Distributed Attack-Robust Submodular Maximization for Multi-Robot Planning

Lifeng Zhou,[1] Vasileios Tzoumas,[2] George J. Pappas,[3] Pratap Tokekar[4]

***Abstract*— We aim to guard swarm-robotics applications against denial-of-service (DoS) failures/attacks that result in withdrawals of robots. We focus on applications requiring the selection of actions for each robot, among a set of available ones, e.g., which trajectory to follow. Such applications are central in large-scale robotic/control applications, e.g., multi-robot motion planning for target tracking. But the current attack-robust algorithms are centralized, and scale quadratically with the problem size (e.g., number of robots). Thus, in this paper, we propose a general-purpose distributed algorithm towards 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 K cliques of robots. The cliques optimize in parallel, independently of each other. That way, `DRM` also offers significant computational speed-ups up to $1/K^2$ the running time of its centralized counterparts. K depends on the robots' communication range, which is given as input to `DRM`. `DRM` also achieves a close-to-optimal performance, equal to the guaranteed performance of its centralized counterparts. We demonstrate `DRM`'s performance in both Gazebo and MATLAB simulations, in scenarios of *active target tracking with swarms of robots*. We observe `DRM` achieves significant computational speed-ups (it is 3 to 4 orders faster) and, *yet*, nearly matches the tracking performance of its centralized counterparts.**

I. Introduction

Safe-critical scenarios of surveillance and exploration often require both mobile agility, and fast capability to detect, localize, and monitor. For example, consider the scenarios:

- *Adversarial target tracking:* Track adversarial targets that move across an urban environment, aiming to escape; [2]
- *Search and rescue:* Explore a burning building to localize any people trapped inside. [3]

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 pushing the frontier on robotic miniaturization and perception [2]–[8], to enable mobile agility and autonomous sensing; and develop distributed coordination algorithms [9]–[13], to enable multi-robot planning, i.e., the joint optimization of robots' actions.

Particularly, distributed planning algorithms (instead of centralized) are especially important when one wishes to deploy large-scale teams of robots; e.g., at the swarm level with tens or hundreds of robots. One reason is the distributed algorithms scale better for larger numbers of robots than their centralized counterparts [9]. And another one, equally important, is that in large-scale teams, not all robots can communicate with each other, but only with the robots within a certain communication range.

However, the safety of the above critical scenarios can still be at peril. For example, robots operating in adversarial scenarios may get cyber-attacked or simply incur failures, both events resulting to a withdrawal of robots from the task. Hence, in such adversarial environments, distributed attack-robust planning algorithms become necessary.[1]

In this paper, we formalize a general framework for *distributed attack-robust multi-robot planning* for tasks that require the maximization of submodular functions, such as in active target tracking with multiple-robots [14].[2] Particularly, we focus on *worst-case* attacks that can result in up to α robot withdrawals from the task.

Attack-robust multi-robot 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 multi-robot planning is NP-hard [16]. All in all, the necessity for distributed attack-robust algorithms, and the inherent computational hardness motivates our goal in this paper: to provide a distributed, provably close-to-optimal approximation algorithm. To this end, we capitalize on recent algorithmic results on centralized attack-robust multi-robot planning [17] and present a distributed attack-robust algorithm.

Related work. Researchers have developed several distributed, but attack-free, planning algorithms, such as [9]–[13]. For example, [9] developed a decentralized algorithm, building on the local greedy algorithm proposed in [18, Section 4], which guarantees a $1/2$ suboptimality bound for submodular objective functions. Particularly, in [9] the robots form a string communication network, and sequentially choose an action, given all the actions of the robots that have chosen so far. Authors of [12] proposed a speed-up of [9]'s

[1]The author is with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061 USA (email: lfzhou@vt.edu).

[2]The author is with the Laboratory of Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (email: vtzoumas@mit.edu).

[3]The author is with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA (email: pappagsg@seas.upenn.edu).

[4]The author was with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061 USA when part of the work was completed. He is currently with the Department of Computer Science, University of Maryland, College Park, MD 20742, USA (email: tokekar@umd.edu).

This work is supported by the ARL CRA DCIST, the National Science Foundation under Grant No. 479615, and the Office of Naval Research under Grant No. N000141812829.

An earlier version of this paper has been accepted as a 2-page extended abstract at the International Symposium on Multi-Robot and Multi-Agent Systems [1].

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

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

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 [19], [20]. Other distributed, attack-free algorithms are also developed in the machine learning literature on submodular maximization, but towards sparse selection (e.g., for data selection, or sensor placement) [21], instead of planning.

Recently, researchers have also developed attack-robust planning algorithms [17], [22]–[24]. With the exception of [22], the algorithms in [17], [23], [24] are centralized. Particularly, [22] provide a distributed attack-resilient algorithm against Byzantine attacks (instead of attacks that result in robot withdrawals). While [17], [23] provide centralized attack-robust algorithms for active information gathering [23] and target tracking [17] with multiple robots. Other attack-robust algorithms, that however apply towards sparse selection instead of planning, are the [25]–[27].

All in all, towards enabling attack-robust planning in multi-robot scenarios, where local intra-robot communication can be necessary, and real-time performance with centralized planning is hard to maintain as the number of robots increases, we make the following contributions on attack-robust distributed multi-robot planning.

Contributions. We introduce the problem of *distributed attack-robust submodular maximization for multi-robot planning*, and provide an algorithm, named *distributed robust maximization* (DRM). DRM distributively partitions the problem among K cliques of robots, where all robots are within communication range. Then, naturally, the cliques optimize in parallel, using [17, Algorithm 1]. We prove for DRM:

a) System-wide attack-robustness: DRM is valid for any number α of worst-case attacks;

b) Superior running time: DRM offers significant computational speed-ups, up to $1/K^2$ the running time of its centralized counterparts. K depends on the inter-robot communication range, which is given as input to DRM.

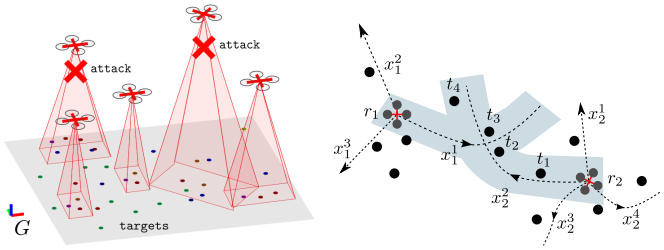
c) Near-to-centralized approximation performance: Even though DRM is a distributed, faster algorithm than its state-of-the-art centralized counterpart [17, Algorithm 1], DRM achieves a near-to-centralized performance, having a suboptimality bound equal to [17, Algorithm 1]'s.

Numerical evaluations. We present Gazebo and MATLAB evaluations of DRM, in scenarios of *active target tracking with swarms of robots*. All simulation results demonstrate DRM's speed-up benefits: DRM runs 3 to 4 orders faster than its centralized counterpart in [17], achieving running times 0.5 to 1.5msec for 100 robots. And, yet, DRM exhibits negligible deterioration in performance (target coverage).

All proofs are given in the appendix.

II. Problem Formulation

We formalize the problem of *distributed attack-robust submodular maximization for multi-robot planning*. At each time-step, the problem asks for assigning actions to the robots, to maximize an objective function despite attacks. For example, in active target tracking with aerial robots (see Fig. 1). The robots' possible actions are their motion



(a) Targets' attacks can block robots' field-of-view

(b) Robots' choose trajectory from a set of motion primitives

Fig. 1. In target tracking with aerial robots, the robots are mounted with down-facing cameras to track mobile targets (depicted as dots in (a) and (b)). The targets have the ability to block some of the robots' cameras. At each time-step, each robot has a set of motion primitives to choose as its trajectory (each possibly covering different targets). For example, in (b) robot 1 has 3 motion primitives, $\{x_1^1, x_1^2, x_1^3\}$, and robot 2 has 4 motion primitives, $\{x_2^1, x_2^2, x_2^3, x_2^4\}$, where x_1^1 covers 2 targets, $\{t_2, t_3\}$, and x_2^2 covers 4 targets, $\{t_1, t_2, t_3, t_4\}$. In combination, however, the two motion primitives totally cover 4 targets, $\{t_1, t_2, t_3, t_4\}$.

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:[3]

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

b) 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 $\mathcal{N}_i$.

c) 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 1 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 1 motion primitive at a time to be its trajectory. For example, in Figure 1-(b) we have 2 robots, where $\mathcal{X}_1 = \{x_1^1, x_1^2, x_1^3\}$ (and robot 1 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 2 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 Figure 1-(b), $\mathcal{S} = \{x_1^1, x_2^2\}$.

d) Objective function: The quality of each $\mathcal{S}$ is quantified by a non-decreasing 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 [16]. As shown in Figure 1-(b), the number of targets covered by the chosen actions, $\mathcal{S} = \{x_1^1, x_2^2\}$, is $f(\mathcal{S}) = 4$.

e) Attacks: At each time, we assume the robots encounter worst-case attacks. We assume the maximum number of anticipated attacks to be known, and denote it by α.

Problem 1 (Distributed attack-robust submodular maximization for multi-robot planning)**.** *The robots, by exchanging information only over the communication graph G, assign*

[3]**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 $\mathcal{B}$.

Algorithm 1: Distributed robust maximization (DRM).

Input: Robots' available actions $\mathcal{X}_i$, $i \in \mathcal{R}$; monotone and submodular 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** $\mathcal{S} = \bigcup_{k=1}^{K} \mathcal{S}_k$.

an action to each robot $i \in \mathcal{R}$ to maximize f against α worst-case attacks/failures:

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

where $\mathcal{A}$ corresponds to the actions of the attacked robots. The first constraint ensures only 1 action is chosen per robot.

Problem 1 is equivalent to a two-stage perfect information sequential game [28, Chapter 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 $\mathcal{A}$.

III. A Distributed Algorithm: DRM

We present *Distributed Robust Maximization* (DRM), a distributed algorithm for Problem 1 (Algorithm 1). DRM executes sequentially two main steps: *distributed clique partition* (DRM's 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).[4] During the second step, each clique computes an attack-robust action assignment (in parallel with the rest), using the centralized algorithm in [17] —henceforth, we refer to the algorithm in [17] 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 below; and quantify its running time and performance in Section IV.

A. Distributed clique partition

We present the first step of DRM, namely, *distributed clique partition* (DRM's line 1, that calls DCP, whose pseudocode is presented in Algorithm 2). Notably, the problem is inapproximable in polynomial time, since even finding a single clique of *maximum* size is inapproximable (unless NP=P) [29] (even in a centralized way).

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

Algorithm 2: Distributed clique partition (DCP).

Input: Robots' positions $\mathcal{P}$; communication range r_c.
Output: Clique partition of graph G.

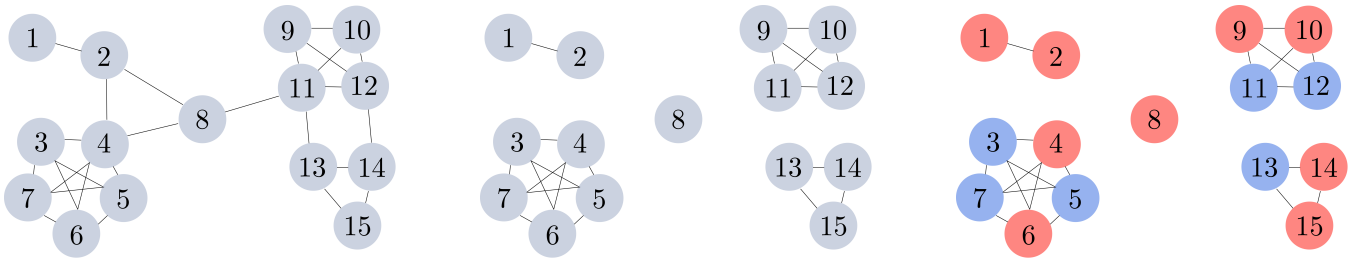
1: Given $\mathcal{P}$ and r_c, find communication graph G;
2: For each $i \in \mathcal{R}$, find a maximal clique $\mathcal{C}^i$ containing i by calling PerVrtx-MaxClique(G);
3: **for** each robot i **do**
4: Share $\mathcal{C}^i$ with each neighbor $j \in \mathcal{N}_i$ (and receive all $\mathcal{C}^j$ from neighbors);
5: **for** all neighbors $j \in \mathcal{N}_i$ **do**
6: **if** $|\mathcal{C}^i| \geq |\mathcal{C}^j|$ **then**
7: $\mathcal{C}^j \leftarrow \mathcal{C}^j \setminus (\mathcal{C}^i \bigcap \mathcal{C}^j)$;
8: **else**
9: $\mathcal{C}^i \leftarrow \mathcal{C}^i \setminus (\mathcal{C}^i \bigcap \mathcal{C}^j)$;
10: **return** Generated cliques.

DCP builds on [30, Algorithm 2], which finds for *each* vertex in a graph G a clique containing the vertex (DCP's line 2). We refer to [30, Algorithm 2] as PerVrtx-MaxClique in DCP. The cliques returned by PerVrtx-MaxClique can overlap with each other, since PerVrtx-MaxClique returns as many cliques as vertices/robots. In order to separate those cliques, in DCP's lines 3-9 each robot communicates with its neighbors once, during which: a) each robot shares its clique with its neighbors (DRM's line 4); b) robot and its neighbor follow a partition rule that, from their two cliques, the smaller one will lose the overlapped robots (DCP's lines 6-9). That way, DCP aims to partition G to fewer and larger cliques. The generated non-overlapping cliques are returned by DCP's line 10.

B. Per clique attack-robust optimization

We now present DRM's second step: *per clique attack-robust optimization* (DRM's lines 2-8). The step calls central-robust as subroutine, and therefore we recall its steps here from [17]: 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 anticipated 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 anticipated 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 1 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 a) assumes the robots in the bait set are removed from $\mathcal{R}'$, and then b) greedily assigns actions to the rest of the robots using the centralized greedy in [18, Section 2] which ensures a near-optimal assignment (at least 1/2 close to the optimal).

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 com-



(a) A communication graph G of 15 robots (b) `DCP` partitions G into 5 cliques (c) Each clique runs `central-robust`

Fig. 2. Qualitative description of `DRM`'s steps over the communication graph G in subfigure (a), composed of 15 robots. The number of anticipated attacks is considered to be $\alpha = 2$. In the first step, we assume `DCP` (`DRM`'s line 1) partitions G into 5 cliques, as shown in subfigure (b). In the second step, all 5 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 3 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.

putes 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 each clique assumes a worst-case scenario where it 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 loss of generality, in `DRM`'s pseudo-code in Algorithm 1 we present the former scenario, where $\alpha_k = \alpha$ across all cliques; notwithstanding, our theoretical results on `DRM`'s performance (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 (see Fig. 2 for an example):

a) `DRM`'s lines 4-5 ($\alpha < |\mathcal{C}_k|$): 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).

b) `DRM`'s lines 6-7 ($\alpha \geq |\mathcal{C}_k|$): 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.

c) `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`).

To close the section, we note that `DRM` is valid for any number of attacks, since `central-robust` is [17].

IV. Performance Analysis

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 [31]). *Consider non-decreasing submodular $f : 2^{\mathcal{X}} \mapsto \mathbb{R}$ such that $f(x) \neq 0$, for any $x \in \mathcal{X} \setminus \{\emptyset\}$ (without 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 1 if, e.g., two robots cover the exact same targets.

B. Running time, and approximation performance

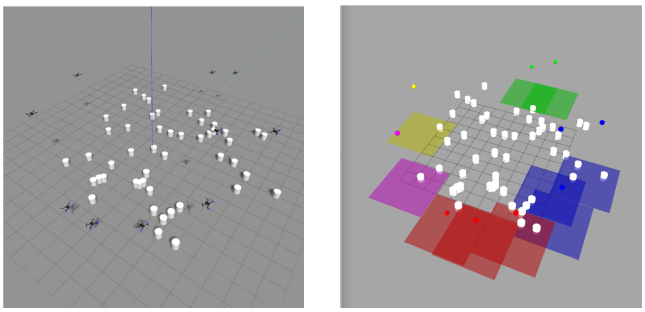
We present `DRM`'s running time and suboptimality bounds. To this end, we use the notation:

- $\mathcal{M}$ is the set of robots composing G's largest clique;
- $\mathcal{X}_{\mathcal{M}}$ is the set of possible actions of all robots in $\mathcal{M}$; that is, $\mathcal{X}_{\mathcal{M}} \triangleq \cup_{i \in \mathcal{M}} \mathcal{X}_i$;
- $f^\star$ is the optimal value of Problem 1;
- $\mathcal{A}^\star(\mathcal{S})$ is a worst-case removal from $\mathcal{S}$ (a removal from $\mathcal{S}$ corresponds to a set of robot/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 $O(|\mathcal{R}|) + O(|\mathcal{X}_{\mathcal{M}}|^2)$ time.*

The part $O(|\mathcal{R}|)$ corresponds to `DRM`'s clique partition step (`DRM`'s line 1), while $O(|\mathcal{X}_{\mathcal{M}}|^2)$ to `DRM`'s attack-robust optimization step (`DRM`'s lines 2-8). Typically, $O(|\mathcal{R}|)$ is smaller than $O(|\mathcal{X}_{\mathcal{M}}|^2)$, since the latter grows quadratically fast, and, as a result, we henceforth ignore the former's contribution in the running time.

In contrast, the centralized [17, Algorithm 1] runs in $O(|\mathcal{X}|^2)$ time. Thus, when $\mathcal{X}_{\mathcal{M}} \subset \mathcal{X}$ (which happens when G is partitioned into at least 2 cliques), then `DRM` offers a significant computational speed-up. The reasons are two-forth: *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 the largest clique $\mathcal{M}$ (Theorem 1). Besides parallelization, the smaller clique size also contributes to the computational reduction. To



(a) Gazebo environment (b) Rviz environment

Fig. 3. Gazebo simulation setup: 10 aerial robots and 50 ground mobile targets: (a) Gazebo environment; and (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.

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 running time is smaller by the factor K^2 (than the running time of its centralized counterpart).

Theorem 2 (Approximation performance)**.** *DRM returns a feasible $\mathcal{S}$ such that if $K \geq 2$, then*

$$\frac{f(\mathcal{S} \setminus \mathcal{A}^{\star}(\mathcal{S}))}{f^{\star}} \geq \max \left[\frac{1-\nu_f}{2}, \frac{1}{(\alpha+1)K|\mathcal{M}|}, \frac{1}{(N-\alpha)K|\mathcal{M}|} \right]. \quad (3)$$

If, instead, $K = 1$, then DRM is the same as its centralized counterpart in [17], in which case the following suboptimality bound holds [17, Theorem 1]:

$$\frac{f(\mathcal{S} \setminus \mathcal{A}^{\star}(\mathcal{S}))}{f^{\star}} \geq \max \left[\frac{1-\nu_f}{2}, \frac{1}{(\alpha+1)}, \frac{1}{(|\mathcal{R}|-\alpha)} \right]. \quad (4)$$

By comparing eq. (3) and eq. (4), and focusing on the ν_f-depended bounds, we conclude that even though DRM is a distributed, faster algorithm than its centralized counterpart, it still achieves a near-to-centralized performance. At the same time, DRM's α-dependent bounds are inversely proportional to the number of cliques, as well as, $\mathcal{M}$'s size.

Generally, Theorem 2 implies DRM guarantees a close-to-optimal value for any submodular f. Specifically, DRM's approximation factor is bounded by the α-depended bounds (rightmost two bounds in eq. (3)), which are non-zero for any finite number of robots. Similarly, the curvature-dependent bound is also non-zero for any f with curvature $\nu_f < 1$.

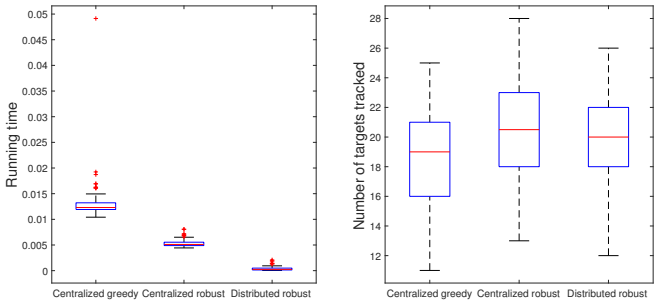
V. Numerical Evaluation

We present DRM's Gazebo and MATLAB evaluations in scenarios of *active target tracking with swarms of robots*. The implementations' code is available online.[5]

Compared algorithms. We compare DRM with two algorithms. First, the centralized counterpart of DRM in [17], named `central-robust` (its near-optimal performance has been extensively demonstrated in [17]). The second algorithm is the centralized greedy algorithm in [18], named `central-greedy`. The difference between the two algorithms is that the former is attack-robust, whereas the latter is attack-agnostic. For this reason, in [17] 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.

[5] https://github.com/raaslab/distributed_resilient_target_tracking.git

A. Gazebo evaluation over multiple steps with mobile targets



(a) Running time (b) Number of targets tracked

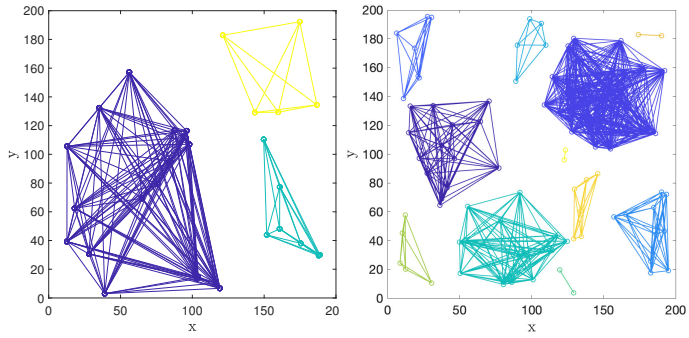
Fig. 4. Gazebo evaluation (averaged across 50 rounds): The tracking performance is captured by the number of covered targets per round.

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. Due to the running efficacy of Gazebo (which is independent of DRM), we focus on small-scale scenarios of 10 robots. In the MATLAB simulation we focus instead in larger-scale scenarios of 100 robots.

Simulation setup. We consider 10 aerial robots that are tasked to track 50 ground mobile targets (Fig. 3-(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 (Fig. 3-(b)). Each robot i has 4 candidates trajectories, $\mathcal{X}_i = \{$`forward, backward, left, right`$\}$, and flies on a different fixed plane (to avoid collision with other robots). Each robot has a square filed-of-view $l_o \times l_o$. Once a robot picks a trajectory, it flies a distance l_f along that trajectory. 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 (per round).

For each of the compared algorithms, at each round each robot picks one of its 4 trajectories. Then, the robot flies a $l_f = 3$ meters along the selected trajectory.

When an attack happens, we assume the attacked robot's tracking sensor (e.g., camera) to be turned-off; nevertheless,



(a) 30 robots with $r_c = 90$ (b) 100 robots with $r_c = 50$

Fig. 5. MATLAB evaluation: Examples of clique formulations (Algorithm 2) across various numbers of robots and communication ranges r_c.

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.[6]

Results. The results are reported in Fig. 4. We observe:

a) Superior running time: DRM runs considerably faster than both central-robust and central-greedy: 3 orders faster than the former, and 4 than the latter, with average running time 0.1msec (Fig. 4-(a)).

b) Near-to-centralized tracking performance: Despite that DRM runs considerably faster, it maintains near-to-centralized performance: DRM covers on average 20 targets per round, while central-robust covers 20.2 (Fig. 4-(b)). As expected, the attack-agnostic central-greedy performs worse than all algorithms, even being centralized.

B. 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.

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 $\lfloor 3N/4 \rfloor$. Similarly to the Gazebo simulations, each robot moves on a fixed plane, and has four possible trajectories: forward, backward, left and right. We set $l_t = 10$ and $l_o = 3$ for all robots. We randomly generate the positions of the robots and targets in a 2D 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 four communication ranges: $r_c = 30, 50, 70, 90$. For a visualization of r_c's effect on the formed cliques, see Fig. 5, where we present two of the generated scenarios. All algorithms are executed for one round in each Monte Carlo run.

[6] https://youtu.be/T0Hb0UURCLM

Notably, since we consider large-scale scenarios (up to $N = 100$ robots, and up to 75 attacks, when $N = 100$, and $\alpha = \lfloor 3N/4 \rfloor$), computing the worst-case attack via a brute-force algorithm is now infeasible (we recall that computing a worst-case attack is NP-hard, and, as a result, to compute one in practice, in small-scale scenarios we need to use a brute-force algorithm, otherwise, in large-scale scenarios we need to use an approximation algorithm). Herein, given a trajectory assignment $\mathcal{S}$ to all robots, the problem of computing a worst-case attack is a monotone submodular optimization problem, which can be solved near-optimally using the greedy algorithm in [15]. Therefore, we henceforth consider greedy attacks, instead of worst-case attacks.

Results. The results are reported in Fig. 6, where we make the same qualitative conclusions as in the Gazebo evaluation:

a) Superior running time: DRM runs several orders faster than both central-robust and central-greedy: 3 to 4 orders, achieving running time from 0.5msec to 1.5msec (Figs. 6-(a-h)). Notably, we also 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 [17]).

b) Near-to-centralized tracking performance: Although DRM runs considerably faster, it retains a tracking performance close to the centralized one (Figs. 6-(e-h)). On the other hand, unsurprisingly, the attack-agnostic greedy performs worse than all algorithms.

To summarize, in all above simulations, DRM offered significant computational speed-ups, and, yet, still achieved a tracking performance that matched the performance of the centralized, near-optimal algorithm in [17].

VI. Conclusion

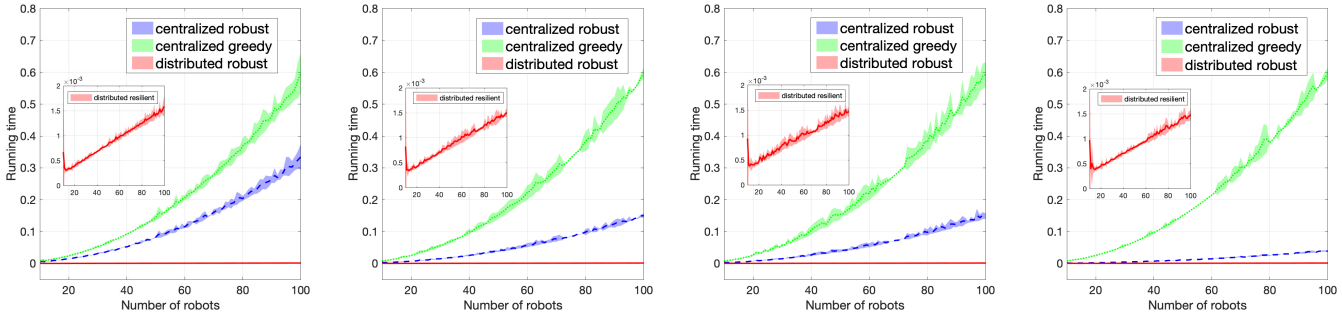
We worked towards securing swarm-robotics applications against worst-case attacks resulting to robot withdrawals. Particularly, we proposed DRM, a distributed robust submodular optimization algorithm. DRM is general-purpose: it applies to any Problem 1's instance. We proved DRM runs considerably faster 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.

A future avenue is to investigate distributed algorithms where each robot communicates with neighboring robots even across different cliques than its own. That way, the robots can utilize more information towards an attack-robust action assignment. Another future avenue is to investigate distributed algorithms against an unknown number of attacks (e.g., captured by stochastic processes [32]).

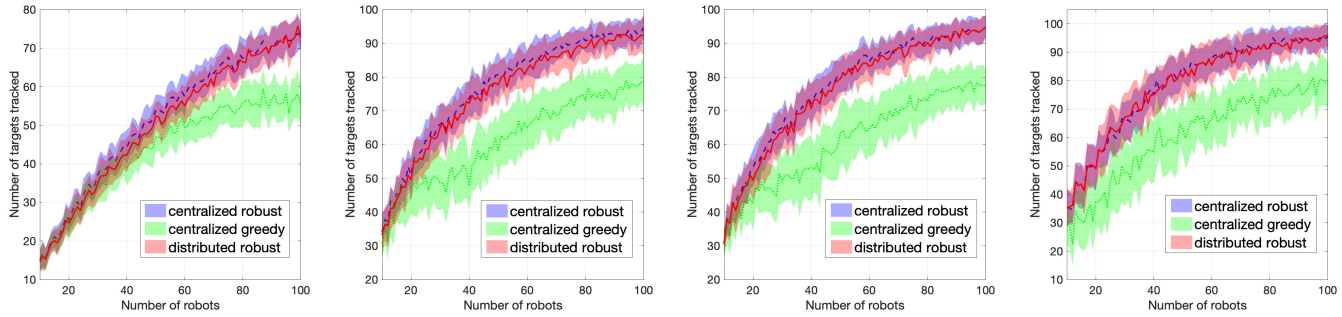
Appendix

A. Proof of Theorem 1

DRM's running time is equal to DCP's running time, plus the running time for all cliques to execute central-robust in parallel. Particularly, in DCP, each robot first finds its maximal clique using PerVrtx-MaxClique, which runs in $O(|\mathcal{R}|)$ time. Then, it shares its maximal clique with its neighbors for graph partition, which also takes $O(|\mathcal{R}|)$ time. Thus, DCP



(a) $\alpha = \lfloor N/4 \rfloor$, $r_c = 30$ (b) $\alpha = \lfloor N/2 \rfloor$, $r_c = 50$ (c) $\alpha = \lfloor N/2 \rfloor$, $r_c = 70$ (d) $\alpha = \lfloor 3N/4 \rfloor$, $r_c = 90$



(e) $\alpha = \lfloor N/4 \rfloor$, $r_c = 30$ (f) $\alpha = \lfloor N/2 \rfloor$, $r_c = 50$ (g) $\alpha = \lfloor N/2 \rfloor$, $r_c = 70$ (h) $\alpha = \lfloor 3N/4 \rfloor$, $r_c = 90$

Fig. 6. MATLAB evaluations (averaged across 30 Monte Carlo runs): (a)-(d) depict running time results, for various α and r_c values; and (e)-(h) depict corresponding tracking performance results ; and (e)-(h) depict corresponding numbers of cliques and clique numbers.

runs in $O(|\mathcal{R}|)$ time. Next, since all cliques perform in parallel, the running time depends on the largest clique, which gives a $O(|\mathcal{X}_\mathcal{M}|^2)$ time (the proof follows the proof of [17, Part 2 of Theorem 1]). Totally, Algorithm 1 runs in $O(|\mathcal{R}|) + O(|\mathcal{X}_\mathcal{M}|^2)$ time.

B. Proof of Theorem 2

The proof of the α-dependent bounds in Theorem 2 follows with similar steps to the α-dependent bound in [24, Theorem 1, eq. (7)] and we omit it. For the proof of the ν_f-dependent bound, 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 only in $\mathcal{R}'$ (i.e., the restriction of $\mathcal{S}'$ to $\mathcal{R}'$). And vise versa: given an action assignment $\mathcal{S}'$ to only a subset $\mathcal{R}'$ of robots, we let $\mathcal{R}(\mathcal{S})$ denote this subset (i.e., $\mathcal{R}(\mathcal{S}') = \mathcal{R}'$). Additionally, 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 ($k \in \{1, \dots, K\}$); evidently, $\mathcal{S} = \bigcup_{k=1}^K \mathcal{S}_k$. Moreover, we let $\mathcal{S}_k^b$ correspond to bait actions chosen by `central-robust` in $\mathcal{C}_k$, and $\mathcal{S}_k^g$ denote the actions for the remaining robots in $\mathcal{C}_k$; that is, $\mathcal{S}_k = \mathcal{S}_k^b \cup \mathcal{S}_k^g$. Notably, if $\alpha \geq |\mathcal{C}_k|$, then $\mathcal{S}_k^g = \emptyset$. Henceforth, we let $\mathcal{S}$ denote 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., $\mathcal{W} \triangleq \mathcal{R} \setminus \mathcal{R}(\mathcal{A}^\star(\mathcal{S}))$. Finally, we let $\mathcal{W}_k \triangleq \mathcal{W} \cap \mathcal{C}_k$, $\mathcal{W}_k^b \triangleq \mathcal{W}_k \cap \mathcal{R}(\mathcal{S}_k^b)$, and $\mathcal{W}_k^g \triangleq \mathcal{W}_k \cap \mathcal{R}(\mathcal{S}_k^g)$.

Now the proof follows from the steps:

$$f(\mathcal{S} \setminus \mathcal{A}^\star(\mathcal{S})) \geq (1-\nu_f) \sum_{r \in \mathcal{W}} f(\mathcal{S}(r)) \tag{5}$$

$$= (1-\nu_f) \sum_{i=1}^K \sum_{r \in \mathcal{W}_k} f(\mathcal{S}(r)) \tag{6}$$

$$= (1-\nu_f) \sum_{i=1}^K \left[\sum_{r \in \mathcal{W}_k^b} f(\mathcal{S}(r)) + \sum_{r \in \mathcal{W}_k^g} f(\mathcal{S}(r)) \right] \tag{7}$$

$$\geq (1-\nu_f) \sum_{i=1}^K \left[\sum_{r \in \mathcal{W}_k^b \setminus (\mathcal{R}(\mathcal{S}_k^g) \setminus \mathcal{W}_k^g)} f(\mathcal{S}(r)) + \sum_{r \in \mathcal{R}(\mathcal{S}_k^g) \setminus \mathcal{W}_k^g} f(\mathcal{S}(r)) + \sum_{r \in \mathcal{W}_k^g} f(\mathcal{S}(r)) \right] \tag{8}$$

$$= (1-\nu_f) \sum_{i=1}^K \left[\sum_{r \in \mathcal{W}_k^b \setminus (\mathcal{R}(\mathcal{S}_k^g) \setminus \mathcal{W}_k^g)} f(\mathcal{S}(r)) + \sum_{r \in \mathcal{R}(\mathcal{S}_k^g)} f(\mathcal{S}(r)) \right] \tag{9}$$

$$\geq (1-\nu_f) \sum_{i=1}^K \left[\sum_{r \in \mathcal{W}_k^b \setminus (\mathcal{R}(\mathcal{S}_k^g) \setminus \mathcal{W}_k^g)} f(\mathcal{S}(r)) + f(\mathcal{S}_k^g) \right] \tag{10}$$

$$\geq (1-\nu_f) \sum_{i=1}^K \left[\sum_{r \in \mathcal{W}_k^b \setminus (\mathcal{R}(\mathcal{S}_k^g) \setminus \mathcal{W}_k^g)} f(\mathcal{S}^\star(r)) + \frac{1}{2} f(\mathcal{S}^\star(\mathcal{R}(\mathcal{S}_k^g))) \right] \tag{11}$$

$$\geq \frac{1-\nu_f}{2} \sum_{i=1}^{K} f(\mathcal{S}^\star(\mathcal{W}_k)) \tag{12}$$

$$\geq \frac{1-\nu_f}{2} f(\mathcal{S}^\star(\mathcal{W})) \tag{13}$$

$$= \frac{1-\nu_f}{2} f(\mathcal{S}^\star \setminus \mathcal{A}^\star(\mathcal{S}^\star)). \tag{14}$$

Ineq. (5) follows from the definition of ν_f (see [24, Proof of Theorem 1]). Eqs. (6) and (7) follow from the notation we introduced above. Ineq. (8) is implied by the fact that any action in $\mathcal{S}(\mathcal{W}_k^b)$ is a bait. Ineq. (10) 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]. Ineq. (11) holds since a), with respect to the left term in the sum, the robots in the sum correspond to robots whose actions are baits; and b), with respect to the right term in the sum, the greedy algorithm that has assigned the actions $\mathcal{S}_k^g$ guarantees at least $1/2$ the optimal [18]. Ineq. (12) holds again due to the submodularity of f, as above. The same for ineq. (13). Eq. (14) follows from the notation, which implies $\mathcal{S}^\star(\mathcal{W}) \equiv \mathcal{S}^\star \setminus \mathcal{A}^\star(\mathcal{S}^\star)$.

REFERENCES

[1] L. Zhou and P. Tokekar, "An approximation algorithm for distributed resilient submodular maximization," in *International Symposium on Multi-Robot and Multi-Agent Systems, in print*, 2019.

[2] C. Nieto-Granda, J. G. Rogers III, and H. Christensen, "Multi-robot exploration strategies for tactical tasks in urban environments," in *Unmanned Systems Technology XV*, vol. 8741, 2013, p. 87410B.

[3] V. Kumar and N. Michael, "Opportunities and challenges with autonomous micro aerial vehicles," in *Robotics Research*, 2017, pp. 41–58.

[4] M. Michini, M. A. Hsieh, E. Forgoston, and I. B. Schwartz, "Robotic tracking of coherent structures in flows," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 593–603, 2014.

[5] S. Karaman and E. Frazzoli, "High-speed flight in an ergodic forest," in *IEEE Intern. Confer. on Robotics and Automation*, 2012, pp. 2899–2906.

[6] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[7] T. Cieslewski, E. Kaufmann, and D. Scaramuzza, "Rapid exploration with multi-rotors: A frontier selection method for high speed flight," in *IEEE/RSJ Int. Conf. on Intel. Robots and Systems*, 2017, pp. 2135–2142.

[8] M. Santos, Y. Diaz-Mercado, and M. Egerstedt, "Coverage control for multirobot teams with heterogeneous sensing capabilities," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 919–925, 2018.

[9] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Decentralized active information acquisition: Theory and application to multi-robot SLAM," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 4775–4782.

[10] B. Schlotfeldt, D. Thakur, N. Atanasov, V. Kumar, and G. J. Pappas, "Anytime planning for decentralized multirobot active information gathering," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1025–1032, 2018.

[11] R. Khodayi-mehr, Y. Kantaros, and M. M. Zavlanos, "Distributed state estimation using intermittently connected robot networks," *IEEE Transactions on Robotics*, 2019.

[12] M. Corah and N. Michael, "Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice," *Autonomous Robots*, vol. 43, no. 2, pp. 485–501, 2019.

[13] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-MCTS: Decentralized planning for multi-robot active perception," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 316–337, 2019.

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

[15] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions–I," *Mathematical programming*, vol. 14, no. 1, pp. 265–294, 1978.

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

[17] L. Zhou, V. Tzoumas, G. J. Pappas, and P. Tokekar, "Resilient active target tracking with multiple robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 129–136, 2019.

[18] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions–II," in *Polyhedral combinatorics*, 1978, pp. 73–87.

[19] B. Gharesifard and S. L. Smith, "Distributed submodular maximization with limited information," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 4, pp. 1635–1645, 2018.

[20] D. Grimsman, M. S. Ali, J. P. Hespanha, and J. R. Marden, "The impact of information in greedy submodular maximization," *IEEE Transactions on Control of Network Systems*, 2018.

[21] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause, "Distributed submodular maximization: Identifying representative elements in massive data," in *Advances in Neural Information Processing Systems*, 2013, pp. 2049–2057.

[22] 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," *Autonomous Robots*, vol. 43, no. 3, pp. 743–768, 2019.

[23] B. Schlotfeldt, V. Tzoumas, D. Thakur, and G. J. Pappas, "Resilient active information gathering with mobile robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 4309–4316.

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

[25] J. B. Orlin, A. S. Schulz, and R. Udwani, "Robust monotone submodular function maximization," *Mathematical Programming*, vol. 172, no. 1-2, pp. 505–537, 2018.

[26] I. Bogunovic, S. Mitrović, J. Scarlett, and V. Cevher, "A distributed algorithm for partitioned robust submodular maximization," in *IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, 2017, pp. 1–5.

[27] V. Tzoumas, K. Gatsis, A. Jadbabaie, and G. J. Pappas, "Resilient monotone submodular function maximization," in *IEEE Conference on Decision and Control*, 2017, pp. 1362–1367.

[28] R. B. Myerson, *Game theory*. Harvard university press, 2013.

[29] D. Zuckerman, "Linear degree extractors and the inapproximability of max clique and chromatic number," in *ACM Symposium on Theory of Computing*, 2006, pp. 681–690.

[30] 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 *International Workshop on Algorithms and Models for the Web-Graph*, 2013, pp. 156–169.

[31] 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 applied mathematics*, vol. 7, no. 3, pp. 251–274, 1984.

[32] H. Park and S. Hutchinson, "Robust rendezvous for multi-robot system with random node failures: an optimization approach," *Autonomous Robots*, pp. 1–12, 2018.