Gradient-based Optimization for Multi-resource Spatial Coverage Problems

Nitin Kamra[1] **Yan Liu**[1]

[1]Department of Computer Science, University of Southern California, Los Angeles, California, USA

Abstract

Resource allocation for coverage of geographical spaces is a challenging problem in robotics, sensor networks and security domains. Conventional solution approaches either: (a) rely on exploiting spatio-temporal structure of specific coverage problems, or (b) use genetic algorithms when targeting general coverage problems where no special exploitable structure exists. In this work, we propose the coverage gradient theorem, which provides a gradient estimator for a broad class of spatial coverage objectives using a combination of Newton-Leibniz theorem and implicit boundary differentiation. We also propose a tractable framework to approximate the coverage objectives and their gradients using spatial discretization and empirically demonstrate the efficacy of our framework on multi-resource spatial coverage problems.

1 INTRODUCTION

Allocation of multiple resources for efficient spatial coverage is an important component in many practical systems, e.g., robotic surveillance, mobile sensor networks and green security domains. Surveillance tasks and sensor node placements generally involve assigning resources e.g. drones or sensors, each of which can monitor physical areas, to various points in a target domain such that a loss function associated with coverage of the domain is minimized [Renzaglia et al., 2012]. Alternatively, green security domains follow a leader-follower game setup between two agents, where a defender defends a continuous target density in a geographical area (e.g. trees in a protected forest) with limited resources to be placed, while an attacker plans an attack after observing the defender's placement strategy using its own resources [Tambe, 2011].

Related Work: Traditional methods used to solve multi-resource surveillance problems often make simplifying assumptions to devise tractable solution techniques. While we will survey these methods briefly, we will primarily focus on addressing a broad class of spatial coverage problems, where special spatial-temporal structure or symmetries cannot be exploited to efficiently allocate resources for coverage.

One of the earliest methods in this field deploys resources for coverage via construction of potential fields [Howard et al., 2002]. The fields are constructed such that each resource is repelled by both obstacles and by other resources, thereby forcing the network of resources to spread itself throughout the target domain to be covered. Poduri and Sukhatme [2004] extend this potential field method to maximize the area coverage of a domain via mobile sensors with the constraint that each sensor node has at least K neighbors in order to ensure good network coverage. If the target domain has uniform target density and the covering resources are assumed to have infinite coverage fields, then one can employ voronoi tessellation based methods [Dirafzoon et al., 2011]. Notably, these approaches and others [Johnson et al., 2012, Huang et al., 2020] make simplifying assumptions on the target domain to be covered or on the covering resources and focus on exploiting the resulting symmetry structures.

Other approaches to coverage and allocation often discretize the domain to be covered and employ specialized decompositions, for instance, Kong et al. [2006] employ the Boustrophedon decomposition in case of a robot coverage problem. Similarly, many exact and approximate approaches proposed in green security game domains to compute strategies against a best responding attacker rely on discretizing the target area into grid cells and restrict the players' actions to discrete sets to find optimal allocations using linear programming (LP) or mixed-integer programming (MIP) [Kiekintveld et al., 2009, Yang et al., 2014, Fang et al., 2013, Haskell et al., 2014, Yin et al., 2014]. However, approaches which directly discretize the action spaces suffer from intractability of computation since the size of the discretized action space grows exponentially with the number of resources to be placed.

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

In such cases, one has to rely on undirected exploration methods such as particle swarm optimization and genetic algorithms. Nazif et al. [2010] propose a mechanism for covering an area by means of a group of homogeneous agents through a single-query roadmap. Saska et al. [2014] propose an algorithm for autonomous deployment of micro-aerial vehicles for cooperative surveillance satisfying motion constraints, environment constraints and localization constraints via particle swarm optimization. Tong et al. [2009] propose a regional service coverage maximization algorithm which solves the problem heuristically using a genetic algorithm. Krishnamurthy and Khorrami [2016] present a solution to the problem of optimal placement of sensors for monitoring a spatial road network based on an iterative genetic algorithm for the optimization of a scalar metric computed from the spatial integration of the sensor influence wave. Similarly, Yakoubi and Laskri [2016] propose an evolutionary approach for vacuum cleaner coverage of a cleaning area. However, since the coverage problem is generally combinatorially hard, such undirected search methods also do not scale well with growing number of resources to be placed.

To address this, recent works in spatial coverage domains have focused on incorporating advances from deep learning and reinforcement learning. For instance, Pham et al. [2018] focus on multi-UAV coverage of a field of interest using a model-free reinforcement learning method. Kamra et al. [2019] have proposed DeepFP, a fictitious play based method to solve green security games in continuous action spaces, which relies on neural networks to provide a differentiable approximation to the coverage objectives. However, these require approximating discontinuous and complex multi-resource coverage objectives using continuous and smooth neural network approximators, which can lead to subsequent inaccuracies in resource placements.

Contributions: To address the above challenges, we propose the *coverage gradient theorem*, which provides a gradient estimator for a broad class of spatial coverage objectives using a combination of Newton-Leibniz theorem and implicit boundary differentiation. This alleviates the need to use function approximators like neural networks to approximate gradients of the coverage objectives. We further present a tractable framework to approximate the coverage objectives and their gradients using spatial discretization of only the target domain, but not the allocated positions of the resources. Hence, we keep the resource allocations amenable to gradient-based optimization thereby leading to scalable and more directed ways of search and optimization for multi-resource coverage problems. Our generalized formulation supports multiple agents, multiple resources, arbitrary (even non-convex) shapes for target domains and arbitrary continuous distributions for targets. By combining our framework with existing optimization methods, we demonstrate successful applications on both surveillance and green security spatial coverage domains.

2 MULTI-RESOURCE SPATIAL COVERAGE PROBLEMS

In this section, we formally introduce notation and definitions for multi-resource spatial coverage problems along with example applications, which will be used for evaluation.

Multi-resource spatial coverage: Spatial coverage problems comprise of a target space $Q \subset \mathbb{R}^d$ (generally $d \in \{2,3\}$) and a set of m resources. **Action**: An action $u \in \mathbb{R}^{m \times \hat{d}}$ is the placement of all m resources in an appropriate coordinate system of dimension $\hat{d}$. **Coverage**: When placed, each resource covers (often probabilistically) some part of the target space Q. Let $\text{cvg} : q \times u \to \mathbb{R}$ be a function denoting the coverage of a target point $q \in Q$ due to action u. We do not assume a specific form for the coverage cvg and leave it to be defined flexibly, to allow many different coverage applications to be amenable to our framework. **Reward**: The scalar coverage reward due to action u is defined as: $r(u) = \int_Q \text{cvg}(q,u)\,\text{imp}(q)\,dq$, where $\text{imp}(q)$ denotes the importance of the target point q. The objective is to optimize the placement reward r w.r.t. action u.

Example 1 (Single-agent Areal Surveillance). *A single agent, namely the defender (D), allocates m areal drones with the i^{th} drone D_i having three-dimensional coordinates $u_{D,i} = (p_{D,i}, h_{D,i}) \in [-1,1]^2 \times [0,1]$ to surveil a two-dimensional forest $Q \subset [-1,1]^2$ of arbitrary shape and with a known but arbitrary tree density $\rho(q)$. Consequently, $u_D \in \mathbb{R}^{m \times 3}$. Each drone has a downward looking camera with a circular lens and with a half-angle θ such that at position $(p_{D,i}, h_{D,i})$, the drone D_i sees the set of points $S_{D,i} = \{q \,|\, ||q - p_{D,i}||_2 \leq h_{D,i} \tan\theta\}$. A visualization of this problem with $m = 2$ drones is shown for a sample forest in Figure 1a. We assume a probabilistic model of coverage with a point q being covered by drone D_i with probability $P_H(h_{D,i}) = e^{K(h_{opt} - h_{D,i})} \left(\frac{h_{D,i}}{h_{opt}}\right)^{Kh_{opt}}$ if $q \in S_{D,i}$ and 0 otherwise. With multiple drones, the probability of a point q being covered can then be written as: $\text{cvg}(q, u_D) = 1 - \prod_{i|q \in S_{D,i}} \bar{P}_H(h_{D,i})$ where $\bar{P}_H$ stands for $1 - P_H$. Hence, the reward function to be maximized is: $r_{D,1p}(u_D) = \int_Q \left(1 - \prod_{i|q \in S_{D,i}} \bar{P}_H(h_{D,i})\right) \rho(q) dq$ with the tree density $\rho(q)$ being the importance of target point q (subscript $1p$ denotes one agent).*

While the above description suffices for single player games, it can be easily extended to multi-agent games with a set of agents (or players). In such a case, the solution concept is to compute the mixed strategy Nash equilibria for all players. For brevity, we provide the extended notation and the Nash equilibria concepts associated with it in appendix A.1 for the interested reader. These can be helpful for understanding our second example domain described below:

Example 2 (Two-agent Adversarial Coverage). *Two agents,* *namely the defender D and the attacker A, compete in a zero-sum game. The defender allocates m areal drones with the same coverage model as in example 1. The attacker controls n lumberjacks each with ground coordinates $u_{A,j} \in [-1,1]^2$ to chop trees in the forest Q. Consequently, $u_A \in \mathbb{R}^{n\times 2}$. Each lumberjack chops a constant fraction κ of trees in a radius R_L around its coordinates $u_{A,j}$. We denote the area covered by the j-th lumberjack as $S_{A,j} = \{q \mid \|q - p_{A,j}\|_2 \leq R_L\}$. A visualization of this problem with $m = n = 2$ is shown for a sample forest in Figure 1b. A drone can potentially catch a lumberjack if its field of view overlaps with the chopping area. For a given resource allocation $u = (u_D, u_A)$, we define $I_j = \{i \mid \|p_{A,j} - p_{D,i}\|_2 \leq R_L + h_{D,i}\tan\theta\}$ as the set of all drones which overlap with the j-th lumberjack. The areal overlap $\alpha_{ij} = \int_{S_{D,i}\cap S_{A,j}} dq$ controls the probability of the j-th lumberjack being caught by the i-th drone: $P_C(h_{D,i}, \alpha_{ij}) = P_H(h_{D,i})\, P_A(\alpha_{ij})$ where P_H is the same as that in example 1 and captures the effect of drone's height on quality of coverage, while $P_A(\alpha_{ij}) = 1 - \exp\left(-\frac{K_a \alpha_{ij}}{\pi R_L^2}\right)$ captures the effect of areal overlap on probability of being caught. Hence, the reward achieved by the j-th lumberjack can be computed as: $r_{A,j}(u_D, u_{A,j}) = \kappa \int_{S_{A,j}\cap Q} \rho(q)dq$ with probability $\prod_{i\in I_j} \bar{P}(h_{D,i}, \alpha_{ij})$, and $-\kappa \int_{S_{A,j}\cap Q} \rho(q)dq$ otherwise i.e. the number of trees chopped if the j-th lumberjack is not caught by any drone or an equivalent negative penalty if it is caught. Hence, the total agent rewards are: $r_{A,2p}(u_D, u_A) = -r_{D,2p}(u_D, u_A) = \sum_j r_{A,j}(u_D, u_{A,j})$ (subscript $2p$ denotes two-agent). Both agents are expected to compute the mixed-strategy Nash equilibria over their respective action spaces.*

Note that in the above examples drones provide best probabilistic coverage at a height h_{opt}. By increasing their height, a larger area can be covered at the cost of deterioration in coverage probability. Further, the defender can increase coverage probability for regions with high tree density by placing multiple drones to oversee them; in which case, the drones can potentially stay at higher altitudes too. Example 2 adds additional interactions due to overlaps between defender and attacker's resources[1]. Hence, these examples form a challenging set of evaluation domains with multiple trade-offs and complex possibilities of coverage involving combinatorial interactions between the players' resources. For both examples, we use the following constants: $\theta = \frac{\pi}{6}$, $h_{opt} = 0.2$, $K = 4.0$, $R_L = 0.1$, $K_a = 3.0$, $\kappa = 0.1$. However, note that these values only serve as practical representative values. The techniques that we introduce in this paper are not specific to the above probabilistic capture models or specific values of game constants, but rather apply to a broad class of coverage problems where the agents act by placing resources with finite coverage fields and agents' rewards are of the form: $r_p(u) = \int_Q f_p(u,q)dq$.

[1]In reality, lumberjacks might act independent of each other and lack knowledge of each others' plans. By allowing them to be placed via a single attacker and letting them collude, we tackle a more challenging problem and ensure that not all of them get caught by independently going to strongly covered forest regions.

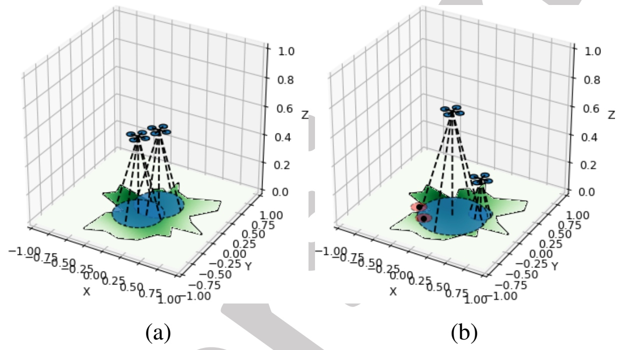


Figure 1: (a) Areal surveillance example with an arbitrary forest and $m = 2$ drones, (b) Adversarial coverage example with $m = 2$ drones and $n = 2$ lumberjacks (red circles).

3 METHODS

The key idea behind our solution approach is to obtain the gradient of the expected coverage reward of the agent/(s) w.r.t. the agents' actions. This can then be used to perform direct gradient ascent to arrive at a (locally) optimal action or as a part of other global search algorithms.

3.1 DIFFERENTIABLE APPROXIMATION FOR COVERAGE OBJECTIVES

First, we propose a method to approximate coverage objectives and their gradients w.r.t. agents' actions. Consider an objective of the form:

$$r(u) = \int_Q f(u,q)\, dq \tag{1}$$

where u denotes actions of one or more agents having multiple resources to place at their disposal and q is any point in the target domain Q. We assume that the action u has m components with u_i representing the location of i-th resource ($i \in [m]$) and $u_{\setminus i}$ representing the locations of all resources other than i. Note that the imp(q) function has been subsumed into $f(u,q)$ in this formulation.

We are interested in computing the gradient: $\frac{\partial r}{\partial u_i}$. However, this is a hard problem since: (a) $r(u)$ involves integration over arbitrary (non-convex shaped) target domains which does not admit a closed-form expression in terms of elementary functions and hence cannot be differentiated with autograd libraries like PyTorch and TensorFlow, and (b) most resources have a finite coverage area, outside of which the coverage drops to zero. This often makes the function $f(u,q)$ discontinuous w.r.t. q given a fixed u especially at the coverage boundaries induced by the resources' coordinates, for e.g., drones have a circular probabilistic coverage area governed by their height and camera half-angle θ, outside which the coverage probability suddenly drops to zero.

At this point, one might wonder why the issue of $f(u, q)$ being discontinuous cannot be solved by just using an approximation to the probabilistic coverage functions, e.g. P_H and P_A in examples 1 and 2, which smooths their boundaries. It turns out that smoothing the coverage function creates "blind spots", such that the solution technique believes that a certain target area is well-covered, e.g., by two or more drones' overlapping (smoothed) boundaries, but in fact the area is completely uncovered since the actual coverage boundaries terminate abruptly. The effects of smoothing are most apparent in two-player adversarial problems where the attacker wants to exploit any loopholes in the defender's allocation. Such blind spots are easily exploited by the attacker to achieve high rewards. In such cases, changing the coverage function to a discontinuous one significantly alters the computed Nash equilibrium allocation distributions for the players. Hence, we propose a method to compute the gradient $\frac{\partial r}{\partial u_i}$ without approximating any discontinuous boundaries.

Theorem 1 (Coverage Gradient Theorem)**.** *Let the objective function be as shown in eq 1: $r(u) = \int_Q f(u, q)\, dq$. Denoting the set of points covered by the i-th resource as S_i, the interior of a set with in$(\cdot)$ and the boundary with $\delta(\cdot)$, the gradient of $r(u)$ w.r.t. the i-th resource's location u_i is given by:*

$$\frac{\partial r(u)}{\partial u_i} = \int_{\mathit{in}(Q \cap S_i)} \frac{\partial f(u, q)}{\partial u_i}\, dq + \int_{Q \cap \delta S_i} \big(f(u, q) - f(u_{\setminus i}, q)\big) \frac{\partial q_{Q \cap \delta S_i}}{\partial u_i}^T n_{q_{Q \cap \delta S_i}}\, dq \quad (2)$$

Proof. We begin by observing that while function f can be potentially discontinuous in q across resources' coverage boundaries due to finite coverage fields of resources, $r(u)$ integrates over $q \in Q$ thereby removing the discontinuities. Hence, instead of directly taking the derivative w.r.t. a particular resource's location u_i inside the integral sign, we first split the integral into two parts - over the i-th resource's coverage area S_i and outside it:

$$r(u) = \int_{Q \cap S_i} f(u, q)\, dq + \int_{Q \setminus S_i} f(u, q)\, dq \quad (3)$$

Splitting the integral at the boundary of the discontinuity allows us to explicitly capture the effect of a small change in u_i on this boundary. Denoting the interior of a set with in$(\cdot)$ and the boundary with $\delta(\cdot)$, the derivative w.r.t. u_i can be expressed using the Newton-Leibniz formula as:

$$\begin{aligned} \frac{\partial r(u)}{\partial u_i} = &\int_{\text{in}(Q \cap S_i)} \frac{\partial f(u, q)}{\partial u_i}\, dq \\ &+ \int_{\delta(Q \cap S_i)} f(u, q) \frac{\partial q_{\delta(Q \cap S_i)}}{\partial u_i}^T n_{q_{\delta(Q \cap S_i)}}\, dq \\ &+ \int_{\text{in}(Q \setminus S_i)} \frac{\partial f(u_{\setminus i}, q)}{\partial u_i}\, dq \\ &+ \int_{\delta(Q \setminus S_i)} f(u_{\setminus i}, q) \frac{\partial q_{\delta(Q \setminus S_i)}}{\partial u_i}^T n_{q_{\delta(Q \setminus S_i)}}\, dq, \end{aligned} \quad (4)$$

where $\frac{\partial q_{\delta(Q \cap S_i)}}{\partial u_i}$ denotes the boundary velocity for $\delta(Q \cap S_i)$ and $n_{q_{\delta(Q \cap S_i)}}$ denotes the unit-vector normal to a point q on the boundary $\delta(Q \cap S_i)$ (similarly for $\delta(Q \setminus S_i)$). Since $f(u_{\setminus i}, q)$ does not depend on u_i, we can set $\frac{\partial f(u_{\setminus i}, q)}{\partial u_i} = 0$. Next observe that the boundaries can be further decomposed as: $\delta(Q \cap S_i) = (\delta Q \cap S_i) \cup (Q \cap \delta S_i)$ and similarly $\delta(Q \setminus S_i) = (\delta Q \setminus S_i) \cup (Q \cap \delta S_i)$. However since u_i does not change the boundary of the target domain δQ, we have:

$$\frac{\partial q_{\delta Q \cap S_i}}{\partial u_i} = 0, \quad \forall q \in \delta Q \cap S_i \quad (5)$$

$$\frac{\partial q_{\delta Q \setminus S_i}}{\partial u_i} = 0, \quad \forall q \in \delta Q \setminus S_i \quad (6)$$

Further on the boundary of S_i, the following unit-vectors normal to the boundary are oppositely aligned:

$$n_{q_{\delta(Q \setminus S_i)}} = -n_{q_{\delta(Q \cap S_i)}} \ \forall q \in Q \cap \delta S_i. \quad (7)$$

Substituting the above results, we can simplify the gradient expression in eq 4 to:

$$\frac{\partial r(u)}{\partial u_i} = \int_{\text{in}(Q \cap S_i)} \frac{\partial f(u, q)}{\partial u_i}\, dq + \int_{Q \cap \delta S_i} \big(f(u, q) - f(u_{\setminus i}, q)\big) \frac{\partial q_{Q \cap \delta S_i}}{\partial u_i}^T n_{q_{Q \cap \delta S_i}}\, dq \quad (8)$$

□

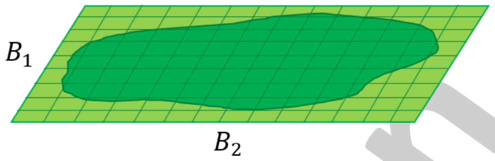
Note that the first term in eq 2 corresponds to the change in f inside the coverage area of resource i due to a small change in u_i, while the second term elegantly factors-in the effects of movement or shape change of the coverage area boundary due to changes in u_i (e.g. when a drone moves or elevates in height). This allows us to mitigate the discontinuities due to finite coverage fields of resources. While we show the general result here, the term $\frac{\partial q_{Q \cap \delta S_i}}{\partial u_i}^T n_{q_{Q \cap \delta S_i}}$ can be simplified further using implicit differentiation of the boundary of S_i, which depends on the particular game under consideration. We show the simplification for our example domains in section A.2 in the appendix.

3.2 DISCRETIZATION-BASED APPROXIMATION FRAMEWORK

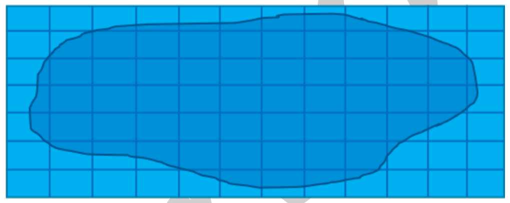
While we now have a general form for $r(u)$ and $\frac{\partial r}{\partial u}$, both forms comprise of non closed-form integrals over the target domain Q or its subsets. While evaluating r and $\frac{\partial r}{\partial u}$ in practice, we adopt a discretization based approach to approximate the integrals. Given a target domain $Q \subset \mathbb{R}^d$ with $d \in \{2, 3\}$, we discretize the full $\mathbb{R}^d$ space into $B_1, \ldots, B_d$ bins respectively in each of the d dimensions (see figure 2a).
Approximating spatial maps: All spatial maps i.e. functions over the target domain Q (e.g. $f(u, q)$), are internally represented as *real tensors* of dimension d with size: $(B_1, \ldots, B_d)$ (see figure 2b).
Approximating sets: All geometric shapes (or sets of points) including S_i for all resources (e.g., the circular coverage areas of drones and lumberjacks) and the target domain Q itself (e.g., the irregular shaped forest) are converted to *binary tensors* each of dimension $d + 1$ with size: $(B_1, \ldots, B_d, 3)$. The final dimension of length 3 denotes interior, boundary and exterior of the geometric shape respectively, i.e. a binary tensor T has $T_{b_1,\ldots,b_d,0} = 1$ if the bin at index $(b_1, \ldots, b_d)$ is inside the geometric shape, $T_{b_1,\ldots,b_d,1} = 1$ if the bin is on the boundary of the geometric shape and $T_{b_1,\ldots,b_d,2} = 1$ if the bin is outside the geometric shape (see figure 2c).
Approximating operators: Doing the above discretization requires an efficient function for computing the *binary tensors* associated with the $\text{in}(\cdot)$ and the $\delta(\cdot)$ operators. This is performed by our efficient divide-and-conquer shape discretizer, which is presented in section A.6 due to space constraints. The other set operations are approximated as follows: (a) set intersections are performed by element-wise *binary tensor* products, (b) integrals of spatial maps over geometric sets are approximated by multiplying (i.e. masking) the *real tensor* corresponding to the spatial map with the *binary tensor* corresponding to the geometric set followed by an across-dimension sum over the appropriate set of axes.

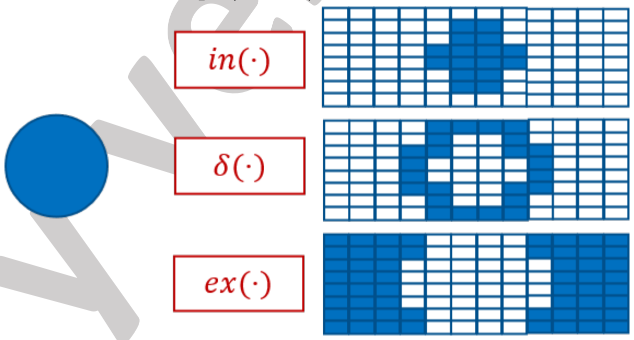
Scaling: While our discretized bins growing exponentially with dimension d of the target domain may come off as a limitation, our method still scales well for most real-world coverage problems since they reside on two or three-dimensional target domains. Note that unlike previous approaches which discretize the target domain and simultaneously restrict the agents' actions to discrete bins [Yang et al., 2014, Haskell et al., 2014], we do not discretize the actions u of agents. Hence, we do not run into intractability induced by discretizing high-dimensional actions of agents owning multiple resources and we keep u amenable to gradient-based optimization.

Using the framework: Our proposed framework essentially acts as an autograd module for $r(u)$ differentiable w.r.t. input u, which provides both the forward and the backward calls (i.e. evaluation and gradients). Hence, it can now



(a) Discretize the target space $\mathbb{R}^d$ (but not action space) into bins


(b) Approximate all spatial maps e.g., $f(u, q)$ as real tensors of shape (B_1, B_2)



(c) Approximate all sets e.g., spatial coverage field S_i of each resource and the target domain Q as binary tensors of shape $(B_1, B_2, 3)$

Figure 2: Illustration of spatial discretization-based framework for 2-D target domains.

be used for direct gradient-based optimization solutions to multi-resource coverage problems. We describe our solution approaches in the next section.

3.3 SOLUTION APPROACHES

For the single agent surveillance domain, we compare the following solution approaches:

1. *Genetic algorithm* [*gen*]: We run a genetic algorithm as shown in algorithm 1 to search for near-optimal resource allocations (with population size $K = 6$ and $max\_itr = 1000$).
2. *Gradient ascent* [*ga*]: We perform gradient ascent on a differentiable approximation to the coverage objective $r_D(u_D)$, thereby converging at a (locally) optimal value of u_D:
   (a) *Neural nets* [*_nn*]: We train feedforward neural networks to approximate the coverage objective and its gradients.

(b) *Graph neural nets* [*_gnn*]: We train graph neural networks to approximate the coverage objective and its gradients.

(c) *Our framework* [*_diff*]: We use our spatial discretization based framework and the coverage gradient theorem to approximate the coverage objective and its gradients.

3. *Augmented genetic algorithm* [*agen*]: We augment the genetic algorithm as shown in algorithm 1, line 11 by having an inner-loop which performs gradient ascent on all population members in every iteration of the algorithm. We use population size $K = 6$, $max\_itr = 1000$ and 100 inner-loop gradient ascent iterations. We again have the three variants: [*_nn*], [*_gnn*] and [*_diff*] based on where the gradients come from.

For two-agent adversarial games, we employ the DeepFP algorithm [Kamra et al., 2019], which is based on fictitious play. Briefly summarized in algorithm 2, it obtains a differentiable approximation to the reward functions $r_{D,2p}$ and $r_{A,2p}$, creates an empty memory to store a non-parametric representation of the agents' mixed strategies $\sigma = (\sigma_D, \sigma_A)$ and initializes best responses for both agents randomly [lines 1-3]. Then it alternatively updates: (a) the agents' strategies, by storing the current best responses in the memory [line 5], and (b) the best responses, by maximizing each agent p's differentiable reward function against a batch of samples drawn from the other agent's strategy σ_{-p} [lines 6-8]. We point the readers to Kamra et al. [2019] for details of the algorithm. In our implementation, we used a modified version of the DeepFP algorithm to apply it to our setting. The modifications made and the reasons behind them have been described in detail in section A.3 in the appendix. The DeepFP hyperparameters used can be found in section A.5 in the appendix. Again we use neural nets [*_nn*], graph neural nets [*_gnn*] and our approximation framework [*_diff*] to obtain the gradients of the coverage objective and compare these variants empirically.

Algorithm 1: A Genetic Algorithm for Resource Allocation in Spatial Coverage Problems

```
Result: Final action $u$
1  Required: Coverage reward $r(u)$ (a.k.a. fitness function);
2  Initialize a population of $K$ actions $u_{1:K}$ each $\in \mathbb{R}^{m \times d}$;
3  for $itr \in \{1, \dots, max\_itr\}$ do
       /* Evaluate population members */
4      Compute fitness $r(u_i)$ of population member $u_i\ \forall i \in 1:K$;
       /* Ranking */
5      Sort all population members in decreasing order of fitness;
       /* Cross-over */
6      Copy the top $K/3$ fittest population members;
7      Make a shuffled copy of these top $K/3$ members;
8      Between each pair of the original and shuffled copies, swap the corresponding resource placements with probability $0.5$ generating 2 new members per pair;
9      Discard the bottom $2K/3$ population and replace them with the newly crossed-over copies;
       /* Perform mutation */
10     Randomly perturb the coordinates of the newly generated $2K/3$ copies by appropriate amounts (we use uniform random numbers between $[-0.1, 0.1]$ per coordinate);
       /* Perform inner-loop gradient ascent if augmented genetic algorithm */
11     In the augmented genetic algorithm variant, apply a fixed number of gradient ascent iterations to each population member using gradients from a differentiable approximation $\hat{r}(u)$ to $r(u)$;
12 Return $\arg\max_{u \in u_{1:K}} \hat{r}(u)$;
```

Algorithm 2: DeepFP

```
Result: Final strategies $\sigma_D, \sigma_A$ in mem
1  Obtain a differentiable approximation $\hat{r} = (\hat{r}_D, \hat{r}_A)$ to the reward functions: $(r_{D,2p}, r_{A,2p})$;
2  Initialize best responses $(br_D, br_A)$ randomly;
3  Create empty memory mem to store $\sigma = (\sigma_D, \sigma_A)$;
4  for $game \in \{1, \dots, max\_games\}$ do
       /* Update strategies */
5      Update $\sigma$ by storing best responses $\{br_D, br_A\}$ in mem;
       /* Update best responses */
6      for agent $p \in \{D, A\}$ do
7          Draw samples $\{u^i_{-p}\}_{i=1:bs}$ from $\sigma_{-p}$ in mem;
8          $br_p := \max_{u_p} \frac{1}{bs} \sum_{i=1}^{bs} \hat{r}_p(u_p, u^i_{-p})$;
```

4 EXPERIMENTS

In our experiments on both our application domains, we differentiably approximate rewards using the following variants: (a) feedforward neural networks [*nn*], (b) graph neural networks [*gnn*], and (c) our approximation framework [*diff*]. For the *nn* and *gnn* baselines, we trained neural networks, one per forest and per value of m (and n for two-agent games), to predict the reward of the defender (and attacker in case of two-agent game) by minimizing the MSE loss using the Adam optimizer. The neural networks take as input the action u_D of the defender (and u_A also for two-agent game) and output a prediction for the reward $\hat{r}_{D,1p}$ ($\hat{r}_{D,2p}$ and $\hat{r}_{A,2p}$ for two-agent game). Please see section A.5 in appendix for network architectures and hyperparameters. We also represent best responses with the following variants: (a)

Table 1: Maximum reward averaged across forest instances achieved for Areal Surveillance domain.

	$m = 1$	$m = 2$	$m = 4$	$m = 8$
gen	**9378.46 ± 660.27**	16061.02 ± 940.34	24857.09 ± 1593.90	33749.89 ± 2949.36
ga_diff	9364.07 ± 660.55	**16086.24 ± 923.84**	**25109.58 ± 1552.05**	**34364.64 ± 3168.55**
ga_nn	9337.57 ± 680.45	14308.12 ± 1070.00	19211.01 ± 2233.19	19127.45 ± 2498.12
ga_gnn	9291.36 ± 665.65	14082.38 ± 1073.62	19075.09 ± 1378.36	19657.22 ± 2346.1
agen_diff	**9374.36 ± 660.56**	**16091.18 ± 927.46**	**25122.13 ± 1555.55**	**34792.45 ± 2924.52**
agen_nn	9351.67 ± 674.55	14348.55 ± 1057.19	19236.34 ± 2229.72	19563.83 ± 2378.31
agen_gnn	9307.41 ± 676.78	14207.96 ± 1044.29	19652.45 ± 1712.13	20286.63 ± 2339.48

stochastic best response nets [*brnet*] as originally done by DeepFP, and (b) our deterministic evolutionary population [*pop*K] with K being the population size (see section A.3 in the appendix for why this modification is useful). We use $d = 2$ dimensional forests and discretize them into $B_1 = B_2 = 200$ bins per dimension for a total of $40K$ bins when using our framework.

4.1 RESULTS ON AREAL SURVEILLANCE DOMAIN

We show the experiment results achieved by using all methods: *gen*, *ga_diff*, *ga_nn*, *ga_gnn*, *agen_diff*, *agen_nn* and *agen_gnn* for different values of $m \in \{1, 2, 4, 8\}$ over 5 different forest instances differing in shape and tree density. The maximum true reward $r_{D,1p}$ achieved by all methods averaged over all the forest instances is shown in Table 1.

It is clear that *agen_diff* always achieves the maximum true reward for nearly all values of m (except $m = 1$ due to stochasticity of genetic algorithms). This is because *gen* only performs undirected global search, while the *ga* variants perform only directed local optimization with gradient ascent. The *agen* variants are the only ones which combine the undirected global search of genetic algorithms with local optimization of gradient-based optimization and hence outperform other baselines. Figure 3 shows the final locations computed for a randomly chosen forest and with $m = 2$ for all methods.

Amongst the *diff*, *nn* and *gnn* variants, the *diff* variants always outperform the other two since our approximation framework is quite precise while neural networks become more inaccurate at approximating the coverage objective and its gradients, especially as m increases and the objective becomes combinatorially harder to approximate. This is also reflected in the plots of true reward achieved vs training iterations shown in Figure 4 for simple gradient ascent (*ga*) variants. Since *diff* variants are unbiased approximators of the true reward[2], the true reward continues to increase till convergence for *diff*. For *nn* and *gnn* variants, the true reward increases initially but eventually goes down as the defender action u_D begins to overfit the potentially inaccurate approximations made by *nn* and *gnn*.

[2]The only bias in *diff* is the discretization bin sizes, which can be made arbitrarily small in principle.

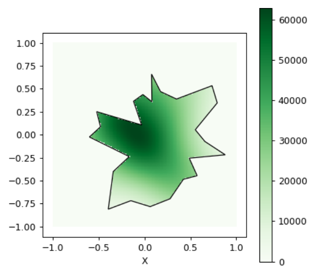
Next, we compare the runtimes of all algorithms for the Areal Surveillance domain in Table 2. Note that all [*ga_*] variants employ 100 iterations of gradient ascent each. The genetic algorithm [*gen*] uses $max\_itr = 1000$ iterations to converge, while the augmented variants [*agen_*] converge much faster and use $max\_itr = 10$ outer loop iterations and 100 inner loop gradient ascent iterations each. We observe that the [*_diff*] variants are always faster than [*_nn*] and [*_gnn*] variants and the best performing [*agen_*] variant, namely [*agen_diff*], only requires a little bit more time than [*ga_*] variants and [*gen*].

Table 2: Runtimes of algorithms (in seconds).

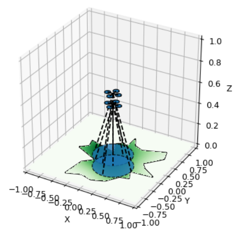
	$m = 1$	$m = 2$	$m = 4$	$m = 8$
gen	3.5	3.5	4.5	6
ga_diff	3.6	4	4.4	5.2
ga_nn	4.2	4.6	4.8	5.2
ga_gnn	3.6	4.4	5	5.4
agen_diff	4.92	4.45	4.62	6.14
agen_nn	5.33	4.55	4.68	8.51
agen_gnn	8.45	12.45	12.6	13.46

4.2 RESULTS ON ADVERSARIAL COVERAGE GAME

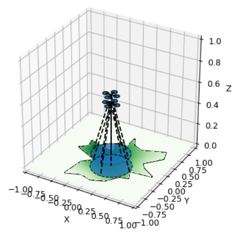
We implemented different variants of DeepFP with variations of differentiable reward models in {*nn*, *gnn*, *diff*} along with variations of best responses in {*brnet*, *pop4*}. We measured the exploitability $\epsilon_D(\sigma_D)$ of the defender strategy found by all methods to compare them against each other. To compute the exploitability of the defender strategy found by any variant of DeepFP, we froze the defender strategy σ_D and directly maximized $\mathbb{E}_{u_D \sim \sigma_D}[\hat{r}_A(u_D, u_A)]$ w.r.t. u_A with $\hat{r}_A$ being approximated by *diff*. This is a single-agent objective and can be directly maximized with gradient ascent. We perform 30 independent maximization runs to avoid reporting local maxima and report the best as the exploitability. Note that nash equilibrium strategies are the least exploitable strategies, hence lower the value of $\epsilon_D(\sigma_D)$ found, the closer σ_D is to the nash equilibrium strategy.



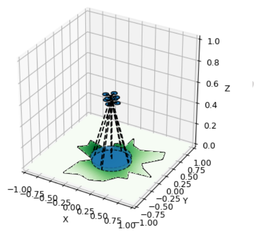
(a) Forest tree density



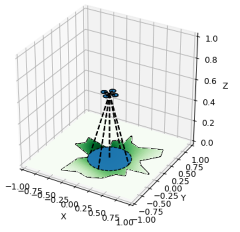
(b) Action found via *gen*



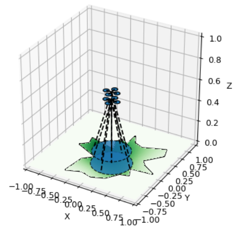
(c) Action found via *ga_diff*



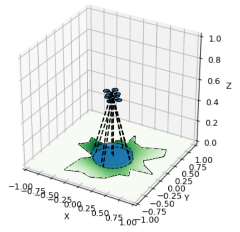
(d) Action found via *ga_nn*



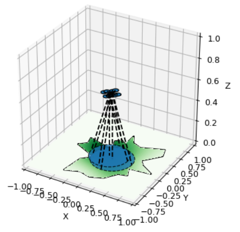
(e) Action found via *ga_gnn*



(f) Action found via *agen_diff*

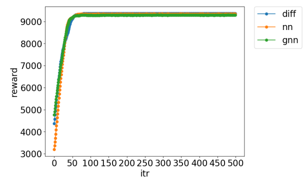


(g) Action found via *agen_nn*

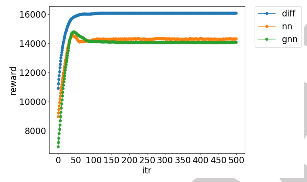


(h) Action found via *agen_gnn*

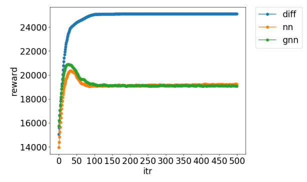
Figure 3: Visualizing final actions for a randomly chosen forest with $m = 2$.



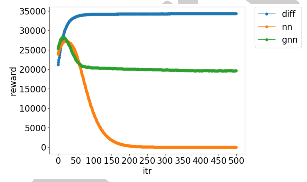
(a) $m = 1$



(b) $m = 2$



(c) $m = 4$



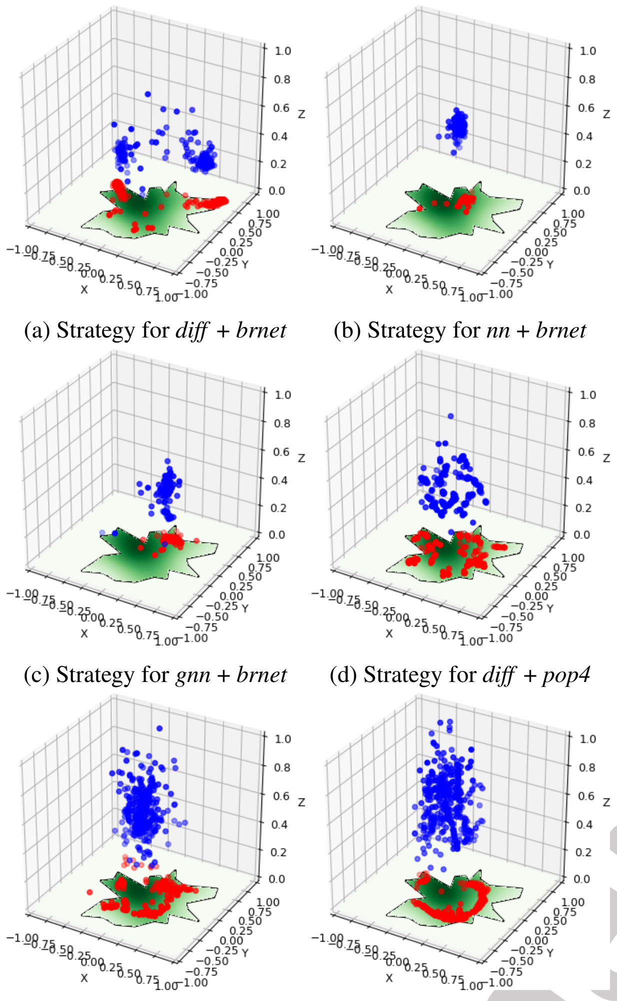
(d) $m = 8$

Figure 4: Plots of true reward achieved by *diff*, *nn* and *gnn* variants over gradient ascent iterations for $m \in \{1, 2, 4, 8\}$.

Table 3 shows the exploitability values for different variants of DeepFP. We observe that the exploitability when best responses are approximated by a population-based variant with $K = 4$ is always lower than that of stochastic best response networks employed by original DeepFP. Further, with few agent resources $m = n = 1$, the exploitability across *diff, nn* and *gnn* is nearly similar but the disparity increases for larger number of agent resources and *diff* dominates over *nn* and *gnn* with less exploitable defender strategies. Notably, the original DeepFP (*nn* + *brnet*) is heavily exploitable while our proposed variant (*diff* + *popK*) is the least exploitable. In Figure 5, we show a visualization of the points sampled from the defender and attacker's strategies for $m = n = 2$ case on the same forest from Figure 3a. The visualization confirms that *diff* + *popK* covers the dense core of the forest with the defender's drones so the attacking lumberjacks attack only the regions surrounding the dense core, while *nn* + *brnet* drones often gets stuck and concentrated in a small region thereby allowing lumberjacks to exploit the remaining dense forest. Please also see section A.4 in the appendix exploring the trade-offs in the choice of population size K.

Table 3: Exploitability of the defender from DeepFP variants averaged across forest instances.

$\epsilon_D(\sigma_D)$	m=n=1	m=n=2	m=n=4
		brnet	
diff	209.78	399.95	559.36
(ours)	±49.94	±57.70	±164.21
nn	203.92	323.00	787.53
	±54.67	±39.55	±194.82
gnn	204.55	307.74	597.23
	±50.72	±62.67	±125.01
		pop4 (ours)	
diff	116.41	**141.09**	**141.54**
(ours)	±15.02	± **13.90**	± **26.60**
nn	**113.61**	208.23	339.31
	± **6.92**	±22.76	±116.77
gnn	113.99	176.25	172.30
	±13.74	±15.21	±34.08



(a) Strategy for *diff* + *brnet*

(b) Strategy for *nn* + *brnet*

(c) Strategy for *gnn* + *brnet*

(d) Strategy for *diff* + *pop4*

(e) Strategy for *nn* + *pop4*

(f) Strategy for *gnn* + *pop4*

Figure 5: Visualizing final strategies found via *diff*, *nn* and *gnn* with best responses of the form *brnet* and *pop4* on a randomly chosen forest with $m = n = 2$. The blue (red) dots are sampled from the defender's (attacker's) strategy for the 2 drones (lumberjacks).

5 DISCUSSION AND CONCLUSION

In this work, we propose the Coverage Gradient Theorem to directly compute the gradients of a large class of multi-resource spatial coverage objectives. We also provide a tractable and scalable spatial discretization-based framework to approximate the resulting gradient expressions. Our generalized formulation supports multiple agents, multiple resources, arbitrary (even non-convex) shapes for target domains and arbitrary continuous distributions for targets. By augmenting existing approaches with our approximation framework, we show improved performance in both single-agent and adversarial two-agent multi-resource spatial coverage problems.

One of the key limitations of the approximation framework is to approximate the integrals using discretization. While this scales well for two or three dimensional target domains, it is harder to scale if we are working with target spaces of larger dimensions and one needs to explore alternative but less accurate methods, e.g., sampling. However, while it is much more manageable to store discretized shape tensors in GPU memory, working with samples from geometric shapes and defining operators on them is generally harder. Further, while our framework scales linearly with the number of resources for single player games, the size of the spatial maps and binary tensors involved depends on the number of bins chosen per dimension of the target domain. This number can be large if a fine-grained discretization is being used or the target space is huge and can require multiple GPUs in parallel to store the full forward and backward models. To obtain the best trade-off between memory and parallelization on GPUs, working on scalable adaptive sampling-based frameworks is a promising next step for future research.

Author Contributions

N. Kamra conceived the idea, performed the experiments and wrote the paper. Y. Liu provided helpful discussions, feedback and funding for the work.

Acknowledgements

This research was supported in part by NSF Research Grant IIS-1254206, MURI Grant W911NF-11-1-0332 and the USC Viterbi Graduate PhD fellowship.

References

Alireza Dirafzoon, Mohammad Bagher Menhaj, and Ahmad Afshar. Decentralized coverage control for multi-agent systems with nonlinear dynamics. *IEICE TRANSACTIONS on Information and Systems*, 94(1):3–10, 2011.

Fei Fang, Albert Xin Jiang, and Milind Tambe. Optimal patrol strategy for protecting moving targets with multiple mobile resources. In *AAMAS*, pages 957–964, 2013.

William Haskell, Debarun Kar, Fei Fang, Milind Tambe, Sam Cheung, and Elizabeth Denicola. Robust protection of fisheries with compass. In *IAAI*, 2014.

Andrew Howard, Maja J Matarić, and Gaurav S Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Distributed Autonomous Robotic Systems 5*, pages 299–308. Springer, 2002.

Taoan Huang, Weiran Shen, David Zeng, Tianyu Gu, Rohit Singh, and Fei Fang. Green security game with community engagement. *arXiv preprint arXiv:2002.09126*, 2020.

Matthew P. Johnson, Fei Fang, and Milind Tambe. Patrol strategies to maximize pristine forest area. In *AAAI*, 2012.

Nitin Kamra, Umang Gupta, Kai Wang, Fei Fang, Yan Liu, and Milind Tambe. Deepfp for finding nash equilibrium in continuous action spaces. In *Decision and Game Theory for Security (GameSec)*, pages 238–258. Springer International Publishing, 2019.

Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. Computing optimal randomized resource allocations for massive security games. In *AAMAS*, pages 689–696, 2009.

Chan Sze Kong, New Ai Peng, and Ioannis Rekleitis. Distributed coverage with multi-robot system. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2423–2429. IEEE, 2006.

Prashanth Krishnamurthy and Farshad Khorrami. Optimal sensor placement for monitoring of spatial networks. *IEEE Transactions on Automation Science and Engineering*, 15(1):33–44, 2016.

Ali Nasri Nazif, Alireza Davoodi, and Philippe Pasquier. Multi-agent area coverage using a single query roadmap: A swarm intelligence approach. In *Advances in practical multi-agent systems*, pages 95–112. Springer, 2010.

Huy Xuan Pham, Hung Manh La, David Feil-Seifer, and Aria Nefian. Cooperative and distributed reinforcement learning of drones for field coverage. *arXiv preprint arXiv:1803.07250*, 2018.

S. Poduri and G. S. Sukhatme. Constrained coverage for mobile sensor networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

Alessandro Renzaglia, Lefteris Doitsidis, Agostino Martinelli, and Elias B Kosmatopoulos. Multi-robot three-dimensional coverage of unknown areas. *The International Journal of Robotics Research*, 31(6):738–752, 2012.

Martin Saska, Jan Chudoba, Libor Přeučil, Justin Thomas, Giuseppe Loianno, Adam Třešňák, Vojtěch Vonásek, and Vijay Kumar. Autonomous deployment of swarms of micro-aerial vehicles in cooperative surveillance. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 584–595. IEEE, 2014.

Milind Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned.* Cambridge University Press, New York, NY, 2011.

Daoqin Tong, Alan Murray, and Ningchuan Xiao. Heuristics in spatial analysis: a genetic algorithm for coverage maximization. *Annals of the Association of American Geographers*, 99(4):698–711, 2009.

Mohamed Amine Yakoubi and Mohamed Tayeb Laskri. The path planning of cleaner robot for coverage region using genetic algorithms. *Journal of innovation in digital ecosystems*, 3(1):37–43, 2016.

Rong Yang, Benjamin Ford, Milind Tambe, and Andrew Lemieux. Adaptive resource allocation for wildlife protection against illegal poachers. In *AAMAS*, 2014.

Yue Yin, Bo An, and Manish Jain. Game-theoretic resource allocation for protecting large public events. In *AAAI*, pages 826–833, 2014.