

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:54:40 UTC from IEEE Xplore. Restrictions apply.

Algorithm 3, we can reduce DRM's conservativeness in inferring the number of attacks in each clique.

B. Performance Analysis of IDRM

The robots of each clique k, using the number of attacks $\hat{\alpha}_k$ generated by Algorithm 3, choose actions $\mathcal{S}_k$ by executing central-robust [21], that is,

$$\mathcal{S}_k = \texttt{central-robust}\left(\bigcup_{i\in\mathcal{C}_k} \mathcal{X}_i, f, \hat{\alpha}_k\right).$$

Approximation Performance of IDRM*:* IDRM has the same approximation bound as that of DRM [cf., (3)]. Notwithstanding, as we demonstrate in our numerical evaluations (cf., Section VI-B), IDRM performs better than DRM in practice, since IDRM utilizes more information (actions' function values shared among all three-hop neighbors). When the communication graph G only has nonoverlapping cliques (i.e., the robots of each clique can communicate only with their neighbors within the clique), IDRM and DRM will exhibit the same performance.

Computational Performance of IDRM*:* Runs in more time than DRM, since each robot needs to verify if it belongs to the top α robots among its three-hop neighbors instead of its one-hop neighbors within its clique as in DRM. But this verification only takes linear time. Thus, the running time of IDRM is similarly dominated by the central-robust operating in all cliques as in DRM. Also, each robot only needs to share with its three-hop neighbors the function value of its best action instead of that of each action. Thus, IDRM keeps the computational advantage of DRM.

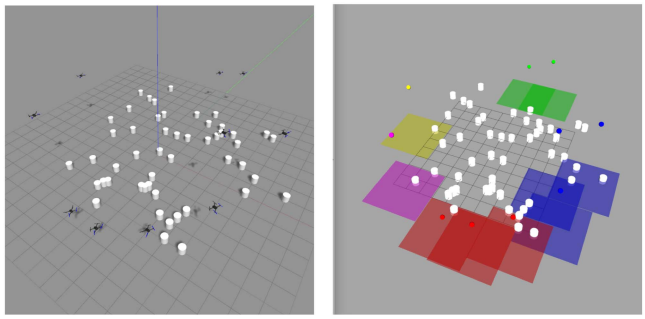
VI. Numerical Evaluations

We first present DRM's Gazebo and MATLAB evaluations in scenarios of *active target tracking with swarms of robots*. Then, we illustrate the advantages of IDRM by comparing it to DRM. The implementations' code is available online.[8] We run the code on a ThinkPad laptop with Intel Core i7 CPU @ 2.6 GHz $\times$ 8 and 62.8 GB Memory by using MATLAB 2018b and ROS Kinetic installed on Ubuntu 16.04. Due to the limited computer resources, we approximate the running time of these distributed algorithms by the total running time divided by the number of cliques, since all the cliques perform in parallel. Notably, the running time of the algorithms contains both the time of communication and the time of computation.

A. Robust Multirobot Target Tracking

Compared Algorithms: We compare DRM with two algorithms. First, the centralized counterpart of DRM in [21], named central-robust (its near-optimal performance has been extensively demonstrated in [21]). The second algorithm is the centralized greedy algorithm in [22], named central-greedy. The difference between the two algorithms is that the former is attack-robust, whereas the latter is attack-agnostic.

[8][Online]. Available: https://github.com/raaslab/distributed_resilient_target_tracking.git



(a) (b)

Fig. 5. Gazebo simulation setup: Ten aerial robots and 50 ground mobile targets. (a) Gazebo environment. (b) Rviz environment. Each robot is color-coded, along with its coverage region. All robots in the same clique have the same color. The targets are depicted as white cylindrical markers. (a) Gazebo environment. (b) Rviz environment.

For this reason, in [21], we demonstrated, unsurprisingly, that central-greedy has inferior performance to central-robust in the presence of attacks. However, we still include central-greedy in the comparison, to highlight the differences among the algorithms both in running time and performance.

1) Gazebo Evaluation Over Multiple Steps With Mobile Targets: We use Gazebo simulations to evaluate DRM's performance across multiple rounds (time-steps). That way, we take into account the kinematics and dynamics of the robots, as well as, the fact that the actual trajectories of the targets, along with the sensing noise, may force the robots to track fewer targets than expected. The motion model for the moving targets is known to the robots but it is corrupted with the Gaussian noise. Therefore, the robots use a Kalman filter to estimate the positions of the targets at each step. Due to the running efficacy of Gazebo (which is independent of DRM), we focus on small-scale scenarios of ten robots. In the MATLAB simulation, we focus instead on larger-scale scenarios of up to 100 robots.

Simulation Setup: We consider ten aerial robots that are tasked to track 50 ground mobile targets [see Fig. 5(a)]. We set the number of attacks α equal to 4, and the robots' communication range to be $r_c = 5$ meters. We also visualize the robots, their field-of-view, their cliques, and the targets using the Rviz environment [see Fig. 5(b)]. Each robot i has five candidates trajectories, $\mathcal{X}_i = \{$forward, backward, left, right, stay$\}$, and flies on a different fixed plane (to avoid collision with other robots). Each robot has a square field-of-view $l_o \times l_o$. Once a robot picks a nonstay trajectory, it flies a distance l_f along that trajectory. If the robot selects the stay trajectory, it stays still (i.e., $l_f = 0$). Thus, each trajectory has a rectangular tracking region with length $l_t = l_f + l_o$ and width l_o. We set the tracking length $l_t = 6$, and tracking width $l_o = 3$ for all robots. We assume robots obtain noisy position measurements of the targets, and then use a Kalman filter to estimate the target's position. We consider f to be the expected number of targets covered, given all robots chosen trajectories.

For each of the compared algorithms, at each round, each robot picks one of its five trajectories. Then, the robot flies a

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:54:40 UTC from IEEE Xplore. Restrictions apply.

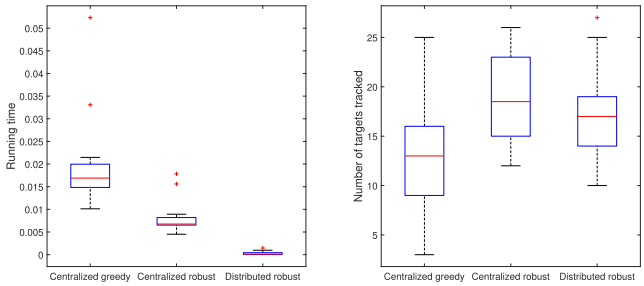


Fig. 6. Gazebo evaluation (averaged across 50 rounds): The tracking performance is captured by the number of covered targets per round. On each box in (a) and (b), the central red mark denotes the median, and the bottom and top edges of the box denote the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme values not considered outliers, and the outliers are plotted individually using the red '+' symbol. If a box does not have outliers [as in (b)], the bottom and top ends of whiskers correspond to the minimum and maximum values. (a) Running time. (b) Number of targets tracked.

$l_f = 3$ meters along the selected `nonstay` trajectory and stays still with the `stay` trajectory.

When an attack happens, we assume the attacked robot's tracking sensor (e.g., camera) to be turned OFF; nevertheless, we assume it can be active again at the next round. The attack is a worst-case attack, per Problem 1's framework. Particularly, we compute the attack via a brute-force algorithm, which is viable for small-scale scenarios (as this one).

We repeat for 50 rounds. A video is available online.[9]

Results: The results are reported in Fig. 6. We observe the following.

1) Superior Running Time: `DRM` runs considerably faster than both `central-robust` and `central-greedy`: 1.8 orders faster than the former, and 2.2 orders faster than the latter, with average running time 0.1 ms [see Fig. 6(a)].

2) Near-to-Centralized Tracking Performance: Despite that `DRM` runs considerably faster, it maintains near-to-centralized performance: `DRM` covers on average 18.8 targets per round, while `central-robust` covers 17.1 [see Fig. 6(b)]. To statistically evaluate `DRM` and `central-robust`, we run a *t*-test with the default 5% significance level on the targets covered by them. The *t*-test gives the test decision $H = 0$ and p-value $p = 0.0805$, which indicates that *t*-test does not reject the null hypothesis that the means of the number of targets covered by `DRM` and `central-robust` are equal to each other at the 5% significance level. This again demonstrates that `DRM` performs very close to `central-robust`.

As expected, the attack-agnostic `central-greedy` performs worse than all algorithms, even being centralized. Similarly, we run a *t*-test on the number of targets covered by `central-greedy` and `DRM` and obtain $H = 1$ and $p = 0.00069$, which indicates that the means of the number of targets covered by `central-greedy` and `DRM` are statistically different at the 5% significance level.

2) MATLAB Evaluation Over One Step With Static Targets: We use MATLAB simulations to evaluate `DRM`'s performance in large-scale scenarios. Specifically, we evaluate `DRM`'s running time and performance for various numbers of robots (from 10 to 100) and communication ranges (resulting from as few as 5 cliques to as many as 30 cliques). We compare all algorithms over a single execution round.

2) Simulation Setup: We consider N mobile robots, and 100 targets. We vary N from 10 to 100. For each N, we set the number of attacks equal to $\lfloor N/4 \rfloor$, $\lfloor N/2 \rfloor$, and $\lfloor 3\,N/4 \rfloor$.

Similarly to the Gazebo simulations, each robot moves on a fixed plane, and has five possible trajectories: forward, backward, left, right, and stay. We set $l_t = 10$ and $l_o = 3$ for all robots. We randomly generate the positions of the robots and targets in a 2-D space of size $[0, 200] \times [0, 200]$. Particularly, we generate 30 Monte Carlo runs (for each N). We assume that the robots have available estimates of targets' positions. For each Monte Carlo run, all compared algorithms are executed with the same initialization (same positions of robots and targets). `DRM` is tested across three communication ranges: $r_c = 30, 60, 90$. For a visualization of r_c's effect on the formed cliques, see Fig. 7, where we present two of the generated scenarios. All algorithms are executed for one round in each Monte Carlo run.

Notably, since we consider large-scale scenarios (up to $N = 100$ robots, and up to 75 attacks, when $N = 100$, and $\alpha = \lfloor 3\,N/4 \rfloor$), computing the worst-case attack via a brute-force algorithm is now infeasible. Particularly, given a trajectory assignment $\mathcal{S}$ to all robots, the problem of computing a worst-case attack is equivalent to minimizing a nonincreasing submodular function, an NP-hard problem [40]. Hence, we consider the attacker to use a greedy heuristic to attack the robots, instead of executing the worst-case attacks. Proposed greedy approaches can be found in [40].[10]

Results: The results are reported in Fig. 8 and we make the similar qualitative conclusions as in the Gazebo evaluation.

1) Superior Running Time: `DRM` runs several orders faster than both `central-robust` and `central-greedy`: 1 to 2 orders on average, achieving running time from 0.5 to 15 ms [see Fig. 8(a) and (d)]. Particularly, in Fig. 8(a) and (b), the algorithms' running time is evaluated with respect to the communication range r_c (with the number of attacks fixed as $\alpha = \lfloor N/2 \rfloor$). We observe that `DRM`'s running time increases as r_c increases. This is because, with a larger r_c, `DCP` generates larger cliques and robots need to communicate more messages and takes more time on computation (Theorem 1), and thus both `DRM`'s communication time and computation time increase, which is shown in Fig. 8(e) and (f). Particularly, `DRM` spends more time on computation with a smaller r_c (e.g., $r_c = 30$) and spends more time on communication with a larger r_c (e.g., $r_c = 90$).

In Fig. 8(c) and (d), the algorithms' running time is evaluated in terms of the number of attacks α (with the communication range fixed as $r_c = 60$). We observe `central-robust` runs faster as α increases, which is due to how `central-robust` works, that causes `central-robust` to become faster as α tends to N [21]. This parallels the observation that `DRM`'s

[9][Online]. Available: https://youtu.be/JA8IOStLndE

[10]Alternative algorithms, along with approximate guarantees, to approximate the worst-case attacks can be found in [41]–[43].

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:54:40 UTC from IEEE Xplore. Restrictions apply.

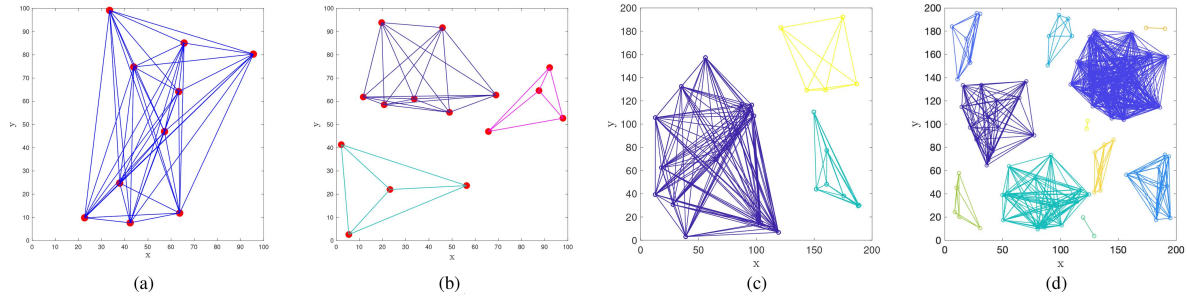


Fig. 7. MATLAB evaluation: Examples of clique formulations (see Algorithm 2) across various numbers of robots and communication ranges r_c. (a) 10 robots with $r_c = 120$. (b) 15 robots with $r_c = 60$. (c) 30 robots with $r_c = 90$. (d) 100 robots with $r_c = 50$.

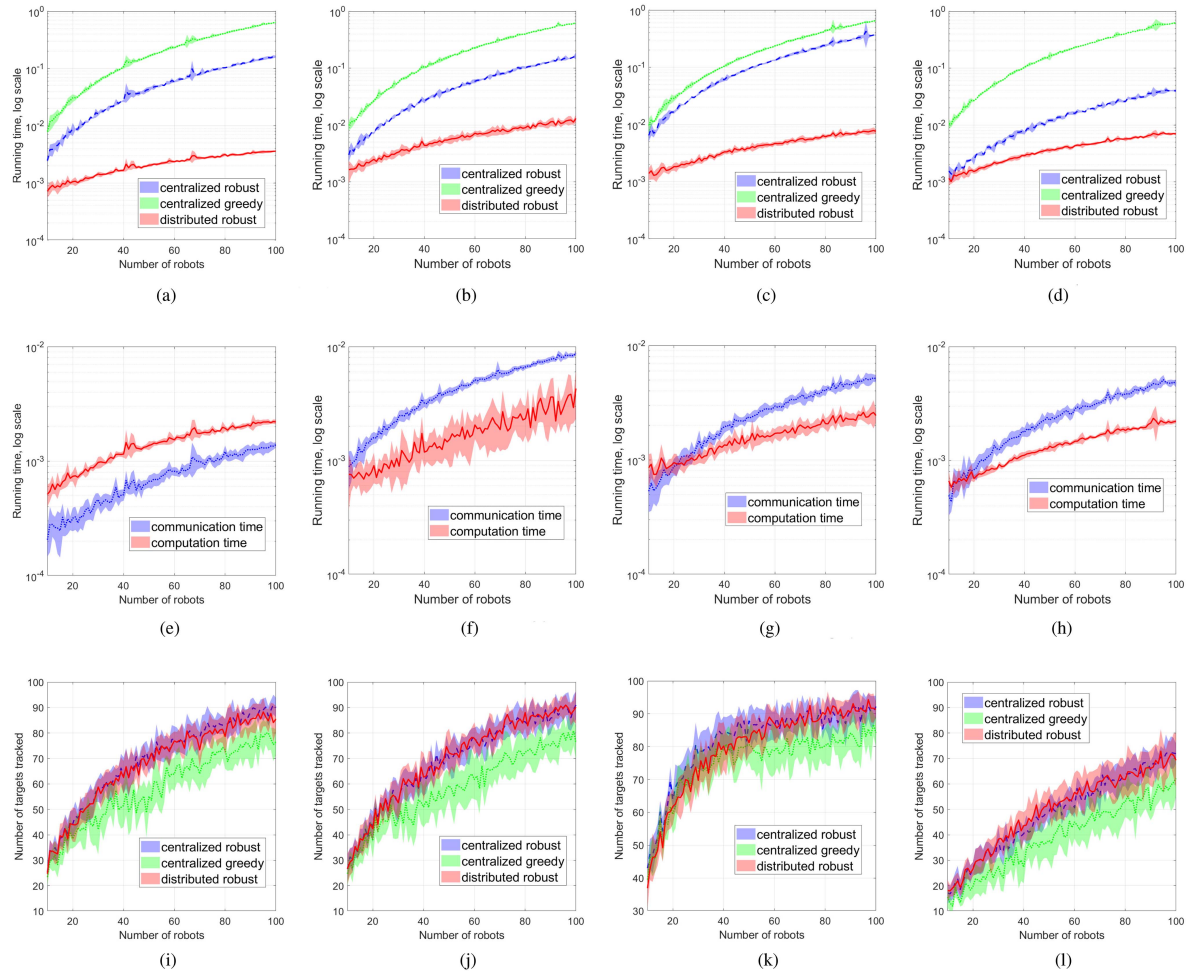


Fig. 8. MATLAB evaluations (averaged across 30 Monte Carlo runs): (a)–(d) depict running time results of three algorithms, for various α and r_c values; (e)–(h) depict the corresponding communication time and the computation time of `DRM` (`distributed-robust`); and (i)–(l) depict corresponding tracking performance results. (a) $r_c = 30$, $\alpha = \lfloor N/2 \rfloor$. (b) $r_c = 90$, $\alpha = \lfloor N/2 \rfloor$. (c) $\alpha = \lfloor N/4 \rfloor$, $r_c = 60$. (d) $\alpha = \lfloor 3\,N/4 \rfloor$, $r_c = 60$. (e) $r_c = 30$, $\alpha = \lfloor N/2 \rfloor$. (f) $r_c = 90$, $\alpha = \lfloor N/2 \rfloor$. (g) $\alpha = \lfloor N/4 \rfloor$, $r_c = 60$. (h) $\alpha = \lfloor 3\,N/4 \rfloor$, $r_c = 60$. (i) $r_c = 30$, $\alpha = \lfloor N/2 \rfloor$. (j) $r_c = 90$, $\alpha = \lfloor N/2 \rfloor$. (k) $\alpha = \lfloor N/4 \rfloor$, $r_c = 60$. (l) $\alpha = \lfloor 3\,N/4 \rfloor$, $r_c = 60$.

computation time decreases as α increases [see Fig. 8(g) and (h)], since `DRM`'s computation time mainly comes from the function evaluations in per clique `central-robust`.

2) Near-to-Centralized Tracking Performance: Although `DRM` runs considerably faster, it retains a tracking performance close to the centralized one [see Fig. 8(i) and (l)]. Unsurprisingly, the attack-agnostic greedy performs worse than all algorithms. Particularly, as shown in Fig. 8(i) and (j) when the communication range r_c increases from 30 to 90, `central-robust`'s tracking performance improves. This is because, with a larger r_c, fewer and larger cliques are generated by DCP, and `DRM` behaves closer to `central-robust`. Fig. 8(k) and (l) show that all

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:54:40 UTC from IEEE Xplore. Restrictions apply.

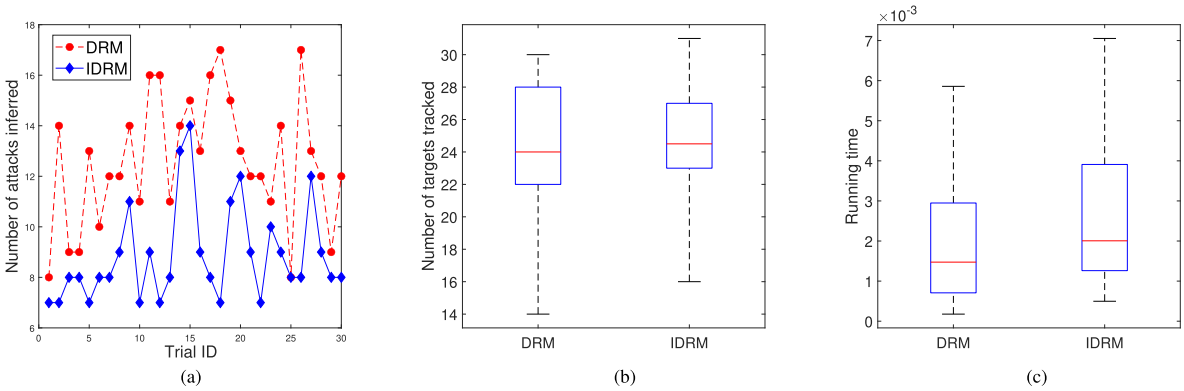


Fig. 9. MATLAB evaluations with $N = 20$, $\alpha = 6$, and $r_c = 120$: Comparison of number of attacks inferred, number of targets covered, and running time for DRM and IDRM in small-scale case. The number of targets tracked of these two algorithms is calculated after applying six worst-case attacks. In (b) and (c), the representations of each box's components follow the corresponding explanations in the caption of Fig. 6. (a) Number of attacks inferred, $\alpha = 6$. (b) Number of targets tracked. (c) Running time.

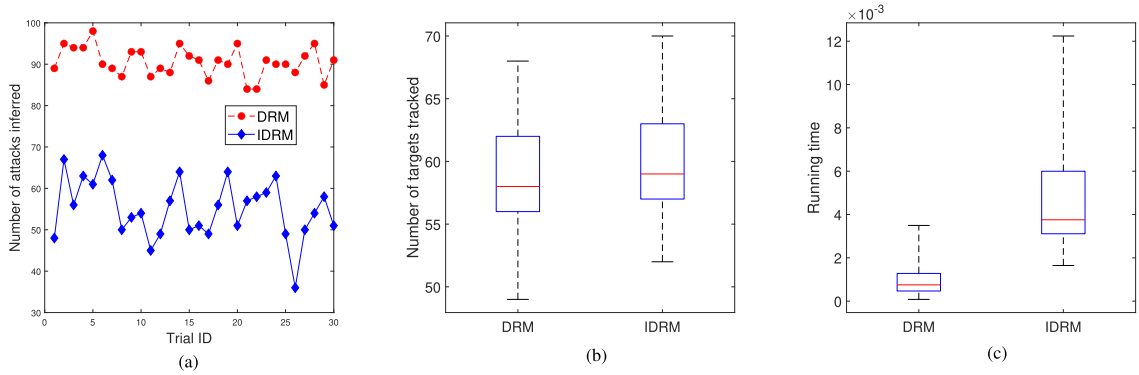


Fig. 10. MATLAB evaluations with $N = 100$, $\alpha = 30$, and $r_c = 70$: Comparison of number of attacks inferred, number of targets covered, and running time for DRM and IDRM in large-scale case. The number of targets tracked of these two algorithms is calculated after applying 30 greedy attacks. In (b) and (c), the representations of each box's components follow the corresponding explanations in the caption of Fig. 6. (a) Number of attacks inferred, $\alpha = 30$. (b) Number of targets tracked. (c) Running time.

algorithms' tracking performance drops when the number of attacks α increases.

To summarize, in all simulations mentioned above, DRM offered significant computational speed-ups, and, yet, still achieved a tracking performance that matched the performance of the centralized, near-optimal algorithm in [21].

B. Improved Multirobot Target Tracking

1) Compared Algorithms: We compare IDRM with DRM. The performance of the algorithms is evaluated through MATLAB simulations on *active target tracking* scenarios. We compare the algorithms in terms of the total number of attacks inferred, running time, and number of targets covered after α attacks. α is known to both IDRM with DRM. The algorithms are compared over a single execution for 30 trials.

Simulation Setup: We consider N mobile robots, and 100 targets. We set N as 20 and 100 for small- and large-scale evaluations, respectively. For evaluating the algorithms in the large-scale case (e.g., $N = 100$), we approximate the worst-case attack by a greedy attack since computing the worst-case attack requires exponential time. The total number of attacks α is set as 6 when $N = 20$, and as 30 when $N = 100$. The communication range r_c is set as $r_c = 120$ for $N = 20$, and $r_c = 70$ for $N = 100$. The remaining settings are the same as in the MATLAB simulation setup of Section VI-A.

Results: The results are reported in Figs. 9 and 10.

1) Conservativeness Relaxing Performance: Figs. 9(a) and 10(a) show that IDRM relaxes the conservativeness of inferring number of attacks α in both small-scale ($N = 20$, $\alpha = 6$) and large-scale ($N = 100$, $\alpha = 30$) cases. Notably, when the communication range is smaller [e.g., $r_c = 70$ in Fig. 10(a)], the inferred number of attacks by DRM is much larger than the real number of attacks ($\alpha = 30$). That is because, with a smaller communication range, the graph is likely to be partitioned into more and smaller cliques by Algorithm 2, which increases the conservativeness of inferring α in DRM. While IDRM gracefully relaxes this conservativeness through three-hop neighboring communications (see Algorithm 3). Particularly, in some trials of both small- and large-scale evaluations, the number of attacks inferred by IDRM is very close to the real number of attacks α.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:54:40 UTC from IEEE Xplore. Restrictions apply.

2) Superior Tracking Performance: Because of the conservativeness relaxing, IDRM tracks more targets than DRM [see Figs. 9(b) and 10(b)] since it reduces the unnecessary coverage overlaps induced by conservative estimate of α. To further evaluate the tracking performance IDRM and DRM, we run a *t*-test with the default 5%-significance level on the targets covered by them. The *t*-test gives the test decision $H = 1$ and $p = 0.025$ for the small-scale case [see Fig. 9(b)] and the test decision $H = 1$ and $p = 0.030$ for the large-scale case [see Fig. 10(b)]. This indicates that, in both cases, *t*-test rejects the null hypothesis that the means of the number of targets covered by IDRM and DRM are equal to each other at the 5% significance level. Therefore, the difference between the means of the number of targets covered by them is statistically significant.

3) Comparative Running Time: Figs. 9(c) and 10(c) show that both DRM and IDRM run very fast (e.g., averaged running time is less than 0.005 s). That is because, in IDRM, after robots share the number of targets covered by their best actions and infer a less conservative α_k (see Algorithm 3), all cliques run central-robust in parallel as well.

VII. Conclusion

We worked toward securing swarm-robotics applications against worst-case attacks resulting in robot withdrawals. Particularly, we proposed DRM, a distributed robust submodular optimization algorithm. DRM is general-purpose: it applies to any Problem 1's instance. We proved DRM runs faster than its centralized counterpart, without compromising approximation performance. We demonstrated both its running time and near-optimality in Gazebo and MATLAB simulations of active target tracking. However, in DRM, each clique assumes the number of attacks α_k to be the total number of attacks α, which can be too conservative if the robots are partitioned into many small-size cliques. To relax this conservativeness, we leveraged the three-hop neighboring communications to present an improved version of DRM, called IDRM. We showed that IDRM improves the target-tracking performance of DRM with comparative running time.

Note that, with DCP (see Algorithm 2), the clique partition solely depends on the positions of robots and the communication range. If robots have a very long communication range, they are likely to be all part of a single clique and then DRM would be the same as central-robust. One way to address this issue is to intelligently manage the number of cliques to control the tradeoff between complexity and optimality. Hence, one future research direction is to design such clique partition approaches that can be complementary to DCP. Another potential improvement on DCP is to relax its three-hop communication constraints. One way is to employ an anytime communication protocol that is tolerant to the failures of communication channels. Another direction is to enable asynchronous communication and achieve tolerance to communication delays.

A second future direction is to secure the team performance when the number of worst-case attacks is unknown. One heuristic approach to infer the number of attacks α_k distributively for cliques without knowing α is presented in the *arXiv* version [44] (cf., Algorithm 5 in Appendix D of [44]). However, this algorithm is a heuristic approach that aims to secure the *expected* worst-case performance and thus may not have guarantees against a specific number of worst-case attacks. Therefore, our future work is focused on designing algorithms that have approximation guarantees against the unknown number of attacks. A third future avenue is to investigate other attack or failure models, e.g., random failures [45], [46], and design corresponding distributed robust algorithms.

Appendix

A. Proof of Theorem 1

Proof: DRM's running time is equal to DCP's plus the time for all cliques to execute central-robust in parallel.

Particularly, the running time of DCP contains the time of three-round communications—finding neighbors (see Algorithm 2, line 3), exchanging neighbor sets (see Algorithm 2, line 4), and exchanging computed cliques (see Algorithm 2, line 6) and the time of neighbor set intersection (see Algorithm 2, line 5). Since robots perform in parallel by DCP, DCP's running time depends on the robot that spends the highest time on three-round communications and neighbor set intersection. Thus, DCP takes $O(1)(t^c_{\text{DCP}} + t^s_{\text{DCP}})$ time. Specifically, in each communication round, each robot i sends its message to and receives the messages from its neighbors, and thus the number of messages exchanged by each robot i is $|\mathcal{N}_i|$. Hence, over three-round communications, each robot exchanges $3|\mathcal{N}_i|$ messages. During the neighbor set intersection, each robot i intersects its neighbor set with those of its neighbors, and thus the number of set intersections (operations) for each robot i is $|\mathcal{N}_i|$.

Next, all cliques perform central-robust [21] in parallel. In each clique $\mathcal{C}_k$, each robot i first communicates and exchanges information collected (e.g., the subsets of targets covered by its candidate actions) with all the other robots. Hence, the number of messages exchanged by each robot i in $\mathcal{C}_k$ is $|\mathcal{C}_k| - 1$. Notably, since robots can communicate with each other in $\mathcal{C}_k$, the communication only takes one round.

After information exchange, the robots in $\mathcal{C}_k$ perform central-robust that executes sequentially two steps—the bait assignment and the greedy assignment. The bait assignment sorts out the best $\min(\alpha, |\mathcal{C}_k|)$ single actions from $|\mathcal{C}_k|$ robots. We denote the joint candidate action set in $\mathcal{C}_k$ as $\mathcal{X}_{\mathcal{C}_k} \triangleq \bigcup_{i\in\mathcal{C}_k} \mathcal{X}_i$. Then, the sorting needs $O(|\mathcal{X}_{\mathcal{C}_k}|\log(|\mathcal{X}_{\mathcal{C}_k}|)$ evaluations of objective function f and thus takes $O(|\mathcal{X}_{\mathcal{C}_k}|\log(|\mathcal{X}_{\mathcal{C}_k}|))t^f$ time. Then, the greedy assignment uses the standard greedy algorithm [22] to choose actions for the remaining robots, which needs $O(|\mathcal{X}_{\mathcal{C}_k}|^2)$ evaluations of f and thus takes $O(|\mathcal{X}_{\mathcal{C}_k}|^2)t^f$ time. Hence, the number of function evaluations (operations) is $O(|\mathcal{X}_{\mathcal{C}_k}|\log(|\mathcal{X}_{\mathcal{C}_k}|) + O(|\mathcal{X}_{\mathcal{C}_k}|^2) = O(|\mathcal{X}_{\mathcal{C}_k}|^2)$.

Since all cliques perform central-robust in parallel, the running time depends on the clique that spends the highest time, i.e., clique $\mathcal{M}$. $\mathcal{M}$'s running time contains the time of

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:54:40 UTC from IEEE Xplore. Restrictions apply.

exchanging information collected, i.e., $O(1)t^c_{\texttt{CRO}}$, and of evaluating function f, i.e., $O(|\mathcal{X}_{\mathcal{C}_k}|\log(|\mathcal{X}_{\mathcal{C}_k}|)t^f + O(|\mathcal{X}_{\mathcal{C}_k}|^2)t^f = O(|\mathcal{X}_{\mathcal{C}_k}|^2)t^f$. Thus, in total, all cliques performing `central-robust` in parallel takes $O(1)t^c_{\texttt{CRO}} + O(|\mathcal{X}_{\mathcal{C}_k}|^2)t^f$ time.

All in all, `DRM` takes $O(1)(t^c_{\texttt{DCP}} + t^s_{\texttt{DCP}})$ + $O(1)t^c_{\texttt{CRO}}$ + $O(|\mathcal{X}_{\mathcal{M}}|^2)t^f$ time. In addition, each robot has four-round communications, including three rounds in `DCP` and one round in per clique `central-robust`, and exchanges $3|\mathcal{N}_i| + |\mathcal{C}_k| - 1$ messages with $i \in \mathcal{C}_k$. Moreover, `DRM` performs $O(|\mathcal{N}_i|)$ operations for set intersections of each robot i in `DCP` and $O(|\mathcal{X}_{\mathcal{C}_k}|^2)$ evaluations of objective function f in `central-robust` for each clique $\mathcal{C}_k$. ■

B. Proof of Theorem 2

Proof: We prove Theorem 2, i.e., `DRM`'s approximation bound, by following the steps of [19, Proof of Th. 1].

We introduce the notation: $\mathcal{S}^\star$ denotes an optimal solution to Problem 1. Given an action assignment $\mathcal{S}$ to all robots in $\mathcal{R}$, and a subset of robots $\mathcal{R}'$, we denote by $\mathcal{S}(\mathcal{R}')$ the actions of the robots in $\mathcal{R}'$ (i.e., the restriction of $\mathcal{S}'$ to $\mathcal{R}'$). And vise versa: given an action assignment $\mathcal{S}'$ to a subset $\mathcal{R}'$ of robots, we let $\mathcal{R}(\mathcal{S})$ denote this subset (i.e., $\mathcal{R}(\mathcal{S}') = \mathcal{R}'$). In addition, we let $\mathcal{S}_k \triangleq \mathcal{S}(\mathcal{C}_k)$; that is, $\mathcal{S}_k$ is the restriction of $\mathcal{S}$ to the clique $\mathcal{C}_k$ selected by `DRM`'s line 1 ($k \in \{1, \ldots, K\}$); evidently, $\mathcal{S} = \bigcup_{k=1}^K \mathcal{S}_k$. Moreover, we let $\mathcal{S}^b_k$ correspond to bait actions chosen by `central-robust` in $\mathcal{C}_k$, and $\mathcal{S}^g_k$ denote the greedy actions for the remaining robots in $\mathcal{C}_k$; that is, $\mathcal{S}_k = \mathcal{S}^b_k \cup \mathcal{S}^g_k$. If $\alpha \geq |\mathcal{C}_k|$, then $\mathcal{S}^g_k = \emptyset$. Henceforth, we let $\mathcal{S}$ be the action assignment given by `DRM` to all robots in $\mathcal{R}$. Also, we let $\mathcal{W}$ be remaining robots after the attack $\mathcal{A}^\star(\mathcal{S})$; i.e., $\mathcal{W} \triangleq \mathcal{R} \setminus \mathcal{R}(\mathcal{A}^\star(\mathcal{S}))$. Further, we let $\mathcal{W}_k \triangleq \mathcal{W} \cap \mathcal{C}_k$ be remaining robots in $\mathcal{C}_k$, $\mathcal{W}^b_k \triangleq \mathcal{W}_k \cap \mathcal{R}(\mathcal{S}^b_k)$ be remaining robots with bait actions in $\mathcal{C}_k$, and $\mathcal{W}^g_k \triangleq \mathcal{W}_k \cap \mathcal{R}(\mathcal{S}^g_k)$ be remaining robots with greedy actions in $\mathcal{C}_k$. Then, we have $\mathcal{S}^b_k \setminus \mathcal{W}^b_k$ and $\mathcal{S}^g_k \setminus \mathcal{W}^g_k$ to denote the attacked robots with bait actions and with greedy actions in $\mathcal{C}_k$, respectively. Then, we have the number of attacks in $\mathcal{C}_k$ as $|\mathcal{S}^b_k \setminus \mathcal{W}^b_k| + |\mathcal{S}^g_k \setminus \mathcal{W}^g_k|$ and

$$|\mathcal{S}^b_k \setminus \mathcal{W}^b_k| + |\mathcal{S}^g_k \setminus \mathcal{W}^g_k| \leq \alpha \tag{4}$$

since the total number of attacks for the robot team is α. Also, we have

$$|\mathcal{S}^b_k \setminus \mathcal{W}^b_k| + |\mathcal{W}^b_k| = \alpha \tag{5}$$

since the number of robots with bait actions in $\mathcal{C}_k$ is α. With (4) and (5), we have

$$|\mathcal{W}^b_k| \geq |\mathcal{S}^g_k \setminus \mathcal{W}^g_k|. \tag{6}$$

With (6), we let $\mathcal{W}^{b'}_k$ denote the remaining robots in $\mathcal{W}^b_k$ after removing from it any subset of robots with cardinality $|\mathcal{R}(\mathcal{S}^g_k) \setminus \mathcal{W}^g_k|$. Then, $|\mathcal{W}^b_k \setminus \mathcal{W}^{b'}_k| = |\mathcal{R}(\mathcal{S}^g_k) \setminus \mathcal{W}^g_k|$ and robots in $\mathcal{W}^b_k \setminus \mathcal{W}^{b'}_k$ have bait actions and robots in $\mathcal{R}(\mathcal{S}^g_k) \setminus \mathcal{W}^g_k$ have greedy actions. Then, $\mathcal{W}^b_k \setminus \mathcal{W}^{b'}_k$ can be used to compensate for the attacked robots $\mathcal{R}(\mathcal{S}^g_k) \setminus \mathcal{W}^g_k$ in $\mathcal{C}_k$.

Now, the proof follows from the steps:

$$f(\mathcal{S} \setminus \mathcal{A}^\star(\mathcal{S})) \geq (1-\nu_f) \sum_{r \in \mathcal{W}} f(\mathcal{S}(r)) \tag{7}$$

$$= (1-\nu_f) \sum_{k=1}^K \sum_{r \in \mathcal{W}_k} f(\mathcal{S}(r)) \tag{8}$$

$$= (1-\nu_f) \sum_{k=1}^K \left[\sum_{r \in \mathcal{W}^b_k} f(\mathcal{S}(r)) + \sum_{r \in \mathcal{W}^g_k} f(\mathcal{S}(r)) \right] \tag{9}$$

$$= (1-\nu_f) \sum_{k=1}^K \left[\sum_{r \in \mathcal{W}^{b'}_k} f(\mathcal{S}(r)) + \sum_{r \in \mathcal{W}^b_k \setminus \mathcal{W}^{b'}_k} f(\mathcal{S}(r)) + \sum_{r \in \mathcal{W}^g_k} f(\mathcal{S}(r)) \right] \tag{10}$$

$$\geq (1-\nu_f) \sum_{k=1}^K \left[\sum_{r \in \mathcal{W}^{b'}_k} f(\mathcal{S}(r)) + \sum_{r \in \mathcal{R}(\mathcal{S}^g_k) \setminus \mathcal{W}^g_k} f(\mathcal{S}(r)) + \sum_{r \in \mathcal{W}^g_k} f(\mathcal{S}(r)) \right] \tag{11}$$

$$= (1-\nu_f) \sum_{k=1}^K \left[\sum_{r \in \mathcal{W}^{b'}_k} f(\mathcal{S}(r)) + \sum_{r \in \mathcal{R}(\mathcal{S}^g_k)} f(\mathcal{S}(r)) \right] \tag{12}$$

$$\geq (1-\nu_f) \sum_{k=1}^K \left[\sum_{r \in \mathcal{W}^{b'}_k} f(\mathcal{S}(r)) + f(\mathcal{S}^g_k) \right] \tag{13}$$

$$\geq (1-\nu_f) \sum_{k=1}^K \left[\sum_{r \in \mathcal{W}^{b'}_k} f(\mathcal{S}^\star(r)) + \frac{1}{2} f(\mathcal{S}^\star(\mathcal{R}(\mathcal{S}^g_k))) \right] \tag{14}$$

$$\geq \frac{1-\nu_f}{2} \sum_{k=1}^K f(\mathcal{S}^\star(\mathcal{W}_k)) \tag{15}$$

$$\geq \frac{1-\nu_f}{2} f(\mathcal{S}^\star(\mathcal{W})) \tag{16}$$

$$\geq \frac{1-\nu_f}{2} f(\mathcal{S}^\star \setminus \mathcal{A}^\star(\mathcal{S}^\star)). \tag{17}$$

In (7) follows from the definition of ν_f [19, Proof of Th. 1]. Equations (8) and (9) follow from the notation we introduced above. Equation (10) holds since $\mathcal{W}^b_k = \mathcal{W}^{b'}_k \cup (\mathcal{W}^b_k \setminus \mathcal{W}^{b'}_k)$. Equation (11) holds since 1) $|\mathcal{W}^b_k \setminus \mathcal{W}^{b'}_k| = |\mathcal{R}(\mathcal{S}^g_k) \setminus \mathcal{W}^g_k|$; 2) robots in $\mathcal{W}^b_k \setminus \mathcal{W}^{b'}_k$ have bait actions and robots in $\mathcal{R}(\mathcal{S}^g_k) \setminus \mathcal{W}^g_k$ have greedy actions, such that for any $r \in \mathcal{W}^b_k \setminus \mathcal{W}^{b'}_k$ and $r' \in \mathcal{R}(\mathcal{S}^g_k) \setminus \mathcal{W}^g_k$, we have $f(\mathcal{S}(r)) \geq f(\mathcal{S}(r'))$. Equation (12) holds from the notation. Equation (13) holds by the submodularity of f, which implies $f(\mathcal{A}) + f(\mathcal{B}) \geq f(\mathcal{A} \cup \mathcal{B})$ for any sets $\mathcal{A}, \mathcal{B}$ [15]. Equation (14) holds since 1) with respect to the left term in the sum, the robots in the sum correspond to robots whose actions are baits; and 2) with respect to the right term in the sum, the greedy algorithm that has assigned the actions $\mathcal{S}^g_k$ guarantees at least $1/2$ the optimal [22]. Equation (11) holds again due to the submodularity of f, as mentioned above. The

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:54:40 UTC from IEEE Xplore. Restrictions apply.

Algorithm 4: Myopic Algorithm for Problem 1.

Input: Robots' available actions $\mathcal{X}_i$, $i \in \mathcal{R}$; monotone and submodular f.
Output: Robots' actions $\mathcal{S}$.
1: $\mathcal{S} \leftarrow \emptyset$;
2: **for** $i \in \mathcal{R}$ **do**
3: $\quad s \leftarrow \text{argmax}_{s \in \mathcal{X}_i} f(s)$;
4: $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{s\}$;
5: **return** $\mathcal{S}$.

same for in (12). Equation (13) follows from [19, Proof of Th. 1] because of the worst-case removal. ■

C. Myopic Optimization Yields Tighter Approximation Performance, yet Worse Practical Performance

The myopic Algorithm 4, according to which each robot selects its best action independently, guarantees a tighter approximation bound than that of DRM.

Theorem 3 (Approximation performance of Algorithm 4): Algorithm 4 returns a feasible action-set $\mathcal{S}$ such that

$$\frac{f(\mathcal{S} \setminus \mathcal{A}^\star(\mathcal{S}))}{f^\star} \geq 1 - \nu_f. \tag{18}$$

Proof: We split $\mathcal{S}$ generated by Algorithm 4 into $\mathcal{S}_1$ and $\mathcal{S}_2$, with $\mathcal{S}_1$ denoting the action set selected by the top α robots, and $\mathcal{S}_2$ denoting the action set selected by the remaining $|\mathcal{R}| - \alpha$ robots. Denote $\mathcal{S}_2^\star$ as the action set selected by $\mathcal{R}(\mathcal{S}_2)$ to maximize $f(\mathcal{A})$, $\mathcal{A} \in \mathcal{I}, \mathcal{A} \subseteq \mathcal{X}(\mathcal{R}(\mathcal{S}_2))$.

$$f(\mathcal{S} \setminus \mathcal{A}^\star(\mathcal{S})) \geq (1 - \nu_f) \sum_{s \in \mathcal{S}_2} f(s) \tag{19}$$

$$\geq (1 - \nu_f) \sum_{s \in \mathcal{S}_2^\star} f(s^\star) \tag{20}$$

$$\geq (1 - \nu_f) f(\mathcal{S}_2^\star) \tag{21}$$

$$\geq (1 - \nu_f) f(\mathcal{S}^\star \setminus \mathcal{A}^\star(\mathcal{S}^\star)). \tag{22}$$

Equation (19) follows from the definition of ν_f (see [19, Proof of Th. 1]). Equation (20) holds since Algorithm 4 selects the best action for each robot (see Algorithm 4, line 3). Equation (21) holds due to the submodularity of f. Equation (22) follows from [19, Proof of Th. 1] because of the worst-case removal. ■

However, Algorithm 4 performs in practice worse than DRM because: 1) it chooses actions for each robot independently of the actions of the rest of the robots (instead, DRM takes into account other robots' actions to intentionally reduce the performance redundancy among these robots); and 2) it is equivalent to `central-robust` *but* under the assumption that the number of attacks is equal to the number of robots, which is, evidently, conservative; DRM instead is less conservative assuming at most α attacks per clique.

An evaluation of the practical performance of Algorithm 4 in comparison to DRM's is made in Fig. 11. The figures clearly show that in both small- and large-scale cases, DRM outperforms Algorithm 4.

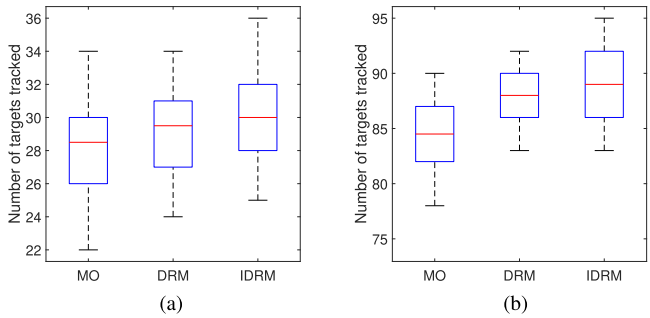


Fig. 11. MATLAB evaluations: comparison of number of targets covered by Algorithm 4 (called MO), DRM, IDRM in (a) small-scale and (b) large-scale cases. The simulation settings follow the MATLAB simulation setup of Section VI-A. In (a) small-scale case with $N = 20, \alpha = 4$, and $r_c = 120$, the number of target tracked by these three algorithms are calculated after applying four worst-case attacks. In (b) large-scale case with $N = 100, \alpha = 10$, and $r_c = 70$, the number of target tracked by these three algorithms are calculated after applying ten greedy attacks. In (a) and (b), the representations of each box's components follow the corresponding explanations in the caption of Fig. 6. (a) $N = 20$. (b) $N = 100$.

ACKNOWLEDGMENT

The authors would like to thank Micah Corah from the Carnegie Mellon University for pointing out that the myopic Algorithm 4, stated in Appendix C, achieves the performance bound $1 - \nu_f$ (see Theorem 3 in Appendix C). The observation led to Remark 2.

REFERENCES

[1] C. Nieto-Granda, J. G. Rogers III, and H. Christensen, "Multi-robot exploration strategies for tactical tasks in urban environments," *Unmanned Syst. Technol. XV*, vol. 8741, 2013, Art. no. 87410B.
[2] V. Kumar and N. Michael, "Opportunities and challenges with autonomous micro aerial vehicles," in *Proc. Robot. Res.*, 2017, pp. 41–58.
[3] M. Michini, M. A. Hsieh, E. Forgoston, and I. B. Schwartz, "Robotic tracking of coherent structures in flows," *IEEE Trans. Robot.*, vol. 30, no. 3, pp. 593–603, Jun. 2014.
[4] S. Karaman and E. Frazzoli, "High-speed flight in an Ergodic forest," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 2899–2906.
[5] C. Cadena *et al.*, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.
[6] T. Cieslewski, E. Kaufmann, and D. Scaramuzza, "Rapid exploration with multi-rotors: A frontier selection method for high speed flight," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 2135–2142.
[7] M. Santos, Y. Diaz-Mercado, and M. Egerstedt, "Coverage control for multirobot teams with heterogeneous sensing capabilities," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 919–925, Apr. 2018.
[8] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Decentralized active information acquisition: Theory and application to multi-robot slam," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 4775–4782.
[9] B. Schlotfeldt, D. Thakur, N. Atanasov, V. Kumar, and G. J. Pappas, "Anytime planning for decentralized multirobot active information gathering," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 1025–1032, Apr. 2018.
[10] R. Khodayi-mehr, Y. Kantaros, and M. M. Zavlanos, "Distributed state estimation using intermittently connected robot networks," *IEEE Trans. Robot.*, vol. 35, no. 3, pp. 709–724, Jun. 2019.
[11] M. Corah and N. Michael, "Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice," *Auton. Robots*, vol. 43, no. 2, pp. 485–501, 2019.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:54:40 UTC from IEEE Xplore. Restrictions apply.

[12] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-MCTS: Decentralized planning for multi-robot active perception," *Int. J. Robot. Res.*, vol. 38, no. 2/3, pp. 316–337, 2019.
[13] P. Dames, P. Tokekar, and V. Kumar, "Detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots," *Int. J. Robot. Res.*, vol. 36, no. 13/14, pp. 1540–1553, 2017.
[14] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *J. Artif. Intell. Res.*, vol. 34, pp. 707–755, 2009.
[15] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions-I," *Math. Program.*, vol. 14, no. 1, pp. 265–294, 1978.
[16] U. Feige, "A threshold of ln n for approximating set cover," *J. Appl. Comput. Mechan.*, vol. 45, no. 4, pp. 634–652, 1998.
[17] P. Tokekar, V. Isler, and A. Franchi, "Multi-target visual tracking with aerial robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 3067–3072.
[18] V. Tzoumas, K. Gatsis, A. Jadbabaie, and G. J. Pappas, "Resilient monotone submodular function maximization," in *Proc. IEEE Conf. Decis. Control*, 2017, pp. 1362–1367.
[19] V. Tzoumas, A. Jadbabaie, and G. J. Pappas, "Resilient non-submodular maximization over matroid constraints," 2018, *arXiv:1804.01013*.
[20] B. Schlotfeldt, V. Tzoumas, D. Thakur, and G. J. Pappas, "Resilient active information gathering with mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4309–4316.
[21] L. Zhou, V. Tzoumas, G. J. Pappas, and P. Tokekar, "Resilient active target tracking with multiple robots," *IEEE Robot. Autom. Lett.*, vol. 4, no. 1, pp. 129–136, Jan. 2019.
[22] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions-II," in *Proc. Polyhedral Combinatorics*, 1978, pp. 73–87.
[23] B. Gharesifard and S. L. Smith, "Distributed submodular maximization with limited information," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 4, pp. 1635–1645, Dec. 2018.
[24] D. Grimsman, M. S. Ali, J. P. Hespanha, and J. R. Marden, "The impact of information in greedy submodular maximization," *IEEE Trans. Control Netw. Syst.*, vol. 6, no. 4, pp. 1334–1343, Dec. 2019.
[25] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause, "Distributed submodular maximization: Identifying representative elements in massive data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2049–2057.
[26] K. Saulnier, D. Saldana, A. Prorok, G. J. Pappas, and V. Kumar, "Resilient flocking for mobile robot teams," *IEEE Robot. Automat. Lett.*, vol. 2, no. 2, pp. 1039–1046, Apr. 2017.
[27] D. Saldana, A. Prorok, S. Sundaram, M. F. Campos, and V. Kumar, "Resilient consensus for time-varying networks of dynamic agents," in *Proc. Amer. Control Conf.*, 2017, pp. 252–258.
[28] A. Mitra, J. A. Richards, S. Bagchi, and S. Sundaram, "Resilient distributed state estimation with mobile agents: Overcoming Byzantine adversaries, communication losses, and intermittent measurements," *Auton. Robots*, vol. 43, no. 3, pp. 743–768, 2019.
[29] B. Schlotfeldt, V. Tzoumas, and G. J. Pappas, "Resilient active information acquisition with teams of robots," *IEEE Trans. Robot.*, vol. 38, no. 1, pp. 244–261, Feb. 2022.
[30] J. B. Orlin, A. S. Schulz, and R. Udwani, "Robust monotone submodular function maximization," *Math. Program.*, vol. 172, no. 1/2, pp. 505–537, 2018.
[31] I. Bogunovic, S. Mitrović, J. Scarlett, and V. Cevher, "A distributed algorithm for partitioned robust submodular maximization," in *Proc. IEEE 7th Int. Workshop Comput. Adv. Multi-Sensor Adaptive Process.*, 2017, pp. 1–5.
[32] V. Tzoumas, A. Jadbabaie, and G. J. Pappas, "Robust and adaptive sequential submodular optimization," *IEEE Trans. Autom. Control*, 2020.
[33] L. Zhou and P. Tokekar, "An approximation algorithm for distributed resilient submodular maximization," in *Proc. IEEE Int. Symp. Multi-Robot Multi-Agent Syst.*, 2019, pp. 216–218.
[34] L. Zhou, V. Tzoumas, G. J. Pappas, and P. Tokekar, "Distributed attack-robust submodular maximization for multi-robot planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 2479–2485.
[35] R. B. Myerson, *Game Theory*. Cambridge, MA, USA: Harvard Univ. Press, 2013.
[36] B. Pattabiraman, M. M. A. Patwary, A. H. Gebremedhin, W.-K. Liao, and A. Choudhary, "Fast algorithms for the maximum clique problem on massive sparse graphs," in *Proc. Int. Workshop Algorithms Models Web-Graph*, 2013, pp. 156–169.
[37] D. Zuckerman, "Linear degree extractors and the inapproximability of max clique and chromatic number," in *Proc. ACM Symp. Theory Comput.*, 2006, pp. 681–690.
[38] M. M. Zanjireh and H. Larijani, "A survey on centralised and distributed clustering routing algorithms for WSNs," in *Proc. IEEE 81st Veh. Technol. Conf.*, 2015, pp. 1–6.
[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 Appl. Math.*, vol. 7, no. 3, pp. 251–274, 1984.
[40] R. Iyer, S. Jegelka, and J. Bilmes, "Fast semidifferential-based submodular function optimization," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 855–863.
[41] D. M. Topkis, "Minimizing a submodular function on a lattice," *Operations Res.*, vol. 26, no. 2, pp. 305–321, 1978.
[42] A. Schrijver, "A combinatorial algorithm minimizing submodular functions in strongly polynomial time," *J. Combinatorial Theory Ser. B*, vol. 80, no. 2, pp. 346–355, 2000.
[43] S. Jegelka, H. Lin, and J. A. Bilmes, "On fast approximate submodular minimization," *Adv. Neural Inf. Process. Syst.*, vol. 24, pp. 460–468, 2011.
[44] L. Zhou, V. Tzoumas, G. J. Pappas, and P. Tokekar, "Distributed attack-robust submodular maximization for multi-robot planning," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2020, pp. 2479–2485,.
[45] H. Park and S. Hutchinson, "Robust rendezvous for multi-robot system with random node failures: An optimization approach," *Auton. Robots*, vol. 42, pp. 1807–1818, 2018.
[46] L. Zhou and P. Tokekar, "An approximation algorithm for risk-averse submodular optimization," in *Proc. Int. Workshop Algorithmic Found. Robot.*, 2018, pp. 144–159.



Lifeng Zhou (Member, IEEE) received the bachelor's degree in automation from the Huazhong University of Science and Technology, Wuhan, China, in 2013, the master's degree in automation from Shanghai Jiao Tong University, Shanghai, China, in 2016, and the Ph.D. degree in electrical and computer engineering from Virginia Tech, Blacksburg, VA, USA, in 2020.

He is currently a Postdoctoral Researcher with the GRASP Lab, University of Pennsylvania, Philadelphia, PA, USA. His research interests include multirobot coordination, approximation algorithms, combinatorial optimization, model predictive control, graph neural networks, and resilient, risk-aware decision making.



Vasileios Tzoumas (Member, IEEE) received the Diploma in electrical and computer engineering from the National Technical University of Athens, Athens, Greece, in 2012, the Master of Arts degree in statistics from the Wharton School of Business, University of Pennsylvania, Philadelphia, PA, USA, in 2016, and the Master of Science degree in electrical engineering and the Ph.D. degree in electrical and systems engineering both from the University of Pennsylvania in 2016 and 2018, respectively.

He is currently an Assistant Professor with the Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI, USA. Previously, he was a Research Scientist with the Department of Aeronautics and Astronautics, and the Laboratory for Information and Decision Systems, both at the Massachusetts Institute of Technology (MIT). In 2017, he was a visiting Ph.D. student with the Institute for Data, Systems, and Society, MIT. He works on control, learning, and perception, as well as combinatorial and distributed optimization, with applications to robotics, cyber–physical systems, and self-reconfigurable aerospace systems. He aims for trustworthy collaborative autonomy. His research interests include foundational results on robust and adaptive combinatorial optimization, with applications to multirobot information gathering for resiliency against robot failures and adversarial removals.

Prof. Tzoumas was a recipient of the Best Paper Award in Robot Vision at the 2020 IEEE International Conference on Robotics and Automation, of a Honorable Mention at the 2020 IEEE Robotics and Automation Letter, and was a Best Student Paper Award finalist at the 2017 IEEE Conference in Decision and Control.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:54:40 UTC from IEEE Xplore. Restrictions apply.



George J. Pappas (Fellow, IEEE) received the Ph.D. degree in electrical engineering and computer sciences from the University of California, Berkeley, CA, USA, in 1998.

He is currently the Joseph Moore Professor and Chair of the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA. He also holds a secondary appointment with the Department of Computer and Information Sciences and the Department of Mechanical Engineering and Applied Mechanics. He is a member of the GRASP Lab and the PRECISE Center. He was previously the Deputy Dean for Research with the School of Engineering and Applied Science. His research interests include control theory and, in particular, hybrid systems, embedded systems, cyber-physical systems, and hierarchical and distributed control systems, with applications to unmanned aerial vehicles, distributed robotics, green buildings, and biomolecular networks.

Dr. Pappas was a recipient of various awards, such as the Antonio Ruberti Young Researcher Prize, the George S. Axelby Award, the Hugo Schuck Best Paper Award, the George H. Heilmeier Award, the National Science Foundation PECASE award, and numerous best student papers awards.



Pratap Tokekar (Member, IEEE) received the Bachelor of Technology degree in electronics and telecommunication from the College of Engineering Pune, India, in 2008, and the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, MN, USA, in 2014.

He is currently an Assistant Professor with the Department of Computer Science, University of Maryland, College Park, MD, USA. Previously, he was a Postdoctoral Researcher with The GRASP Lab, University of Pennsylvania. Between 2015 and 2019, he was an Assistant Professor with the Department of Electrical and Computer Engineering, Virginia Tech.

Prof. Tokekar was a recipient of the NSF CISE Research Initiation Initiative award. He is an Associate Editor for the IEEE Robotics and Automation Letters and IEEE TRANSACTIONS OF AUTOMATION SCIENCE AND ENGINEERING.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:54:40 UTC from IEEE Xplore. Restrictions apply.