Robotics: Science and Systems 2020
Corvalis, Oregon, USA, July 12-16, 2020

Robust Multiple-Path Orienteering Problem: Securing Against Adversarial Attacks

Guangyao Shi[†] Lifeng Zhou[‡] Pratap Tokekar[††]

***Abstract*—The multiple-path orienteering problem asks for paths for a team of robots that maximize the total reward collected while satisfying budget constraints on the path length. This problem models many multi-robot routing tasks such as exploring unknown environments and information gathering for environmental monitoring. In this paper, we focus on how to make the robot team robust to failures when operating in adversarial environments. We introduce the Robust Multiple-path Orienteering Problem (RMOP) where we seek worst-case guarantees against an adversary that is capable of attacking at most α robots. Our main contribution is a general approximation scheme with bounded approximation guarantee that depends on α and the approximation factor for single robot orienteering. In particular, we show that the algorithm yields a (i) constant-factor approximation when the cost function is modular; (ii) $\log$ factor approximation when the cost function is submodular; and (iii) constant-factor approximation when the cost function is submodular but the robots are allowed to exceed their path budgets by a bounded amount. In addition to theoretical analysis, we perform simulation study for an ocean monitoring application to demonstrate the efficacy of our approach.**

I. Introduction

The Orienteering Problem (OP) is that of determining a path, whose length is less than a given budget, from a given starting vertex that maximizes the total reward collected along the path [1]. The reward depends on the vertices visited along the path. The OP[1] naturally models informative-path planning a robot is tasked to gather as much information from the environment as possible within a give time or energy budget. For example, in [2]–[4], ocean monitoring, opportunistic surveillance, and 3D reconstruction tasks are formulated as the OP or its variants. In general, the OP is NP-hard but there are constant-factor approximation algorithms for many variants [5]. This includes the Multiple-path Orienteering Problem (MOP) [5] where the goal is to design paths for N robots such that the sum of the rewards collected by all the robots is maximized. In this paper, we introduce the robust variant of OP. Specifically, we introduce the Robust Multiple-Path Orienteering Problem (RMOP) motivated by scenarios where robots operate in adversarial or failure-prone environments.

Figure 1 shows a motivating scenario where a team of underwater robots are tasked with gathering data in an ocean. However, some robots in the team may fail to complete their paths either due to adversarial attacks [6] or hardware malfunction [7]. If a robot fails, then the data gathered by it is lost. Our goal is to provide efficient planning and coordination algorithms that are resilient to such failures.

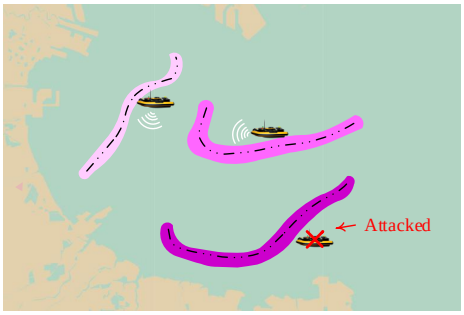


Fig. 1. Case study of monitoring a marine environment with aquatic robots. The robots are tasked with finding informative paths to gather data. The darker the color of path is, the more valuable path since it gathers information from a more important region. We investigate the question of how the robots should plan their paths if we expect some of the robots to fail due to adversarial elements or natural causes?

Building robot teams that are robust to adversarial attacks is emerging as an important research area [8]–[11]. Our approach differs from classical fault-tolerant frameworks [12]–[14] that focus on making individual robots robust to failures. Instead, we focus on the question of how should the team coordinate their actions to improve redundancy in their plans such that even if some robots fail, the overall performance of the team will not drop significantly. As such, our work is completely different to the work on making individual robots robust.

In this paper, we focus on the RMOP to make progress towards the aforementioned broader goal. The RMOP seeks plans for a team of N robots that guard against worst-case failures. Of course, in the worst-case all N robots may fail. To make it more meaningful we study the case where at most a given number $\alpha < N$ robots may fail. What we seek is to understand how the performance of the team will be affected as a function of α. Our main contribution is an algorithmic scheme that uses a single robot OP solution as a subroutine. Choosing an appropriate subroutine allows us to investigate three variants of the original problem. In the general version, the reward collected by an individual robot is a submodular function of the vertices along the path. Submodularity is the property of diminishing returns [15]. Many information gathering measures such as mutual information [16] and

[†]Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA email:gyshi@terpmail.umd.edu.
[‡]Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061 USA e-mail: lfzhou@vt.edu.
[††]Department of Computer Science, University of Maryland, College Park, MD 20742 USA email: tokekar@umd.edu

[1]Unless specified otherwise, OP refers to single robot orienteering.

coverage area [17] are known to be submodular. We also study special cases where the reward function is strictly modular (i.e., additive) and where the budget constraint for each robot can be relaxed by a bounded amount.

A. Related work

The orienteering problem has been researched extensively by both theoretical computer science and operations research communities. The review by Vansteenwegen [1] summarizes various algorithms for OP and its variants. We highlight the results most closely related to our work. Blum et al. [5] presented a polynomial time 4–approximation for OP when the objective function is modular. This result is then extended to yield a 5–approximation for the MOP assuming all robots start at different vertices. If the reward function is submodular, Chekuri and Pal [18] present a recursive greedy algorithm for a single robot that yields a $O(\log(OPT))$ approximation algorithm, where OPT is the reward collected by the optimal algorithm. The algorithm runs in quasi-polynomial time.

Singh et al. [19] showed how to use OP and MOP for active information gathering to learn a spatial model of the environment represented by Gaussian Processes. Their algorithms sequentially finds paths for each robot using the single-robot algorithms by Blum et al. [5] and Chekuri and Pal [18] as subroutines. Atanasov et al. [20] recently presented a decentralized version for multi-robot information gathering along similar lines as [19]. They use a submodular objective function but solve a finite horizon planning problem as opposed to OP. However, none of these works account for potential failures of the robots, as we do in RMOP.

Recently, Jorgensen et al. introduced the Matroids Team Surviving Orienteers Problem (MTSO) [2] which does account for individual robot failures. They assume that there is some given probability of failure associated with every edge in the environment. The goal is to maximize the expected rewards while ensuring each path satisfies some survival chance constraints. MTSO is appropriate when the failures of robots are random and follow a known distribution. The version we study, the RMOP, accounts for worst-case failures which makes it better suited when operating in adversarial conditions or in stochastic conditions when worst-case guarantees are sought due to unknown probability distributions.

Our work builds on recent work on robust submodular maximization [21]–[26] which select sets that are robust to worst-case removal of some subset of items. The challenge in this framework is to solve the trade-off between too much overlap, thereby not enough coverage (i.e., reward) and too little overlap, thereby not enough redundancy. The conceptual idea in these papers is similar — the final solution consists of two subsets, one that has enough redundancy to ensure robustness against worst-case removal and the other that has enough coverage to get good overall performance. Orlin et al. [21] term the former as "copies" whereas it is called as "baits" in [25]. The robust submodular maximization formulation has been applied for multi-robot, multi-target tracking in centralized [25] and decentralized settings [26] as well as for active information gathering with multi-robot teams [24].

We seek similar robustness guarantees as in the works mentioned in the previous paragraph. The key technical advancement we make is that these prior work solve a single step selection problem whereas we solve a multi-step planning problem. As a result, the single robot problem in the prior work can be trivially solved optimally (amounts to selecting the best amongst a finite set of options), whereas in the RMOP the single robot problem (OP) itself is NP-Hard. While both [25] and [24] use their results for planning over a finite horizon, they make key assumptions that are limiting. Schlotfeldt [24] assume that the single robot information gathering problem can be solved optimally (c.f. Proposition 1) whereas Zhou et al. [25] repeatedly solve the problem at each time step. Instead, we show how to use an approximate solution to the OP to yield a bounded approximation solution to the RMOP.

B. Contributions

The main contributions of this paper are as follows. We introduce the Resilient Multiple-Path Orienteering Problem. We present a general approximation scheme to solve RMOP. We analyze the running time and the performance of the algorithm. In particular, we show that the approximation ratio is a constant of the approximation factor for single robot OP. We show how to employ three single robot algorithms for modular and submodular OP as subroutines in our algorithm and analyze their performance. We evaluate the performance of our algorithm using simulations involving a case study of ocean monitoring with a team of robots. In addition, we give an alternative and more complete proof on the bound of sequential algorithm in [19] which is of independent interest.

The rest of the paper is organized as follows. We provide the necessary background on submodular functions and formally introduce the RMOP in Sec. II. Next, the approximation algorithm is proposed in Sec. III. Detailed analysis of the proposed algorithm is given in Sec. IV. Finally, simulation results are given in Sec. V.

II. Problem Description

In this section, we formally describe the Robust Multiple Orienteering Problem. We start by introducing notations and conventions used in the paper.

We use calligraphic fonts to denote sets (e.g. $\mathcal{A}$). Given a set $\mathcal{A}$, $2^{\mathcal{A}}$ denotes the power set of $\mathcal{A}$ and $|\mathcal{A}|$ denotes the cardinality of $\mathcal{A}$. Given another set $\mathcal{B}$, the set $\mathcal{A}\backslash\mathcal{B}$ denotes the set of elements in $\mathcal{A}$ but not in $\mathcal{B}$. Given a set $\mathcal{V}$, a set function $f : 2^{\mathcal{V}} \mapsto \mathbb{R}$, and an element $x \in \mathcal{V}$, $f(x)$ is a shorthand that denotes $f(\{x\})$. We use $f_{\mathcal{A}}(\mathcal{B})$ to denote $f(\mathcal{A} \cup \mathcal{B}) - f(\mathcal{A})$.

We now define two useful properties of set functions.

Definition 1 (Normalized Monotonicity)**.** For a set $\mathcal{V}$, a function $f : 2^{\mathcal{V}} \mapsto \mathbb{R}$ is called as normalized, monotonically nondecreasing if and only if for any $\mathcal{A} \subseteq \mathcal{A}' \subseteq \mathcal{V}, f(\mathcal{A}) \leq f(\mathcal{A}')$ and $f(\mathcal{A}) = 0$ if and only if $A = \emptyset$.

As a short-hand, we refer to a normalized, monotonically non-decreasing function as simply a monotone function.

Definition 2 (Submodularity)**.** For a set $\mathcal{V}$, a function $f : 2^{\mathcal{V}} \mapsto \mathbb{R}$ is submodular if and only if for any sets $\mathcal{A} \subseteq \mathcal{V}$ and $\mathcal{A}' \subseteq \mathcal{V}$, we have $f(\mathcal{A}) + f(\mathcal{A}') \geq f(\mathcal{A} \cup \mathcal{A}') + f(\mathcal{A} \cap \mathcal{A}')$;

Let $G(\mathcal{V}, \mathcal{E})$ be a graph. A path $\mathcal{P}$ in G is an ordered sequence of non-repeated vertices. As a shorthand, we use $\mathcal{P}$ to denote both the path (ordered set) as well as the unordered set of vertices along the path. Let $\mathcal{T} = 2^{\mathcal{V}}$ denote the power set of $\mathcal{V}$. Intuitively, $\mathcal{T}$ is the superset of all possible sets of vertices that a robot may visit along its path. The cost of a path $\mathcal{P}$, denoted by $C(\mathcal{P})$, is the sum of the edge weights along the path. We assume that the edge weights are metric. We study the *rooted* version of the problem where the path for robot i, denoted by $\mathcal{P}_i$, must begin at a specific vertex v_{s_i}.

We consider the case that the *reward function*, $g(\mathcal{P}) : \mathcal{T} \rightarrow \mathbb{R}_+$, of a single robot is a monotone submodular function. We also study the special version where the function is modular (i.e., reward of a path is the sum of rewards of the vertices along the path).

Let $\mathcal{S}$ be some collection of N paths corresponding to the N robots in the team, $\mathcal{S} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N\}$. The team reward collected by any subset $\mathcal{S}' \subseteq \mathcal{S}$ is given by,

$$f(\mathcal{S}') = g\left(\bigcup_{\mathcal{P}_i \in \mathcal{S}'} \mathcal{P}_i\right). \tag{1}$$

Note that reward function of the team $f(\mathcal{S}')$ is a submodular function irrespective of whether the single robot reward function $g(\mathcal{P}_i)$ is submodular or not. Multiple robots can visit the same vertex but only one visit is accounted for when computing the reward of the team. That is, there can be no double counting of the rewards.

We are now ready to formally define our problem.

Problem 1 (RMOP)**.** *Given a metric graph $G(\mathcal{V}, \mathcal{E})$, N robots with starting positions $\{v_{s_1}, v_{s_1}, \ldots, v_{s_N}\}$, budget constraint B, a single robot reward function $g(\mathcal{P}) : \mathcal{T} \rightarrow \mathbb{R}_+$, and a team reward function f as defined in Equation 1, the Robust Multiple-Path Orienteering Problem seeks to find a collection of N paths, $\mathcal{S} = \{\mathcal{P}_1, \ldots, \mathcal{P}_N\}$ that are robust to the worst-case failure of α robots:*

$$\begin{aligned} &\max_{\mathcal{S} \subseteq \mathcal{T}} \min_{\mathcal{A} \subseteq \mathcal{S}} f(\mathcal{S} \setminus \mathcal{A}) \\ \textit{s.t.} \quad &|\mathcal{A}| \leq \alpha, 0 < \alpha < N \\ &|\mathcal{S}| = N \\ &C(\mathcal{P}_j) \leq B. \end{aligned} \tag{2}$$

where additionally v_{s_j} must be the starting vertex when constructing a path $\mathcal{P}_j$ for robot j.

The RMOP can be interpreted as a two-stage perfect information sequential game, where the first player (i.e., the team of robots) chooses a set $\mathcal{S}$, and the second player (i.e., the adversary), knowing $\mathcal{S}$, chooses a subset $\mathcal{A}$ to remove from $\mathcal{S}$. We seek worst-case guarantees — in practice, the adversary may not know the paths for each robot. By playing against this stronger adversary, we guarantee that the performance against a weaker one will be even better. We evaluate this empirically by considering attack models other than the worst one.

The adversarial model considered in this paper is the same as that in prior work on robust submodular optimization [22], [23], [25]. However, RMOP is even harder since even at the single robot level, the optimization problem we need to solve (i.e., OP) is NP-Hard. Nevertheless, we present a constant-factor approximation algorithm for this problem next.

III. Algorithm for RMOP

In this section, we present the general algorithm to solve RMOP (Algorithm 1). The algorithm uses a generic subroutine for solving OP. In the next section, we show examples of three subroutines that can be used and show how they affect the performance of the algorithm.

Before we describe the algorithm, we present additional notation. If $\mathcal{S}'$ is a set of $N' \leq N$ paths, then let $\mathcal{R}(\mathcal{S}')$ denote the set of corresponding N' robots whose paths are contained in $\mathcal{S}'$. We use $\mathcal{A}^*(\mathcal{S}) \triangleq \arg\min_{\mathcal{A}} f(\mathcal{S} \setminus \mathcal{A})$ to denote the worst-case set of paths that are removed from a given set of path $\mathcal{S}$. Therefore, $\mathcal{S} \setminus \mathcal{A}^*(\mathcal{S})$ denotes the set of paths that are not attacked from $\mathcal{S}$ with $|\mathcal{A}^*(\mathcal{S})| \leq \alpha$.

The algorithm consists of two main steps: first, it calls a subroutine for solving OP N times to compute a path for each robot independently. It then chooses α paths (denoted by $\mathcal{S}_1$) with highest individual rewards without considering overlap with other robots; Second, it uses sequential greedy assignment to find paths for the rest of the robots (denoted by $\mathcal{S}_2$) by querying the OP subroutine $N - \alpha$ times. The **while** loop is used to maintain an invariant that the paths in $\mathcal{S}_1$ are always better than the paths in $\mathcal{S}_2$.

As described earlier, there is a tradeoff between redundancy and coverage in RMOP. The two set of paths are constructed so that $\mathcal{S}_1$ adds redundancy and $\mathcal{S}_2$ adds coverage, together yields a provably good solution for RMOP. We explain each step in Algorithm 1 next.

Constructing $\mathcal{S}_1$: Each of the α paths in $\mathcal{S}_1$ are better than those in $\mathcal{S}_2$. The paths in $\mathcal{S}_1$ may overlap with each other and also overlap with those in $\mathcal{S}_2$. Thus, these paths serve to add redundancy to the team. Constructing the best α paths with respect to f itself is NP-hard. Therefore, Algorithm 1 firstly solves orienteering problem for each robot independently and stores in $\mathcal{M}$ the (approximately optimal) paths for individual robots (lines 2–5). Then Algorithm 1 sorts the paths in $\mathcal{M}$ based on their collected rewards (line 8) and chooses the α best paths to be $\mathcal{S}_1$ (line 9).

Constructing $\mathcal{S}_2$: After finding $\mathcal{S}_1$, Algorithm 1 needs to find the best paths for the rest of robots $\mathcal{R} \setminus \mathcal{R}(\mathcal{S}_1)$. Unlike $\mathcal{S}_1$, here the algorithm explicitly considers overlap when finding the paths. Thus, $\mathcal{S}_2$ serves to add coverage to the solution.

However, selecting optimal paths for $\mathcal{R} \setminus \mathcal{R}(\mathcal{S}_1)$ is a multiple-path orienteering problem and is also NP-hard. Therefore, Algorithm 1 approximates the solution by employing the

Algorithm 1: Algorithm for Problem 1

Input : Per problem 1 requires following inputs:
- set of robots $\mathcal{R} = \{1, \ldots, N\}$
- metric graph G
- starting vertices $\{v_{s_1}, v_{s_2}, \ldots, v_{s_N}\}$
- number of maximum potential attacks α and budget B

Output: Set $\mathcal{S}$ of paths for each robot

1 $\mathcal{S}_1 \leftarrow \emptyset, \mathcal{S}_2 \leftarrow \emptyset, \mathcal{M} \leftarrow \emptyset$
2 **for** $i \leftarrow 1$ **to** N **do**
3 $\quad \mathcal{P}_i \leftarrow OP(G, v_{s_i}, B)$
4 $\quad \mathcal{M} \leftarrow \mathcal{M} \cup \{\mathcal{P}_i\}$
5 **end**
6 *flag* $\leftarrow$ True
7 **while** *flag* **do**
8 $\quad$ Sort elements in $\mathcal{M}$ such that $\tilde{\mathcal{M}} = \{\mathcal{P}'_1, \mathcal{P}'_2, \ldots, \mathcal{P}'_N\}$ and $f(\{\mathcal{P}'_1\}) \geq f(\{\mathcal{P}'_2\}) \geq \ldots \geq f(\{\mathcal{P}'_N\})$
9 $\quad \mathcal{S}_1 \leftarrow \{\mathcal{P}'_1, \mathcal{P}'_2, \ldots, \mathcal{P}'_\alpha\}$
10 $\quad$ //extract starting positions for the rest of robots $\tilde{v}_s \leftarrow \{v_{s_j} | \forall j \in \mathcal{R} \setminus \mathcal{R}(\mathcal{S}_1)\}$
11 $\quad$ //Sequential greedy assignment $\mathcal{S}_2 \leftarrow SGA(G, \tilde{v}_s, B)$
12 $\quad$ //while loop control
13 $\quad$ **if** $f(\mathcal{P}_i) \geq f(\mathcal{P}_j), \forall \mathcal{P}_i \in \mathcal{S}_1, \mathcal{P}_j \in \mathcal{S}_2$ **then**
14 $\quad\quad flag \leftarrow$ False
15 $\quad$ **else**
16 $\quad\quad$ Find all robots $j \in \mathcal{R}(\mathcal{S}_2)$ such that
17 $\quad\quad \exists i \in \mathcal{R}(\mathcal{S}_1), f(\{\mathcal{P}_j \in \mathcal{S}_2\}) > f(\{\mathcal{P}_i \in \mathcal{S}_1\})$
18 $\quad\quad$ Replace the path stored in $\mathcal{M}$ corresponding to robot j with the better path found when constructing $\mathcal{S}_2$
19 $\quad$ **end**
20 **end**
21 $\mathcal{S} \leftarrow \mathcal{S}_1 \cup \mathcal{S}_2$

sequential greedy algorithm (line 11). For completeness, we present the pseudocode for SGA in Algorithm 2.

Specifically, for robots in $\mathcal{R} \setminus \mathcal{R}(\mathcal{S}_1)$, Algorithm 2 finds a path using an approximation algorithm for OP (line 4). Then, Algorithm 2 sets the reward for the vertices visited by that robot to be zero (lines 6–8). This process repeats until we find a path for all robots. Here we implicitly assume that there is at least one path for each robot satisfying the budget constraints.

The paths in $\mathcal{S}_1 \cup \mathcal{S}_2$ form the solution to RMOP. However, we also have an outer **while** loop which we explain next.

Invariant: Our analysis requires the paths in $\mathcal{S}_1$ and $\mathcal{S}_2$ to have the following property: $f(\{\mathcal{P}_i\}) \geq f(\{\mathcal{P}_j\}), \forall \mathcal{P}_i \in \mathcal{S}_1, \mathcal{P}_j \in \mathcal{S}_2$. This condition is trivially met if the single robot problem has to just choose the best amongst a fixed set of trajectories as in the prior work [22], [25]. However, when solving RMOP, we employ a subroutine for solving OP which gives us the paths in $\mathcal{S}_1$ and $\mathcal{S}_2$. Since the subroutine uses an approximation algorithm for OP instead of an exact optimal one, we cannot guarantee that this invariant holds. For example, if the subroutine uses randomness, then running the same algorithm twice may give different results. In any case, all we can guarantee is that the paths found by the subroutine will be no more than a constant from the optimal.

We fix this problem by utilizing a **while** loop (lines 7–0). When the condition of the **while** loop holds (lines 13–15), the loop flag is set to be false and the while loop terminates. Otherwise (lines 16–19), Algorithm 1 will find those robots that violate the above inequality and update their corresponding paths in the set $\mathcal{M}$. Recall that $\mathcal{M}$ is used to store the best path corresponding to each robot. Then, while loop will restart to construct $\mathcal{S}_1$ using the updated $\mathcal{M}$ and $\mathcal{S}_2$ for the remaining robots, again. We show that this loop will eventually terminate.

Corollary 1. *The **while** loop in Algorithm 1 will terminate in a finite number of steps.*

Proof. If the flag is not set to false after an iteration of the **while** loop, then it must mean that at least one new path found when constructing $\mathcal{S}_2$, say for robot j, is better than some path in $\mathcal{S}_1$. Suppose this better path is $\mathcal{P}'_j$. Note that the set $\mathcal{S}$ consists of a path for the robot j, say $\mathcal{P}_j$. Since $\mathcal{S}_1$ consists of the best α paths in $\mathcal{M}$ and $j \notin \mathcal{S}_1$, then it must mean that the path $\mathcal{P}'_j$ is strictly better than $\mathcal{P}_j$. Thus, after every iteration of the **while** loop, if the flag is not set, then at least one path in $\mathcal{M}$ has improved. For each robot given a fixed budget, there is a maximum amount of reward that it can collect. We cannot keep increasing the rewards of paths in $\mathcal{M}$. Therefore, while loop must terminate after finite iterations. ■

Algorithm 2: Sequential Greedy Assignment

1 function SGA (G, v_s, B);
Input :
- A graph G representing environment
- Budget B for each robot
- Starting positions v_s

Output: a collection $\mathcal{A}$ of paths

2 $\mathcal{A} \leftarrow \emptyset, G' \leftarrow G, N \leftarrow length(v_s)$
3 **for** $j \leftarrow 1$ **to** N **do**
4 $\quad \mathcal{P}_j \leftarrow OP(G', v_{s_j}, B)$;
5 $\quad \mathcal{A} \leftarrow \mathcal{A} \cup \{\mathcal{P}_j\}$
6 $\quad$ **foreach** $v \in \mathcal{P}_j$ **do**
7 $\quad\quad$ Set the reward of $v \in G'$ to be zero
8 $\quad$ **end**
9 **end**
10 return $\mathcal{A}$

Remark 1. In practice, the loop in Algorithm 1 typically terminates after just one iteration. Paths in $\mathcal{S}_1$ are found without considering overlap. On the other hand, when solving SGA the robots find their paths by taking into account overlap with the previously found paths. The conditions in the latter are a subset of the former. Furthermore, none of the three subroutines that we employ for OP include any randomness.

Therefore, it is unlikely that the paths in $\mathcal{S}_2$ will be better than that in $\mathcal{S}_1$. As such, it is unlikely that the **while** loop will take more than one iteration. Nevertheless, we give the full algorithm for completeness.

So far, we have not discussed the subroutine used to solve OP. In the next section, we present the analysis of the algorithm and then present the three subroutines.

IV. Performance Analysis

In this section, we quantify the performance of proposed Algorithm 1. We first present a new analysis for the Sequential Greedy Assignment (SGA) and then show the performance bound for our algorithm. The performance is based on the notion of curvature of the set functions.

Definition 3 (Curvature)**.** Consider a finite ground set $\mathcal{V}$ and a monotone submodular set function $h : 2^{\mathcal{V}} \mapsto \mathbb{R}$. The curvature of h is defined as,

$$k_h = 1 - \min_{v \in \mathcal{V}} \frac{h(\mathcal{V}) - h(\mathcal{V} \setminus v)}{h(v)} \tag{3}$$

The curvature takes values $0 \leq k_h \leq 1$ and measures how far h is from modularity. When $k_h = 0$, h is modular since for all $v \in \mathcal{V}$, we get $h(\mathcal{V}) - h(\mathcal{V} \setminus \{v\}) = h(v)$. On the other extreme, when $k_h = 1$ there exists some element v that makes no unique contribution to the rest of the set, since we get $h(\mathcal{V}) = h(\mathcal{V} \setminus \{v\})$. We assume that the curvature k_g of the reward function and that k_f of objective function is strictly less than 1. This is reasonable since it implies every vertex and path in the environment makes some non-zero unique contribution over the rest.

We first analyze the SGA and then use that analysis for proving the performance bound of our algorithm.

A. Sequential Greedy Assignment

SGA was first proposed in [19] to solve the MOP. Note that the MOP is the same as RMOP if we consider $\alpha = 0$. SGA solves the problem by finding the path for the i^{th} robot in the i^{th} iteration, by considering the paths found in the previous $i-1$ iterations.

Let $\mathcal{Q}^* = \{\mathcal{Q}_1^*, \mathcal{Q}_2^*, \ldots, \mathcal{Q}_M^*\}$ be the optimal solution to the MOP with M robots. It is easy to see that these paths will be non-overlapping in a metric graph.

Proposition 1. *There exists an optimal solution for the MOP consisting of no overlapping paths, i.e., $\mathcal{Q}_i^* \cap \mathcal{Q}_j^* = \emptyset, \forall i \neq j$.*

The proof is given in the supplementary document.

We use an approximation algorithm to find the path for each robot. Let $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_M\}$ be the set of paths returned by SGA. Note that SGA runs for M iterations. Let $\mathcal{X}_{1:j}$ denote the set $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_j\}$, which is the collection of paths returned by SGA after the first j iterations. We have $\mathcal{X}_{1:0} = \emptyset$. With slight abuse of notation, we use $\mathcal{X}_{1:j}$ to refer to the (unordered set of) vertices visited by the paths in $\mathcal{X}_{1:j}$.

Every iteration of SGA requires solving an NP-Hard problem (c.f. line 4 in Algorithm 2). Let $\mathcal{P}_j^*$ be the optimal solution for the problem in iteration j, i.e.,

$$\mathcal{P}_j^* = \text{argmax}_{\pi_j \in \Pi_j} f_{\mathcal{X}_{1:j-1}}(\pi_j)$$

where Π_j is the set of all feasible paths for robot j and $f_X(\mathcal{P}) \triangleq f(X \cup \mathcal{P}) - f(X)$. Let $\mathcal{O}_j$ be the set $\{\mathcal{P}_1^*, \mathcal{P}_2^*, \ldots, \mathcal{P}_j^*\}$, which is the collection of optimal paths for the problems in iterations 1 through j. We also set $\mathcal{O}_0 = \emptyset$.

The analysis in [19] gives the relation between $f(\mathcal{X}_{1:M})$ and $f(\mathcal{O}_M)$ which is not necessarily be the same as $f(\mathcal{Q}^*)$. In fact, the underling relationship between $f(\mathcal{O}_M)$ and global optimal $f(\mathcal{Q}^*)$ is unclear. Recall that $\mathcal{Q}^*$ is the optimal set of M paths for MOP and $\mathcal{O}_M$ is the set of M paths where the j^{th} path is the optimal solution to the problem in iteration j of SGA. These are not necessarily the same. One way to see this is to note that the order of the robots in SGA is arbitrary. If we shuffle the order in which the robots select their paths, then the paths found for each robot as well as the per stage optimal paths $\mathcal{O}_M$ will change but the optimal solution for MOP $\mathcal{Q}^*$ will still be the same. In the following, we will directly establish the relation between $f(\mathcal{X}_{1:M})$ and $f(\mathcal{Q}^*)$. Our proof uses ideas from [5].

For now, assume that we use an η approximation for OP,

$$f_{\mathcal{X}_{1:j-1}}(\mathcal{P}_j) \geq \frac{1}{\eta} f_{\mathcal{X}_{1:j-1}}(\mathcal{P}_j^*).$$

Theorem 1. *Algorithm 2 (SGA) gives a $\frac{1+\eta}{1-k_g}$ approximation for MOP, where η is the approximation factor for OP and k_g is the curvature for the reward function $g(\cdot)$.*

Proof. Let $\Delta_i \triangleq \mathcal{X}_{1:i-1} \cap \mathcal{Q}_i^*$ be the set of vertices visited by both the optimal path for the robot i and paths found using SGA for robots 1 through $i-1$. Let $\Delta \triangleq \bigcup_i \Delta_i$. When we construct a path for robot i in the i-th iteration, there is a feasible path for robot i that visits all vertices in $\mathcal{Q}_i^* \setminus \Delta_i$. That is, if we remove the vertices from the optimal path $\mathcal{Q}_i^*$ for robot i that are also in $\mathcal{X}_{1:i-1}$, the remaining vertices in $\mathcal{Q}_i^*$ still form a feasible path for robot i (since it cannot be longer). Therefore, the optimal path $\mathcal{P}_i^*$ in the iteration i for robot i should satisfy

$$\begin{aligned} f_{\mathcal{X}_{1:i-1}}(\{\mathcal{P}_i^*\}) &\geq f_{\mathcal{X}_{1:i-1}}(\{\mathcal{Q}_i^* \setminus \Delta_i\}) \\ &= f(\mathcal{X}_{1:i-1} \cup \{(\mathcal{Q}_i^* \setminus \Delta_i)\}) - f(\mathcal{X}_{1:i-1}) \\ &= g(\cup_{j=1}^{i-1} \mathcal{P}_j \cup (\mathcal{Q}_i^* \setminus \Delta_i)) - g(\cup_{j=1}^{i-1} \mathcal{P}_j) \end{aligned} \tag{4}$$

Given two sets $\mathcal{Y}, \mathcal{Z} \subseteq \Pi$, using the inequality presented in the footnote in [27], we have

$$g(\mathcal{Z} \cup \mathcal{Y}) - g(\mathcal{Z}) + \sum_{j \in \mathcal{Y} \cap \mathcal{Z}} g_{\mathcal{Z} \cup \mathcal{Y} \setminus \{j\}}(j) \geq (1 - k_g) g(\mathcal{Y}). \tag{5}$$

Note that $\cup_{j=1}^{i-1} \mathcal{P}_j \cap \{\mathcal{Q}_i^* \setminus \Delta_i\} = \emptyset$. Applying inequality (5) to (4) and using submodularity, we have

$$f_{\mathcal{X}_{1:i-1}}(\{\mathcal{P}_i^*\}) \geq f_{\mathcal{X}_{1:i-1}}(\{\mathcal{Q}_i^* \setminus \Delta_i\}) \tag{6}$$

$$\geq (1 - k_g) g(\mathcal{Q}_i^* \setminus \Delta_i) \tag{7}$$

$$\geq (1 - k_g)(g(\mathcal{Q}_i^*) - g(\Delta_i)). \tag{8}$$

Using the η approximation algorithm for OP, we have

$$f_{\mathcal{X}_{1:i-1}}(\{\mathcal{P}_i\}) \geq \frac{1}{\eta}(1-k_g)(g(\mathcal{Q}_i^*) - g(\Delta_i)).$$

Summing over all i, we get

$$\sum_{i=1}^{M} \frac{\eta}{1-k_g} f_{\mathcal{X}_{1:i-1}}(\{\mathcal{P}_i\}) \geq \sum_{i=1}^{M} g(\mathcal{Q}_i^*) - \sum_{i=1}^{M} g(\Delta_i). \quad (9)$$

The left hand side is equal to $\frac{\eta}{1-k_g} f(\mathcal{X}_{1:M})$ by definition. Furthermore, by submodularity,

$$\sum_{i=1}^{M} g(\mathcal{Q}_i^*) \geq g(\bigcup_{i=1}^{M} \mathcal{Q}_i^*). \quad (10)$$

We also have,

$$g(\Delta) \geq (1-k_g) \sum_{\delta \in \Delta} g(\{\delta\}) \quad (11)$$

$$= (1-k_g) \sum_{i=1}^{M} \sum_{\delta \in \Delta_i} g(\{\delta\}) \quad (12)$$

$$\geq (1-k_g) \sum_{i=1}^{M} g(\Delta_i) \quad (13)$$

where Eq. (11) holds due to Lemma 1 in [22]; Eq. (12) follows from Proposition 1 that states $\Delta_i \cap \Delta_j = \emptyset, \forall i \neq j$; Eq. (13) is due to submodularity of g. Rearranging the terms and using the definition of Δ,

$$-\sum_{i=1}^{M} g(\Delta_i) \geq -\frac{1}{1-k_g} g(\bigcup_{i=1}^{M} \Delta_i). \quad (14)$$

Since $\Delta_i \in \mathcal{X}_{1:i-1}$, by monotonicity we have

$$g(\bigcup_{i=1}^{M} \Delta_i) \leq g(\bigcup_{i=1}^{M} \mathcal{X}_{1:i-1}) \leq g(\mathcal{X}_{1:M}) = g(\bigcup_{i=1}^{M} \mathcal{P}_i). \quad (15)$$

Using Eq. (10), (14), and (15), we have,

$$\frac{\eta}{1-k_g} f(\mathcal{X}_{1:M}) \geq g(\bigcup_{i=1}^{M} \mathcal{Q}_i^*) - \frac{1}{1-k_g} g(\bigcup_{i=1}^{M} \mathcal{P}_i).$$

By definition of the objective function f,

$$\frac{\eta}{1-k_g} f(\mathcal{X}_{1:M}) \geq g(\bigcup_{i=1}^{M} \mathcal{Q}_i^*) - \frac{1}{1-k_g} g(\bigcup_{i=1}^{M} \mathcal{P}_i)$$

$$= f(\mathcal{Q}^*) - \frac{1}{1-k_g} f(\mathcal{X}_{1:M}).$$

That is, $f(\mathcal{X}_{1:M}) \geq \frac{1-k_g}{1+\eta} f(\mathcal{Q}^*)$. ■

We now use this result in proving our main result.

B. Analysis for Algorithm 1

Theorem 2. *Algorithm 1 returns a set $\mathcal{S}$ such that*

$$f(\mathcal{S} \setminus \mathcal{A}^*(S)) \geq \frac{\max[1-k_f, \frac{1}{\alpha+1}, \frac{1}{N-\alpha}]}{\frac{1+\eta}{1-k_g}} f^*$$

where η, k_g are the same as that defined in Theorem 1; k_f is the curvature of objective function f; and $\mathcal{A}^*(S)$ is the optimal removal set of $\mathcal{S}$; and f^* is the optimal solution to RMOP.

The omitted proofs can be found in here.

Now, we describe the three subroutines that can be employed for solving OP. We start with the most general case where the reward function g is submodular and the budget for each robot must be strictly enforced.

Corollary 2. *If recursive greedy algorithm [18] is used as subroutine to solve OP and additionally each robot has a predefined terminal vertex, then η in Theorem 2 equals to $\log(OPT)$. Here OPT is the reward collected by the optimal algorithm. The running time of the resulting algorithm is quasi-polynomial since the running time of recursive greedy is quasi-polynomial.*

Next, consider the variant where g is still submodular, but each robot is allowed to exceed its predefined budget by a bounded amount.

Corollary 3. *If General Cost-Benefit (GCB) approximation algorithm [28] is used as subroutine for OP and we are allowed to relax given budget to $\frac{\psi(n)K_c}{\beta(1+\beta(K_c-1)(1-k_c))}B$, then η in Theorem 2 equals to $2(1-e^{-1})^{-1}$. Here, $\psi(n), \beta, K_c, k_c$ as defined in [28]. The GCB algorithm runs in polynomial time.*

Finally, consider the case where g is modular. Here, we get the strongest guarantee with no relaxations to RMOP.

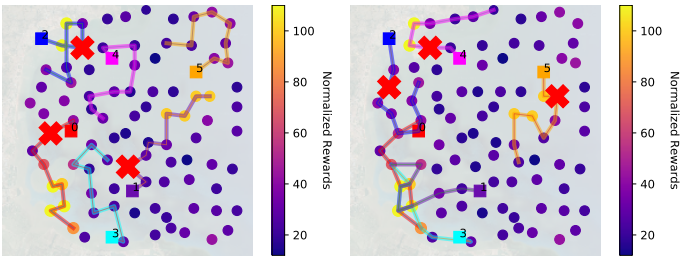
Corollary 4. *If the reward function g is modular, then by using the approximation algorithm for OP in [5] yields an $\eta = 4$ in Theorem 2. The running time of the algorithm in [5] is polynomial.*

V. Numerical Simulations

In this section, we validate the performance of Algorithm 1 through numerical simulations. In particular, (1) we compare the performance of our algorithm with two baseline strategies; (2) demonstrate the robustness of the proposed algorithm against attacks that are not necessarily the worst-case ones; and (3) investigate the running time as a function of the size of the input graph and number of robots. All experiments were performed on a Windows 64-bit laptop with 16 GB RAM and an 8-core Intel i5-8250U 1.6GHz CPU using Python 3.7.

A. Simulation Setup

Application case study: We use the application of monitoring a marine environment for mapping oil leaks, macroalgal blooms, or pH values. Specifically, as explained in [29], such tasks usually calls for collaboration of multiple sensors including satellites, which can provide coarse prior information



(a) SGA paths (b) RMOP paths

Fig. 2. Case study of monitoring macroalgal blooms using $N = 6$ robots assuming $\alpha = 3$ failures. Colored dots indicate locations to be monitored along with their importance (i.e., rewards). Red crosses indicates worst-case attacks found using brute-force. Paths returned by the proposed algorithm manages to cover one of the three important areas (lower left corner) while SGA loses all three. The background map is part of Yellow Sea, where green tides prevails every summer since 2007.

on the concentration of the phenomenon of interest, and mobile robotic sensors, which can use the prior information for targeted data collection. Using this as motivation, we consider a scenario where prior information from satellites (for example) can be used to define an importance map over the environment to be monitoring. Fig. 2 shows the setup which consists of 96 vertices placed in the environment. The color of the vertex reflects the importance of that vertex which gives the reward associated with visiting that vertex. Here, the single robot function, $g(\mathcal{P}_i)$, is a modular reward. The cost along the edges is the Euclidean distance between the vertices. Assuming unit speed of travel, the cost of a path reflects the travel time of the robot. In all the instances, each robot is given a budget $B = 60$ units.

Baseline algorithms: Since we introduce RMOP in this paper, there is no other efficient algorithm to directly compare the performance with. One option is to compute the optimal solution (using for example, brute-force enumeration) which quickly becomes intractable. Instead, we choose two approximation algorithms for MOP, the non-adversarial version, as baselines. The first one uses the sequential greedy assignment for all N robots (we refer to it as SGA) where the path for robot i is based on the paths computed for robots 1 through $i-1$. The second baseline is the naive greedy algorithm where each robot naively (without considering the travel cost) and greedily (without considering other robots) maximize their rewards (we refer to it as NG).

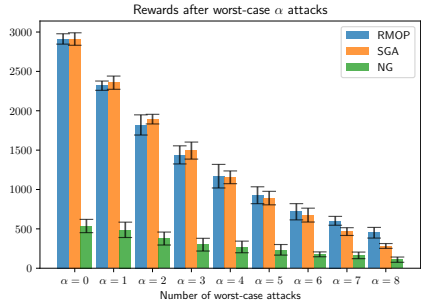
For both SGA and the proposed algorithm, we use the GCB algorithm solving OP due to its efficiency and ease of implementation. Specifically, we implement GCB using details provided in [28]. When running GCB, we simply set the relaxed budget itself to be B.

Attack models: Our algorithm is designed to give performance guarantees against worst-case attacks. However, in practice, we would like for any algorithm to be robust to not just the worst-case attacks but also other attacks. Therefore, we evaluate two other types of attacks besides worst-case attacks. The details are provided in the next subsection.

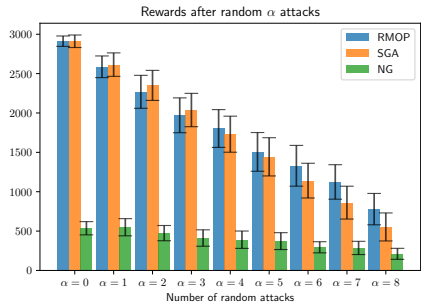
B. Results

Fig. 2 shows a qualitative example comparing our proposed algorithm and SGA with $N = 6$ and $\alpha = 3$. Not surprisingly, the six paths found by SGA does not have any overlap but the ones found by our algorithm does. As a result, the worst-case attack takes away all three robots covering the important regions in SGA, whereas one of the three regions is still covered with our algorithm. The worst-case attacks were computed using brute-force.

Next, we present quantitative results. In all following figures, the errorbar shows the variance of 20 trials where the starting robot positions are randomly chosen.



(a)



(b)

Fig. 3. (a) Rewards after worst-case attack with increasing α and $N = 10$. (b) Rewards after random α attacks and $N = 10$.

Fig. 3a shows the comparison between our algorithm for RMOP with SGA and NG as α increases with $N = 10$. The bars show the rewards collected by the robots after attacks. Our algorithm returns paths that are slightly worse than SGA when α is small. This is not surprising since our algorithm will have overlapping paths whereas SGA will not. NG is the other extreme since each robot plans for itself which can lead to high degree of overlap. As α increases, our algorithm outperforms SGA. For example, when $\alpha = 8$ our algorithm yields a reward of 451 whereas SGA only yields 283 on average.

Next, we evaluate the performance of our algorithm when the attack model does not match the worst-case one assumed during planning. The goal is to verify the robustness of the algorithm to other attack models. Fig. 3b shows the comparison between our algorithm and the two baselines as α varies when the attacked robots are randomly chosen. Our algorithm still plans assuming worst-case attacks. We observe

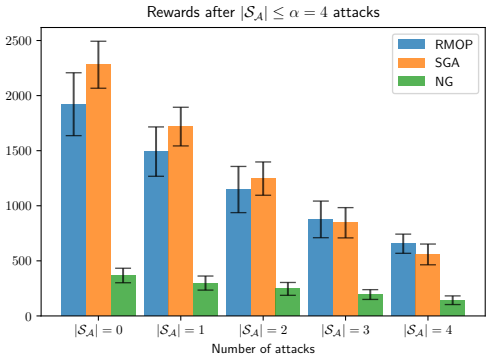
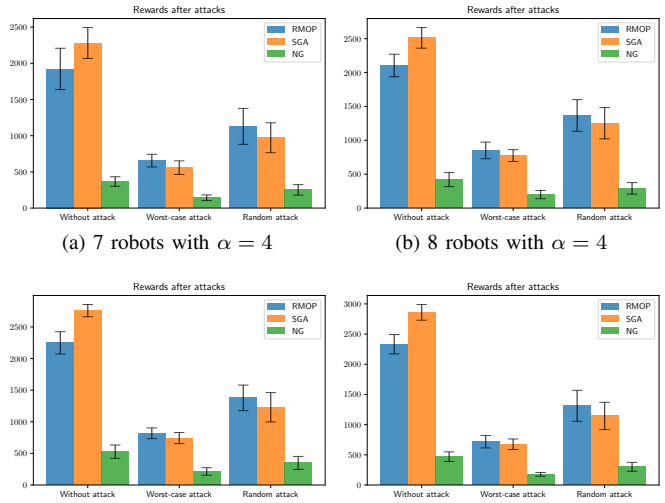


Fig. 4. Rewards after worst-case attacks of increasing sizes, $|\mathcal{S}_\mathcal{A}| \leq \alpha$. Here the planner uses $N = 7$ and $\alpha = 4$.

the same trend with random attacks as with the worst-case ones — as α increases, our algorithm outperforms SGA.

Fig. 4 shows results for the case where we construct paths assuming $\alpha = 4$ robots will be attacked but in practice only $|\mathcal{S}_\mathcal{A}| \leq \alpha$ robots suffer from worst-case attacks. SGA performs better than our algorithm when the number of robots actually attacked $|\mathcal{S}_\mathcal{A}|$ is far from the designed value of α. As the actual number of robots attacked increases and $|\mathcal{S}_\mathcal{A}|$ approaches α, our algorithm outperforms SGA.

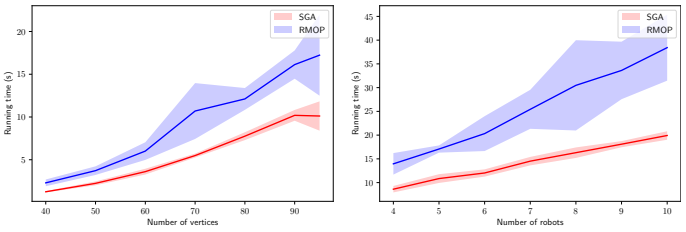


(a) 7 robots with $\alpha = 4$ (b) 8 robots with $\alpha = 4$

(c) 9 robots with $\alpha = 5$ (d) 10 robots with $\alpha = 6$

Fig. 5. Rewards after: (1) no attacks; (2) α out of N robots under worst-case attack; (3) α out of N robots under random attack.

Fig. 5 shows the evaluation when there are (1) no attacks; (2) worst-case attacks; and (3) random attacks for four configurations of N and α. In all three cases, our algorithm still plans the paths assuming worst-case attacks for the given value of α. When there are no attacks (first set of bars in each subfigure), SGA outperforms our algorithm as observed in previous charts. When worst-case attacks do happen (middle set of bars), the average rewards collected by the unattacked robots employing our algorithm is better than that of SGA. This is also the case when the α attacked robots are chosen randomly (third set of bars). This trend holds for various values of N and α as shown.



(a) Running time w.r.t. vertices (b) Running time w.r.t. robots

Fig. 6. Running time of Algorithm 1 and SGA

The running time comparisons between our algorithm and SGA are shown in Fig. 6. We vary the number of robots as well as the size of the graph. Our algorithm takes longer than SGA which is expected since it uses SGA as a subroutine. Nevertheless, we observe similar trends in the runtime.

Discussion of Results: The results show the proposed algorithm works in practice as intended. As the number of attacked robots increases (either $\mathcal{S}_\mathcal{A}$ or α), it outperforms SGA. Furthermore, we observe that the margin between our algorithm and SGA increases as the number of attacked robots increases. Even when our algorithm finds worse paths than SGA, they are still comparable to SGA and are significantly better than NG. We also observe that our algorithm is robust to the actual attack models — even if the attacks are not the worst-case ones, we see similar trends.

VI. Conclusion

We introduced a new problem, termed Robust Multiple-Path Orienteering Problem, in which we seek to construct a set of paths for robots such that even if a subset of robots fail, the rest of the team still performs well. We provided an approximation algorithm for RMOP, which builds on bounded approximation algorithm for OP and the sequential greedy assignment framework. We showed three variants of the general algorithm that use three different subroutines for OP and still yield a bounded approximation for RMOP. In addition to theoretical results, we presented empirical results showing that our algorithm is robust to attacks other than the worst-case ones. We also compare our performance with baseline algorithms and show that our algorithm yields better performance as more and more robots are attacked. An immediate future direction to extend our work to the online variant of RMOP in which robots can replan once they know which (if any) robots are attacked.

VII. Acknowledgement

This work is supported by the the National Science Foundation under Grant No. 1943368, and the Office of Naval Research under Grant No. N000141812829

References

[1] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011.

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

[3] D. Thakur, M. Likhachev, J. Keller, V. Kumar, V. Dobrokhodov, K. Jones, J. Wurz, and I. Kaminer, "Planning for opportunistic surveillance with multiple robots," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 5750–5757.

[4] A. Sadeghi, A. B. Asghar, and S. L. Smith, "On minimum time multi-robot planning with guarantees on the total collected reward," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2019, pp. 16–22.

[5] A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, and M. Minkoff, "Approximation algorithms for orienteering and discounted-reward tsp," in *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, Oct 2003, pp. 46–55.

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

[7] J. Carlson, R. R. Murphy, and A. Nelson, "Follow-up analysis of mobile robot failures," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 5. IEEE, 2004, pp. 4987–4994.

[8] A. Prorok, B. M. Sadler, M. Egerstedt, and V. Kumar, "Guest editorial: Special section on "foundations of resilience for networked robotic systems"," *Autonomous Robots*, vol. 43, no. 3, pp. 741–741, 2019.

[9] L. Guerrero-Bonilla, A. Prorok, and V. Kumar, "Formations for resilient robot teams," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 841–848, 2017.

[10] A. Prorok, "Redundant robot assignment on graphs with uncertain edge costs," in *Distributed Autonomous Robotic Systems*. Springer, 2019, pp. 313–327.

[11] K. Saulnier, D. Saldana, A. Prorok, G. J. Pappas, and V. Kumar, "Resilient flocking for mobile robot teams," *IEEE Robotics and Automation letters*, vol. 2, no. 2, pp. 1039–1046, 2017.

[12] M. W. Spong, "On the robust control of robot manipulators," *IEEE Transactions on automatic control*, vol. 37, no. 11, pp. 1782–1786, 1992.

[13] Y. Ting, S. Tosunoglu, and B. Fernandez, "Control algorithms for fault-tolerant robots," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. IEEE, 1994, pp. 910–915.

[14] D. Crestani and K. Godary-Dejean, "Fault tolerance in control architectures for mobile robots: Fantasy or reality?" in *CAR: Control Architectures of Robots*, 2012.

[15] A. Clark, B. Alomair, L. Bushnell, and R. Poovendran, *Submodularity in dynamics and control of networked systems*. Springer, 2016.

[16] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *Journal of Machine Learning Research*, vol. 9, no. Feb, pp. 235–284, 2008.

[17] A. Krause and C. Guestrin, "Submodularity and its applications in optimized information gathering," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 4, pp. 1–20, 2011.

[18] Chandra Chekuri and M. Pal, "A recursive greedy algorithm for walks in directed graphs," in *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, Oct 2005, pp. 245–253.

[19] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, vol. 34, pp. 707–755, 2009.

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

[21] J. B. Orlin, A. S. Schulz, and R. Udwani, "Robust monotone submodular function maximization," in *Proceedings of the 18th International Conference on Integer Programming and Combinatorial Optimization-Volume 9682*. Springer-Verlag, 2016, pp. 312–324.

[22] V. Tzoumas, K. Gatsis, A. Jadbabaie, and G. J. Pappas, "Resilient monotone submodular function maximization," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 1362–1367.

[23] V. Tzoumas, A. Jadbabaie, and G. J. Pappas, "Resilient monotone sequential maximization," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 7261–7268.

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

[25] 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, 2018.

[26] ——, "Distributed attack-robust submodular maximization for multi-robot planning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, to appear.

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

[28] H. Zhang and Y. Vorobeychik, "Submodular optimization with routing constraints," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[29] Q. Xing, D. An, X. Zheng, Z. Wei, X. Wang, L. Li, L. Tian, and J. Chen, "Monitoring seaweed aquaculture in the yellow sea with multiple sensors for managing the disaster of macroalgal blooms," *Remote Sensing of Environment*, vol. 231, p. 111279, 2019.

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