2018 IEEE Conference on Decision and Control (CDC)
Miami Beach, FL, USA, Dec. 17-19, 2018

Resilient Monotone Sequential Maximization

Vasileios Tzoumas,[1] Ali Jadbabaie,[2] George J. Pappas[3]

***Abstract*—Applications in machine learning, optimization, and control require the sequential selection of a few system elements, such as sensors, data, or actuators, to optimize the system performance across multiple time steps. However, in failure-prone and adversarial environments, sensors get attacked, data get deleted, and actuators fail. Thence, traditional sequential design paradigms become insufficient and, in contrast, *resilient* sequential designs that adapt against system-wide attacks, deletions, or failures become important. In general, resilient sequential design problems are computationally hard. Also, even though they often involve objective functions that are monotone and (possibly) submodular, no scalable approximation algorithms are known for their solution. In this paper, we provide the first scalable algorithm, that achieves the following characteristics: *system-wide resiliency*, i.e., the algorithm is valid for any number of denial-of-service attacks, deletions, or failures; *adaptiveness*, i.e., at each time step, the algorithm selects system elements based on the history of inflicted attacks, deletions, or failures; and *provable approximation performance*, i.e., the algorithm guarantees for monotone objective functions a solution close to the optimal. We quantify the algorithm's approximation performance using a notion of curvature for monotone (not necessarily submodular) set functions. Finally, we support our theoretical analyses with simulated experiments, by considering a control-aware sensor scheduling scenario, namely, *sensing-constrained robot navigation*.**

I. Introduction

Problems in machine learning, optimization, and control [1]–[9] require the design of systems in applications such as:

- *Car-congestion prediction*: Given a flood of driving data, collected from the drivers' smart-phones, which *few* drivers' data should we process at *each time of the day* to enable the accurate prediction of car traffic? [1]
- *Adversarial-target tracking*: At a flying robot, that uses on-board sensors to navigate itself, which *few* sensors should we activate *at each time step* both to maximize the robot's battery life, and to ensure its ability to track targets moving in a cluttered environment? [2]
- *Hazardous environmental-monitoring*: In a team of mobile robots, which *few* robots should we choose *at each time step* as actuators (leaders) to guarantee the team's capability to monitor the radiation around a nuclear reactor despite intro-robot communication noise? [3]

In particular, all the aforementioned applications [1]–[9] motivate the *sequential* selection of *a few* system elements, such as sensors, data, or actuators, to optimize the system performance across multiple time steps, subject to a resource constraint, such as limited battery for sensor activation. More formally, each of the above applications motivate the solution to a sequential optimization problem of the form:

$$\begin{aligned} &\max_{\mathcal{A}_1\subseteq\mathcal{V}_1} \cdots \max_{\mathcal{A}_T\subseteq\mathcal{V}_T} f(\mathcal{A}_1,\ldots,\mathcal{A}_T),\\ &\quad\text{such that:}\\ &\quad|\mathcal{A}_t| = \alpha_t, \text{ for all } t = 1,\ldots,T, \end{aligned} \tag{1}$$

where T represents the number of design steps in time; the objective function f is monotone and (possibly) submodular —submodularity is a diminishing returns property;— and the cardinality bound α_t captures a resource constraint at time t. The problem in eq. (1) is combinatorial, and, specifically, it is NP-hard [10]; notwithstanding, several approximation algorithms have been proposed for its solution, such as the greedy [11].

But in all the above critical applications, sensors can get cyber-attacked [12]; data can get deleted [13]; and actuators can fail [14]. Hence, in such failure-prone and adversarial scenarios, *resilient* sequential designs that *adapt* against denial-of-service attacks, deletions, or failures become important.

In this paper, we formalize for the first time a problem of *resilient monotone sequential maximization*, that goes beyond the traditional objective of the problem in eq. (1), and guards adaptively against real-time attacks, deletions, or failures. In particular, we introduce the following resilient reformulation of the problem in eq. (1):

$$\begin{aligned} &\max_{\mathcal{A}_1\subseteq\mathcal{V}_1} \min_{\mathcal{B}_1\subseteq\mathcal{A}_1} \cdots \max_{\mathcal{A}_T\subseteq\mathcal{V}_T} \min_{\mathcal{B}_T\subseteq\mathcal{A}_T} f(\mathcal{A}_1\setminus\mathcal{B}_1,\ldots,\mathcal{A}_T\setminus\mathcal{B}_T),\\ &\quad\text{such that:}\\ &\quad|\mathcal{A}_t| = \alpha_t \text{ and } |\mathcal{B}_t| \leq \beta_t, \text{ for all } t = 1,\ldots,T, \end{aligned} \tag{2}$$

where the number β_t represents the number of possible attacks, deletions, or failures —in general, it is $\beta_t \leq \alpha_t$. Overall, the problem in eq. (2) maximizes the function f despite real-time *worst-case* failures that compromise the consecutive maximization steps in eq. (1). Therefore, the problem formulation in eq. (2) is suitable in scenarios where there is no prior on the removal mechanism, as well as, in scenarios where protection against worst-case failures is essential, such as in expensive experiment designs, or missions of adversarial-target tracking.

[1] At the time the paper was written, the author was with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA. Currently, the author is with the Laboratory for Information & Decision Systems (LIDS), Massachusetts Institute of Technology (MIT), Cambridge, MA 02139 USA (email: vtzoumas@mit.edu).
[2] The author is with the Institute for Data, Systems and Society (IDSS), Massachusetts Institute of Technology (MIT), Cambridge, MA 02139 USA (email: jadbabai@mit.edu).
[3] The author is with the Department of Electrical and Systems Engineering (ESE), University of Pennsylvania (UPenn), Philadelphia, PA 19104 USA (email: pappasg@seas.upenn.edu).
This work was partially supported by the AFOSR Complex Networks program, by the ARL CRA DCIST W911NF-17-2-0181 program, and the Rockefeller Foundation.

In more detail, the problem in eq. (2) may be interpreted as a T-stage perfect information sequential game between two players [15, Chapter 4], namely, a "maximization" player (designer), and a "minimization" player (attacker), who play sequentially, *both observing all past actions of all players*, and with the designer starting the game. That is, at each time $t = 1, \ldots, T$, both the designer and the attacker *adapt* their set selections to the history of all the players' selections so far, and, in particular, the attacker adapts its selection also to the current (t-th) selection of the designer (since at each step t, the attacker plays after it observes the selection of the 'designer).

In sum, the problem in eq. (2) goes beyond traditional (non-resilient) optimization [16]–[20] by proposing *resilient* optimization; beyond single-step resilient optimization [21]–[23] by proposing *multi-step* (sequential) resilient optimization; beyond memoryless resilient optimization [24] by proposing *adaptive* resilient optimization; and beyond protection against non-adversarial attacks [13], [25] by proposing protection against *worst-case* attacks. Hence, the problem in eq. (2) aims to protect the system performance over extended periods of time against real-time denial-of-service attacks or failures, which is vital in critical applications, such as multi-target surveillance with teams of mobile robots [9].

Contributions. In this paper, we make the contributions:

- (*Problem definition*) We formalize the problem of *resilient monotone sequential maximization* against denial-of-service removals, per eq. (2). This is the first work to formalize, address, and motivate this problem.
- (*Solution*) We develop the first algorithm for the problem of resilient monotone sequential maximization in eq. (2), and prove it has the following properties:
  - *system-wide resiliency*: the algorithm is valid for any number of removals;
  - *adaptiveness*: the algorithm adapts the solution to each of the maximization steps in eq. (2) to the history of realized (inflicted) removals;
  - *minimal running time*: the algorithm terminates with the same running time as state-of-the-art algorithms for (non-resilient) set function optimization, per eq. (1);
  - *provable approximation performance*: the algorithm ensures for all $T \geq 1$, and for objective functions f that are monotone and (possibly) submodular —as it holds true in all aforementioned applications [1]–[9],— a solution finitely close to the optimal.

  To quantify the algorithm's approximation performance, we used a notion of curvature for monotone (not necessarily submodular) set functions.
- (*Simulations*) We conduct simulations in a variety of sensor scheduling scenarios for autonomous robot navigation, varying the number of sensor failures. Our simulations validate the benefits of our approach.

Overall, the proposed algorithm in this paper enables the resilient reformulation and solution of all the aforementioned applications [1]–[9] against worst-case attacks, deletions, or failures, over multiple design steps, and with provable approximation guarantees. All proofs can be found in the full version of this paper, located at the authors' websites.

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

II. Resilient Monotone Sequential Maximization

We formally define resilient monotone sequential maximization. We start with the basic definition of monotonicity.

Definition 1 (Monotonicity) *Consider any finite ground set $\mathcal{V}$. The set function $f : 2^{\mathcal{V}} \mapsto \mathbb{R}$ is non-decreasing if and only if for any sets $\mathcal{A} \subseteq \mathcal{A}' \subseteq \mathcal{V}$, it holds $f(\mathcal{A}) \leq f(\mathcal{A}')$.*

We define next the main problem in this paper.

Problem 1 (Resilient Monotone Sequential Maximization) *Consider the parameters: an integer T; finite ground sets $\mathcal{V}_1, \ldots, \mathcal{V}_T$; a non-decreasing set function $f : 2^{\mathcal{V}_1} \times \cdots \times 2^{\mathcal{V}_T} \mapsto \mathbb{R}$ such that, without loss of generality, it holds $f(\emptyset) = 0$ and f is non-negative; finally, integers α_t and β_t such that $0 \leq \beta_t \leq \alpha_t \leq |\mathcal{V}_t|$, for all $t = 1, 2, \ldots, T$.*

The problem of resilient monotone sequential maximization *is to maximize the objective function f through a sequence of T maximization steps, despite compromises to the solutions of each of the maximization steps; in particular, at each maximization step $t = 1, \ldots, T$ a set $\mathcal{A}_t \subseteq \mathcal{V}_t$ of cardinality α_t is selected, and is compromised by a worst-case set removal $\mathcal{B}_t$ of cardinality β_t. Formally:*

$$\max_{\mathcal{A}_1 \subseteq \mathcal{V}_1} \min_{\mathcal{B}_1 \subseteq \mathcal{A}_1} \cdots \max_{\mathcal{A}_T \subseteq \mathcal{V}_T} \min_{\mathcal{B}_T \subseteq \mathcal{A}_T} f(\mathcal{A}_1 \setminus \mathcal{B}_1, \ldots, \mathcal{A}_T \setminus \mathcal{B}_T),$$

$$\text{such that: } |\mathcal{A}_t| = \alpha_t \text{ and } |\mathcal{B}_t| \leq \beta_t, \text{ for all } t = 1, \ldots, T. \tag{3}$$

As we mentioned in this paper's Introduction, Problem 1 may be interpreted as a T-stage perfect information sequential game between two players [15, Chapter 4], a "maximization" player, and a "minimization" player, who play sequentially, both observing all past actions of all players, and with the "maximization" player starting the game. In the following paragraphs, we describe this game in more detail:

- *1st round of the game in Problem 1*: the "maximization" player selects the set $\mathcal{A}_1$; then, the "minimization" player observes $\mathcal{A}_1$, and selects the set $\mathcal{B}_1$ against $\mathcal{A}_1$;
- *2nd round of the game in Problem 1*: the "maximization" player, who already knows $\mathcal{A}_1$, observes $\mathcal{B}_1$, and selects the set $\mathcal{A}_2$, given $\mathcal{A}_1$ and $\mathcal{B}_1$; then, the "minimization" player, who already knows $\mathcal{A}_1$ and $\mathcal{B}_1$, observes $\mathcal{A}_2$, and selects the set $\mathcal{B}_2$ against $\mathcal{A}_2$, given $\mathcal{A}_1$ and $\mathcal{B}_1$.

$$\vdots$$

- *T-th round of the game in Problem 1*: the "maximization" player, who already knows the history of selections

$\mathcal{A}_1, \ldots, \mathcal{A}_{T-1}$, as well as, removals $\mathcal{B}_1, \ldots, \mathcal{B}_{T-1}$, selects the set $\mathcal{A}_T$, given $\mathcal{A}_1, \ldots, \mathcal{A}_{T-1}$ and $\mathcal{B}_1, \ldots, \mathcal{B}_{T-1}$; then, the "minimization" player, who also already knows the history of selections $\mathcal{A}_1, \ldots, \mathcal{A}_{T-1}$, as well as, removals $\mathcal{B}_1, \ldots, \mathcal{B}_{T-1}$, observes $\mathcal{A}_T$, and selects the set $\mathcal{B}_T$ against $\mathcal{A}_T$, given $\mathcal{A}_1, \ldots, \mathcal{A}_{T-1}$ and $\mathcal{B}_1, \ldots, \mathcal{B}_{T-1}$.

III. Adaptive Algorithm for Problem 1

We present the first algorithm for Problem 1, show it is adaptive, and, finally, describe the intuition behind it. The pseudo-code of the algorithm is described in Algorithm 1.

A. Intuition behind Algorithm 1

The goal of Problem 1 is to ensure a maximal value for an objective function f through a sequence of T maximization steps, despite compromises to the solutions of each of the maximization steps. In particular, at each maximization step $t = 1, \ldots, T$, Problem 1 aims to select a set $\mathcal{A}_t$ towards a maximal value of f, despite that each $\mathcal{A}_t$ is compromised by a worst-case set removal $\mathcal{B}_t$ from $\mathcal{A}_t$, resulting to f being finally evaluated at the sequence of sets $\mathcal{A}_1 \setminus \mathcal{B}_1, \ldots, \mathcal{A}_T \setminus \mathcal{B}_T$ instead of the sequence of sets $\mathcal{A}_1, \ldots, \mathcal{A}_T$. In this context, Algorithm 1 aims to fulfil the goal of Problem 1 by constructing each set $\mathcal{A}_t$ as the union of the sets $\mathcal{S}_{t,1}$, and $\mathcal{S}_{t,2}$ (line 9 of Algorithm 1), whose role we describe in more detail below:

a) Set $\mathcal{S}_{t,1}$ approximates worst-case set removal from $\mathcal{A}_t$: Algorithm 1 aims with the set $\mathcal{S}_{t,1}$ to capture the worst-case removal of β_t elements among the α_t elements that Algorithm 1 is going to select in $\mathcal{A}_t$; equivalently, the set $\mathcal{S}_{t,1}$ is aimed to act as a "bait" to an attacker that selects to remove the *best* β_t elements from $\mathcal{A}_t$ (*best* with respect to the elements' contribution towards the goal of Problem 1). However, the problem of selecting the *best* β_t elements in $\mathcal{V}_t$ is a combinatorial and, in general, intractable problem [10]. For this reason, Algorithm 1 aims to *approximate* the best β_t elements in $\mathcal{V}_t$, by letting $\mathcal{S}_{t,1}$ be the set of β_t elements with the largest marginal contributions in the value of the objective function f (lines 3-4 of Algorithm 1).

b) Set $\mathcal{S}_{t,2}$ is such that $\mathcal{S}_{t,1} \cup \mathcal{S}_{t,2}$ approximates optimal set solution to t-th maximization step of Problem 1: Assuming that $\mathcal{S}_{t,1}$ is the set of β_t elements that are going to be removed from Algorithm 1's set selection $\mathcal{A}_t$, Algorithm 1 needs next to select a set $\mathcal{S}_{t,2}$ of $\alpha_t - \beta_t$ elements to complete the construction of $\mathcal{A}_t$, since it is $|\mathcal{A}_t| = \alpha_t$ per Problem 1. In particular, for $\mathcal{A}_t = \mathcal{S}_{t,1} \cup \mathcal{S}_{t,2}$ to be an optimal solution to t-th maximization step of Problem 1 (assuming the removal of $\mathcal{S}_{t,1}$ from $\mathcal{A}_t$), Algorithm 1 needs to select $\mathcal{S}_{t,2}$ as a *best* set of $\alpha_t - \beta_t$ elements from $\mathcal{V}_t \setminus \mathcal{S}_{t,1}$. Nevertheless, the problem of selecting a *best* set of $\alpha_t - \beta_t$ elements from $\mathcal{V}_t \setminus \mathcal{S}_{t,1}$ is a combinatorial and, in general, intractable problem [10]. As a result, Algorithm 1 aims to *approximate* such a best set, using the greedy procedure in the lines 5-8 of Algorithm 1.

Overall, Algorithm 1 constructs the sets $\mathcal{S}_{t,1}$ and $\mathcal{S}_{t,2}$ to approximate an optimal solution $\mathcal{A}_t$ to the t-th maximization step of Problem 1 with their union (line 9 of Algorithm 1).

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

Algorithm 1 Adaptive algorithm for Problem 1.

Input: Per Problem 1, Algorithm 1 receives two input types:

- (*Off-line*) Integer T; finite ground sets $\mathcal{V}_1, \ldots, \mathcal{V}_T$; set function $f : 2^{\mathcal{V}_1} \times \cdots \times 2^{\mathcal{V}_T} \mapsto \mathbb{R}$ such that f is non-decreasing, non-negative, and $f(\emptyset) = 0$; integers α_t and β_t such that $0 \leq \beta_t \leq \alpha_t \leq |\mathcal{V}_t|$, for all $t = 1, \ldots, T$.
- (*On-line*) At each step $t = 2, 3, \ldots, T$: realized set removal $\mathcal{B}_{t-1}$ from Algorithm 1's set selection $\mathcal{A}_{t-1}$.

Output: At each step $t = 1, 2, \ldots, T$, set $\mathcal{A}_t$.

```
1: for all $t = 1, \ldots, T$ do
2:     $\mathcal{S}_{t,1} \leftarrow \emptyset$;   $\mathcal{S}_{t,2} \leftarrow \emptyset$;
3:     Sort elements in $\mathcal{V}_t$ such that $\mathcal{V}_t \equiv \{v_{t,1}, \ldots, v_{t,|\mathcal{V}_t|}\}$ and $f(v_{t,1}) \geq \ldots \geq f(v_{t,|\mathcal{V}_t|})$;
4:     $\mathcal{S}_{t,1} \leftarrow \{v_{t,1}, \ldots, v_{t,\beta}\}$;
5:     while $|\mathcal{S}_{t,2}| < \alpha_t - \beta_t$ do
6:         $x \in \arg\max_{y \in \mathcal{V}_t \setminus (\mathcal{S}_{t,1} \cup \mathcal{S}_{t,2})} f(\mathcal{A}_1 \setminus \mathcal{B}_1, \ldots, \mathcal{A}_{t-1} \setminus \mathcal{B}_{t-1}, \mathcal{S}_{t,2} \cup \{y\})$;
7:         $\mathcal{S}_{t,2} \leftarrow \{x\} \cup \mathcal{S}_{t,2}$;
8:     end while
9:     $\mathcal{A}_t \leftarrow \mathcal{S}_{t,1} \cup \mathcal{S}_{t,2}$;
10: end for
```

B. Description of steps in Algorithm 1

Algorithm 1 executes four steps for each $t = 1, \ldots, T$, where T is the number of maximization steps in Problem 1:

a) Initialization (line 2 of Algorithm 1): Algorithm 1 defines two auxiliary sets, namely, the $\mathcal{S}_{t,1}$ and $\mathcal{S}_{t,2}$, and initializes each of them with the empty set (line 2 of Algorithm 1). The purpose of $\mathcal{S}_{t,1}$ and of $\mathcal{S}_{t,2}$ is to construct the set $\mathcal{A}_t$, which is the set Algorithm 1 selects as a solution to Problem 1's t-th maximization step; in particular, the union of $\mathcal{S}_{t,1}$ and of $\mathcal{S}_{t,2}$ constructs $\mathcal{A}_t$ at the end of the t-th execution of the algorithm's "for loop" (lines 1-10 of Algorithm 1).

b) Construction of set $\mathcal{S}_{t,1}$ (lines 3-4 of Algorithm 1): Algorithm 1 constructs the set $\mathcal{S}_{t,1}$ such that $\mathcal{S}_{t,1}$ contains β_t elements from the ground set $\mathcal{V}_t$ and, for any element $s \in \mathcal{S}_{t,1}$ and any element $s' \notin \mathcal{S}_{t,1}$, the marginal value of $f(s)$ is at least that of $f(s')$; that is, among all elements in $\mathcal{V}_t$, the set $\mathcal{S}_{t,1}$ contains a collection of β_t elements that correspond to the highest marginal values of f. In detail, Algorithm 1 constructs $\mathcal{S}_{t,1}$ by first sorting and indexing all elements in $\mathcal{V}_t$ such that $\mathcal{V}_t = \{v_{t,1}, \ldots, v_{t,|\mathcal{V}_t|}\}$ and $f(v_{t,1}) \geq \ldots \geq f(v_{t,|\mathcal{V}_t|})$ (line 3 of Algorithm 1), and, then, by including in $\mathcal{S}_{t,1}$ the fist β_t elements among the $\{v_{t,1}, \ldots, v_{t,|\mathcal{V}_t|}\}$ (line 4 of Algorithm 1).

c) Construction of set $\mathcal{S}_{t,2}$ (lines 5-8 of Algorithm 1): Algorithm 1 constructs the set $\mathcal{S}_{t,2}$ by picking greedily $\alpha_t - \beta_t$ elements from the set $\mathcal{V}_t \setminus \mathcal{S}_{t,1}$, and by accounting for the effect that the history of set selections and removals $(\mathcal{A}_1 \setminus \mathcal{B}_1, \ldots, \mathcal{A}_{t-1} \setminus \mathcal{B}_{t-1})$ has on the objective function f of Problem 1. Specifically, the greedy procedure in Algorithm 1's "while loop" (lines 5-8 of Algorithm 1) selects an element $y \in \mathcal{V}_t \setminus (\mathcal{S}_{t,1} \cup \mathcal{S}_{t,2})$ to add in $\mathcal{S}_{t,2}$ only if y maximizes the value of $f(\mathcal{A}_1 \setminus \mathcal{B}_1, \ldots, \mathcal{A}_{t-1} \setminus \mathcal{B}_{t-1}, \mathcal{S}_{t,2} \cup \{y\})$.

d) Construction of set $\mathcal{A}_t$ (line 9 of Algorithm 1): Algorithm 1 proposes the set $\mathcal{A}_t$ as a solution to Problem 1's t-th maximization step. To this end, Algorithm 1 constructs $\mathcal{A}_t$ as the union of the previously constructed sets $\mathcal{S}_{t,1}$ and $\mathcal{S}_{t,2}$.

In sum, Algorithm 1 enables an adaptive solution of Problem 1: for each $t = 1, 2, \ldots$, Algorithm 1 constructs a solution set $\mathcal{A}_t$ to the t-th maximization step of Problem 1 based on both the history of selected solutions up to step $t-1$, namely, the sets $\mathcal{A}_1, \ldots, \mathcal{A}_{t-1}$, and the corresponding history of set removals from $\mathcal{A}_1, \ldots, \mathcal{A}_{t-1}$, namely, the $\mathcal{B}_1, \ldots, \mathcal{B}_{t-1}$.

IV. Performance Guarantees for Algorithm 1

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

A. Curvature and total curvature of non-decreasing functions

We present the notions of *curvature* and of *total curvature* for non-decreasing set functions. We start by describing the notions of *modularity* and *submodularity* for set functions.

Definition 2 (Modularity) *Consider any finite set $\mathcal{V}$. The set function $g : 2^{\mathcal{V}} \mapsto \mathbb{R}$ is modular if and only if for any set $\mathcal{A} \subseteq \mathcal{V}$, it holds $g(\mathcal{A}) = \sum_{v\in\mathcal{A}} g(v)$.*

In words, a set function $g : 2^{\mathcal{V}} \mapsto \mathbb{R}$ is modular if through g all elements in $\mathcal{V}$ cannot substitute each other; in particular, Definition 2 of modularity implies that for any set $\mathcal{A} \subseteq \mathcal{V}$, and for any element $v \in \mathcal{V}\setminus\mathcal{A}$, it holds $g(\{v\}\cup\mathcal{A}) - g(\mathcal{A}) = g(v)$.

Definition 3 (Submodularity [26, Proposition 2.1]) *Consider any finite set $\mathcal{V}$. The set function $g : 2^{\mathcal{V}} \mapsto \mathbb{R}$ is* submodular *if and only if for any sets $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$, and any element $v \in \mathcal{V}$, it holds $g(\mathcal{A}\cup\{v\}) - g(\mathcal{A}) \geq g(\mathcal{B}\cup\{v\}) - g(\mathcal{B})$.*

Definition 3 implies that a set function $g : 2^{\mathcal{V}} \mapsto \mathbb{R}$ is submodular if and only if it satisfies a diminishing returns property where for any set $\mathcal{A} \subseteq \mathcal{V}$, and for any element $v \in \mathcal{V}$, the marginal gain $g(\mathcal{A}\cup\{v\}) - g(\mathcal{A})$ is non-increasing. In contrast to modularity, submodularity implies that the elements in $\mathcal{V}$ *can* substitute each other, since Definition 3 of submodularity implies the inequality $g(\{v\} \cup \mathcal{A}) - g(\mathcal{A}) \leq g(v)$; that is, in the presence of the set $\mathcal{A}$, the element v may lose part of its contribution to the value of $g(\{x\} \cup \mathcal{A})$.

Definition 4 (Curvature of monotone submodular functions [20]) *Consider a finite set $\mathcal{V}$, and a non-decreasing submodular set function $g : 2^{\mathcal{V}} \mapsto \mathbb{R}$ such that (without loss of generality) for any element $v \in \mathcal{V}$, it is $g(v) \neq 0$. The curvature of g is defined as follows:*

$$\kappa_g \triangleq 1 - \min_{v\in\mathcal{V}} \frac{g(\mathcal{V}) - g(\mathcal{V}\setminus\{v\})}{g(v)}. \tag{4}$$

Definition 4 of curvature implies that for any non-decreasing submodular set function $g : 2^{\mathcal{V}} \mapsto \mathbb{R}$, it holds $0 \leq \kappa_g \leq 1$. In particular, the value of κ_g measures how far g is from modularity, as we explain next: if $\kappa_g = 0$, then for all elements $v \in \mathcal{V}$, it holds $g(\mathcal{V}) - g(\mathcal{V}\setminus\{v\}) = g(v)$, that is, g is modular. In contrast, if $\kappa_g = 1$, then there exist an element $v \in \mathcal{V}$ such that $g(\mathcal{V}) = g(\mathcal{V} \setminus \{v\})$, that is, in the presence of $\mathcal{V} \setminus \{v\}$, v loses all its contribution to the value of $g(\mathcal{V})$.

Definition 5 (Total curvature of non-decreasing functions [27, Section 8]) *Consider a finite set $\mathcal{V}$, and a monotone set function $g : 2^{\mathcal{V}} \mapsto \mathbb{R}$. The total curvature of g is defined as follows:*

$$c_g \triangleq 1 - \min_{v\in\mathcal{V}} \min_{\mathcal{A},\mathcal{B}\subseteq\mathcal{V}\setminus\{v\}} \frac{g(\{v\}\cup\mathcal{A}) - g(\mathcal{A})}{g(\{v\}\cup\mathcal{B}) - g(\mathcal{B})}. \tag{5}$$

Definition 5 of total curvature implies that for any non-decreasing set function $g : 2^{\mathcal{V}} \mapsto \mathbb{R}$, it holds $0 \leq c_g \leq 1$. To connect the notion of total curvature with that of curvature, we note that when the function g is non-decreasing and submodular, then the two notions coincide, i.e., it holds $c_g = \kappa_g$; the reason is that if g is non-decreasing and submodular, then the inner minimum in eq. (5) is attained for $\mathcal{A} = \mathcal{B} \setminus \{v\}$ and $\mathcal{B} = \emptyset$. In addition, to connect the above notion of total curvature with the notion of modularity, we note that if $c_g = 0$, then g is modular, since eq. (5) implies that for any elements $v \in \mathcal{V}$, and for any sets $\mathcal{A}, \mathcal{B} \subseteq \mathcal{V} \setminus \{v\}$, it holds:

$$(1 - c_g)\,[g(\{v\} \cup \mathcal{B}) - g(\mathcal{B})] \leq g(\{v\} \cup \mathcal{A}) - g(\mathcal{A}), \tag{6}$$

which for $c_g = 0$ implies the modularity of g. Finally, to connect the above notion of total curvature with the notion of monotonicity, we mention that if $c_g = 1$, then eq. (6) implies that g is merely non-decreasing (as it is already assumed by the Definition 5 of total curvature).

Definition 6 (Approximate submodularity) *Consider a finite set $\mathcal{V}$, and a non-decreasing set function $g : 2^{\mathcal{V}} \mapsto \mathbb{R}$, whose total curvature c_g is such that $c_g < 1$. Then, we say that g is* approximately submodular.

B. Performance analysis for Algorithm 1

We quantify Algorithm 1's approximation performance, as well as, its running time per maximization step in Problem 1.

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

- *let the number $f^\star$ be the (optimal) value to Problem 1;*
- *given sets $\mathcal{A}_{1:T} \triangleq (\mathcal{A}_1, \ldots, \mathcal{A}_T)$ as solutions to the maximization steps in Problem 1, let $\mathcal{B}^\star(\mathcal{A}_{1:T})$ be the collection of optimal (worst-case) set removals from each of the $\mathcal{A}_t$, where $t = 1, \ldots, T$, per Problem 1, i.e.:*

$$\mathcal{B}^\star(\mathcal{A}_{1:T}) \in \arg \min_{\mathcal{B}_1\subseteq\mathcal{A}_1,|\mathcal{B}_1|\leq\beta_1} \cdots \min_{\mathcal{B}_T\subseteq\mathcal{A}_T,|\mathcal{B}_T|\leq\beta_T} f(\mathcal{A}_1 \setminus \mathcal{B}_1, \ldots, \mathcal{A}_T \setminus \mathcal{B}_T);$$

The performance of Algorithm 1 is bounded as follows:

1) (Approximation performance) *Algorithm 1 returns the sequence of sets $\mathcal{A}_{1:T} \triangleq (\mathcal{A}_1, \ldots, \mathcal{A}_T)$ such that, for all $t = 1, \ldots, T$, it holds $\mathcal{A}_t \subseteq \mathcal{V}_t$, $|\mathcal{A}_t| \leq \alpha_t$, and:*
   - *if the objective function f is non-decreasing and submodular, then:*

$$\frac{f(\mathcal{A}_{1:T} \setminus \mathcal{B}^\star(\mathcal{A}_{1:T}))}{f^\star} \geq (1 - \kappa_f)^4, \tag{7}$$

where κ_f is the curvature of f (Definition 4).

- *if the objective function f is non-decreasing, then:*

$$\frac{f(\mathcal{A}_{1:T} \setminus \mathcal{B}^{\star}(\mathcal{A}_{1:T}))}{f^{\star}} \geq (1 - c_f)^5, \tag{8}$$

where c_f is the total curvature of f (Definition 5).

2) (Running time) *Algorithm 1 constructs each set $\mathcal{A}_t$, for each $t = 1, \ldots, T$, to solve the t-th maximization step of Problem 1, with $O(|\mathcal{V}_t|(\alpha_t - \beta_t))$ evaluations of f.*

Provable approximation performance. Theorem 1 implies on the approximation performance of Algorithm 1:

a) Near-optimality: Both for monotone submodular objective functions f with curvature $\kappa_f < 1$, and for merely monotone objective functions f with total curvature $c_f < 1$, Algorithm 1 guarantees a value for Problem 1 finitely close to the optimal. In particular, per ineq. (7) (case of submodular objective functions), the approximation factor of Algorithm 1 is bounded by $(1 - \kappa_f)^4$, which is non-zero for any monotone submodular function f with $\kappa_f < 1$; similarly, per ineq. (8) (case of approximately-submodular functions), the approximation factor of Algorithm 1 is bounded by $(1 - c_f)^5$, which is non-zero for any monotone function f with $c_f < 1$ —notably, although it is known for the problem of (non-resilient) set function maximization that the approximation bound $(1 - c_f)$ is tight [27, Theorem 8.6], the tightness of the bound $(1 - c_f)^5$ in ineq. (8) for Problem 1 is an open problem.

We discuss classes of functions f with curvatures $\kappa_f < 1$ or $c_f < 1$, along with relevant applications, in the remark below.

Remark 1 (Classes of functions f with $\kappa_f < 1$ or $c_f < 1$, and applications) Classes of functions f with $\kappa_f < 1$ *are the concave over modular functions [17, Section 2.1], and the* $\log\det$ *of positive-definite matrices [28], [29].* Classes of functions f with $c_f < 1$ *are support selection functions [30], and estimation error metrics such as the average minimum square error of the Kalman filter [2, Theorem 4]*

The aforementioned classes of functions f with $\kappa_f < 1$ or $c_f < 1$ appear in applications of facility location, machine learning, and control, such as sparse approximation and feature selection [4], [5], sparse recovery and column subset selection [6], [7], and actuator and sensor scheduling [2], [8]; as a result, Problem 1 enables applications such as resilient experiment design, resilient actuator scheduling for minimal control effort, and resilient multi-robot navigation with minimal sensing and communication.

b) Approximation performance for low curvature: For both monotone submodular and merely monotone objective functions f, when the curvature κ_f and the total curvature c_f, respectively, tend to zero, Algorithm 1 becomes exact since for $\kappa_f \to 0$ and $c_f \to 0$ the terms $(1 - \kappa_f)^4$ and $(1 - c_f)^5$ in ineq. (7) and ineq. (8) tend to 1. Overall, Algorithm 1's curvature-dependent approximation bounds make a first step towards separating the classes of monotone submodular and merely monotone functions into functions for which Problem 1 can be approximated well (low curvature functions), and functions for which it cannot (high curvature functions).

A machine learning problem where Algorithm 1 guarantees an approximation performance close to 100% the optimal is that of Gaussian process regression for processes with RBF kernels [31], [32]; this problem emerges in applications of sensor deployment and scheduling for temperature monitoring. The reason that in this class of regression problems Algorithm 1 performs almost optimally is that the involved objective function is the entropy of the selected sensor measurements, which for Gaussian processes with RBF kernels has curvature value close to zero [29, Theorem 5].

c) Approximation performance for no failures or attacks: Both for monotone submodular objective functions f, and for merely monotone objective functions f, when the number of attacks, deletions, and failures is zero ($\beta_t = 0$, for all $t = 1, \ldots, T$), Algorithm 1's approximation performance is the same as that of the state-of-the-art algorithms for (non-resilient) set function maximization. In particular, when for all $t = 1, \ldots, T$ it is $\beta_t = 0$, then Algorithm 1 is the same as the local greedy algorithm, proposed in [11, Section 4] for (non-resilient) set function maximization, whose approximation performance is optimal [27, Theorem 8.6].

Minimal running time. Theorem 1 implies that Algorithm 1, even though it goes beyond the objective of (non-resilient) multi-step set function optimization, by accounting for attacks, deletions, and failures, it has the same order of running time as state-of-the-art algorithms for (non-resilient) multi-step set function optimization. In particular, such algorithms for (non-resilient) multi-step set function optimization [11, Section 4] [27, Section 8] terminate with $O(|\mathcal{V}_t|(\alpha_t - \beta_t))$ evaluations of the objective function f per maximization step $t = 1, \ldots, T$, and Algorithm 1 also terminates with $O(|\mathcal{V}_t|(\alpha_t - \beta_t))$ evaluations of the objective function f per maximization step $t = 1, \ldots, T$.

Summary of theoretical results. In sum, Algorithm 1 is the first algorithm for Problem 1, and it enjoys:

- *system-wide resiliency*: Algorithm 1 is valid for any number of denial-of-service attacks, deletions, and failures;
- *adaptiveness*: Algorithm 1 adapts the solution to each of the maximization steps in Problem 1 to the history of inflicted denial-of-service attacks and failures;
- *minimal running time*: Algorithm 1 terminates with the same running time as state-of-the-art algorithms for (non-resilient) multi-step submodular function optimization;
- *provable approximation performance*: Algorithm 1 ensures for all monotone objective functions f that are either submodular or approximately submodular ($c_f < 1$), and for all $T \geq 1$, a solution finitely close to the optimal.

Notably, Algorithm 1 is also the first to guarantee for any number of failures, and for monotone approximately-submodular functions f, a provable approximation performance for the one-step version of Problem 1 where $T = 1$.

V. Numerical Experiments

In this section, we demonstrate a near-optimal performance of Algorithm 1 via numerical experiments. In particular, we consider a control-aware sensor scheduling scenario, namely, *sensing-constrained robot navigation*.[1] According to this scenario, an unmanned aerial vehicle (UAV), which has limited remaining battery and measurement-processing power, has the objective to land, and to this end, it schedules to activate at each time step only a subset of its on-board sensors, so to localize itself and enable the generation of a control input for landing; specifically, at each time step, the UAV generates its control input by implementing an LQG-optimal controller, given the measurements collected by the activated sensors up to the current time step [2], [33].

In more detail, in the following paragraphs we present a Monte Carlo analysis for an instance of the aforementioned sensing-constrained robot navigation scenario, in the presence of worst-case sensor failures, and observe that Algorithm 1 results to a near-optimal sensor selection schedule: in particular, the resulting navigation performance of the UAV matches the optimal in all tested instances for which the optimal selection could be computed via a brute-force approach.

Simulation setup. We adopt the same instance of the sensing-constrained robot navigation scenario adopted in [2, Section V.B]. Specifically, a UAV moves in a 3D space, starting from a randomly selected initial location. The objective of the UAV is to land at position $[0,\ 0,\ 0]$ with zero velocity. The UAV is modelled as a double-integrator with state $x_t = [p_t\ v_t]^\top \in \mathbb{R}^6$ at each time $t = 1, 2, \dots$ (p_t is the 3D position of the UAV, and v_t is its velocity), and can control its own acceleration $u_t \in \mathbb{R}^3$; the process noise is chosen as $W_t = \mathbf{I}_6$. The UAV is equipped with multiple sensors, as follows: it has two on-board GPS receivers, measuring the UAV position p_t with a covariance $2 \cdot \mathbf{I}_3$, and an altimeter, measuring only the last component of p_t (altitude) with standard deviation 0.5m. Moreover, the UAV can use a stereo camera to measure the relative position of 10 landmarks on the ground; we assume the location of each landmark to be known only approximately, and we associate to each landmark an uncertainty covariance, which is randomly generated at the beginning of each run. The UAV has limited on-board resource-constraints, hence it can only activate a subset of sensors (possibly different at each time step). For instance, the resource-constraints may be due to the power consumption of the GPS and the altimeter, or due to computational constraints that prevent to run object-detection algorithms to detect all landmarks on the ground.

Among the aforementioned 13 possible sensor measurements available to the UAV at each time step, we assume that the UAV can use only $\alpha = 11$ of them. In particular, the UAV chooses the sensors to activate at each time step so to minimize an LQG cost with cost matrices Q (which penalizes the state vector) and R (which penalizes the control input vector), per the problem formulation in [2, Section II]; specifically, in this simulation setup we set $Q = \text{diag}\left([1e^{-3},\ 1e^{-3},\ 10,\ 1e^{-3},\ 1e^{-3},\ 10]\right)$ and $R = \mathbf{I}_3$. Note that the structure of Q (which penalizes the magnitude of the state vector) reflects the fact that during landing we are particularly interested in controlling the vertical direction and the vertical velocity (entries with larger weight in Q), while we are less interested in controlling accurately the horizontal position and velocity (assuming a sufficiently large landing site). Given a time horizon T for landing, in [2] it is proven that the UAV selects an optimal sensor schedule and generates an optimal LQG control input with cost matrices Q and R if it selects the sensors set $\mathcal{S}_t$ to activate at each time $t = 1, \dots, T$ by minimizing an objective function of the form:

$$\sum_{t=1}^{T} \text{trace}[M_t \Sigma_{t|t}(\mathcal{S}_1, \dots, \mathcal{S}_t)], \tag{9}$$

where M_t is a positive semi-definite matrix that depends on the LQG cost matrices Q and R, as well as, on the UAV's model dynamics; and $\Sigma_{t|t}(\mathcal{S}_1, \dots, \mathcal{S}_t)$ is the error covariance of the Kalman filter given the sensor selections up to time t.

In the remaining paragraphs, we present results averaged over 10 Monte Carlo runs of the above simulation setup, where in each run we randomize the covariances describing the landmark position uncertainty, and where we vary the number β of sensors failures at each time step t: in particular, we consider β to vary among the values $1, 4, 7, 10$.

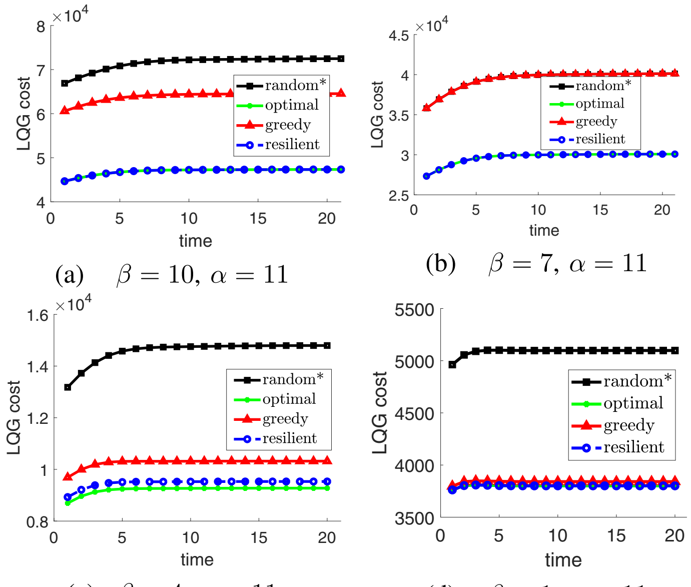
Compared algorithms. We compare four algorithms. All algorithms only differ in how they select the sensors used. The first algorithm is the optimal sensor selection algorithm, denoted as `optimal`, which attains the minimum of the cost function in eq. (9); this brute-force approach is viable since the number of available sensors is small. The second approach is a pseudo-random sensor selection, denoted as `random`*, which selects one of the GPS measurements and a random subset of the lidar measurements; note that we do not consider a fully random selection since in practice this often leads to an unobservable system. The third approach, denoted as `greedy`, selects sensors to greedily minimize the cost function in eq. (9), *ignoring the possibility of sensor failures*, per the problem formulation in eq. (1). The fourth approach uses Algorithm 1 to solve the resilient reformulation of eq. (9), per Problem 1, and is denoted as `resilience`.

At each time step, from each of the selected sensor sets, selected by any of the above four algorithms, we consider an optimal sensor removal, which we compute via a brute-force.

Results. The results of our numerical analysis are reported in Fig. 1. In particular, Fig. 1 shows the LQG cost for increasing time, for the case where the number of selected sensors at each time step is $\alpha = 11$, while the number of sensor failures β at each time step varies across the values $10, 7, 4, 1$. The following observations are due:

- (*Near-optimality of Algorithm 1*) Algorithm 1 (blue colour in Fig. 1) performs close to the optimal brute-force algorithm (green colour in Fig. 1); in particular, across all scenarios in Fig. 1, Algorithm 1 achieves an approximation performance at least 97% the optimal.
- (*Performance of greedy algorithm*) The greedy algorithm (red colour in Fig. 1) performs poorly as the number β

[1]The scenario of sensing-constrained robot navigation is introduced in [2, Section V.B], yet in the absence of sensor failures.



(a) $\beta = 10$, $\alpha = 11$ (b) $\beta = 7$, $\alpha = 11$ (c) $\beta = 4$, $\alpha = 11$ (d) $\beta = 1$, $\alpha = 11$

Fig. 1: LQG cost for increasing time, where across all sub-figures (a)-(d) it is $\alpha = 11$ (number of active sensors per time step). The value of β (number of sensor failures at each time step among the α active sensors) varies across the sub-figures.

of sensor failures increases, which was expected, given that this algorithm greedily minimizes the cost function in eq. (9) ignoring the possibility of sensor failures.

- (*Performance of random algorithm*) Expectedly, also the performance of the random algorithm (black colour in Fig. 1) is poor across all scenarios in Fig. 1.

Overall, in the above numerical experiments, Algorithm 1 demonstrates a near-optimal approximation performance, and the necessity for the resilient reformulation of the problem in eq. (1) per Problem 1 is exemplified.

VI. Concluding remarks & Future work

We made the first step to ensure the success of critical missions in machine learning and control, that involve the optimization of systems across multiple time-steps, against persistent failures or denial-of-service attacks. In particular, we provided the first algorithm for Problem 1, which, with minimal running time, adapts to the history of the inflicted failures and attacks, and guarantees a close-to-optimal performance against system-wide failures and attacks. To quantify the algorithm's approximation performance, we exploited a notion of curvature for monotone (not necessarily submodular) set functions, and contributed a first step towards characterizing the curvature's effect on the approximability of resilient *sequential* maximization. Our curvature-dependent characterizations complement the current knowledge on the curvature's effect on the approximability of simpler problems, such as of non-sequential resilient maximization [22], [23], and of non-resilient maximization [17], [18], [20]. Finally, we supported our theoretical analyses with simulated experiments.

This paper opens several avenues for future research, both in theory and in applications. Future work in theory includes the extension of our results to matroid constraints, to enable applications of set coverage and of network design [34], [35]. Future work in applications includes the experimental testing of the proposed algorithm in applications of motion-planning for multi-target covering with mobile vehicles [9], and in applications of control-aware sensor scheduling for multi-agent autonomous navigation [2], to enable resiliency in critical scenarios of surveillance, and of search and rescue.

VII. Acknowledgements

We thank Luca Carlone for inspiring discussions.

References

[1] D. Golovin and A. Krause, "Adaptive submodularity: Theory and applications in active learning and stochastic optimization," *Journal of Artificial Intelligence Research*, vol. 42, pp. 427–486, 2011.
[2] V. Tzoumas, L. Carlone, G. J. Pappas, and A. Jadbabaie, "Control and Sensing Co-design," *ArXiv e-prints: 1802.08376*, 2018.
[3] A. Clark, L. Bushnell, and R. Poovendran, "A supermodular optimization framework for leader selection under link noise in linear multi-agent systems," *IEEE Trans. on Aut. Contr.*, vol. 59, no. 2, pp. 283–296, 2014.
[4] A. Das and D. Kempe, "Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection," in *International Conference on Machine Learning*, 2011, pp. 1057–1064.
[5] R. Khanna, E. Elenberg, A. Dimakis, S. Negahban, and J. Ghosh, "Scalable greedy feature selection via weak submodularity," in *Artificial Intelligence and Statistics*, 2017, pp. 1560–1568.
[6] E. J. Candes, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on pure and applied mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.
[7] C. Boutsidis, M. W. Mahoney, and P. Drineas, "An improved approximation algorithm for the column subset selection problem," in *ACM-SIAM Symposium on Discrete algorithms*, 2009, pp. 968–977.
[8] Y. Zhao, F. Pasqualetti, and J. Cortés, "Scheduling of control nodes for improved network controllability," in *IEEE 55th Conference on Decision and Control*, 2016, pp. 1859–1864.
[9] 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.
[10] U. Feige, "A threshold of $ln(n)$ for approximating set cover," *Journal of the ACM*, vol. 45, no. 4, pp. 634–652, 1998.
[11] 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.
[12] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," *Computer*, vol. 35, no. 10, pp. 54–62, 2002.
[13] B. Mirzasoleiman, A. Karbasi, and A. Krause, "Deletion-robust submodular maximization: Data summarization with 'the right to be forgotten'," in *International Conference on Machine Learning*, 2017, pp. 2449–2458.
[14] A. S. Willsky, "A survey of design methods for failure detection in dynamic systems," *Automatica*, vol. 12, no. 6, pp. 601–611, 1976.
[15] R. B. Myerson, *Game theory*. Harvard University Press, 2013.
[16] D. Golovin and A. Krause, "Adaptive submodularity: A new approach to active learning and stochastic optimization," in *Annual Conference on Learning Theory*, 2010, pp. 333–345.
[17] R. K. Iyer, S. Jegelka, and J. A. Bilmes, "Curvature and optimal algorithms for learning and minimizing submodular functions," in *Advances in Neural Inform. Processing Systems*, 2013, pp. 2742–2750.
[18] A. A. Bian, J. M. Buhmann, A. Krause, and S. Tschiatschek, "Guarantees for greedy maximization of non-submodular functions with applications," in *Int. Conf. on Machine Learning*, 2017, pp. 498–507.
[19] G. L. Nemhauser and L. A. Wolsey, "Best algorithms for approximating the maximum of a submodular set function," *Mathematics of operations research*, vol. 3, no. 3, pp. 177–188, 1978.
[20] M. Conforti and G. Cornuéjols, "Submodular set functions, matroids and the greedy algorithm," *Discrete Applied Mathematics*, vol. 7, no. 3, pp. 251 – 274, 1984.
[21] J. B. Orlin, A. S. Schulz, and R. Udwani, "Robust monotone submodular function maximization," *arXiv preprint:1507.06616*, 2015.

[22] 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.
[23] I. Bogunovic, J. Zhao, and V. Cevher, "Robust maximization of non-submodular objectives," *ArXiv e-prints:1802.07073*, 2018.
[24] S. Mitrovic, I. Bogunovic, A. Norouzi-Fard, J. M. Tarnawski, and V. Cevher, "Streaming robust submodular maximization," in *Advances in Neural Information Processing Systems*, 2017, pp. 4560–4569.
[25] E. Kazemi, M. Zadimoghaddam, and A. Karbasi, "Deletion-robust submodular maximization at scale," *ArXiv e-prints:1711.07112*, 2017.
[26] G. Nemhauser, L. Wolsey, and M. Fisher, "An analysis of approximations for maximizing submodular set functions – I," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.
[27] M. Sviridenko, J. Vondrák, and J. Ward, "Optimal approximation for submodular and supermodular optimization with bounded curvature," *Math. of Operations Research*, vol. 42, no. 4, pp. 1197–1218, 2017.
[28] ——, "Optimal approximation for submodular and supermodular optimization with bounded curvature," in *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2015, pp. 1134–1148.
[29] D. Sharma, A. Kapoor, and A. Deshpande, "On greedy maximization of entropy," in *Inter. Conf. on Machine Learning*, 2015, pp. 1330–1338.
[30] E. R. Elenberg, R. Khanna, A. G. Dimakis, and S. Negahban, "Restricted strong convexity implies weak submodularity," *ArXiv e-prints:1612.00804*, 2016.
[31] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *J.l of Machine Learning Research*, vol. 9, pp. 235–284, 2008.
[32] C. Bishop, *Pattern recognition and machine learning*. Springer, 2006.
[33] D. P. Bertsekas, *Dynamic programming and optimal control, Vol. I.* Athena Scientific, 2005.
[34] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1740–1766, 2011.
[35] A. Clark, B. Alomair, L. Bushnell, and R. Poovendran, "Scalable and distributed submodular maximization with matroid constraints," in *International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, 2015, pp. 435–442.