Improved Resilient Coverage Maximization with Multiple Robots

Ishat E Rabban
Department of Computer Science
University of Maryland
College Park, MD, USA
ier@umd.edu

Pratap Tokekar
Department of Computer Science
University of Maryland
College Park, MD, USA
tokekar@umd.edu

***Abstract*—The task of maximizing coverage using multiple robots has several applications such as surveillance, exploration, and environmental monitoring. A major challenge of deploying such multi-robot systems in a practical scenario is to ensure resilience against robot failures. A recent work [1] introduced the Resilient Coverage Maximization (*RCM*) problem where the goal is to maximize a submodular coverage utility when the robots are subject to adversarial attacks and/or failures. The *RCM* problem is known to be NP-hard. The state-of-the-art solution of the *RCM* problem [1] employs a greedy approximation strategy with theoretical performance guarantees. In this paper, we propose two approximation algorithms for the *RCM* problem, both of which empirically outperform the existing solution in terms of accuracy and/or execution time. To demonstrate the effectiveness of our proposed solution, we empirically compare our proposed algorithms with the existing solution and a brute force optimum algorithm.**

***Index Terms*—Coverage Maximization, Multi-Robot Systems, Resilience**

I. Introduction

Tasks such as surveillance [3], tracking [2], and motion planning [7] can be formulated as an optimization problem that aims to maximize the coverage of a set of target objects. These coverage maximization tasks can be benefited by the use of multiple robots as opposed to a single robot. Indeed, the advancements in robotic mobility, sensing, and communication technology have led to the use of multiple collaborating robots to support such tasks [4]–[6]. But a major challenge for practical deployment of such multi-robot systems is to make the robots resilient to failures. For example, the robots may undergo adversarial attacks [10], or the field-of-view of some robots may get occluded due to environmental hazards [11], or the sensors may stop working due to technical malfunction [12]. In this paper, our goal is to devise coverage maximization algorithms that are resilient to such failures.

The standard coverage maximization problem has been well studied in literature [15]–[17]. But an adversarial variant of the coverage maximization problem that ensures robustness against robot failures has gained attention among the research community only recently [8], [13], [14]. In a recent work, Zhou et al. [1] introduced a new variant of the coverage maximization problem that takes into account the resilience of the multi-robot system. In their proposed problem setup, a team of robots aim to cover a set of targets (Figure 1).

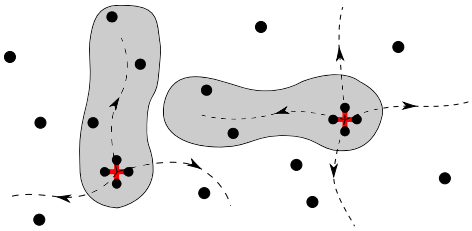

Fig. 1. Two robots (drone signs) are tracking targets (black dots). The left (right) robot has 3 (4) available trajectories (dotted arrow). Coverage region of one trajectory of each robot is shown in gray. The highlighted trajectory of the left (right) robot covers 3 (2) targets.

For each robot, there is a set of candidate trajectories, one of which the robot will follow. The list of targets covered by each robot trajectory is provided. It is assumed that at most α robots may fail. But it is unknown which robots are going to fail. The objective of the problem is to select one trajectory for each robot such that the target coverage is maximized in case of a worst case failure of α robots. We call this problem *Resilient Coverage Maximization* (*RCM*) problem. The *RCM* problem is known to be NP-hard [9].

Inspired by a recent work of Tzoumas et al. [13] that explores generalized resilient optimization subject to matroid constraints, Zhou et al. [1] proposed an approximation algorithm for the *RCM* problem that involves two phases. In the first phase, the algorithm determines the worst case subset of α robots that could fail, and selects their trajectories. In the next phase, assuming that the robots selected in the first phase will actually fail, the rest of the robot trajectories are selected greedily such that, for each greedy selection, the marginal gain in target coverage is maximized. We call this algorithm the *2 Phase Greedy* (*2PG*) algorithm. The execution time of the *2PG* algorithm is $O(P^2)$, where P is the sum of the number of trajectories of all the robots.

In this paper, we propose two algorithms for the *RCM* problem that perform better than the *2PG* algorithm in terms of accuracy and/or execution time. Here, by accuracy of a solution, we mean how much target coverage the solution achieves with respect to an optimum solution. Our proposed algorithms are called *Ordered Greedy* (*OrG*) algorithm and *Local Search* (*LS*) algorithm.

The *OrG* algorithm produces an ordering of the robots according to some sorting criteria, and greedily selects the trajectories for each robot one after another according to the above sorted order such that, for each robot, incremental target coverage is maximized. The execution time of the *OrG* algorithm is O(P). Experimental results show that the accuracy of the *OrG* algorithm is slightly better than the *2PG* algorithm, and it runs much faster than the *2PG* algorithm.

In the *LS* algorithm, we start with an initial solution of the *RCM* problem. Then, at each iteration, we estimate the accuracy of the neighbors of the current solution and select a neighbor solution with higher estimated accuracy. The algorithm terminates when we find a local optima. Empirical studies show that the accuracy of the *LS* algorithm is significantly better than the *2PG* algorithm, while the two algorithms are close to each other in terms of execution time.

In case of both algorithms, we consider several design choices and compare the accuracy of the variants of the algorithms that arise from different design choices. In case of the *OrG* algorithm, the design choice is the sorting criteria used to sort the robots. In case of the *LS* algorithm, design choices include the initial solution and the attack model.

In summary, we make the following contributions:

- We propose two algorithms for the *RCM* problem, namely, the *OrG* algorithm and the *LS* algorithm, which perform better than the state-of-the-art *2PG* algorithm in terms of accuracy and/or execution time.
- We conduct extensive experiments with synthetic datasets to evaluate the accuracy and execution time of our proposed algorithms with respect to the *2PG* algorithm and a brute force optimum algorithm.

The organization of the rest of the paper is as follows. In Section II, we provide the formal definition of the *RCM* problem. Next, in Section III and Section IV, we describe the *OrG* and *LS* algorithms respectively. The experimental results are presented in Section V. Finally, in Section VI, we make some concluding remarks.

II. Problem Formulation

In this section, we describe the framework of the resilient coverage maximization problem (Section II-A), present the formal definition of the problem (Section II-B), and provide some supplementary definitions (Section II-C).

A. Framework

We adopt the framework introduced by Zhou et al. [1] in their work on resilient multi-target tracking. According to the proposed framework, we consider a set of targets, $\mathcal{X}$, which are being tracked/covered by a set of mobile robots, $\mathcal{R}$. The targets can be mobile or stationary, distinguishable or indistinguishable, and can have known or unknown motion model. It is assumed that the robots have perfect localization and can communicate with each other at all times. Using sensors, communication, and filtering techniques, the robots are able to calculate the estimated position of the targets.

Time is divided into rounds of finite duration. We consider each round independently. At the beginning of a round, each robot generates a set of candidate trajectories, one of which will be followed in the current round. The set of trajectories of a robot r is denoted be $\mathcal{T}_r$. The set of all robots' trajectories is denoted by $\mathcal{T}_\mathcal{R}$, i.e., $\mathcal{T}_\mathcal{R} := \cup_{r\in\mathcal{R}}\mathcal{T}_r$. Let P be the number of all trajectories, i.e., $P = |\mathcal{T}_\mathcal{R}|$. Here the notation $|\mathcal{A}|$ denotes the cardinality of set $\mathcal{A}$.

Target Coverage function: The *coverage* of a trajectory τ is defined as the set of targets that τ covers, which we denote by $\mathrm{C}(\tau)$. The *target coverage function*, F, takes as input a set of trajectories $\mathcal{T}$ and returns the number of distinct targets covered by the trajectories in $\mathcal{T}$, i.e., $\mathrm{F}(\mathcal{T}) := |\cup_{\tau\in\mathcal{T}}\mathrm{C}(\tau)|$. The target coverage function F is *monotone* and *submodular* [16]. Other examples of monotone and submodular target coverage functions are mutual information and entropy [18].

Attack Model: We assume that at most α robots/sensors can fail at a time. We consider the worst case/optimum attack model as defined below. Given a set of trajectories $\mathcal{T}$, the target coverage function F, and an integer α denoting the maximum attack size, an *optimal attack* on $\mathcal{T}$ of size α is defined as follows.

$$\mathrm{A}^*_\alpha(\mathcal{T}) := \operatorname*{argmin}_{\mathcal{A}\subseteq\mathcal{T}} \mathrm{F}(\mathcal{T}\backslash\mathcal{A}) \quad s.t.\ |\mathcal{A}| \leq \alpha$$

In other words, an optimal attack on $\mathcal{T}$ of size α is a subset of $\mathcal{T}$ of size at most α such that removal of the subset from $\mathcal{T}$ results in maximum decrease of the target coverage. In the above definition, the notation $\mathcal{A}\backslash\mathcal{B}$ denotes the set of elements in $\mathcal{A}$ that are not in $\mathcal{B}$.

B. Problem Definition

Given a set of targets, a set of robots $\mathcal{R}$, the trajectories for the robots $\mathcal{T}_\mathcal{R}$, the attack size α, and a target coverage function F, the *Resilient Coverage Problem* problem aims to select a set of trajectories according to the following objective function.

$$\operatorname*{argmax}_{\mathcal{S}\subseteq\mathcal{T}_\mathcal{R}} \mathrm{F}(\mathcal{S}\backslash\mathrm{A}^*_\alpha(\mathcal{S})) \quad s.t.\ |\mathcal{S}\cap\mathcal{T}_r| = 1,\ \forall r\in\mathcal{R} \qquad (1)$$

In other words, the solution subset contains one trajectory per robot (enforced by the constraints), such that in case of an optimal attack of size α, the target coverage of the remaining robots is maximized.

C. Supplementary Definitions

A *Feasible Solution* is a subset of $\mathcal{T}_\mathcal{R}$ that satisfies the constraints in (1). A feasible solution corresponds to a valid assignment of trajectories to robots, i.e., one trajectory per robot. The feasible solution that maximizes the objective function (1) is called the *Optimum Solution*. We denote the optimum solution by $\mathcal{S}^*$. The *Residual Coverage* of a feasible solution $\mathcal{S}$ is the number of targets covered by $\mathcal{S}$ after the optimal attack set is removed from $\mathcal{S}$. The residual coverage of $\mathcal{S}$ is denoted by $\mathrm{R}(\mathcal{S})$. According to the above definition, $\mathrm{R}(\mathcal{S}) = \mathrm{F}(\mathcal{S}\backslash\mathrm{A}^*_\alpha(\mathcal{S}))$, where α is the attack size.

III. Ordered Greedy Algorithm

In this section, we present a class of greedy algorithms that require $O(P)$ evaluations of the target coverage function F. The algorithm is named Ordered Greedy Algorithm (*OrG*) and is presented below (Algorithm 1). In this algorithm, first we sort the robots according to some *sorting criteria* (Line 1). Then we consider the robots in the sorted order (Line 3). For each robot, we greedily select the trajectory that maximizes the incremental coverage of the targets (Line 4-5).

Algorithm 1 Ordered Greedy Algorithm

Input: $\mathcal{R}$, $\mathcal{T}_{\mathcal{R}}$, α, F
Output: Set of trajectories, $\mathcal{S}$

1: $< r_1, r_2, \ldots, r_{|\mathcal{R}|} > \leftarrow \text{sort}(\mathcal{T}_{\mathcal{R}}, \text{F})$
2: $\mathcal{S} \leftarrow \emptyset$
3: **for** $i \leftarrow 1$ to $|\mathcal{R}|$ **do**
4: $\quad \tau^* \leftarrow \text{argmax}_{\tau \in \mathcal{T}_{r_i}} \text{F}(\mathcal{S} \cup \tau)$
5: $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \tau^*$
6: **end for**
7: **return** $\mathcal{S}$

To perform the sorting of the robots, for each robot r, we calculate a numerical value $\text{V}(r)$, according to some sorting criteria, and then sort the robots in increasing and/or decreasing order of the assigned numerical value. We use the following sorting criteria to evaluate the accuracy and performance of the ordered greedy algorithms. Each criteria results in two different variants of the ordered greedy algorithms: one for increasing order, and one for decreasing order.

- *Sum of Target Coverage*: The numerical value of robot r is the sum of the number of targets covered by all the trajectories of r, i.e., $\text{V}(r) = \sum_{\tau \in \mathcal{T}_r} |\text{C}(\tau)|$. The resultant ordered greedy algorithms are named *OrG-S-I* and *OrG-S-D* (for increasing and decreasing order of sorting, respectively).
- *Size of Union of Target Coverage*: The numerical value of robot r is the number of distinct targets covered by all the trajectories of r, i.e., $\text{V}(r) = |\cup_{\tau \in \mathcal{T}_r} \text{C}(\tau)|$. The resultant algorithms are named *OrG-U-I* and *OrG-U-D*.
- *Maximum Individual Target Coverage*: The numerical value of robot r is the cardinality of the trajectory of r that covers the maximum number of targets, i.e., $\text{V}(r) = \max_{\tau \in \mathcal{T}_r} |\text{C}(\tau)|$. The resultant algorithms are named *OrG-M-I* and *OrG-M-D*.

We also consider another variant where the ordering of the robots is random (*OrG-R*). Note that, each of the above algorithms requires $O(P)$ evaluations of F. Here, P is the sum of number of trajectories of all robots. To calculate the numerical values of the robots, we need P evaluations of F. Also, in Line 4 of Algorithm 1, the call to F is executed P times in total. Thus, the total number of evaluations of F for the ordered greedy algorithm is $O(P)$.

IV. Local Search Algorithm

In this section, we describe a class of algorithms based on local search technique. In a traditional local search algorithm, we start with an initial solution. At each iteration, we make small local changes to the current solution to form a set of candidate neighboring solutions. Then we evaluate the objective function on the neighboring solutions to find if any improvement over the current solution is possible. If a better solution is found, the search moves to that direction. Otherwise, the algorithm terminates.

A tricky aspect of adopting a local search based approach to the resilient coverage maximization problem is that evaluating the objective function for a given solution is not straightforward. In this problem, the objective value of a solution $\mathcal{S}$ is $\text{F}(\mathcal{S} \backslash \text{A}^*_\alpha(\mathcal{S}))$. Thus, given a candidate solution $\mathcal{S}$, in order to evaluate the objective function for $\mathcal{S}$, we are required to construct an optimal attack on $\mathcal{S}$. But constructing an optimal attack on $\mathcal{S}$ is a combinatorially hard problem, which requires exponential computation time with respect to the size of $\mathcal{S}$. Consequently, in this algorithm, we use computationally feasible greedy attack models, instead of an optimal attack model, to drive the local search. We denote the greedy attack function by A, which is further discussed later in this section.

Algorithm 2 Local Search Algorithm

Input: $\mathcal{R}$, $\mathcal{T}_{\mathcal{R}}$, α, F
Output: Set of trajectories, $\mathcal{S}$

1: $\mathcal{S} \leftarrow \text{INIT}(\mathcal{T}_{\mathcal{R}}, \text{F})$
2: $z \leftarrow \text{F}(\mathcal{S} \backslash \text{A}_\alpha(\mathcal{S}))$
3: **while** TRUE **do**
4: $\quad f \leftarrow \text{FALSE}$
5: $\quad$ **for** all neighbor $\dot{\mathcal{S}}$ of $\mathcal{S}$ **do**
6: $\quad\quad \mathcal{A} \leftarrow \text{A}_\alpha(\dot{\mathcal{S}})$
7: $\quad\quad \dot{z} \leftarrow \text{F}(\dot{\mathcal{S}} \backslash \mathcal{A})$
8: $\quad\quad$ **if** $\dot{z} > z$ **then**
9: $\quad\quad\quad f, \mathcal{S}, z \leftarrow \text{TRUE}, \dot{\mathcal{S}}, \dot{z}$
10: $\quad\quad\quad$ **break**
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: $\quad$ **if** $f = \text{FALSE}$ **then**
14: $\quad\quad$ **break**
15: $\quad$ **end if**
16: **end while**
17: **return** $\mathcal{S}$

Now we describe the Local Search algorithm (Algorithm 2) in details. We start with an initial solution $\mathcal{S}$ (Line 1) and the corresponding objective value z (Line 2). At each iteration of the local search (Line 3-16), we consider all neighbors of the current solution $\mathcal{S}$ (Line 5). Any feasible solution which differs with $\mathcal{S}$ by exactly one trajectory is defined to be a *neighbor* of $\mathcal{S}$. For each neighbor $\dot{\mathcal{S}}$ of $\mathcal{S}$, we construct a greedy attack on $\dot{\mathcal{S}}$, denoted by $\mathcal{A}$ (Line 6), and calculate the corresponding objective value $\dot{z}$ (Line 7). If the candidate solution is better than the current solution (Line 8), we restart the iteration with updated solution and objective value (Line 9-10). If no neighbor leads to a solution better than the current solution, the local search terminates (Line 4, 13-14), and the

current solution is returned (Line 17) as the solution of the resilient coverage maximization problem.

Several variants of the local search algorithm arise when we use different attack models (Line 2, 6) and different initial solutions (Line 1). We consider the following two greedy attack models. In both models, the attacker makes α greedy choices, one after another, obeying the constraint that at most one trajectory can be selected per robot.

- *Attack Model 1 (A1)*: At each iteration, the attacker selects the trajectory which maximizes the increase in target coverage.
- *Attack Model 2 (A2)*: Let U be the set of targets which are covered by exactly one of the remaining trajectories. At each iteration, the attacker selects the trajectory which covers the maximum number of targets in U.

We consider two initial solutions as follows.

- *Initial Solution 1 (I1)*: The output of the Oblivious Greedy algorithm (to be described in Section V-A).
- *Initial Solution 2 (I2)*: The output of the *OrG-U-I* algorithm (described in Section III).

Two attack models and two initial solutions yield four variants of the *LS* algorithm. We append the attack type and initial solution type to name the variants. For example, *LS-A1-I2* is the variant of the local search algorithm that uses attack model 1 and initial solution 2.

Note that, in the local search algorithm, the number of evaluations of F is dominated by the number of calls to the attack function in Line 6. In case of both of the above attack models, constructing a greedy attack requires $O(\alpha R)$ calls to F, where R is the number of robots. Also, there are $O(P)$ neighbors of the current solution $\mathcal{S}$. Consequently, in case of both attack models, the local search algorithm requires $O(I\alpha RP)$ evaluations of F, where I is the number of times the local search iterates. But, by using some hash-based data structures, we reduce the execution time of the local search algorithm by a factor of T, where T is the number of targets. We describe the acceleration technique in Appendix A.

V. Experiments

In this section, we empirically evaluate our proposed algorithms and present the experimental results. First, we discuss the experimental setup (Section V-A). Next, we compare the accuracy (Section V-B) and execution time (Section V-C) of our proposed algorithms with the *2PG* algorithm. Finally, we evaluate the performance of our proposed algorithms in case of a non-optimal attack model (Section V-D).

A. Experimental Setup

Evaluation Metric: We use two metrics to empirically evaluate our proposed algorithms: accuracy and execution time. The *accuracy* of a feasible solution $\mathcal{S}$ is the ratio of the residual coverages of $\mathcal{S}$ and $\mathcal{S}^*$, where $\mathcal{S}^*$ is the optimum solution. Thus, the accuracy of a feasible solution $\mathcal{S}$ is a measure of the quality of $\mathcal{S}$ with respect to the optimum solution $\mathcal{S}^*$. If, in an experiment, the optimum solution is known, we directly report the accuracy of the solutions found by the algorithms which are being compared. On the other hand, if the optimum solution is unknown, we compute the residual coverages of the solutions found by the algorithms and report the relative accuracy with respect to the *2PG* algorithm. Note that, a higher residual coverage corresponds to higher accuracy, and vice versa, since the ratio of residual coverages of two feasible solutions equals the ratio of their accuracy.

Compared Algorithms: We empirically compare the performance of our proposed algorithms (*OrG* and *LS*) with the *2PG* algorithm. We additionally consider two straightforward algorithms as follows:

- *Brute Force algorithm*: The Brute Force (*BF*) algorithm determines the optimum solution of the *RCM* problem. In this algorithm, we consider all feasible solutions, construct an optimal attack on each feasible solution, and report the feasible solution with maximum residual coverage. Thus, the *BF* algorithm has exponential execution time with respect to the number of robots.
- *Oblivious Greedy algorithm*: In the Oblivious Greedy (*ObG*) algorithm, we select, for each robot, the trajectory that covers maximum number of targets. Formally, the solution found by this algorithm is $\bigcup_{r\in\mathcal{R}} \operatorname{argmax}_{\tau\in\mathcal{T}_r} \mathrm{F}(\{\tau\})$. The *ObG* algorithm makes P calls to the target coverage function F.

Dataset: In our experiments, we use a synthetic dataset generated as follows. First, we select the locations of the targets and robots within a 10000×10000 2D grid with uniform probability. For each robot, we consider 4 trajectories: forward, backward, left, and right as shown in Figure 2. We assume that the regions covered by the 4 trajectories are rectangular (shown in gray). The targets falling inside a rectangle are assigned to the corresponding trajectory.

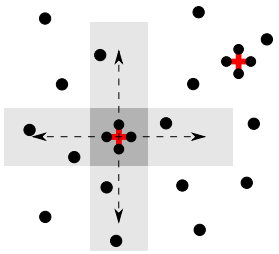

Fig. 2. Generation of Dataset.

Parameter Set: We use different sets of parameter values for different experiments. For example, in the experiments where we compute a brute force solution, we use only 6 robots to keep the total execution time in check. But for other experiments we use higher number of robots. The values of the parameters used in each experiment is mentioned in the corresponding subsection. Each experiment is conducted 100 times and the average is reported. We assume that the number of robot failures is equal to the attack size, α.

Platform: The algorithms are implemented using C++ programming language. The experiments are conducted on a core-i7 2GHz PC with 8GB RAM, running Microsoft Windows 10.

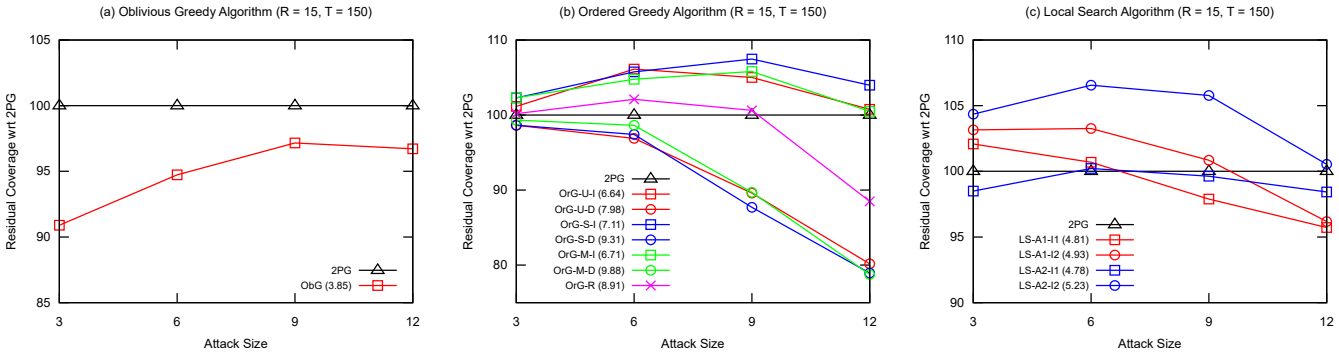

Fig. 3. Comparison of accuracy of (a) *ObG* algorithm, (b) *OrG* algorithm, and (c) *LS* algorithm with *2PG* algorithm.

B. Comparison of Accuracy

Relative Accuracy with respect to *2PG* Algorithm: In this experiment, we construct a dataset with 15 robots and 150 targets. We vary the attack size in increments of 3 and report the average relative accuracy (in percentage) of our proposed algorithms with respect to the *2PG* algorithm (Figure 3). The average standard deviation is shown in the legends.

The experimental results (Figure 3(a)) show that the accuracy of the *ObG* algorithm is consistently lower than the *2PG* algorithm. In case of the *OrG* algorithm, the *OrG-I* variants (i.e., variants of *OrG* algorithm sorted in *increasing* order) have better accuracy in comparison with their *OrG-D* counterparts (Figure 3(b)). The accuracy of the *2PG* and *OrG-R* algorithms lie in between the *OrG-I* and *OrG-D* variants.

We argue that the increasing sorting order leads to an even distribution of the targets to trajectories. Consequently, the reduction of target coverage after an optimal attack is smaller in case of the *OrG-I* variants as opposed to the *OrG-D* ones. We empirically verify the correctness of the above claim by conducting an experiment where we compare the standard deviations of the incremental coverages at each greedy iteration of *OrG-U-I* and *OrG-U-D*. We find that the standard deviation of the *OrG-D* variant is on average 60% higher than the *OrG-I* variant, which provides empirical evidence in support of our claim.

In case of the *LS* algorithm, experimental results (Figure 3(c)) show that attack model 2 (*A2*) leads to better accuracy than attack model 1 (*A1*). Also, initial condition 2 (*I2*) gives higher accuracy in comparison to initial condition 1 (*I1*). Thus, *LS-A2-I2* has the highest accuracy among the *LS* variants. Also, the accuracy of *LS-A2-I2* is significantly better than the *2PG* algorithm across all attack sizes.

Comparison with Brute Force Algorithm: In this experiment, we determine the accuracy of our proposed algorithms. Note that, the accuracy of a feasible solution $\mathcal{S}$ is the ratio of the residual coverages of $\mathcal{S}$ and $\mathcal{S}^*$, the optimum solution. We compute the optimum solution using the *BF* algorithm, which has exponential running time. Consequently, in this experiment, we consider small instances of the *RCM* problem with 6 robots and 60 targets, and use attack sizes 2, 3, and 4.

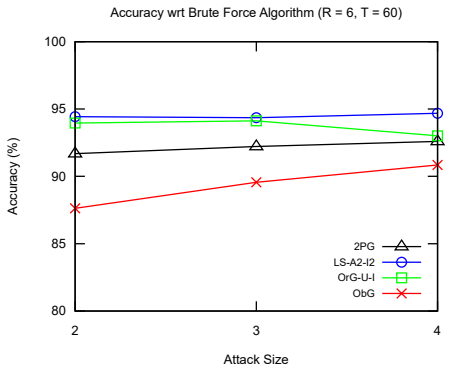

Fig. 4. Comparison of accuracy of proposed algorithms with *BF* algorithm.

For brevity of presentation, from now on, instead of reporting the accuracy of all the variants of our proposed algorithms, we only report the results for the *OrG* variant *OrG-U-I* and *LS* variant *LS-A2-I2* along with the oblivious greedy algorithm *ObG* and *2PG* algorithm. The experimental results (Figure 4) show that the local search algorithm has the highest accuracy, followed by the ordered greedy algorithm, *2PG* algorithm, and the oblivious greedy algorithm, which is in accordance with the experimental results presented in the previous section. Note that, the accuracy of *BF* algorithm is 1 or 100%.

C. Comparison of Execution Time

In this experiment, we vary the number of robots/targets from 100 to 5000, and use an attack size of 10. The experimental results (Figure 5) show that the oblivious greedy algorithm has the lowest execution time, followed by the ordered greedy algorithm. The local search algorithm and the *2PG* algorithm are slower than the former two algorithms, and the local search algorithm outperforms the *2PG* algorithm as the number of robots/targets goes past 1000. The experimental results are in accordance with the time complexity analysis of the compared algorithms presented in the previous sections. Note that, other variants of the *OrG* and *LS* algorithm have similar execution time as the counterpart compared above.

Fig. 5. Comparison of execution time of the proposed algorithms with 2PG algorithm.

D. Evaluation of Accuracy for Large Problem Instances

In this section, we evaluate the accuracy of our proposed algorithms for large instances of the *RCM* problem. In case of large problem instances, it is not feasible to compute the residual coverage, because constructing an optimal attack requires exponential time with respect to the number of robots. Consequently, we resort to a non-optimal greedy attack model to compute an estimation of the residual coverage. We use attack model 2 (*A2*), outlined in Section IV, which is a greedy approximate attack model computable in polynomial time.

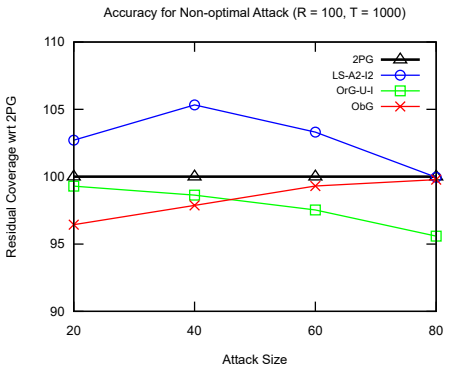

Fig. 6. Comparison of accuracy with Non-optimal Attack Model.

In this experiment, we use 100 robots and 1000 targets, vary the attack size in increments of 20, and report the relative accuracy with respect to the *2PG* algorithm. The experimental results (Figure 6) show that the accuracy of the proposed algorithms using non-optimal attack model *A2* is equivalent to the accuracy found in previous sections using an optimal attack model. The local search algorithm still has the best accuracy among the compared algorithms.

VI. Conclusion

In this work, we have proposed two algorithms for the coverage maximization problem with multiple robots in an adversarial setting. Our proposed algorithms have outperformed the state-of-the-art algorithm in terms of accuracy and/or execution time. We have demonstrated the effectiveness of our proposed solutions by conducting empirical studies.

In future, we intend to perform real world testing of our proposed algorithms in the context of practical applications of surveillance and patrolling. One may also consider reformulating the problem with a computationally feasible non-optimal attack model, and reevaluate the performance of the existing and proposed algorithms.

VII. Acknowledgement

We are grateful to Dr. Lifeng Zhou, and other members of RAAS lab at UMD for their feedback on this paper.

References

[1] L. Zhou, V. Tzoumas, G. J. Pappas, and P. Tokekar, "Resilient active target tracking with multiple robots," IEEE Robotics and Automation Letters, vol. 4, pp. 129–136, 2018.

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

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

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

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

[6] C. Robin and S. Lacroix, "Multi-robot target detection and tracking: taxonomy and survey," Autonomous Robots, vol. 40, pp. 729–760, 2016.

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

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

[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] E. Sless, N. Agmon, and S. Kraus, "Multi-robot adversarial patrolling: Facing coordinated attacks," in International Conference on Autonomous Agents and Multi-agent Systems, 2014, pp. 1093–1100.

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

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

[13] V. Tzoumas, A. Jadbabaie, and G. J. Pappas, "Resilient Non-Submodular Maximization over Matroid Constraints," arXiv: 1804.01013, 2018.

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

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

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

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

[18] A. Krause, and D. Golovin, "Submodular function maximization," 2014.

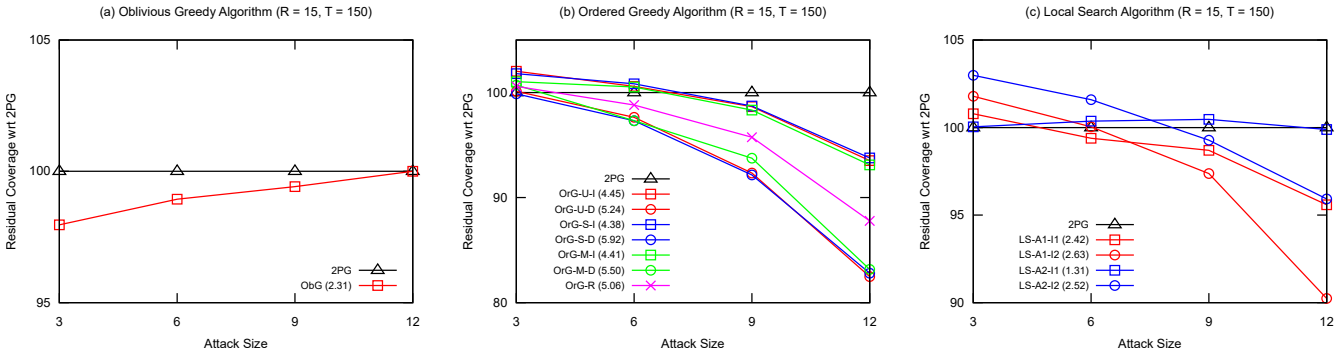

Fig. 7. Comparison of accuracy of (a) *ObG* algorithm, (b) *OrG* algorithm, and (c) *LS* algorithm with *2PG* algorithm for random dataset.

APPENDIX A
ACCELERATION OF THE LOCAL SEARCH ALGORITHM

In this section, we present a technique to make the local search algorithm (described in Section IV) execute faster. The main idea is to use a hash-based data structure so that we can reduce the time required to construct an attack.

First we compute the running time of a straightforward implementation of the attack function A. We use R and T to denote the number of robots and targets respectively. To construct the attack, we make α greedy choices. At each greedy iteration, we make R calls to the target coverage function F. F can be computed in O(T) time by maintaining a boolean array of size T that stores the coverage status of all the targets. Thus a straightforward implementation of A takes O(αRT) time.

Now we present how we can accelerate the process of constructing an attack. We present the methodology for both attack models $A1$ and $A2$ separately below. Here, S denotes the feasible solution that is passed as input parameter to the attack function A, and L_τ denotes the set of targets covered by the trajectory τ.

- Attack model 1 ($A1$): In case of $A1$, at each greedy iteration, the attacker selects the trajectory that maximizes the increase in target coverage. In our implementation of $A1$, we maintain a boolean array of size T, which stores, for each target t, if t is covered by at least one of the trajectories greedily selected thus far. Using the array, we determine the incremental coverage of a trajectory τ in O($|L_\tau|$) time.
- Attack model 2 ($A2$): In case of $A2$, at each greedy iteration, the attacker selects the trajectory that covers the maximum number of targets in U, where U is the set of targets which are covered by exactly one of the remaining trajectories. In our implementation of $A2$, for each target t, we store the set of trajectories in S which cover t. Let's denote the collection of all such sets by X. We implement U and the sets in X using hash-tables, which permit find, insertion, and delete operations in constant time.

  In the implementation of $A2$, when we greedily select a trajectory τ, we remove τ from all sets in X which include τ. If the size of any set in X reduces to one, we update U accordingly. Using the hash-based implementation of U and X, we determine the number of targets that are covered by exactly one trajectory after removal of a trajectory τ in O($|L_\tau|$) time.

In case of both attack models, using the above mentioned data structures, we can make each greedy choice in time O(Rt^*), where t^* denotes the maximum number of targets covered by one trajectory. This makes the overall computation time of the attack function O(αRt^*). Thus, using the above described acceleration technique, we achieve a performance speedup of a factor of $\frac{T}{t^*}$ over the straightforward algorithm, which is O(T) if we assume that t^* is a constant.

APPENDIX B
EXPERIMENTAL RESULTS FOR RANDOM DATASET

In this section, we present the experimental results for a synthetic dataset that is generated in a different way than the one mentioned in Section V. We call the new dataset *random* dataset. To generate a random dataset, we consider 4 trajectories for each robot. For each trajectory τ, we generate an integer n uniformly within the range [5, 15], which indicates the number of targets covered by τ. Then, from the set of targets, we select n targets with uniform probability and assign them to the trajectory τ.

Let's call the dataset used in the experiments in Section V *geometric* dataset. The random dataset is fundamentally different from the geometric dataset, because it does not have any underlying geometric structure. Consequently, experimentations using the random dataset demonstrate slightly different outcomes in comparison with geometric dataset.

Now we present the results of empirical evaluation of our proposed algorithms with random dataset. We conduct the same set of experiments as in Section V using the same values of the parameters. In Figure 7, we compare the accuracy of our proposed algorithms with the *2PG* algorithm. The experimental results differ from the geometric dataset in two ways. We observe that the accuracy of all *OrG* and *LS* variants

decreases with increase in attack size. Also, the average standard deviation of the relative accuracy is significantly lower in case of random dataset in comparison with geometric dataset.

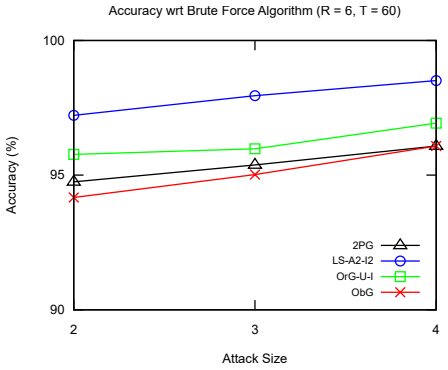

Fig. 8. Comparison of accuracy of proposed algorithms with *BF* algorithm for random dataset.

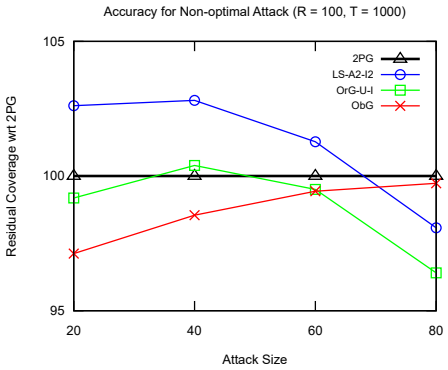

Fig. 9. Comparison of accuracy with Non-optimal Attack Model for random dataset.

Figure 8 shows the accuracy of our proposed algorithms in comparison with the brute force algorithm. We observe that the accuracy of the algorithms being compared is significantly higher in case of random dataset than the geometric dataset. But the order of the accuracy of the compared algorithms is preserved across the datasets. Figure 9 shows that the experimental results with non-optimal attack model for random dataset is similar to the results found using geometric dataset.

APPENDIX C
SENSITIVITY ANALYSIS

In the above experiments, we have assumed that the number of robot failures equals the attack size, α. But in reality, the two quantities will differ in most cases. For example, in a practical deployment of 10 robots which assumes a maximum attack size of 2 (i.e., $\alpha = 2$), there may be no robot failure. Hence, we present experimental results where the number of robot failures differs from the attack size.

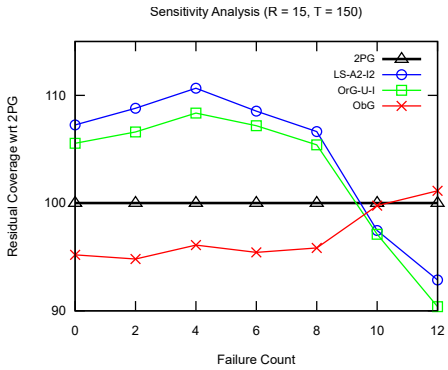

Fig. 10. Sensitivity analysis for geometric dataset.

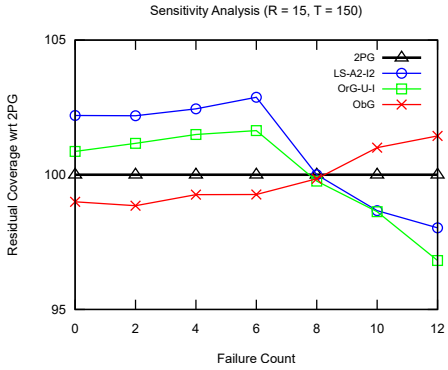

Fig. 11. Sensitivity analysis for random dataset.

In this experiment, we consider a scenario with 15 robots, 150 targets, and attack size, $\alpha = 6$. We vary the number of robot failures (worst case failure) in increments of 2 and report the relative accuracy of our proposed algorithms with respect to the *2PG* algorithm for both geometric (Figure 10) and random (Figure 11) datasets.

The experimental results show that, for both datasets, the accuracy of the proposed algorithms (*OrG-U-I* and *LS-A2-I2*) drops sharply if the number of robot failure exceeds α. If the number of robot failures is within the attack size (i.e., less than or equal to α), the proposed algorithms give better accuracy than the *2PG* algorithm.