

Defending a City from Multi-Drone Attacks: A Sequential Stackelberg Security Games Approach

Dolev Mutzari^{a,*}, Tonmoay Deb^b, Cristian Molinaro^c, Andrea Pugliese^c, V.
S. Subrahmanian^b, Sarit Kraus^a

^a*Department of Computer Science, Bar Ilan University, Israel*

^b*Department of Computer Science, Northwestern University, IL, USA*

^c*DIMES Department, University of Calabria, Italy*

Abstract

To counter an imminent multi-drone attack on a city, defenders have deployed drones across the city. These drones must intercept/eliminate the threat, thus reducing potential damage from the attack. We model this as a Sequential Stackelberg Security Game, where the defender first commits to a mixed sequential defense strategy, and the attacker then best responds. We develop an efficient algorithm called S2D2, which outputs a defense strategy. We demonstrate the efficacy of S2D2 in extensive experiments on data from 80 real cities, improving the performance of the defender in comparison to greedy heuristics based on prior works. We prove that under some reasonable assumptions about the city structure, S2D2 outputs an approximate Strong Stackelberg Equilibrium (SSE) with a convenient structure.

Keywords: Multi-Drone Attacks, Security Games, Sequential Games

1. Introduction

There has been a lot of recent concern about multi-drone attacks [1, 2, 3, 4, 5, 6, 7, 8], especially in highly populated urban areas where not all countermeasures can be used [7]. Drones can target centers of government

*Corresponding author

Email addresses: dolevmu@gmail.com (Dolev Mutzari),
tonmoay.deb@northwestern.edu (Tonmoay Deb), cmolinaro@dimes.unical.it
(Cristian Molinaro), andrea.pugliese@unical.it (Andrea Pugliese),
vss@northwestern.edu (V. S. Subrahmanian), sarit@cs.biu.ac.il (Sarit Kraus)

and severely damage critical infrastructure (e.g., utilities). It has been proposed [1, 5, 4, 7] that the city can be defended with drones to counter the attacks and reduce damage to life and property. As drones are cheap, accessible, and can maneuver above city buildings, effective defense should be equally affordable, and free from ground-based constraints.

Therefore, in this work we focus on defending against multi-drone attacks on large-scale cities, using defense drones. It is clear that certain locations in the city are more attractive to attack for the attacker and hence more critical for the defender to protect. The goal is therefore to minimize damage rather than to catch the attacker drones as fast as possible. Finally, while aerial drones can be relatively easy to purchase, they are subject to battery and payload constraints.

Stackelberg security games (SSGs) offer a framework to optimize the allocation of defense resources against strategic adversaries. [9, 10] provide an extensive overview of SSG applications successfully deployed to date. An SSG consists of a defender with some defense resources protecting multiple targets against a strategic attacker. The defender commits to a mixed allocation strategy, and the attacker best responds by attacking the target that maximizes her utility.

Many extensions of the original SSG model [11] exist today, e.g. bounded rationality attackers [12, 13, 14], partial information [15], defense schedules [16], heterogeneous resources [16], multiple defenders [17, 18] and attackers [19], attackers with multiple resources [20], and repeated SSGs [21]. Nevertheless, most research is on Stackelberg equilibria in normal-form games: the defender commits to a mixed strategy, the attacker best responds, and the expected utilities are then directly determined. In particular, the attacker has a single opportunity to attack.

To defend a city from multi-drone attacks¹, we use *sequential* SSGs, in which the targets are nodes in a graph, which both players' drones traverse. In particular, we model this as an extensive-form game. The attacker's drones are subject to payload and battery capacity constraints.

1.1. Related Work

Defending against *swarm* aerial drone attacks has been studied extensively — [4, 7] provide a recent overview. In a drone swarm, each drone

¹Our framework also applies to land-based attacks by a coordinated set of attackers, targeting a city with simultaneous or sequential attacks by traversing its roads.

acts in real-time based on its local observation of the environment, including neighboring drones. Modeling attacker drones as a swarm is limiting since an attacker with sufficient computational and technological resources can conduct coordinated attacks to increase its utility. For similar reasons, while defense using a drone swarm is more scalable with the number of drones, both computationally and from practical perspectives, it is less effective than a coordinated multi-drone defense mechanism.

Past research on drone swarm attacks can be roughly split into three domains: (i) detection mechanisms focusing on identifying an incoming attack, tracking and classifying air-drones [3, 22], (ii) quickly assessing whether a tracked drone is threatening or not [23], and (iii) defense mechanisms that seek to counter and protect against threatening drone attacks [1, 2, 4, 6]. The growing body of work on detection mechanisms is complementary to this work, justifying the assumption that attack drones can be monitored.²

Next, we briefly cover the gaps and limitations of defensive mechanisms other than using defense drones. GPS jamming / spoofing (used e.g. in [2]) cannot tackle drones that use other navigation methods (visual, radar, etc.), and RF jamming is not effective against autonomous malicious drones. Furthermore, anti-jamming/spoofing techniques may undermine their effectiveness. In addition, these methods may jam civilian applications (e.g., mobile phone communications). We refer to [7] for further discussion and focus on the defensive drone swarm literature.

[1, 5] and [24] study defense using a drone swarm. These works mostly focus on coordinating defensive drones, and the attacker model is limited. First, only a single attacker drone is considered. Second, it is assumed that the attacker drone is nearby, and was detected before causing any damage. This might work for protecting a facility of interest, but spreading them would enable covering much more ground. Third, once it is detected, the defensive drone swarm assumes the attacker drone follows a straight projectile³ to predict its future location and catch it rapidly. Obstacles might hinder such movement of the attacker, and more importantly, the attacker is interested not only in evading the defensive swarm but also in striking targets, otherwise it would not take off to begin with. [25, 26] alleviated

²In fact, we make the weak assumption that the location of a drone is known only after its first strike takes place.

³[24] adds a brief discussion on other strategies the attacker might choose.

the assumption of straight-line movement by learning from simulations using Deep Reinforcement Learning.

The above works fall under multi-pursuer multi-evader differential games [27], where each player decides on a continuous function over time, called *control* that must admit certain constraints. [28] pairs the pursuers and evaders thereby reducing the problem into a single pursuer single evader game, and we follow a similar approach. Differential games (DGs) can be roughly divided into two forms: *open-loop* DGs where the controls depend only on time and initial game state and there is no dependence on the current game state, and *closed-loop* DGs where controls may be a function of the continuously evolving state.

In our setting, we want the defender to be closed-loop and utilize recent work on detecting and monitoring attack drones, whereas the attacker should be open-loop as it does not know the defense drone locations. Another well-studied family of evasion games are cops and robbers [29], traversing a graph. The locations of each cop and robber are typically visible. There are works on invisible robbers [30, 31, 32], but not on invisible cops. [30] also considers a drunk robber, which effectively does not take the cops' locations into account, but instead takes a random walk, and we are interested in a rational attacker. Moreover, the goal in evasion games (both on graphs and differential games) is to catch the evaders as fast as possible. In particular, they do not take into account rewards and penalties from successful attacks.

Finally, there has been some work on sequential security games (which is the approach taken in this paper) to model the problem at hand. This should not be confused with repeated SSGs, which are one-shot games, played multiple times to enable players to gain information. For instance, [33] studies repeated SSGs with unknown attacker type to handle deception, and [21] studies repeated SSGs where the attacker does not know the defense mixed strategy initially.

In sequential SSGs [34], the defender and attacker simultaneously traverse a graph. The attacker can attack multiple targets on her path. As in classical SSGs, the defender commits to a mixed strategy, and the attacker best responds. Unlike traditional SSGs, the strategy space is huge. [34] assumes: (i) the attacker has one drone, (ii) drones carry unbounded payload, (iii) a solution is offered only against two sequential strikes, (iv) solutions assume that either defense movement is unrestricted or is prohibited completely. [35] extended [34] by alleviating (iii), but assumes a zero-sum finite game, where SSE and NE are equivalent [36].

General sequential SGs were first considered in [37]. Exact methods [38, 39] do not scale to our setting as they are at best linear in the game tree. Heuristic algorithms (e.g., [40]), being generic, perform poorly in our setting. They do not exploit the graph structure of the problem and lack basic tools (e.g., shortest path and TSP solvers). [41] considers sequential SGs and develops an MCTS-based heuristic algorithm. Nevertheless, this method is not suitable for finding a Strong Stackelberg Equilibrium (SSE). [42] considered a discrete-time stochastic Stackelberg game where the attacker has a private type that evolves as a controlled Markov process. They compute a Stackelberg equilibrium by solving lower dimensional fixed-point equations for each time t . Their technique assumes the state to be small.

1.2. Contributions

The main contributions we make are summarized below.

1. We extend sequential SSGs to handle multiple attack/defense drones with payload/battery constraints.
2. We propose *Sequential Stackelberg Drone Defense (S2D2)*, an efficient algorithm to output a defense strategy.
3. We identify conditions for the underlying graph, under which S2D2 outputs an approximate Strong Stackelberg Equilibrium (SSE), along with an upper bound on the error. We also develop an algorithm to check if a given graph admits such a structure.
4. Though our theoretical results make assumptions to guarantee the existence of approximate SSEs, not all real-world situations satisfy these conditions. Thus:
 - We ran extensive experiments on a dataset of 80 famous world cities (1000s to $\sim 250K$ nodes) using two distributions (Zipf and log-normal) to assign utilities to neighborhoods of the city.
 - We conducted a detailed case study of 6 cities (one small and two big US cities, a large and a small city in the Middle East, a megacity in Asia) using utilities provided by experts, rather than random assignment. Our experiments compare S2D2 to a heuristic algorithm based on prior works that trades off runtime and defender utility.
 - We studied the robustness of the computed approximate SSEs by perturbing the utilities and looking at performance variations.

Our results showed that slightly perturbing game parameters (e.g., penalties and rewards) led to proportional changes in defender utility.

We conclude that even when theoretical assumptions do not hold, S2D2 still yields good results.

Section 7 contains a deeper discussion of the rationale behind our model design, including justifications for key choices, alternative approaches with their trade-offs, and other relevant questions. This section also presents non-trivial arguments that further support our modeling decisions.

1.3. Organization

Section 2 provides a high-level, birdseye view of the overall S2D2 architecture and decision. In particular, it explains how the different parts of this paper fit together. Section 3 presents the problem of interest, modeled as a sequential SSG. A deeper discussion of the rationale behind our model design, and comparison with alternative approaches is presented in Section 7. Section 4 then describes our *S2D2* algorithm, which has three steps. First, a “coarsening” algorithm (cf. Section 4.1) partitions an input city graph into clusters (“*neighborhoods*” — clusters of vertices). Then, an approximate solution is computed (cf. Section 4.2), assuming both the attacker and the defender have one drone and play in one neighborhood. This algorithm is an extension of [16]’s method to sequential games, where the attacker strategy space becomes overwhelmingly large. We then discuss how to use the solution for the single drone game to find an approximate solution for the multi-drone game (cf. Section 4.3). This is achieved by generalizing [20]’s work on multi-resource attacker SSGs to support non-linear utilities. S2D2 uses this method to decide the allocation of defense drones into neighborhoods. Section 5 proves that under a set of conditions on a coarsened graph, S2D2 is sure to output an approximate SSE, and Section 6 presents experimental results. Finally, Section 8 outlines our conclusions.

2. Birdseye View of S2D2

In this section, we present a birdseye view of the S2D2 system and describe its architecture (cf. Figure 1). S2D2 contains the following components.

- **Cities represented as graphs.** We represent cities being protected as a graph. Each node in the graph represents a region on the ground.

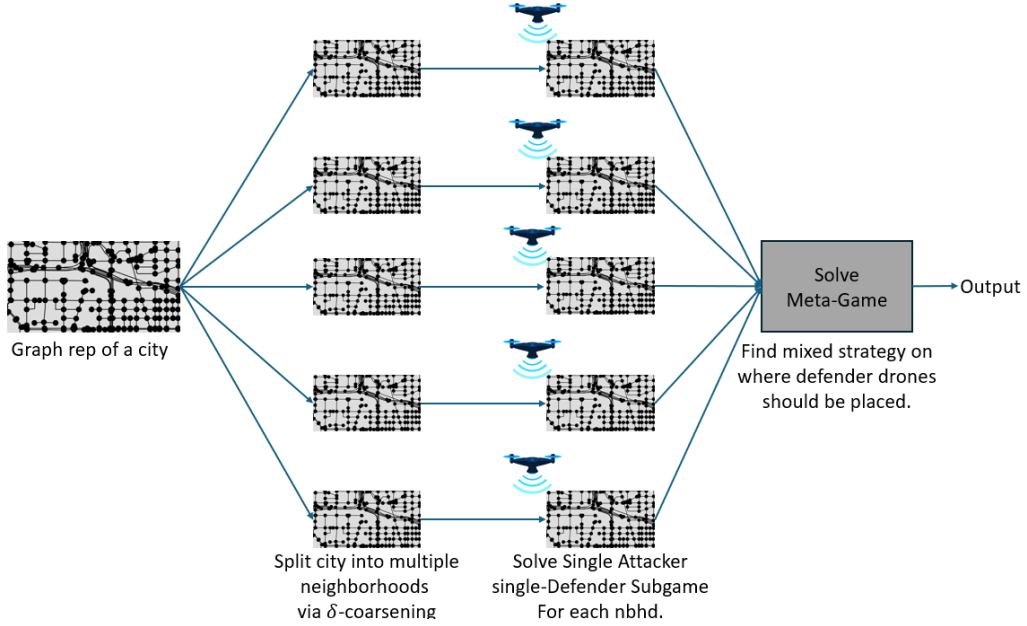


Figure 1: S2D2 Architecture.

Adjacent nodes in the graph represent adjacent regions on the ground, i.e., regions that share a common border.

- **Coarsening a graph for scalability.** Because cities can be huge (the number of vertices in our dataset vary from 2.2K to 277K and the number of edges vary from 3.4K to 405K), game-theoretic models will not scale. Because of this, we *coarsen* a graph into *neighborhoods*. A neighborhood consists of a connected set of nodes in the city graph. We will require coarsenings to satisfy some desired properties (discussed further below). An algorithm to find a good coarsening is described in Algorithm 2 in Section 4.1.
- **Single-Defender, Single-Attacker Game Per Neighborhood.** Next, we look and ask the question: if a single defender and a single attacker drone are in a given neighborhood, what strategy would maximize their respective utilities? We solve this problem by building on top of the results of [16]. However, fixing the coarsening first and then solving a single attacker single defender problem could lead to suboptimal solu-

tions. The attacker is not formally restricted to place each drone in a single neighborhood throughout the game, and it may also be suboptimal for the defender to do so. The coarsening algorithm is therefore responsible to correctly capture the attacker and defender incentives, and provide a corresponding coarsening of the graph. We propose the concept of a δ -coarsening that ensures several desirable properties of the coarsening. We then design an algorithm to find a δ -coarsening (Algorithm 2).

- **Solving the Meta Game.** Once we understand the utilities of the single attacker, single-defender game, one in each neighborhood, we need to determine where the defender must place his/her defender drones. The third part of the S2D2 algorithm addresses this problem (Algorithm 6) using a mixed strategy. This will be discussed further in Section 4.4.

3. Sequential SSGs

We start by briefly overviewing sequential SSGs in the context of our problem. In sequential SSGs, the defender may re-distribute its defense drones after a successful attack. While doing so, the defender knows the attacker drones' location and which targets were destroyed. Meanwhile, the attacker may select and start moving toward other potential targets. The game continues until all attacker drones are either caught, out of battery, or out of payload. The attacker may only attack targets close to her current position. Formally, the game consists of:

1. An undirected graph $G = (V, E)$, where:
 - $V = \{1, \dots, m\}$ is a set of m target nodes.
 - E is a set of undirected edges between targets.
2. $R^a : V \rightarrow \mathbb{N}$ and $P^d : V \rightarrow \mathbb{Z}_{<0}$ map each target to the attacker reward and defender penalty, respectively⁴, from an attack on a given node $v \in V$.

⁴Unlike traditional SSGs, we set attacker penalties and defender rewards to zero ($P^a = R^d = 0$) since the attacker is already penalized when caught, as it cannot attack any more targets. Similarly, the defenders are rewarded when they catch the attacker as doing so prevents future strikes.

3. $A, D \in \mathbb{N}$ are the number of attacker and defender drones, respectively.
4. The payload $P \in \mathbb{N}$ each attacker drone is able to carry. This equals the maximal number of attacks each drone can pull-off (if not caught or run out of battery).
5. The battery capacity $B \in \mathbb{N}$ each attacker drone has. This equals the maximal total distance it can travel (if not caught). We assume traversing an edge $e \in E$ takes one unit of battery (adding 0-rewarded/penalized nodes along a long edge if necessary), as well as staying (or loitering) over a node.

Assumptions. We assume the defender knows (A, P, B) and the current location of each attacker drone at all times after the first strike by that drone. Defense drones also have a battery capacity B . Hence, without loss of generality, the game ends after B steps. The attacker only knows the number of defense drones D at the beginning of the game. Attacker drones do not know the locations of defense drones unless they meet at a node — this is reasonable as a defender can deploy sensor and other assets in her city. When this occurs, the attacker drone is destroyed. Attacker drones are not informed when other attacker drones are eliminated.

3.1. Defender and Attacker Strategies

The defender knows the location of some attacker drones and can leverage this information. Formally, a pure defender strategy $s_d \in \mathcal{S}^d$ is a B -tuple of functions (s_1^d, \dots, s_B^d) , specifying its strategy at each time-step. The first strategy $s_1^d \in V^D$ specifies the start position of each defense drone. At any step $1 < t \leq B$, the function s_t^d determines the next step of each drone given the current state of the game, which includes:

- Last location of each defense drone $(v_1^d, \dots, v_D^d) \in V^D$.
- Last location of each observed attack drone $(v_1^a, \dots, v_A^a) \in (V \cup \{\perp, \dagger\})^A$. We use the special symbol \perp for unknown location (no strike yet), and \dagger for eliminated.
- Subset of destroyed targets $I_{t-1} \subseteq V$ (where $I_0 = \emptyset$).

In a single step, a drone at location $v \in V$ can only reach neighboring locations in graph G , i.e. $N[v] := \{v' \in V : \{v, v'\} \in E\} \cup \{v\}$. The function s_t^d outputs the new location of each defense drone $(\tilde{v}_1^d, \dots, \tilde{v}_D^d)$ where

$\tilde{v}_i^d \in N[v_i^d]$ for each $1 \leq i \leq D$.⁵ Figure 2(a) provides a visualization (from our S2D2 system) of the defender’s strategy overlaid over a map of a city. The locations of defender drones (blue) and attacker drones (red) as well as the destroyed parts of the city are shown as icons. The defender’s strategy specifies a function that answers the following question: given a picture like the one depicted, where should the blue drones move to next?



Figure 2: (a) Visualization of Defender Strategy. (b) Visualization of Attacker Strategy.

Figure 2(b) shows the attacker strategy. For each attacker drone (shown in red), a flight path is specified (shown for one red drone in Figure 2(b) as a red arrow). In addition, the strategy specifies where each attacker drone will actually target with one unit of payload. In Figure 2(b), we see two locations where payload is used by this attacker, marked by an explosion icon. To keep the figure simple, we do not show these flight paths and payload utilization for the other attacker drones depicted. The pure strategies for the attacker are related to B -length paths in the graph. We use $\mathcal{P}_B := \{(v_1, \dots, v_B) \in V^B \mid \forall 1 \leq t < B : \{v_t, v_{t+1}\} \in E \vee v_t = v_{t+1}\}$ to denote the set of all paths of length B in G , and let $\mathcal{P}_0 = \{\emptyset\}$. Recall that traversing each edge requires one battery unit, as well as hovering over a node ($v_{t+1} = v_t$).⁶

⁵Note that the game is Markovian: the history of how drones ended up in their last observed location, or the order in which targets have been destroyed, cannot be utilized against a rational attacker.

⁶The sequential SSG has a few natural extensions which we may consider in future work. These include: (i) *Heterogeneous drones*: The attacker may have drones of different types, $(B_1, P_1), \dots, (B_A, P_A)$. (ii) *Distances*: The edges may be weighted as well, by the distance between its endpoints. Adding $d - 1$ vertices along an edge with distance d will not yield a reduction. Indeed, one has to define a reward over these new vertices, say 0.

Furthermore, each attacker drone must decide which targets to attack. Let $\mathcal{T}_{P,B} = \{I \subseteq \{1, \dots, B\} \mid |I| \leq P\}$ denote the set containing sets of at most P indices along the path of length B to be attacked. The set of pure strategies of the attacker is therefore $\mathcal{S}^a = (\mathcal{T}_{P,B} \times \mathcal{P}_B)^A$.

Utility. Given an attacker (resp. defender) strategy $s_a \in \mathcal{S}^a$ (resp. $s_d \in \mathcal{S}^d$), we can recursively compute utilities at time t . Initially, $u_0^a = u_0^d = 0$. At time $t > 0$, we compute the position of all surviving drones from the specified strategies and the previous drone locations. We update the utilities $u_t^a = u_{t-1}^a + r_t^a$ and $u_t^d = u_{t-1}^d + p_t^d$ where r_t^a (p_t^d) is the sum of rewards (resp. penalties) from successful attacks at step t for the attacker (defender). We then nullify the rewards for targets that were successfully attacked at time step t , and eliminate any attacker that is either caught or out of payload. Finally, we set $u^a(s_d, s_a) = u_B^a$, $u^d(s_d, s_a) = u_B^d$.

3.2. Mixed Strategies

The defender may use a *mixed* strategy. In other words, it may sample its strategy from a distribution $\mathbf{x}_d \in \Delta(\mathcal{S}^d)$, where $\Delta(\mathcal{S}^d)$ is the set of all probability distributions over \mathcal{S}^d . For the special case where $B = 1$ (the non-sequential SSG), we can use a compact representation $\mathcal{C}_D := \{\mathbf{x} \in [0, 1]^m : \sum_{v \in V} x_v \leq D\}$ of the set of defender mixed strategies. A vector $\mathbf{x} \in \mathcal{C}_D$ is called a *coverage vector*, and it denotes the probability that each node $v \in V$ is covered by some defense drone. Coverage vectors can provably be implemented by a distribution over deterministic allocation strategies, each using at most D resources. This distribution can also be found efficiently, see [16], Theorem 1.

In Stackelberg games, the attacker can conduct surveillance on the defender's (mixed) strategy \mathbf{x}_d beforehand and best respond to it. Assume now the defender and the attacker play mixed strategies over $\mathcal{S}^a, \mathcal{S}^d$, respectively. Given mixed strategies $\mathbf{x}_d, \mathbf{x}_a$, the utility of the attacker (and similarly the

Still, the defender will know the attacker's position in the first step along the split edge. (iii) *Velocities*: Different drones may fly with different velocities. The velocity may also depend on the percentage of loaded payload. (iv) *Defense schedules*: Allocating a defense drone to some target v may also protect its neighbors $N(v)$.

defender) is given by

$$\begin{aligned}
u^a(\mathbf{x}_d, \mathbf{x}_a) &:= \mathbb{E}_{(s_d, s_a) \sim \mathbf{x}_d \times \mathbf{x}_a} [u^a(s_d, s_a)] \\
&= \sum_{(s_d, s_a) \in \mathcal{S}^d \times \mathcal{S}^a} \mathbf{x}_d(s_d) \mathbf{x}_a(s_a) \cdot u^a(s_d, s_a)
\end{aligned} \tag{1}$$

Example 1 (Sequential SSG: Toy Example). *Consider a toy graph $G = (V, E)$ with $m = 41$ vertices and edges depicted in Figure 3. Suppose we set*

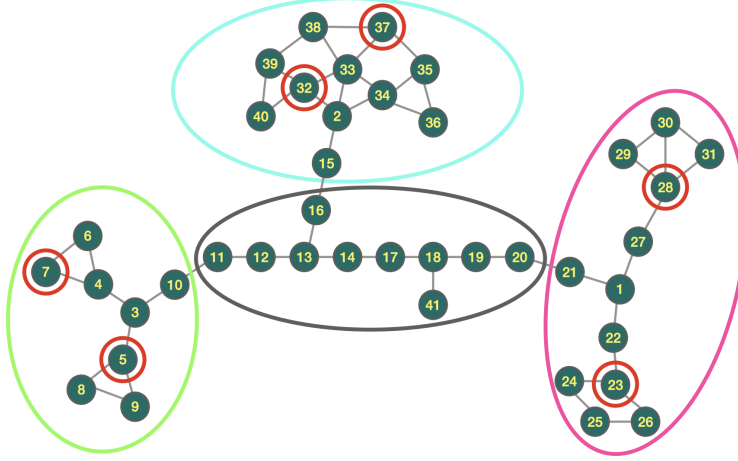


Figure 3: A graph and its coarsening into neighborhoods.

$P^d \equiv -R^a$ in our example, and the attacker rewards are set to one for targets $v_5, v_7, v_{23}, v_{28}, v_{32}, v_{37}$, and zero for all the rest. Suppose the defender and the attacker both have $A = D = 2$ drones, and that $B = 4, P = 2$ for attacker drones.

A defender pure strategy may first place the defense drones on v_3, v_1 respectively. Then, given the attacker position, the strategy would let each defense drone follow the closest path towards the closest attacker drone. Denote this strategy by s_d^1 . Suppose the attacker plays strategy s_a where her drones are at v_{37}, v_{28} . The first drone follows path $v_{37} \rightarrow v_{38} \rightarrow v_{39} \rightarrow v_{32}$, and attacks v_{37} and v_{32} . The second drone follows $v_{28} \rightarrow v_{27} \rightarrow v_1 \rightarrow v_{22} \rightarrow v_{23}$ and attacks v_{28}, v_{23} . In this case, the defense drone starting at v_3 will not do much, but the defense drone starting at v_1 will catch the drone that started at v_{28} before v_{23} is attacked. We can verify that when facing pure strategies, the attacker may always successfully attack two meaningful targets using one

of her drones. Instead, the defender may use a mixture \mathbf{x}_d of 3 strategies, each for instance with probability $1/3$. Suppose $\mathbf{x}_d(s_d^1) = 1/3$, $\mathbf{x}_d(s_d^2) = 1/3$ and $\mathbf{x}_d(s_d^3) = 1/3$. In s_d^2 , the defense drones start from v_2, v_3 , and in s_d^3 , they start from v_1, v_2 . By doing so, there is always a probability ($2/3$ in this case) that a defense drone is “in the hood”.

In SSGs, the attacker knows the defender’s mixed strategy $\mathbf{x}_d \in \Delta(\mathcal{S}^d)$, and then best responds to it with $s_a \in \text{BR}^a(\mathbf{x}_d)$. Since the utility of the attacker from a mixed strategy is the weighted average of the utilities from each pure strategy, she may always choose a pure strategy that yields the maximal utility. Therefore, w.l.o.g., the attacker’s best response set consists of pure strategies only:

$$\text{BR}^a(\mathbf{x}_d) := \arg \max_{s_a \in \mathcal{S}^a} u^a(\mathbf{x}_d, s_a)$$

When there are multiple targets in $\text{BR}^a(\mathbf{x})$, we take the standard approach [43] and assume that the attacker breaks ties in favor of the defender. The reason is that by reducing the coverage of the desired target by an arbitrarily small amount, the attacker will attack the desired target and the defender will suffer an arbitrarily small utility loss. We therefore define

$$\text{BR}^d(\mathbf{x}_d) = \arg \max_{s_a \in \text{BR}^a(\mathbf{x}_d)} u^d(\mathbf{x}_d, s_a).$$

The set of strategies in $\text{BR}^a(\mathbf{x})$ are the ones that are best for the defender. We may then define $u^d(\mathbf{x}_d) := u^d(\mathbf{x}_d, s_a)$, $u^a(\mathbf{x}_d) := u^a(\mathbf{x}_d, s_a)$ for $s_a \in \text{BR}^d(\mathbf{x}_d)$. This is well-defined as the value is independent of the choice of s_a .

The typical solution concept for SSGs is Strong Stackelberg Equilibrium (SSE).

Definition 1 (Strong Stackelberg Equilibrium). *A strategy profile $(\mathbf{x}_d, s_a) \in \Delta(\mathcal{S}^d) \times \mathcal{S}^a$ is a Strong Stackelberg Equilibrium iff*

$$\mathbf{x}_d \in \arg \max_{\mathbf{x}'_d \in \Delta(\mathcal{S}^d)} u^d(\mathbf{x}'_d) \text{ and } s_a \in \text{BR}^d(\mathbf{x}_d).$$

Approximate SSE’s are defined analogously.

Definition 2 (ϵ -approximate SSE). *A strategy profile $(\mathbf{x}_d, s_a) \in \Delta(\mathcal{S}^d) \times \mathcal{S}^a$ is an ϵ -approximate SSE (ϵ -SSE) iff*

$$\begin{aligned} u^a(\mathbf{x}_d, s_a) + \epsilon &\geq \max_{s'_a \in \mathcal{S}^a} u^a(\mathbf{x}_d, s'_a) \text{ and} \\ u^d(\mathbf{x}_d, s_a) + \epsilon &\geq \max_{\mathbf{x}'_d \in \Delta(\mathcal{S}^d)} u^d(\mathbf{x}'_d, \text{BR}^d_\epsilon(\mathbf{x}'_d)), \end{aligned} \tag{2}$$

where $\text{BR}_\epsilon^a(\mathbf{x}'_d)$ consists of all strategies s_a satisfying (2), and $\text{BR}_\epsilon^d(\mathbf{x}'_d) \subseteq \text{BR}_\epsilon^a(\mathbf{x}'_d)$ consists of all strategies in $\text{BR}_\epsilon^a(\mathbf{x}'_d)$ that maximize defender utility (breaking ties optimistically).

Finding SSE Efficiently by Solving Linear Programs. Equation (1) suggests that the defender's utility is linear with respect to the coverage vector \mathbf{x}_d . Furthermore, the defender's strategy space $\Delta(\mathcal{S}^d)$ is a polytope. This suggests using linear programming. We extend the approach in [16] for $B = 1$ to general sequential games as detailed below. We want to compute:

$$\mathbf{x}_d \in \arg \max_{\mathbf{x}'_d \in \Delta(\mathcal{S}^d)} u^d(\mathbf{x}'_d) = \arg \max_{\mathbf{x}'_d \in \Delta(\mathcal{S}^d)} u^d(\mathbf{x}'_d, \text{BR}^d(\mathbf{x}'_d)).$$

The only problem is that the $\text{BR}^d(\mathbf{x}_d)$ is not linear. Our idea is to solve, for each potential s_a^* candidate for $\text{BR}^d(\mathbf{x}_d)$, the LP (linear program):

- Maximize $u^d(\mathbf{x}_d, s_a^*)$, subject to:
 1. $\mathbf{x}_d \in \mathcal{C}_D$.
 2. $\forall s_a \in \mathcal{S}^a, u^a(\mathbf{x}_d, s_a) \leq u^a(\mathbf{x}_d, s_a^*)$.

That is, we add $|\mathcal{S}^a|$ linear constraints to ensure that $s_a^* \in \text{BR}^a(\mathbf{x}_d)$, and enumerate over s_a^* . At the end, we pick the solution that gives the defender the greatest utility.

Multiple Attack Resources. In the sequential SSG, we consider multiple attacker drones, that is, multiple attacker resources. In this case [20] showed that finding SSE is NP-hard. This also implies that the problem of finding sequential SSGs is NP-hard via a reduction from finding SSE in SSGs with multiple attacker resources. Simply let each attacker drone have a single unit of battery, to make the game effectively a non-sequential SSG. Nevertheless, NP-hard problems like MILPs (Mixed Integer Linear Programs) are well-studied and practical solutions have been developed previously. Indeed, S2D2 involves a reduction to a MILP.

Table 1 summarizes the symbols used in this paper. A comprehensive discussion regarding our proposed model is provided in Section 7.

4. The S2D2 Algorithm

The S2D2 algorithm generates a mixed defense strategy through three steps:

Symbol	Meaning
$G = (V, E)$ $(R^a, P^d) : V \rightarrow \mathbb{N} \times \mathbb{Z}_{<0}$ $A, D \in \mathbb{N}$ $P, B \in \mathbb{N}$ $(v_1^d, \dots, v_D^d) \in V^D$ $(v_1^a, \dots, v_A^a) \in (V \cup \{\perp, \dagger\})^A$ $s_d = (s_1^d, \dots, s_B^d) \in \mathcal{S}^d$ $s_a \in \mathcal{S}^a$ $\mathbf{x}_d \in \Delta(\mathcal{S}^d)$ $u^a(s_d, s_a), u^d(s_d, s_a)$ $u^a(\mathbf{x}_d, s_a), u^d(\mathbf{x}_d, s_a)$ $\text{BR}^a(\mathbf{x}_d) \subseteq \mathcal{S}^a$	SSSG Model: City graph, where $V = \{1, \dots, m\}$ is the set of nodes and E is the set of undirected edges Attacker reward and defender penalty functions Number of attacker and defender drones Payload and battery capacity of attacker drones Locations of defender drones Locations of attacker drones (\perp means unknown location, \dagger means eliminated) Pure defender strategy, where s_t^d is the policy at time t Pure attacker strategy Mixed defender strategy Attacker and defender utilities under s_d and s_a Attacker and defender utility under $\mathbf{x}_d \in \Delta(\mathcal{S}^d)$ and $s_a \in \mathcal{S}^a$ Set of best attacker responses to defender's mixed strategy $\mathbf{x}_d \in \Delta(\mathcal{S}^d)$
δ $\hat{V} = \{\hat{v}_1, \dots, \hat{v}_k\}$ $\mathcal{S}_{\hat{V}}^a, \mathcal{S}_{\hat{V}}^d$	Coarsening: Scale parameter, rewards smaller than δ are neglected Coarsening of G , a set of disjoint neighborhoods $\hat{v}_i \subseteq V$ Set of pure attacker and defender strategies that respect the coarsening \hat{V}
λ $u^d(\mathbf{x}_d, s_a, \lambda), u^a(\mathbf{x}_d, s_a, \lambda)$	Single Drone Parameterized Sub-game: A parameter, fixing the probability that a defender is present in a neighborhood Attacker and defender utilities in a single drone game at a given neighborhood, under single drone strategies \mathbf{x}_d, s_a , and defender presence probability λ
$\hat{f}_a : \{1, \dots, A\} \mapsto \hat{V}$ $\hat{p}_d(\hat{v})$ $(\hat{f}_a, \hat{s}_a), (\hat{p}_d, \hat{x}_d)$	Multi-Drone Meta Game: Mapping from attacker drones to attacked neighborhoods Probability that a defender is present in \hat{v} Attacker and defender multi-drone strategies

Table 1: Symbols used in the paper.

1. **Coarsening** the graph, which involves partitioning it into artificial *neighborhoods*. The goal is to output a partition such that both the attacker and the defender are incentivized to spread their drones across different neighborhoods and stay there throughout the game. The defense (and attack) strategies can then be decomposed into the following two components.
2. **Single-Attacker Single-Defender Game per Neighborhood.** For each neighborhood, we solve a Single-Attacker Single-Defender sub-game and compute an approximate SSE. In reality, there is a probability $\hat{p}_d(\hat{v}) \in [0, 1]$ that a defender is present in a neighborhood. Since this probability is unknown a-priori, it is treated as an unknown variable λ , provided as an additional input parameter. S2D2 discretizes the interval $[0, 1]$ into evenly spaced intervals and solves the problem

for each $\lambda_i \in [0, 1]$.

3. **Solving the Meta-Game.** Once we know the defender utilities for each neighborhood, we can solve the problem of assigning a defender drone to each neighborhood. Basically, each neighborhood is considered as one “meta”-target. In this step, S2D2 **determines a mixed strategy for allocating defense drones to neighborhoods** via a reduction to a non-sequential SSG between a multi-resource defender and a multi-resource attacker. The utility functions for both the attacker and defender are approximated by piece-wise linear functions, derived from solving the single-defender single-attacker sub-game within each neighborhood, as a function of the defender presence probability λ .

The high-level pseudocode of the S2D2 algorithm is provided in Algorithm 1.

In reality, the attacker may opt to ignore the coarsening found by S2D2. This may happen either since the attacker is not rational, or because a “good coarsening” does not exist. In such a case, S2D2 randomly picks, for each defense drone, an attacker drone in its neighborhood, and ignores the rest. In addition, whenever an attacker drone leaves a neighborhood, the defender drone in that neighborhood halts. When the coarsening admits certain properties, we show in Section 5 that this does not result in a major utility loss for the defender.

Approximations. S2D2 tries to find an ϵ -approximate SSE, balancing the defender’s computational resources with the approximation error ϵ . To achieve this, S2D2 introduces a scale parameter $0 < \delta < \max_v R^a v$, effectively disregarding rewards smaller than δ . As δ increases, fewer nodes are deemed valuable, allowing S2D2 to focus on smaller subset of nodes to protect. Consequently, while this simplification reduces computational complexity, it also decreases the accuracy of S2D2’s view of the game, leading to an expected increase in the approximation error $\epsilon(\delta)$. However, under certain conditions for the underlying graph, $\epsilon(\delta)$ can be bounded, which provides theoretical guarantees for our algorithm. Even when these conditions are not met, empirical results demonstrate that S2D2 performs effectively in practice.

The next 3 subsections describe the three components listed above.

4.1. Coarsening the Graph

A *coarsening* of $G = (V, E)$ is a set $\hat{V} = \{\hat{v}_1, \dots, \hat{v}_k\}$ such that $\hat{v}_i \subseteq V$ for each $1 \leq i \leq k$ and $\hat{v}_i \cap \hat{v}_j = \emptyset$ for any $i \neq j$. Each subset in \hat{V} is

Algorithm 1 S2D2

Require: An undirected graph $G = (V, E)$;
numbers of attacker and defender drones $A, D \in \mathbb{N}$;
attacker drone's payload $P \in \mathbb{N}$;
drone's battery capacity $B \in \mathbb{N}$;
attacker rewards $R^a \in \mathbb{N}^{|V|}$;
defender penalties $P^d \in \mathbb{Z}_{<0}^{|V|}$.
Discretization parameters $\#\lambda, \lambda_c$ for the piece-wise linear approximation of the single-drone utility sub-games within each neighborhood.

Ensure: An ε -SSE defense strategy $\mathbf{x}^d = (\hat{V}, \hat{p}_d, \hat{x}_d, \varepsilon)$ and ε , or \mathbf{x}^d and \perp , where:
 \hat{V} is a coarsening of G and $(\hat{p}_d, \hat{x}_d) \in \Delta(\mathcal{S}_{\hat{V}}^d)$;
 $\hat{p}_d \in \mathcal{C}_{\hat{V}}^{\hat{V}}$ is the allocation strategy of D drones into neighborhoods of \hat{V} ;
 \hat{x}_d is the single-drone defense strategy within each neighborhood of \hat{V} ;

- 1: Compute a coarsening $(\delta, \hat{V}) \leftarrow \text{Coarsening}(G, \dots)$;
- 2: **for each** neighborhood $\hat{v} \in \hat{V}$ **do**
- 3: Compute piece-wise linear approximations of $u_v^a(\lambda), u_v^d(\lambda)$, the attacker and defender utilities for the single-drone game in neighborhood \hat{v} , where the defender is present with probability λ ;
- 4: **for** $\lambda = \frac{0}{\#\lambda}, \frac{1}{\#\lambda}, \dots, \lambda_c$ **do**
- 5: Set $S^a = \text{ScanAttackStrategies}(\hat{v}, \lambda, \dots)$ (reduced attack strategy space).
- 6: Set $S^d = \text{ScanDefenseStrategies}(\hat{v}, S^a, \dots)$ (reduced def. strategy space).
- 7: Compute $u_{\hat{v}}^d(\lambda), u_{\hat{v}}^a(\lambda)$ as in [16] and corresponding mixed strategy $\hat{x}^d(\hat{v})$, when restricting the attacker and defender strategy space to S^a, S^d .
- 8: **end for**
- 9: **end for**
- 10: Invoke $\langle \hat{p}^d, \hat{f}^a \rangle \leftarrow \text{SolveMetaGame}(\hat{V}, A, D, \{u_{\hat{v}}^d(\lambda), u_{\hat{v}}^a(\lambda)\}_{\hat{v} \in \hat{V}})$, to get the mixed allocation strategy \hat{p}^d of D drones into the neighborhoods of \hat{V} , by solving the static, multi-resource SSG with respect to the approximate utility functions $u_{\hat{v}}^d(\lambda), u_{\hat{v}}^a(\lambda)$.
- 11: In case $\delta \neq \perp$, compute ε as in Theorem 2, otherwise set $\varepsilon = \perp$.
- 12: **return** $\mathbf{x}^d, \varepsilon$;

a *neighborhood*. A good coarsening is akin to “zooming-out”, where nearby nodes are merged into a single neighborhood.

Ideally, a “good” coarsening (Step 1 of the S2D2 algorithm) cannot be found without simultaneously computing the utilities of the defender for that coarsening which is only considered in Step 2 of the S2D2 algorithm. One way to do this is to generate all possible coarsenings, then find the best

defender strategy for each coarsening, and then pick the coarsening and defender strategy that yields the best utility for the defender. Unfortunately, this is not practical to compute. We therefore introduce the concept of a δ -coarsening to ensure that a coarsening is “good” and has some desirable properties.

The scale parameter δ controls the granularity of the coarsening. Since S2D2 neglects rewards smaller than δ , increasing δ reduces the number of nodes the coarsening algorithm considers. A node v is deemed δ -valuable if $R^a(v) > \delta$. The coarsening algorithm then clusters these δ -valuable nodes. In each cluster, all the δ -valuable nodes are relatively close, while the clusters themselves remain relatively separated. The resulting coarsening then consists of a set of neighborhoods, each centered around a cluster of δ -valuable nodes (see Figure 3).

S2D2 coarsens via two steps, as depicted in Algorithm 2. First, it attempts to detect a “high-quality” coarsening, referred to as δ -coarsening. When a δ -coarsening exists, we prove in Section 5 that S2D2 approximates SSE. A δ -coarsening must satisfy four conditions: (i) getting from outside a neighborhood to a δ -valuable node within it takes too much battery; (ii) there are sufficiently many valuable neighborhoods; (iii) a single attacker can collect most δ -valuable rewards in its neighborhood; (iv) the presence of a defender significantly impacts both attacker and defender expected utility. When a δ -coarsening exists, the first step aims to minimize δ , and does so efficiently by applying a binary search. Indeed, if any of conditions (i)-(iv) are not met for some δ_{low} , they cannot be met for any $\delta < \delta_{low}$.

If $\delta_{low} > \delta_{up}$, a δ -coarsening may not exist at all. To this end, if S2D2 fails to detect a δ -coarsening in the first step, it proceeds to the second step, where it coarsens the graph using a greedy heuristic. *It is important to note that S2D2 works even when no δ -coarsening exists — but in this case, the theoretical guarantees do not hold.* In the following Example 2 we provide an illustrative example of a coarsening.

Example 2 (Coarsening). *Consider the graph in Example 1, Figure 3. The gray neighborhood has no valuable nodes and so is removed. Next, getting from one neighborhood to a valuable node of another requires going through the grey neighborhood, which takes a prohibitive amount of battery (i). Note that we only consider nodes circled in red when evaluating this condition as other nodes have no reward. Next, note that a single drone can tackle both red nodes within each neighborhood (iii). Unfortunately, the other two*

conditions (ii) and (iv) are not met with the desired constants required for the theoretical proof to hold. As for (ii), since we present a toy graph as an illustrative example, it only has 3 neighborhoods (and 4 are required). Splitting some neighborhoods into two may potentially violate (i). Similarly, for (iv), a defender can always stay put on one red node and block the attacker from successfully attacking both valuable nodes within every neighborhood, yet in this case, it yields a factor of 2 between the utility from a protected neighborhood and an unprotected one. In more complex games with larger B, P values and larger neighborhoods, the gap could be significantly larger.

Consider an SSG $(G, R^a, P^d, A, D, P, B)$ and let $\delta \in \mathbb{N}$. Given $v, v' \in V$, we write $v \sim_\delta v'$ iff $R^a(v') > \delta$ and $d(v, v') \leq B$. Intuitively, $v \sim_\delta v'$ means that v and v' must belong to the same neighborhood of a coarsening of G in order to satisfy Condition 1. Let \approx_δ denote the reflexive, symmetric, and transitive closure of \sim_δ . Since \approx is an equivalence relation, V/\approx_δ is a partition of V (into equivalence classes). Hence, it is a coarsening that maximizes $|\hat{V}|$ (for Condition 2a) while satisfying Condition 1. To meet Condition 2b, we sort the neighborhoods in V/\approx by $u_{1,0}^{\hat{v},a}$, and remove poor neighborhoods until Condition 2b holds.

As $u_{1,0}^{\hat{v},a}$ requires solving an NP-hard problem [44], we use TSP (Traveling Salesman Problem)-solvers to get lower bounds, and use **best-path** (\hat{v}, δ) to refer to the procedure which looks for a shortest path going through all δ -valuable nodes in \hat{v} . Hence, the algorithm may fail to find a δ -coarsening although one exists, and instead return a $\hat{\delta}$ -coarsening for some greater $\hat{\delta}$. In turn, the resulting coarsening will only be $\epsilon(\hat{\delta})$ -tight. On the other hand, the algorithm is efficient, optimizing on δ with a simple binary search. Moreover, it returns an upper bound on δ , which translates (by Theorem 1) to a concrete bound on the loss from respecting the coarsening, instead of playing an SSE defense strategy. Lastly, the algorithm solves the single-attacker single-defender game in each neighborhood, as described in Section 4.2, to ensure that defending a neighborhood results with a significant utility change for both players.

Lines 4–9 of Algorithm 2 return a partition of V that satisfies (i), i.e. incentivizing drones to stay in their starting neighborhoods throughout the game. Line 9 removes “poor” neighborhoods that the attacker doesn’t care about. Lines 11, 13, 15, and 17 check if a δ -coarsening exists by checking the other three conditions (ii), (iii), (iv), respectively. The algorithm performs binary search on δ , to find the smallest one for which a δ -coarsening exists,

Algorithm 2 Coarsening

Require: An undirected graph $G = (V, E)$;
numbers of attacker and defender drones $A, D \in \mathbb{N}$;
attacker drone's payload $P \in \mathbb{N}$;
drone's battery capacity $B \in \mathbb{N}$;
attacker rewards $R^a \in \mathbb{N}^{|V|}$;
defender penalties $P^d \in \mathbb{Z}_{<0}^{|V|}$.

Ensure: (δ) -Coarsening \hat{V} and δ , or failure.

- 1: $\delta_{low} \leftarrow 1, \delta_{up} \leftarrow 1 + \max_{v \in V} R^a(v)$;
- 2: **while** $\delta_{low} < \delta_{up}$ **do**
- 3: $\delta \leftarrow \lfloor (\delta_{low} + \delta_{up})/2 \rfloor$;
- 4: $\hat{V} \leftarrow V / \approx_{\delta}$;
- 5: init table T ;
- 6: **for each** $\hat{v} \in \hat{V}$ **do**
- 7: $T[\hat{v}] \leftarrow \sum_{v \in \hat{v}.top(P, by=R)} R^a(v)$; {sum of top- P rewards}
- 8: **end for**
- 9: $\hat{V} \leftarrow \{\hat{v} \in \hat{V} \mid \frac{4}{3}T[\hat{v}] \geq T.max()\}$; {Remove poor neighborhoods}
- 10: **if** $|\hat{V}| < 4 \max\{A, D\}$ **then**
- 11: $\delta_{low} \leftarrow \delta + 1$; {Not enough neighborhoods}
- 12: **else if** $\exists \hat{v} \in \hat{V} : |\{v \in \hat{v} \mid R^a(v) > \delta\}| > P$ **then**
- 13: $\delta_{low} \leftarrow \delta + 1$; {Insufficient attacker payload}
- 14: **else if** $\exists \hat{v} \in \hat{V} : \text{best-path}(\hat{v}, \delta) > B$ **then**
- 15: $\delta_{low} \leftarrow \delta + 1$; {Insufficient attacker battery}
- 16: **else if** $\exists \hat{v} \in \hat{V} : \frac{3}{64|\hat{V}|}u_{1,0}^{\hat{v},a} \leq u_{1,1}^{\hat{v},a}$ **or** $\frac{3}{8|\hat{V}|}|u_{1,0}^{\hat{v},d}| \leq |u_{1,1}^{\hat{v},d}| + \delta P$ **then**
- 17: $\delta_{low} \leftarrow \delta + 1$; {Defender presence is ineffective}
- 18: **else**
- 19: $sol \leftarrow (\hat{V}, \delta)$;
- 20: $\delta_{up} \leftarrow \delta$;
- 21: **end if**
- 22: **end while**
- 23: **if** $\delta = 1 + \max_{v \in V} R^a(v)$ **then**
- 24: $\hat{V} \leftarrow \text{K-Means}(V, \text{num_clusters} \propto D, \text{weights} \propto |P^d|)$;
- 25: $sol \leftarrow (\hat{V}, \perp)$;
- 26: **end if**
- 27: **return** sol ;

as the SSE approximation error is linear in δ (as shown in Section 5). A formal definition of a δ -coarsening is given in Section 5.

If the condition in Line 23 holds, it means that no δ -coarsening exists. In this case, S2D2 uses weighted K-Means [45], which has three advantages: (i) it is efficient and simple; (ii) it leverages the planar structure of the graph, and the coordinate-based location of each vertex in the graph; (iii) it takes the penalties into account, by setting them as the weights. The parameter δ can be viewed as a cut-off, where any node with a smaller reward is considered negligible. Hence, S2D2 heuristically assigns δ as the $|\hat{V}|P$ most rewarding target, so that each neighborhood has P rewards $> \delta$ on average. The number of neighborhoods $|\hat{V}|$ is set to be proportional to the number of available defense drones D . We test the performance of this algorithm by conducting experiments on real-world cities in Section 6. Therefore, in what follows, we will seek defense strategies that *respect* a given coarsening, whether it admits the strict theoretical requirements or not, as defined below.

Definition 3 (Strategy Respecting a Coarsening). *A defense (attack) strategy respects the coarsening \hat{V} when the following conditions are met:*

1. *Every defense (attack) drone stays within its starting neighborhood throughout the game.*
2. *Every neighborhood contains up to a single defense (attack) drone.*

$\mathcal{S}_{\hat{V}}^d, \mathcal{S}_{\hat{V}}^a$ denote the sets of pure strategies that respect the coarsening \hat{V} , for the defender and the attacker, respectively.

4.2. Single-Attacker Single-Defender Solution

S2D2 approximates SSE for a single-attacker single-defender game within each neighborhood. Crucially, in the broader multi-drone, multi-neighborhood setting, the defender’s presence in a given neighborhood is probabilistic. In large cities with limited defense resources, it is generally expected that neighborhoods are not protected indefinitely. This probability must be taken into account when considering the single drone game within a given neighborhood, and is therefore introduced as an additional input parameter, denoted by λ .

Brute Force Solution. Since the problem is NP-hard⁷, we use smart enumeration as P, B are small.⁸ We begin with a naive approach which linearizes the problem. We compute the matrices U_λ^a, U_λ^d of the attacker and defender utility for each pair of pure strategies. Note that those values depend on λ , the defender's presence probability. We then omit any dominated pure strategies, and find SSE (\mathbf{x}_d^*, s_a^*) in a similar manner to the single-attacker single-defender SSG (cf. [16]), i.e., we enumerate the set of attacker pure strategies, and for each pure strategy s'_a , we then solve the following LP that maximizes the defender utility, under the constraint that s'_a is the best response:

- Maximize $u^d(\mathbf{x}_d, s'_a, \lambda)$, subject to:
 1. $\mathbf{x}_d \in \mathcal{C}_D$ – Now it is the set of combinations over all non-dominated defense strategies.
 2. For each $s_a \in \mathcal{S}^a$, $u^a(\mathbf{x}_d, s_a, \lambda) \leq u^a(\mathbf{x}_d, s'_a, \lambda)$.

Note that $u^d(\mathbf{x}_d, s_a, \lambda)$ is a linear combination of values from U_λ^d , according to \mathbf{x}_d , and the same holds for u^a and U_λ^a .

Finally, we pick \mathbf{x}_d^*, s_a^* that maximizes the defender utility. The complexity is $|\mathcal{S}^a| \times \text{LP}(|\mathcal{S}^d|, |\mathcal{S}^a| + |\mathcal{S}^d|)$. Namely, for each attacker strategy, we solve a linear program with $|\mathcal{S}^d|$ variables and $|\mathcal{S}^a| + |\mathcal{S}^d|$ constraints. Next, we improve by reducing the relevant strategy space for both the attacker and the defender.

Reducing the Attacker Strategy Space. By narrowing down the strategy space, we expect to move away from the optimal solution and trade-off run time vs. solution quality.

When λ is small, we know that s_a^* is more greedy, as the $(1 - \lambda)$ term dominates. Hence, s_a^* largely ignores the defender. This may eliminate most of the attacker's possible strategies. λ should anyway be small when there are sufficiently many neighborhoods that are attractive to the attacker. When

⁷The problem is NP-hard even for $\lambda = 0$, i.e., when solving the optimization problem for the attacker facing no defender. For example, if $P = B = |\hat{v}|$, deciding whether the attacker has a strategy with utility $u = \sum_{\hat{v} \in \hat{V}} R(\hat{v})$ is equivalent to deciding whether a Hamiltonian path exists in graph $(\hat{v}, E|_{\hat{v}})$.

⁸This assumption is reasonable as most drone attacks take small amounts of time. For instance, [23] tracked all drone flights over The Hague over 8 months and found the average duration to be 298 seconds and the max duration to be 720 seconds.

this is not true, the problem is smaller, and S2D2 takes a random sample of the strategy space, trading-off runtime and quality of the solution. So we may only enumerate a smaller space of possible attacker strategies. To some extent, this can be done without damaging performance. Suppose $s_a, s'_a \in \mathcal{S}^a$ so that $u^a(\perp, s'_a) \leq (1 - \lambda) \cdot u^a(\perp, s_a)$. Then for any strategy $\mathbf{x}_d \in \Delta\mathcal{S}^d$, $u^a(\mathbf{x}_d, s'_a, \lambda) \leq u^a(\perp, s'_a) \leq (1 - \lambda) \cdot u^a(\perp, s_a) \leq u^a(\mathbf{x}_d, s_a, \lambda)$. Therefore, if the attacker's utility from s_a when facing a defender with probability λ is at least the utility from playing s'_a against no defender, we can strike out the strategy s'_a , as s_a strictly dominates it.

When there is a small subset of crucial nodes in each neighborhood which are far apart so that an attacker drone must follow an almost optimal path in order to pass through a couple of them, the number of candidate attacker strategies drops significantly. When this is not the case though, S2D2 randomly samples from the large space of possible strategies. This is depicted in Algorithm 3.

Algorithm 3 ScanAttackStrategies

Require: A weighted, undirected graph $(\hat{v}, E|_{\hat{v}}, R|_{\hat{v}})$;

Defender presence probability λ ;

attacker drone battery capacity and payload $B, P \in \mathbb{N}$;

Threshold th on the number of output attack strategies;

Ensure: attacker drone possible strategies $S^a \subset \mathcal{S}^a$.

- 1: Compute $u_{\max}^a = \max_{s_a \in \mathcal{S}^a} u_{\hat{v}}^a(s_a, \perp)$, the maximal attacker utility at \hat{v} when facing no defender;
 - 2: Set $S^a := \{s_a \in \mathcal{S}^a \mid u_{\hat{v}}^a(s_a, \perp) \geq (1 - \lambda)u_{\max}^a\}$;
 - 3: **if** $|S^a| > \text{th}$ **then**
 - 4: **return** A random sample of size th from S^a ;
 - 5: **end if**
 - 6: **return** S^a ;
-

Narrowing Down Defender Strategy Space. As the attacker's set of best response pure strategies is now small, the dominating set of defense strategies is also expected to be small. Algorithm 5's goal is to output a small subset of dominating defense strategies, as explained below.⁹

⁹Narrowing down the defender strategy space is complex: as there are multiple possible attack strategies, the defender might want to cover many of them with a single strategy, rather than considering the optimal strategy against every potential attack strategy.

Suppose the defender and attacker drones' starting positions are v_d, v_a , respectively, and S^a is the (narrowed) set of possible attack strategies starting from v_a . For each strategy $s_a \in S^a$, up to P nodes are attacked, $v_1(s_a), \dots, v_P(s_a)$, at times $t_1(s_a), \dots, t_P(s_a)$. To further reduce runtime, we may only consider targets with a significant (i.e. less than $-\delta$) defender penalty.

Algorithm 4 catch

Require: An undirected graph $(\hat{v}, E|_{\hat{v}})$;
Attacker pure strategy s_a ;
Defender start position $v_d \in \hat{v}$;
Ensure: $1 \leq i \leq P+1$, the index of the first target the defender is able to protect;
($i = P+1$ indicates the defender is not in time to protect any target)

- 1: Define $(v_1(s_a), \dots, v_{P'}(s_a))$ as the ordered list of targeted nodes in s_a ;
- 2: Remove nodes with an absolute penalty less than δ ;
- 3: Re-index the remaining nodes, and update P' ;
- 4: Define $(t_1(s_a), \dots, t_{P'}(s_a))$ as the planned time steps for each node to be attacked;
- 5: **for** i **from** 1 **to** P' **do**
- 6: Find shortest path π_i from v_d to $v_i(s_a)$;
- 7: Denote its length by t_i^d ;
- 8: **if** $t_i^d \leq t_i(s_a)$ **then**
- 9: **return** i ;
- 10: **end if**
- 11: **end for**
- 12: **return** $P+1$;

We can then compute for the defender, the minimal time to get to each such node (t_1^d, \dots, t_P^d) , and let $1 \leq i \leq P$ be the first target the defender can protect. This is the output of $\text{catch}(v_d, s_a)$ (Algorithm 4) which corresponds to the best strategy when the attacker's pure strategy is known.¹⁰

It should be observed that, at each time point in Algorithm 5, it suffices to decide the set of possible next steps for the defender. We can then explore these using DFS, and eventually return all non-dominated pure strategies.

¹⁰Note that we only find the first node targeted by the attacker that is feasible to protect, not the first node we can catch the attacker at. This is because following a longer path may cover other potential paths the attacker may take, without losing utility from not following the shortest path, when considering the given attacker path.

Algorithm 5 ScanDefenseStrategies

Require: A weighted, undirected graph $(\hat{v}, E|_{\hat{v}}, R|_{\hat{v}})$;
attacker drone battery capacity and payload $B, P \in \mathbb{N}$;
attacker drone start position v_a ;
defense drone start position v_d ;
attacker drone possible strategies $S^a \subset \mathcal{S}^a$.
Ensure: Defense drone possible strategies $S^d \subset \mathcal{S}^d$.

```
1: if  $|S^a| = 1$  then
2:   return  $\text{catch}(v_d, S^a)$ ; {Compute first strike feasible to prevent (and respec-
   tive path).}
3: end if
4: init  $T$ ;
5: for each  $v'_d \in N(v_d) \cup \{v_d\}$  and  $s'_a \in S^a$  do
6:    $T[v'_d, s'_a] \leftarrow \text{catch}(v'_d, s'_a)$ ;
7: end for
8:  $\text{next\_step} \leftarrow \text{prune}(T)$ ; {Omit dominated neighbors}
9:  $S^d \leftarrow \emptyset$ ;
10: for each  $v'_d \in \text{next\_step}$  do
11:   init  $T_S$ ;
12:   for each  $v'_a \in N(v_a) \cup \{v_a\}$  do
13:     {DFS visit}
14:      $\text{update}(S^a)$ ; {Consider only strategies in  $S^a$  that goes from  $v_a$  to  $v'_a$ }
15:      $\tilde{S}^d \leftarrow \text{ScanDefenseStrategies}(v'_d, v'_a, B - 1)$ ;
16:      $T_S[v'_a] \leftarrow \tilde{S}^d$ 
17:   end for
18:    $S^d \leftarrow S^d \cup \text{lift\_strategies}(v'_d, T_S)$ ; {Combine strategies from recursion}
19: end for
20: return  $S^d$ .
```

The more steps the attacker takes (recursion depth), the narrower its strategy space gets, so the search should converge relatively quickly.

The ScanDefenseStrategies algorithm has 3 steps:

1. For each possible next step $v'_d \in N(v_d) \cup \{v_d\}$, compute $\text{catch}(v_d, s_a)$ for each $s_a \in S^a$. Then prune any dominated strategy (where for any strategy of the attacker, it catches the attacker later or at the same targeted node).
2. For each v'_d that survived, and for each possible next attacker step v'_a , recursively call ScanDefenseStrategies and retrieve the set $T_S[v'_a]$ of

non-dominated pure strategies (with $B - 1$, and updated S^a).

3. Lastly, lift pure strategies from (v'_d, \cdot) to a strategy from (v_d, v_a) of the form: “go to v'_d , and for each possible attacker next step v'_a , pick a pure strategy from $T_S[v'_a]$ ”.

The recursion ends either when $B = 0$ or when the attacker strategy space is a singleton — we then use **catch**. To save space, we leverage dynamic programming, and start by solving the problem for $B = 0$ and increment the battery capacity by 1 at every step, solving each instance problem once. After this, we get a reduced matrix U_λ , which only considers a smaller subset of defense and attack strategies.

4.3. The Meta Game: Multi-Drone Solution

The third step in the S2D2 algorithm is to solve the Meta Game, once we know the optimal defender strategy for each neighborhood. The MetaGame looks at the question of which neighborhoods to deploy a defense drone to. This is done via a mixed strategy. The pseudo-code of the MetaGame is in Algorithm 6 and can be described at a high level as follows.

1. We translate the meta-game of allocating defense drones to neighborhoods into a multi-resource attacker defender SSG with nonlinear utilities.
2. An approximation of utilities is given as an input. This is a piece-wise linear approximation derived from solving the single-drone neighborhood game for different λ values.
3. Next, we translate the SSG problem into a MIP.
4. We then build on past work [46] to translate the MIP into a MILP. Their technique allows to replace piecewise linear functions with a linear one by adding a linear number of continuous variables and a logarithmic number of binary variables.
5. Finally, we solve the above MILP and extract the attacker and defender solutions.

Next, we delve into the technical details of the high-level structure of the MetaGame algorithm described above.

Recall that we only consider strategies that respect \hat{V} , i.e., drones stay within their starting neighborhood, and there is up to one attacker and one defender per neighborhood. Therefore, an attacker pure strategy naturally decomposes into an injection $\hat{f}_a : \{1, \dots, A\} \mapsto \hat{V}$ mapping each attacker

Algorithm 6 SolveMetaGame

Require: A set of neighborhoods \hat{V} ;
 numbers of attacker and defender drones $A, D \in \mathbb{N}$;
 (Approximate) attacker and defender utility functions $\{u_v^a(\lambda), u_v^d(\lambda)\}_{v \in \hat{V}}$;

Ensure: An SSE $\langle \hat{p}_d, \hat{f}_a \rangle$, where:
 $\hat{p}_d \in \mathcal{C}_D^{\hat{V}}$, a coverage vector of D drones over the neighborhoods of \hat{V} ;
 \hat{f}_a maps each attacker drone to a neighborhood of \hat{V} ;

- 1: Compute piece-wise linear approximations of the attacker and defender utility functions $\tilde{u}_v^a(\lambda), \tilde{u}_v^d(\lambda)$;
- 2: Initialize a MIP with the objective of maximizing $\sum_{\hat{v}} x_a(\hat{v}) \cdot \tilde{u}_{\hat{v}}^d(\mathbf{x}_d(\hat{v}))$;
- 3: Add constraints on attacker and defender resources: $\sum_{\hat{v}} x_a(\hat{v}) = A, \sum_{\hat{v}} \mathbf{x}_d(\hat{v}) = D$;
- 4: Require variables $x_a(\hat{v}) \in \{0, 1\}$ to be binary and limit continuous variables $0 \leq \mathbf{x}_d(\hat{v}) \leq 1$;
- 5: Add a continuous variable θ_a for attacker threshold;
- 6: Add the following inequality constraints, forcing attacker best response:
- 7: **for each** neighborhood $\hat{v} \in \hat{V}$ **do**
- 8: (i) $\tilde{u}_{\hat{v}}^a \mathbf{x}_d(\hat{v}) \geq x_a(\hat{v}) \cdot \theta_a$;
- 9: (ii) $\tilde{u}_{\hat{v}}^a \mathbf{x}_d(\hat{v}) \leq (1 - x_a(\hat{v})) \cdot \theta_a + x_a(\hat{v}) \tilde{u}_{\hat{v}}^a(0)$;
- 10: **end for**
- 11: Linearize the above MIP (using [46]).
- 12: Let $\langle \hat{p}_d, \hat{f}_a \rangle$ be MILP solution.
- 13: **return** $\langle \hat{p}_d, \hat{f}_a \rangle$;

drone to a neighborhood which it will attack, and for each drone $1 \leq i \leq A$, a pure strategy $\mathcal{S}_i^a = \mathcal{T}_{P,B} \times \mathcal{P}_B^{\hat{f}_a(i)}$, where $\mathcal{P}_B^{\hat{f}_a(i)}$ considers only paths within the neighborhood $\hat{f}_a(i)$.

Similarly, each defender strategy decomposes into a mapping of each defense drone to a neighborhood, and a strategy within this neighborhood. Since all defense drones are identical, when considering mixed defense strategies, it suffices to specify (i) within each neighborhood $\hat{v} \in \hat{V}$ a mixed single-drone defense strategy $\hat{x}^d(\hat{v}) \in \Delta(\mathcal{S}_v^d)$; (ii) for each neighborhood the probability of it being protected, as a *coverage vector* $\hat{p}_d \in \mathcal{C}_D^{\hat{V}}$, where $\mathcal{C}_D^{\hat{V}} := \{\mathbf{x} \in [0, 1]^{|\hat{V}|} : \sum_{\hat{v} \in \hat{V}} x_{\hat{v}} \leq D\}$. Therefore, the defender mixed strategy space decomposes to $\Delta(\mathcal{S}^d) = \mathcal{C}_D^{\hat{V}} \times \prod_{\hat{v} \in \hat{V}} \Delta(\mathcal{S}_v^d)$.

When solving the single-attacker single-defender instance for a neighborhood \hat{v} , $\hat{p}_d(\hat{v})$ denotes the probability $\lambda_{\hat{v}}$ that “a defender is in the hood”. \hat{p}_d is

a coverage vector, representing the probability of presence of a defense drone in each neighborhood. Recall that any vector with entries in $[0, 1]$ that sums up to $\leq D$ is feasible to implement with some mixed strategy of assigning defense drones to neighborhoods.

Given defender (resp. attacker) strategy (\hat{p}_d, \hat{x}_d) (resp. (\hat{f}_a, \hat{s}_a)), where $\hat{s}_a(\hat{f}_a(i)) = (T_a^i, \pi_a^i) \in \mathcal{T}_{P,B} \times \mathcal{P}_B$, the expected utility is the sum of expected utilities from each neighborhood $\hat{f}_a(i)$ attacked, for $1 \leq i \leq A$. The expected utility from neighborhood \hat{v} is the average of the sum of the rewards over the attacker drone set of chosen targets, and the utility when facing a single defender with strategy $\hat{x}_d(\hat{v})$, weighted by $\hat{p}_d(\hat{v})$. That is, for $u \in \{u^a, u^d\}$:

$$u \left(\langle \hat{p}_d, \hat{x}_d \rangle, \langle \hat{f}_a, \hat{s}_a \rangle \right) = \sum_{\substack{1 \leq i \leq A \\ \hat{v}_i = \hat{f}_a(i)}} \left[\hat{p}_d(\hat{v}_i) \cdot u(\hat{x}_d(\hat{v}_i), \hat{s}_a(\hat{v}_i)) + (1 - \hat{p}_d(\hat{v}_i)) u(\perp, \hat{s}_a(\hat{v}_i)) \right]$$

Thus, as the probability a defender is “in the hood” $\hat{p}_d(\hat{v})$ decreases, the attacker drone is better off taking a greedy action. This implies that it is not sufficient to compute SSE for single defender attacker game within each neighborhood to solve the overall multi-drone game. Focusing on a neighborhood, we can extend the utility definition $u(\mathbf{x}_d, s_a, \lambda) := \lambda u(\mathbf{x}_d, s_a) + (1 - \lambda)u(\perp, s_a)$, to consider the probability λ , denoting the probability a defender is in the hood. This may remind the reader of the SSG model with penalties where, even when the attacker is caught, it gets a penalty $P > 0$. We may effectively tune the parameters of the game so that rewards are scaled by λ , and the penalties are the rewards scaled by $(1 - \lambda)$. Section 4.2 discusses how to approximate SSE in a single-attacker single-defender game with parameter λ . We next focus on allocation to neighborhoods and assume an oracle returns (an approximation of) optimal \hat{x}_d^*, \hat{s}_a^* strategies within each neighborhood given \hat{p}_d, \hat{f}_a . This is possible as Section 4.2 shows how to implement the oracle, and Lemma 3 below shows that an approximation suffices. Therefore we get for $u \in \{u^a, u^d\}$:

$$u(\hat{p}_d, \hat{f}_a) = u \left(\langle \hat{p}_d, \hat{x}_d^* \rangle, \langle \hat{f}_a, \hat{s}_a^* \rangle \right) = \sum_{i=1}^A u \left(\hat{x}_d^*(\hat{f}_a(i)), \hat{s}_a^*(i), \hat{p}_d(\hat{f}_a(i)) \right).$$

We next pick a distribution \hat{p}_d which minimizes the utility above when \hat{f}_a is the best response to \hat{p}_d . Hence, we get a typical SSG, with an attacker

with multiple (A) resources, with one important detail: the utility of each neighborhood \hat{v} is not necessarily linear with the coverage $\hat{p}^d(\hat{v})$, although it is monotonic decreasing.

4.4. Generalization of Multi-Resource SSGs

In this section, we show how to generalise the work of [20] to handle a non-linear dependency of the attacker and utility functions on \mathbf{x}_d , the defense probability on each target.

When both utilities are linear with \mathbf{x}_a and \mathbf{x}_d , there is a complete characterization of the Nash equilibrium of the game. Indeed, best-responding simply means attacking (defending) the D (A) targets with the highest (marginal) utility for the defender (attacker). Therefore:

Lemma 1. *If u^a, u^d are linear with \mathbf{x}_d and \mathbf{x}_a , let $v^d(t, x_a(t)) = a_t(R^d(t) - P^d(t))$ be the defender marginal utility from attacking target t . Then $(\mathbf{x}_d, \mathbf{x}_a)$ is a Nash equilibrium iff there exist thresholds θ_a, θ_d such that:*

- $\mathbf{x}_a \in \text{BR}(\mathbf{x}_d)$. *Equivalently:*
 - $u^a(t, \mathbf{x}_d(t)) < \theta_a \Rightarrow x_a(t) = 0$.
 - $u^a(t, \mathbf{x}_d(t)) > \theta_a \Rightarrow x_a(t) = 1$.
 - $\sum_t x_a(t) = A$.
- $\mathbf{x}_d \in \text{BR}(\mathbf{x}_a)$. *Equivalently:*
 - $v^d(t, x_a(t)) < \theta_d \Rightarrow \mathbf{x}_d(t) = 0$.
 - $v^d(t, x_a(t)) > \theta_d \Rightarrow \mathbf{x}_d(t) = 1$.
 - $\sum_t \mathbf{x}_d(t) = D$.

When u^d is linear with \mathbf{x}_d , the defender's marginal utility $\frac{\partial u^d}{\partial \mathbf{x}_d}$ is a constant, and in particular, is independent of \mathbf{x}_d . Therefore, the utility the defender gets from protecting target t with probability “budget” $\mathbf{x}_d(t)$ is $\mathbf{x}_d(t) \cdot v^d(t, x_a(t))$. Therefore, best responding means first covering the top D targets, and when there are ties for the D^{th} place, any randomization over the corresponding targets will result in a valid best response.

However, when $\frac{\partial u^d}{\partial \mathbf{x}_d}$ is a function of \mathbf{x}_d , this is not the case any longer. Indeed, the above condition would be necessary, suggesting \mathbf{x}_d to be a *local* maximum of u^d , as otherwise (assuming u^d is continuously differentiable) one

could make small changes and increase the defender's utility. Nevertheless, it will not ensure a *global* maximum of u^d , meaning a best response. If u^d was concave with \mathbf{x}_d , any local maximum would also be global and therefore [20]'s algorithm would still work. Unfortunately, we cannot make such an assumption in our game.

Nevertheless, we are not interested in computing a Nash equilibrium, but a SSE. Therefore, we first show that the criterion for the attacker to best respond remains intact:

Lemma 2. *Assume u^a is linear with \mathbf{x}_a , and that $|V| > A$, and that $P^a(t) < R^a(t)$ for every target t . Let \mathbf{x}_d be a defense mixed strategy. Then $\mathbf{x}_a \in \text{BR}^a(\mathbf{x}_d)$ iff there exist a threshold θ_a such that:*

- $u^a(t, \mathbf{x}_d(t)) < \theta_a \Rightarrow x_a(t) = 0.$
- $u^a(t, \mathbf{x}_d(t)) > \theta_a \Rightarrow x_a(t) = 1.$
- $\sum_t x_a(t) = A.$

Proof. (\Leftarrow) Suppose \mathbf{x}_a admits the above conditions. Then, the marginal attacker utility from attacking target t is $u^a(t, \mathbf{x}_d(t))$, therefore, independent of \mathbf{x}_a . Hence, best responding would first protect the targets with the highest attacker utility given \mathbf{x}_d , and any randomization over the A^{th} target will result with the same overall attacker utility. (\Rightarrow) Assume by way of contradiction that one of the above conditions doesn't hold. If there are two targets t_1, t_2 such that $u^a(t_1, \mathbf{x}_d(t_1)) < u^a(t_2, \mathbf{x}_d(t_2))$, and $0 < x_a(t_1), x_a(t_2) < 1$, the attacker's utility will increase by shifting attacker probability mass from t_1 to t_2 until either $x_a(t_1)$ gets to 0 or $x_a(t_2)$ gets to 1. Last, if not all of the attacker resources are utilized, we can increase the attack probability on all targets, and increase the attacker's overall utility as well. Note that this is why we need to assume $|V| > A$ and $P^a(t) < R^a(t)$ on every target t . \square

Next, we opt to transform the SSE computation into a mixed integer program, which is a well-studied problem. We start from the following optimization problem:

$$\begin{aligned}
& \text{maximize: } \sum_t x_a(t) \cdot u^d(t, \mathbf{x}_d(t)) \\
& \text{subject to: } \sum_t x_a(t) = A, \sum_t \mathbf{x}_d(t) = D, \\
& \quad x_a(t) \in \{0, 1\}, 0 \leq \mathbf{x}_d(t) \leq 1, \\
& \quad u^a(t, \mathbf{x}_d(t)) \geq x_a(t) \cdot \theta_a, \\
& \quad u^a(t, \mathbf{x}_d(t)) \leq (1 - x_a(t)) \cdot \theta_a + x_a(t) R^a(t).
\end{aligned} \tag{3}$$

Evidently, a solution to the above MIP is SSE. Indeed, the objective is to maximize the defender's utility, over all possible coverage vectors \mathbf{x}_d of the defender. Demanding $\sum_t \mathbf{x}_d(t) = D$ is okay because the utilities are monotonically increasing. Finally, in the SSE framework, we can assume that the attacker's strategy is pure, that is, $x_a(t) \in \{0, 1\}$ which enables us to write the condition for the attacker to best respond (described in Lemma 2) with linear inequalities over the variables $\mathbf{x}_a(t)$.

The only problem is that $u^d(t, \mathbf{x}_d(t))$ and $u^a(t, \mathbf{x}_d(t))$ are non-linear w.r.t. $\mathbf{x}_d(t)$ in general. However, this can be handled using standard techniques to approximate the utility functions with piece-wise linear approximations \tilde{u}^d, \tilde{u}^a . This is inevitable as we don't have closed form formulas for the utilities — rather, they are derived from the algorithm for the single attacker/single defender drone problem in Step 2 of the S2D2 algorithm). We refer to [46] for an overview of the technique. In principle, we can add a logarithmic number of integer variables, and linear number of continuous variables, and replace the utilities with linear expressions using the new variables.

We can bound the error from approximating the utilities by the following lemma:

Lemma 3. *Let $G = (\hat{V}, A, D, u^a, u^d)$ be a (non-sequential) attacker SSG. Let $\epsilon > 0$ and let \tilde{u}^a, \tilde{u}^d be different attacker and defender utility functions, such that $\|(u^a, u^d) - (\tilde{u}^a, \tilde{u}^d)\|_\infty < \epsilon$. That is, on every pair of strategies (\mathbf{x}_d, s_a) , the attacker and defender utility outputs differ by up to ϵ , using the other utility functions. Then if (\mathbf{x}_d, s_a) is an ϵ -SSE of G , it is also a 2ϵ -approximate SSE of \tilde{G} where the utilities are replaced with \tilde{u}^a, \tilde{u}^d .*

Proof. Indeed, assume that for any pair of strategies, $(\mathbf{x}_d, \mathbf{x}_a)$, we have that $|u^a(\mathbf{x}_d, \mathbf{x}_a) - \tilde{u}^a(\mathbf{x}_d, \mathbf{x}_a)| < \epsilon$ and $|u^d(\mathbf{x}_d, \mathbf{x}_a) - \tilde{u}^d(\mathbf{x}_d, \mathbf{x}_a)| < \epsilon$.

Let (\mathbf{x}_d, s_a) be an ϵ -SSE with respect to (u^a, u^d) . Then $s_a \in \text{BR}_{\epsilon, u^a, u^d}^d(\mathbf{x}_d)$, and therefore, $s_a \in \text{BR}_{2\epsilon, \tilde{u}^a}^a(\mathbf{x}_d)$. Thus:

$$\tilde{u}^d(\mathbf{x}_d, \text{BR}_{2\epsilon, \tilde{u}^a, \tilde{u}^d}^d(\mathbf{x}_d)) \geq \tilde{u}^d(\mathbf{x}_d, s_a).$$

Next, let $s'_a \in \text{BR}_{2\epsilon, \tilde{u}^a, \tilde{u}^d}^d(\mathbf{x}_d)$. Then, the above inequality says $\tilde{u}^d(\mathbf{x}_d, s'_a) \geq \tilde{u}^d(\mathbf{x}_d, s_a)$. Analogously, let $(\tilde{\mathbf{x}}_d, \tilde{s}_a)$ be an ϵ -SSE with respect to $(\tilde{u}^a, \tilde{u}^d)$, and let $\tilde{s}'_a \in \text{BR}_{2\epsilon, \tilde{u}^a, \tilde{u}^d}^d(\tilde{\mathbf{x}}_d)$. Then $u^d(\tilde{\mathbf{x}}_d, \tilde{s}'_a) \geq u^d(\tilde{\mathbf{x}}_d, \tilde{s}_a)$. Thus:

$$\begin{aligned} \tilde{u}^d(\mathbf{x}_d, s'_a) - \tilde{u}^d(\tilde{\mathbf{x}}_d, \tilde{s}_a) &\geq \tilde{u}^d(\mathbf{x}_d, s_a) - \tilde{u}^d(\tilde{\mathbf{x}}_d, \tilde{s}_a) \geq \\ u^d(\mathbf{x}_d, s_a) - u^d(\tilde{\mathbf{x}}_d, \tilde{s}_a) - 2\epsilon &\geq u^d(\mathbf{x}_d, s_a) - u^d(\tilde{\mathbf{x}}_d, \tilde{s}'_a) - 2\epsilon \geq 0 - 2\epsilon. \end{aligned}$$

Finally, since $s_a \in \text{BR}_{\epsilon, u^a, u^d}^d(\mathbf{x}_d)$, $s_a \in \text{BR}_{2\epsilon, \tilde{u}^a}^a(\mathbf{x}_d)$, as desired. Therefore, (\mathbf{x}_d, s_a) is a 2ϵ -approximate SSE with respect to $(\tilde{u}^a, \tilde{u}^d)$. \square

Finally, we can use standard techniques, such as the one described in [47], to linearize the resulted MIP, and solve a MILP.

5. Theoretical Analysis: SSE Approximation

In this section, we prove that if Algorithm 2 outputs a δ -coarsening, then it is an $\epsilon(\delta)$ -approximate SSE.

First, a formal definition of a δ -coarsening is provided in Definition 4. While this definition provides a precise framework, it is somewhat restrictive, and the choice of constants may impose limitations. We stress that this definition is only needed for the rigorous correctness proof of S2D2 (Theorem 1). *Nevertheless, it is important to note that S2D2 yields good results in practice on real-world large-scale cities, even if such a δ -coarsening does not exist, as demonstrated via exhaustive experimentation described in Section 6.*

Definition 4 (δ -Coarsening). Let $G_{\hat{v}} = (\hat{v}, E|_{\hat{v}})$ be some neighborhood. We denote by $u_{A,D}^{\hat{v},d}$ ($u_{A,D}^{\hat{v},a}$), the (maximal) utility of a defender (an attacker) at SSE in $G_{\hat{v}}$ given A attacker drones and D defense drones. Let $\delta > 0$. A δ -coarsening \hat{V} is a coarsening that satisfies the following conditions:

1. For each $\hat{v} \in \hat{V}$, $v' \notin \hat{v}$, and $v \in \hat{v}$ with $R^a(v) > \delta$, it is the case that $d(v, v') > B$, where d is shortest path length.
2. (a) Number of neighborhoods $|\hat{V}| > 4 \max\{A, D\}$.

- (b) For each $\hat{v}, \hat{v}' \in \hat{V}$: $\frac{4}{3}u_{1,0}^{\hat{v},a} > u_{1,0}^{\hat{v}',a} - \delta P$.
3. For each $\hat{v} \in \hat{V}$: $u_{A,0}^{\hat{v},a} < u_{1,0}^{\hat{v},a} + \delta AP$.
4. For each $\hat{v} \in \hat{V}$: $\frac{3}{64|\hat{V}|}u_{1,0}^{\hat{v},a} > u_{1,1}^{\hat{v},a}$ and $\frac{3}{8|\hat{V}|}|u_{1,0}^{\hat{v},d}| > |u_{1,1}^{\hat{v},d}| + \delta P$.

Conditions 1-4 formalize conditions (i)-(iv) in Section 4.1 respectively. Condition 1 suggests that it takes a prohibitive amount of time to move from any node outside a neighborhood into a valuable node within that neighborhood.¹¹ This condition incentivizes drones to stay within their starting neighborhoods throughout the game. It is also a practical political reality — city security officials need to be *seen* to be distributing defensive assets in a fair way across the city rather than appearing to give “preference” to certain places, even if they are high utility locations.

Condition 2a suggests that there are not enough defense/attack drones to protect/attack each neighborhood with probability $\geq 1/4$, as security resources are limited. If not, one may consider partitioning the neighborhoods further, though this may violate Condition 1. This condition incentivizes the attacker to be more greedy, as neighborhoods with no defender with probability $\geq 3/4$ are sure to exist, and Condition 1 ensures defender drones will not reach an unprotected neighborhood in time. In turn, Condition 2b says that since there are many neighborhoods, the attacker will not go to a low value neighborhood regardless of the defense strategy, and therefore there is no reason for defending it either. As a result, we may ignore this neighborhood altogether, and simplify the graph.¹²

As for Condition 3, note that $u_{A,0}^{\hat{v},a} \leq A \cdot u_{1,0}^{\hat{v},a}$ always holds. However, when there is variability in the rewards and valuable rewards are sparse, we expect a smaller gap between the two, since one cannot exploit the same target twice. In particular, Condition 3 holds if a single attacker can collect all rewards in \hat{v} with $R^a(\cdot) > \delta$. This should be the case when valuable targets are sparse and lie in the interior of neighborhoods rather than near the periphery. This condition incentivizes the attacker to spread her drones across

¹¹We don’t require $R^a(v') > \delta$ because the goal of defense drones is to catch the attacker before it causes more damage. Therefore, if there is a node $v \in \hat{v}$ with $R^a(v) = 0$, that is close to valuable nodes of multiple different neighborhoods, placing a defense drone at v could be a good strategy. After the attacker places her drones, the defensive drone will decide which neighborhood to go to in order to catch the attacker.

¹²For this reason, we do not require the coarsening \hat{V} to be a partition of V (i.e., $\bigcup_{i \in [1,k]} \hat{v}_i \neq \hat{V}$).

different neighborhoods to increase the chance of attacking an unprotected neighborhood, as by Condition 2a the chance of a neighborhood being unprotected is not negligible. At the same time, it incentivizes the defender to spread her drones across different neighborhoods to decrease the chance of a successful attack on an unprotected neighborhood. However, this argument holds only if neighborhoods are comparably valuable, which is captured by the following condition.

Condition 4 suggests that the presence of a defense drone in a neighborhood makes a significant impact on defender and attacker drone utility. The constraints ensure that the damage done by the attacker facing an undefended neighborhood is significantly larger than the damage done when facing a single defender, where $u_{A,D}^{\hat{v},d}$ is defined analogously to $u_{A,D}^{\hat{v},a}$ (cf. Condition 3) for the defender. The intuition is that a defender can always start at the center of a neighborhood, and thus be able to catch the attacker relatively quickly, whereas by Condition 3, the attacker has enough battery and payload to destroy all crucial spots of a neighborhood when no defender is present.

Sufficiency. When a δ -coarsening exists, we will show that an ϵ -SSE can be computed efficiently. The reason is that both the attacker and defender are incentivized to spread their drones out across different neighborhoods, which results in a decomposition of the multi-drone game into multiple single-attacker single-defender drone sub-games. To show this, we start with a definition:

Definition 5 (ϵ -tight coarsening of a Graph). *We say that \hat{V} is an ϵ -tight coarsening if there exist $\mathbf{x}_d \in \Delta(\mathcal{S}_{\hat{V}}^d)$ and $s_a \in \mathcal{S}_{\hat{V}}^a$ such that (\mathbf{x}_d, s_a) is ϵ -approximate SSE.*

We emphasize again that the S2D2 algorithm works even when a tight coarsening does not exist. The above definition is only needed for the formal proof that yields theoretical results on the quality of the output strategy. Specifically, we prove a theoretical bound on the loss of the defender and the attacker caused by restricting their strategies to respect a given coarsening \hat{V} , which is ϵ for an ϵ -tight coarsening. Therefore, we will need to compute $\epsilon(\delta)$ for a δ -coarsening.

Our restrictions on δ -coarsening enable us to prove some nice properties, e.g. that a δ -coarsening is always ϵ -tight. To show this, we first analyze the loss of the attacker from respecting a coarsening \hat{V} .

Lemma 4. Let $\mathbf{x}_d \in \Delta(\mathcal{S}^d)$, $s_a \in \mathcal{S}^a$, $\delta > 0$, and \hat{V} be a δ -coarsening. Then, there exists a strategy $s'_a \in \mathcal{S}_{\hat{V}}^a$ such that $u^a(\mathbf{x}_d, s'_a) \geq u^a(\mathbf{x}_d, s_a) - \epsilon$, and $u^d(\mathbf{x}_d, s'_a) \geq u^d(\mathbf{x}_d, s_d)$, for $\epsilon = 2\delta AP$.

Proof. Strategy s'_a is constructed in two steps. First, in s_a^1 each drone stays within its starting neighborhoods. Then, in $s'_a := s_a^2$, in addition there is a single attacker drone in each neighborhood. The attacker loss is then bounded by the sum of the losses from the two steps.

First, consider the following strategy s_a^1 . All attacker drones are placed as in s_a , and follow the same paths. Whenever an attacker drone in s_a crosses a neighborhood, the corresponding drone in s_a^1 halts. Note that since attacker drones are not coordinated after initial allocation, s_a^1 is well defined. Specifically, the strategy of other attacker drones is not affected. By Condition 1, when an attacker drone moves across neighborhoods, it can only get negligible rewards. Therefore, following s_a^1 may have a utility loss of up to δAP for the attacker drone compared to s_a . Indeed, for every attack drone and attack payload unit, it could be that in s_a it picked a reward smaller than δ , and in s_a^1 it doesn't collect this reward.

Next, assume $A' > 1$ attacker drones were assigned the same neighborhood \hat{v}' in s_a^1 . For each neighborhood $\hat{v} \in \hat{V}$, let $\lambda_{\hat{v}}$ be the probability that at least one defense drone is allocated to neighborhood \hat{v} at time $t = 0$, with respect to \mathbf{x}_d . Among all neighborhoods that are not occupied with any attacker drone, let $\hat{V}_{A'}$ be the A' least protected neighborhoods with respect to \mathbf{x}_d . Then at $t = 0$, each neighborhood $\hat{v}_{a'} \in \hat{V}_{A'}$ is protected with probability at most $\lambda_{\hat{v}_{a'}} \leq \frac{D}{|\hat{V}| - A}$. Indeed, assume for purposes of contradiction that they are protected with probability $> \frac{D}{|\hat{V}| - A}$. Then since those are the least protected, all unoccupied neighborhoods are protected with probability $> \frac{D}{|\hat{V}| - A}$, and there are at least $|\hat{V}| - A$ such neighborhoods. However, even protecting $|\hat{V}| - A$ neighborhoods with probability $\frac{D}{|\hat{V}| - D}$ already requires D defense resources, hence such a defense coverage vector is not feasible $\mathbf{x}_d \notin \mathcal{C}_D$, a contradiction.

Now, by Condition 2a, $\frac{D}{|\hat{V}| - A} \leq \frac{D}{3D + A - A} = \frac{1}{3}$. Consider spreading the attacker drones from \hat{v} to $\hat{V}_{A'}$, and play greedily, that is, maximize the attacker utility when facing no defender. Denote this strategy by s_a^2 .

At worst, the utility of the attacker drones from $\hat{V}_{A'}$ is $\frac{2}{3} \sum_{\hat{v}_{a'} \in \hat{V}_{A'}} u_{1,0}^{\hat{v}_{a'}} - \delta PA'$. Indeed, with probability $\geq \frac{2}{3}$, there are no defenders in $\hat{v}_{a'}$ at $t = 0$.

Assume by way of contradiction that a defender catches an attacker drone in $\hat{v}_{a'}$ at v_m , before it reaches some valuable node $v \in \hat{v}_{a'}$ with reward $R^a(v) > \delta$. Then, let v_a be the start node for the attacker and v_d be the start node for the defender. Since they both begin at $t = 0$ and meet at v , we know that $d(v_d, v_m) = d(v_a, v_m)$. By triangular inequality, $d(v_d, v) \leq d(v_d, v_m) + d(v_m, v) = d(v_a, v_m) + d(v_m, v) \leq B$, as the attacker moves from v_a to v_m and then to v in less than B units of battery. However, $d(v_d, v) \leq B$ and $R^a(v) > \delta$ contradicts Condition 1. Therefore, the defenders can cause a utility loss for each attacker of up to $P\delta$.

On the other hand, at best, the utility of the A' drones in s_a^1 is $u_{A',0}^{\hat{v}'}$. Therefore, the utility loss of the attacker is at most: $\delta PA' + u_{A',0}^{\hat{v}'} - \frac{2}{3} \sum_{\hat{v}_{a'} \in \hat{V}_{A'}} u_{1,0}^{\hat{v}_{a'}}$. By Condition 3, this is less than $2\delta PA' + u_{1,0}^{\hat{v}',a} - \frac{2}{3} \cdot 2 \min_{\hat{v}_{a'} \in \hat{V}_{A'}} u_{1,0}^{\hat{v}_{a'}}$. Then, by Condition 2b, the overall utility loss is bounded by $2\delta PA'$. Repeating the above for all neighborhoods in s_a^1 that were initially assigned with multiple attacker drones will result with a total attacker utility loss of up to $\epsilon = 2\delta PA$.

Thus, a greedy strategy $s'_a = s_a^2 \in \mathcal{S}_{\hat{V}}^a$ of spreading the attacker drones unoccupied neighborhoods and playing greedily, ignoring the defender, results with a negligible loss in utility, regardless of the defense strategy. \square

It follows that the attacker respects the coarsening.

Corollary 1. *Let \hat{V} be a δ -coarsening. Then:*

$$\begin{aligned} \max_{\mathbf{x}_d \in \Delta(\mathcal{S}_{\hat{V}}^d)} u^d(\mathbf{x}_d, \text{BR}_{\epsilon}^d(\mathbf{x}_d)) &\geq \max_{\mathbf{x}_d \in \Delta(\mathcal{S}_{\hat{V}}^d)} u^d(\mathbf{x}_d, \text{BR}_{\epsilon, \hat{V}}^d(\mathbf{x}_d)) \\ \max_{\mathbf{x}_d \in \Delta(\mathcal{S}^d)} u^d(\mathbf{x}_d, \text{BR}_{\epsilon, \hat{V}}^d(\mathbf{x}_d)) &\geq \max_{\mathbf{x}_d \in \Delta(\mathcal{S}^d)} u^d(\mathbf{x}_d, \text{BR}^d(\mathbf{x}_d)) \end{aligned}$$

where $\text{BR}_{\epsilon, \hat{V}}^d(\mathbf{x}_d) := \arg \max_{s_a \in \text{BR}_{\epsilon}^a(\mathbf{x}_d) \cap \mathcal{S}_{\hat{V}}^a} u^d(\mathbf{x}_d, s_a)$ considers only strategies from $\text{BR}_{\epsilon}^a(\mathbf{x}_d)$

that respect the coarsening, and only then takes the strategy that favors the defender.

Proof. Let $\mathbf{x}_d \in \Delta(\mathcal{S}^d)$ be any strategy. Then by definition, $\text{BR}_{\epsilon, \hat{V}}^a(\mathbf{x}_d) = \text{BR}_{\epsilon}^a(\mathbf{x}_d) \cap \mathcal{S}_{\hat{V}}^a \subseteq \text{BR}_{\epsilon}^a(\mathbf{x}_d)$. Therefore, $\text{BR}_{\epsilon}^d(\mathbf{x}_d)$ can only increase defender utility:

$$\begin{aligned} u^d(\mathbf{x}_d, \text{BR}_{\epsilon}^d(\mathbf{x}_d)) &= \max_{s_a \in \text{BR}_{\epsilon}^a(\mathbf{x}_d)} u^d(\mathbf{x}_d, s_a) \geq \\ \max_{s_a \in \text{BR}_{\epsilon, \hat{V}}^a(\mathbf{x}_d)} u^d(\mathbf{x}_d, s_a) &= u^d(\mathbf{x}_d, \text{BR}_{\epsilon, \hat{V}}^d(\mathbf{x}_d)). \end{aligned}$$

Note that $\text{BR}_{\epsilon, \hat{V}}^d(\mathbf{x}_d) \neq \emptyset$ by Lemma 4. Since the inequality above holds for every $\mathbf{x}_d \in \mathcal{S}^d$, it also holds when maximizing over $\mathbf{x}_d \in \Delta(\mathcal{S}_{\hat{V}}^d)$. Conversely, let $s_a \in \text{BR}^d(\mathbf{x}_d)$. Then by Lemma 4, there exist $s'_a \in \text{BR}_{\epsilon, \hat{V}}^a(\mathbf{x}_d)$ such that $u^d(\mathbf{x}_d, s'_a) \geq u^d(\mathbf{x}_d, s_a)$. Therefore:

$$\begin{aligned} u^d(\mathbf{x}_d, \text{BR}_{\epsilon, \hat{V}}^d(\mathbf{x}_d)) &= \max_{s'_a \in \text{BR}_{\epsilon, \hat{V}}^a(\mathbf{x}_d)} u^d(\mathbf{x}_d, s'_a) \geq \\ \max_{s_a \in \text{BR}^a(\mathbf{x}_d)} u^d(\mathbf{x}_d, s_a) &= \max_{\mathbf{x}_d \in \Delta(\mathcal{S}^d)} u^d(\mathbf{x}_d, \text{BR}^d(\mathbf{x}_d)). \end{aligned}$$

Again, since the above holds for every \mathbf{x}_d , it also holds when maximizing over $\mathbf{x}_d \in \Delta(\mathcal{S}^d)$. \square

We now derive bounds on the utility loss (gain) of the attacker (defender) from increasing the protection of a neighborhood.

Lemma 5. *Consider any neighborhood \hat{v} , protected with some probability λ . Then, increasing the probability a defender is in the hood \hat{v} by $\eta > 0$, will result in the following lower bounds on the attacker utility loss and defender utility gain:*

1. $u^{\lambda, a} - u^{\lambda+\eta, a} \geq \eta \cdot u_{1,0}^a - u_{1,1}^a.$
2. $u^{\lambda+\eta, d} - u^{\lambda, d} \geq -\eta \cdot u_{1,0}^d + u_{1,1}^d.$

Recall that $u_{A,D}^a$ is the maximal attacker utility from attacking a neighborhood with A attack drones, facing D defense drones. Analogously, $u_{A,D}^d$ is the maximal defender utility when protecting a neighborhood against A attack drones using D defense drones.

Proof. The attacker utility is bounded as follows:

$$(1 - \lambda)u_{1,0}^a \leq u^{\lambda, a} \leq (1 - \lambda)u_{1,0}^a + \lambda u_{1,1}^a.$$

The defender utility is bounded as follows:

$$\lambda \cdot u_{1,1}^d + (1 - \lambda)u_{1,0}^d \leq u^{\lambda, d} \leq (1 - \lambda)u_{1,0}^d.$$

Subtracting the lower and upper bounds appropriately yields the above lower bounds. \square

Theorem 1. *A δ -coarsening is ϵ -tight, for $\epsilon = 2\delta AP$.*

Proof. First, by Corollary 1, we know that:

$$\max_{\mathbf{x}_d \in \Delta(\mathcal{S}_{\hat{V}}^d)} u^d(\mathbf{x}_d, \text{BR}_{\epsilon}^d(\mathbf{x}_d)) \geq \max_{\mathbf{x}_d \in \Delta(\mathcal{S}_{\hat{V}}^d)} u(\mathbf{x}_d, \text{BR}_{\epsilon, \hat{V}}^d(\mathbf{x}_d)).$$

Therefore, we will assume w.l.o.g that the attacker respects the coarsening \hat{V} , and compare the RHS with:

$$\max_{\mathbf{x}_d \in \Delta(\mathcal{S}^d)} u^d(\mathbf{x}_d, \text{BR}_{\epsilon, \hat{V}}^d(\mathbf{x}_d)) \geq \max_{\mathbf{x}_d \in \Delta(\mathcal{S}^d)} u^d(\mathbf{x}_d, \text{BR}^d(\mathbf{x}_d))$$

where the last inequality is again by Corollary 1.

That is, it is enough to bound the loss of the defender from respecting the coarsening \hat{V} , when assuming that the attacker respects the coarsening. Let $\mathbf{x}_d \in \Delta \mathcal{S}^d$.

Similarly to the attacker, consider strategy \mathbf{x}_d^1 where defense drones stop before crossing a neighborhood. Suppose that with probability > 0 , playing \mathbf{x}_d , defense drone $1 \leq i_D \leq D$ catches an attacker drone $1 \leq i_A \leq A$. Let v_D be the start position of defense drone i_D , v_M be the meeting point, at time $1 < t \leq B$, and consider any path π_A of length $B - t$ starting from v_M within the attacker's starting neighborhood, ending at some node v_A , for attacker drone i_A . Then the path from v_D to v_A is of length B , and therefore, by Condition 1, all of the nodes along π_A yield a reward $\leq \delta$. Hence, playing \mathbf{x}_d^1 results in an attacker utility gain of up to δAP . Therefore, to this end, drones stay within their neighborhoods. Again, we stress that \mathbf{x}_d^1 is only defined in case the defense drones' movement is not coordinated after initial allocation.

Next, we claim that the defender should not allocate two defense drones to the same neighborhood. Let $p < 1$ be the probability that each neighborhood is protected with a single drone, in \mathbf{x}_d^1 . We can consider a strategy \mathbf{x}_d^2 that coincides with \mathbf{x}_d^1 when the coarsening is respected (which happens with probability p), and then the utility loss will be bounded by a factor of $1 - p < 1$. Thus, w.l.o.g, assume $p = 0$.

Denote by $\mathbf{c} \in \mathcal{C}_{D-1}$ the coverage of the neighborhoods, with respect to \mathbf{x}_d^1 . That is, $c_{\hat{v}}$ is the probability that neighborhood \hat{v} is protected with at least a single defender. Denote by $\hat{\text{BR}}^d$ the set of A targets the attacker is attacking when facing \mathbf{x}_d^1 (this is well defined since we may now safely assume the attacker respects the coarsening).

Next, we want to introduce a strategy \mathbf{x}_d^2 that respects the coarsening and has comparable defender utility. Denote by $\mathbf{c}' \in \mathcal{C}_D$ the coverage vector for

\mathbf{x}_d^2 . We are interested in increasing the coverage of $\hat{\mathbf{B}}\mathbf{R}^d$, while maintaining the condition that these targets are in $\mathbf{B}\mathbf{R}^d$. We have an extra defense unit to allocate, since in \mathbf{x}_d^2 , only $D - 1$ neighborhoods are covered in each pure strategy.

Let $0 < r < 1$. By Lemma 5, using $5r$ defense resources on each target outside of $\hat{\mathbf{B}}\mathbf{R}^d$, we can decrease the attacker's utility by at-least $5r \cdot u_{1,0}^{\hat{v},a} - u_{1,1}^{\hat{v}}$.

We will do the same for targets in $\hat{\mathbf{B}}\mathbf{R}^d$, increasing their coverage by $3r$. By Lemma 5, the attacker's utility will decrease by at most $3r \cdot u_{1,0}^{\hat{v},a} + u_{1,1}^{\hat{v}}$. By Condition 2b, the set $\hat{\mathbf{B}}\mathbf{R}^d$ remains the attacker's best response, if $ru_{1,0}^{\hat{v}} > 2 \max_{\hat{v} \in \hat{V}} u_{1,1}^{\hat{v},a}$. Meanwhile, again by Lemma 5, the defender's utility on every $\hat{v} \in \hat{\mathbf{B}}\mathbf{R}^d$ increases by at least $-3r \cdot u_{1,0}^{\hat{v},d} + u_{1,1}^{\hat{v},d}$.

We therefore take $r = \frac{1}{8|\hat{V}|}$, and require the following:

1. $\frac{1}{8|\hat{V}|} \frac{3}{4} \max_{\hat{v} \in \hat{V}} u_{1,0}^{\hat{v},a} > 2 \max_{\hat{v} \in \hat{V}} u_{1,1}^{\hat{v},a}$.
2. $\frac{3}{8|\hat{V}|} |u_{1,0}^{\hat{v},d}| > |u_{1,1}^{\hat{v},d}| + \delta P$.

The first condition ensures that the attacker's best response set is maintained, and the second condition ensures the defender utility is decreased by up to δAP in total. Both of these hold by Condition 4, which states that (and quantifies how) the presence of a defense drone in a neighborhood significantly affects the attacker and defender utilities.

Therefore, strategy \mathbf{x}_d^2 results with up to an additional δAP utility loss for the defender. This completes the proof, as:

$$\begin{aligned} \max_{\mathbf{x}_d \in \Delta(\mathcal{S}_{\hat{V}}^d)} u^d(\mathbf{x}_d, \mathbf{B}\mathbf{R}_{\epsilon}^d(\mathbf{x}_d)) &\geq \max_{\mathbf{x}_d \in \Delta(\mathcal{S}_{\hat{V}}^d)} u(\mathbf{x}_d, \mathbf{B}\mathbf{R}_{\epsilon, \hat{V}}^d(\mathbf{x}_d)) \geq \\ \max_{\mathbf{x}_d \in \Delta(\mathcal{S}^d)} u^d(\mathbf{x}_d, \mathbf{B}\mathbf{R}_{\epsilon, \hat{V}}^d(\mathbf{x}_d)) - \epsilon &\geq \max_{\mathbf{x}_d \in \Delta(\mathcal{S}^d)} u(\mathbf{x}_d, \mathbf{B}\mathbf{R}^d(\mathbf{x}_d)) - \epsilon \end{aligned}$$

□

We now explain how to bound the error of S2D2 algorithm, assuming a δ -coarsening exists.

Theorem 2. *Let $\delta > 0$ and assume \hat{V} is a δ -coarsening. Then S2D2 outputs an ϵ -SSE for $\epsilon = 2AP\delta + 2\epsilon'$, where ϵ' is an upper bound on the error of the single-attacker single-defender oracle.*

Recall that our proposed S2D2 algorithm consists of 3 steps. In the coarsening step, the algorithm outputs, along with the coarsening \hat{V} , a parameter δ . By Theorem 1, the optimal strategy that respects the coarsening is an ϵ_1 -SSE, for $\epsilon_1 = 2AP\delta$.

In the second step, we solve the single-attacker single-defender game on each neighborhood. We can bound the utility of the attacker by $(1 - \lambda)u_{1,0}^{\hat{v},a} \leq u_{\lambda}^{\hat{v},a} \leq u_{1,0}^{\hat{v},a}$. The lower bound is reached by setting the attacker strategy as $s_a \in \arg \max_{s'_a} u^{\hat{v}}(\perp, s'_a)$ a greedy strategy, regardless of the defense strategy. The upper bound is reached by setting the defense strategy to be \perp . Similarly, the utility of the defender can be bounded by

$$u_{1,0}^{\hat{v},d} \leq u_{\lambda}^{\hat{v},d} \leq (1 - \lambda) \cdot u_{1,0}^{\hat{v},d}.$$

Indeed, the lower bound is when $\mathbf{x}_d = \perp$, and the upper bound is reached if we assume that a defender in the neighborhood protects all targets completely (and λ is small, so that a greedy strategy is approximately BR).

As a consequence, the error of the utility estimation from step 2 can be bounded by $\epsilon_2 = \lambda_{\max} P \max_{v \in V} R^a(v)$.

In the third step, the algorithm solves the multi-drone meta-game, using the approximated utility function from step 2 as an input. By Lemma 3, given the error for the second step, the third step outputs a $2\epsilon_2$ -SSE. The final solution is thus ensured to be an $(\epsilon_1 + \epsilon_3)$ -SSE. Nevertheless, to make sure the error is small, the error δ_2 must be small as well.

Whenever $\lambda_{\hat{v}} > \lambda_c$ for some cutoff λ_c , in order to get a meaningful bound for the error, we must approximate the utility more accurately. As suggested in Section 4.2, we should consider all strategies s_a for the attacker in \hat{v} , such that $u^{\hat{v}}(\perp, s_a) \geq (1 - \lambda_{\hat{v}})u_{1,0}^{\hat{v},a}$, which may consist of more than all greedy strategies. In particular, let $r_1 \geq r_2 \geq \dots \geq r_P > \delta$ be the top P rewarding nodes. Then the attacker strategy space consists of all paths that pass through enough of these nodes so that the overall utility when there is no defender is more than $(1 - \lambda_{\hat{v}})u_{1,0}^{\hat{v},a}$. Doing so will result with an exact solution, and will allow us to replace λ_{\max} with λ_c , as desired.

6. Experiments

Our experiments were aimed at assessing the efficacy of S2D2 by comparing runtime and defender utilities with a baseline. We first synthetically generated utilities for nodes in 80 world cities from all continents (except

Africa and Antarctica), including several capitals. Then, we used manual annotations for different facilities through a survey of 7 security and defense experts. Finally, we did a detailed case study that qualitatively assessed the defenses recommended for a single city.

All the experiments were run on an Intel(R) Core(TM) i9-10980XE CPU with 256 GB RAM.

Implementation of S2D2, baseline defense strategy, and the code for the experiments presented below, are all publicly available on Github <https://github.com/tonmoay/S2D2-Experiments>.

6.1. Setting

Dataset and Parameters. We created a dataset of 80 cities, ranging from a few thousands nodes up to a few hundreds of thousands of nodes. For each city, the street networks were sourced from the *OpenStreetMap* platform via the OSMnx library [48].

The number of neighborhoods was fixed to $|\hat{V}| = 8D$, that is, proportional to the number of defense drones. Taking a large constant will result in a graph that is mostly unprotected, and neighborhoods that are too small. Taking a small constant would mean that there are enough defense drones to cover all neighborhoods with probability 1, yet those neighborhoods will be too big to protect.

As the dataset lacked rewards/penalties for nodes, we assigned those parameters in two ways: (i) sampling them independently from a distribution (log-normal/Zipf); (ii) using security experts to manually annotate 6 cities. Defender penalties were then assigned by randomly perturbing the rewards. This maintains some degree of correlation while circumventing a zero-sum game scenario.

For the synthetic data, we assigned rewards to city nodes by sampling independently from a distribution over the $[0, \infty)$ interval. The reason is that (i) rewards should be non-negative and (ii) we expect the set of nodes with high rewards to be sparse. Otherwise, the game essentially becomes an evasion game where the goal is to catch the attacker as soon as possible. Specifically, we sampled from the log-normal distribution with $\mu = 0, \sigma = 4$, as well as a Zipf distribution with $s = 2$.

For the manually annotated data, we asked 7 senior defense and security officials from the US, EU, Asia, and the Middle East to rate the importance of different facilities in city neighborhoods. The 6 cities included three major U.S. cities, an Asian megacity with a population of over 20M people, and two

smaller cities in the Middle East. In all, the cities included 3 world capitals. We asked the experts to imagine a city that they knew well when filling out the survey without telling them which city to look at. We asked questions related to the following types of facilities: Local/Municipality Buildings (e.g., the office of the mayor or city administration), National Government Leadership Buildings (e.g., the White House in Washington DC or 10 Downing Street in London), National Government Operational Buildings (e.g., the office of a Ministry), Security Installation Buildings (e.g., Ministry of Defense or Europol Headquarters), Hospitals, Electricity/Natural Gas Plants, Sanitation and Water Plants, Industrial and Hazardous Materials Areas, Transportation Hubs (e.g., airports, train stations, etc.), Tourist Sites, Financial Districts, Shopping and Entertainment Areas, Sports Arenas, and High Density Areas. Each type of facility was to be ranked on a 1 to 5 scale with 1 meaning it was of very low importance and 5 meaning it was of critical importance. The median values obtained are summarized in Table 2. All the

Facility	Median Utility
National Government Leadership Buildings	5
Security Installation Buildings	5
Electricity/Natural Gas Plants	5
National Government Operational Buildings	4
Hospitals	4
Sanitation and Water Plants	4
Industrial and Hazardous Materials Areas	4
Transportation Hubs	4
High Density Areas	4
Local/Municipality Buildings	3
Tourist Sites	3
Financial Districts	3
Shopping and Entertainment Areas	3
Sports Arenas	3

Table 2: Median utilities assigned to various types of facilities by security experts.

experts agreed that security installations and major national government buildings would have top priority followed by utilities (e.g., power, water). The vast majority of the cities (e.g., residential areas) would have much lower

rewards. A specifically designed annotation interface (Figure 4) was used by the experts to draw rectangles and/or polygons and provide a utility value for each polygon, i.e., value of the region. To avoid risks to real cities, the figure has been intentionally blurred to show the overall use of the interface without making it possible to identify the specific city.

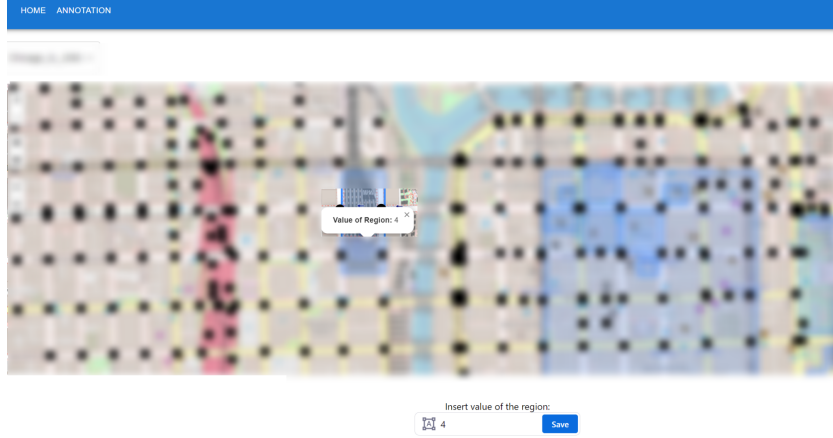


Figure 4: Annotation interface (intentionally blurred).

We fixed the number of defender and attacker drones to be $D = A = 4$, the battery capacity $B = 6$ and the payload $P = 4$. However, in some experiments, we also varied A , D , B , and P . The outcomes presented pertain to 100 iterations in all reported results.

Baseline. We compared S2D2’s runtime and expected utility with a greedy baseline (Algorithm 7) that employs a drone-swarm defense mechanism. The baseline allocates protection to each neighborhood with a drone, doing so proportionally to the cumulative absolute penalties of the top P attacker rewarding nodes. This is done by letting each defense drone sample its starting neighborhood independently from the distribution \hat{p}_d (Line 6).

Within each neighborhood, the baseline drone starts at a random node. The output of `move_towards_closest_node(\hat{v})` is a function that assigns a random start node from \hat{v} in each execution. It then follows a greedy next-step function, defined for each time step $1 \leq t \leq B$ as follows. The defender looks for the target $v \in \hat{V}$ that is closest to the attacker drone position, is of interest to the attacker (amongst the top P), is valuable to the defender (penalty $< -\delta$), and the defender can reach there before the attacker, and

Algorithm 7 Greedy Baseline

Require: Undirected graph $G = (V, E)$;
coarsening \hat{V} of G , and δ ;
numbers of attacker and defender drones $A, D \in \mathbb{N}$;
attacker drone’s payload $P \in \mathbb{N}$;
drone’s battery capacity $B \in \mathbb{N}$;
attacker rewards $R^a \in \mathbb{N}^{|V|}$;
defender penalties $P^d \in \mathbb{Z}_{<0}^{|V|}$.

Ensure: Defense mixed strategy $(\hat{p}_d, \hat{x}_d) \in \Delta(\mathcal{S}_{\hat{V}}^d)$.

- 1: $\text{weights_d} \leftarrow []$;
- 2: **for each** $\hat{v} \in \hat{V}$ **do**
- 3: $\hat{v}_a \leftarrow \hat{v}.\text{get_top_attack_rewards}(P, R^a, P^d)$; {Breaks ties in favor of defender}
- 4: $\text{weights_d}[\hat{v}] \leftarrow \hat{v}_a.\text{sum_abs_penalties}(P^d)$;
- 5: **end for**
- 6: $\hat{p}_d \leftarrow D * \text{weights_d} / \text{weights_d}.\text{sum}()$;
- 7: **for each** $\hat{v} \in \hat{f}_a$ **do**
- 8: $s_1^d \leftarrow \text{goto_random_node}(\hat{v})$;
- 9: **for each** $2 \leq t \leq B$ **do**
- 10: $s_t^d \leftarrow \text{move_towards_closest_node}(\hat{v}, \delta)$;
- 11: **end for**
- 12: $\hat{x}_d[\hat{v}] \leftarrow (s_1^d, \dots, s_t^d)$; {a pure strategy}
- 13: **end for**
- 14: **return** (\hat{p}_d, \hat{x}_d) ;

moves one step along the shortest respective path. If there are multiple attackers in sight (in \hat{v}), it picks the closest one to hunt down, breaking ties randomly. Importantly, the algorithm returns a strategy, so the output of `move_towards_closest_node` is actually a function, that determines the next step for the defender given the defense current position, attacker last observed position, updated penalties and rewards, and the neighborhood graph structure. This procedure is reiterated based on the attacker’s updated position. The above baseline is an adaptation of [28] to our setting. As in [28], each defender is paired with the closest observed attacker drone. In addition, chasing a drone is done by predicting its projectile. While [28] assumes a straight-line projectile, our heuristic takes into account the attacker rewards, and also rather than straight-line the defender takes the shortest path along the graph.

Notably, the defense strategy above can be implemented by using a defense drone swarm. As the experiments will demonstrate, such a defense strategy is more scalable in terms of run-time, but the expected defender utility is going to be smaller.

Attacker. Both S2D2 and baseline defenders were paired with the S2D2 attacker, as we are interested in handling a strong attacker approximating the best response.

6.2. Results

Figure 5 reports the results regarding runtime and defender utility (expressed as ratios between the baseline and S2D2).

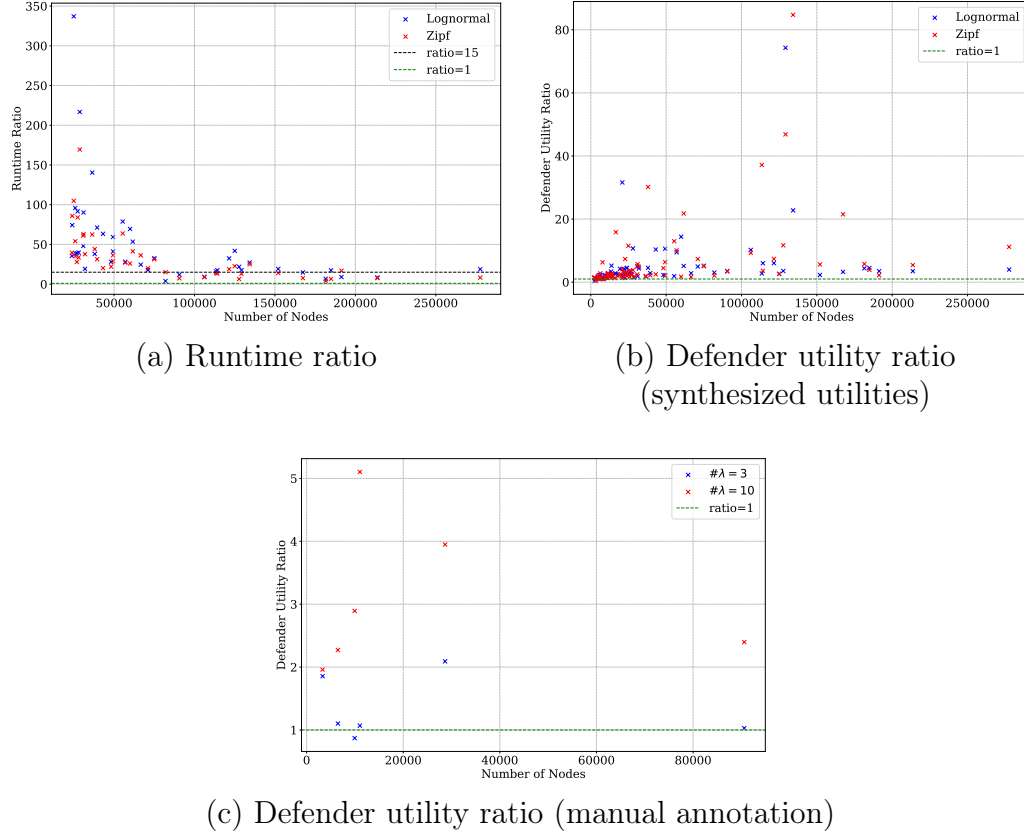


Figure 5: Comparison of S2D2 with the baseline.

Figure 5(a) shows that the runtime ratio between S2D2 and the baseline approaches a constant factor of about 15. Indeed, for small cities S2D2 run-

time is dominated by the multi-drone stage, but this cost becomes negligible as neighborhood size grows. Accordingly, S2D2 can be run within times that are reasonable even for the largest cities (3.5 hours with one CPU for the largest one). The more expensive runtime is amply rewarded by the fact that S2D2 decreases the expected defender loss and the attacker’s expected utility by an average of 2.84 times that of the baseline, as shown in Figure 5(b). Interestingly, the utility is less dependent on graph size and more on the structure of the graph and distribution of rewards.

As for manually annotated cities, Figure 5(c) shows that the baseline is better for one city. This could be due to the piece-wise linear approximation of single-drone neighborhood utility function, where we only used $\#\lambda = 3$ points $\lambda_i \in \{0, 0.5, 1\}$ (which performed good enough for the synthetic data). We therefore repeated the experiment with S2D2 using $\#\lambda = 10$ points. Indeed, S2D2 convincingly outperforms the baseline in all manually annotated cities.

Figure 6 reports results regarding the dependency of runtime and defender utility on the ratio between the number of attacker to defender drones (synthetic utilities, log-normal distribution). The number of attacker drones is $A = \lfloor D \times \text{ADR} \rfloor$ where ADR is the *attacker-defender ratio*. S2D2 yields

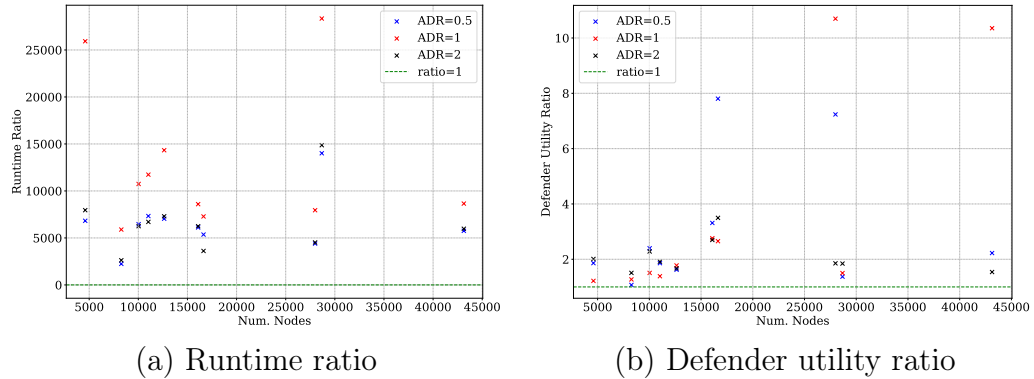


Figure 6: Comparison of S2D2 with the baseline, with $\text{ADR} \in \{0.5, 1, 2\}$.

a higher defender utility at the expense of increased run-time compared to the baseline. The runtime ratio is relatively big since the cities are relatively small (up to 45,000 nodes), and so the multi-drone phase of the solution takes a significant portion of time. However, we do not observe a strong correlation between ADR and runtime, so handling more attackers does not incur a higher computational cost. We also do not observe a correlation between

ADR and defender utility ratio. This is probably because both the baseline and the S2D2 defender utilities are similarly affected from the change of ADR.

Figure 7 shows how defender utility varied when we perturbed manually annotated utilities. The latter were perturbed by $\pm 10\%$, i.e. they were fixed to $\pm 10\%$ of the true value. In the case of one city, the defender utility ratio

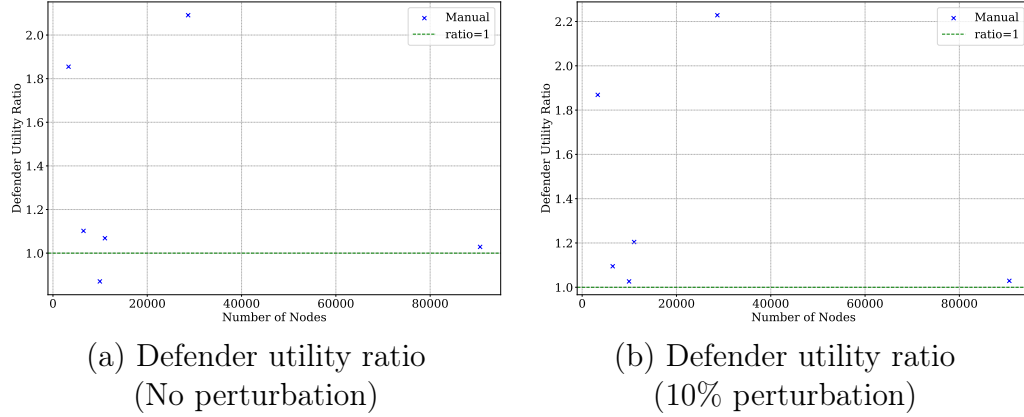


Figure 7: Defender utility ratios when the defender’s estimate of the attacker’s utility is off by 0 (a) or by $\pm 10\%$ (b).

is less than 1, so the baseline outperforms S2D2. This is explained by the coarse approximation of the single-drone utility function as a piece-wise linear function, which can be improved if necessary at the expense of runtime. In all the other 5 cases, the utility ratio ranges from around 1.05 to 2.15. In two cases, S2D2 yields almost double the utility of the baseline, and in 3 other cases, it outperforms the baseline by a smaller margin. It should be noted that, if we average the defender’s utility ratio across the 6 cities, the average is 1.34. Thus, on average, S2D2 provides 34% improvement over the baseline algorithm in terms of the utility to the defender. Even though that comes at the cost of an increased runtime, most cities would be happy to make this tradeoff: saving 34% more of the utility of the city (lives and property damage).

Figure 7(b) shows that in every single case, the defender utility ratio is over 1, so the S2D2 algorithm outperforms the baseline. Even when the defender’s assumption about the attacker’s utility is slightly incorrect, the average improvement over the baseline is 41%. Most cities would prefer to save an additional 41% of the utility of the city compared to saving some

runtime.

Finally, Figure 8 reports the results we obtained when we fixed $A = 5$ attacker drones and selected 5 cities based on a balanced size distribution, from 2,283 nodes to 125,013 nodes (synthetic utilities, Zipf distribution). Battery capacity and payloads were fixed to $B = 6$ and $P = 4$.

Figure 8(a) reports the impact on defender utility of varying the number of defender drones from 2 to 10 against 5 attacker drones.

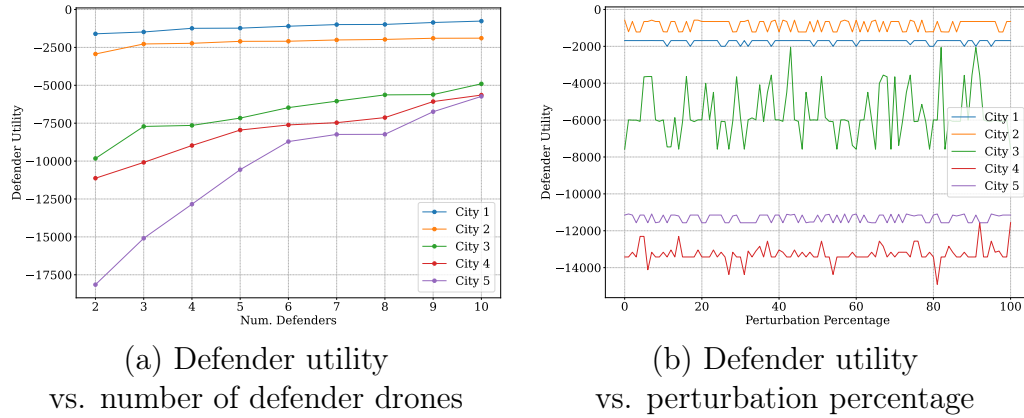
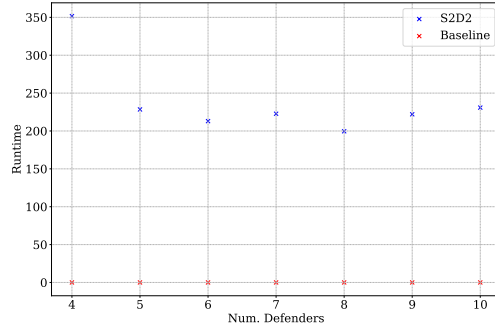


Figure 8: Defender utilities with 5 attacker drones.

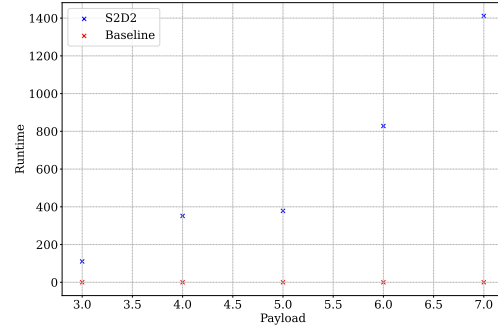
As expected, increasing the number of defenders consistently enhances defender utility. Figure 8(b) shows the impact of perturbing utilities from 0% to 100% in 1% increments, with penalties unchanged, when we fixed $D = 5$ defender drones. We sampled a defender strategy from the mixed strategy set, perturbed the rewards, and then evaluated the attacker’s response and the utilities of both sides across all perturbation levels. The results show that, despite the perturbations, the defender strategy remains effective, demonstrating robustness against the attacker’s adaptations. Although some noise was observed, the overall utility trends were stable.

Case Study of One Major City. We now describe a detailed case study of one major city from the Americas with a population of over 2M. The city had 28,671 nodes and was manually annotated. Unless stated otherwise, the experimental parameters described previously were used in this case study.

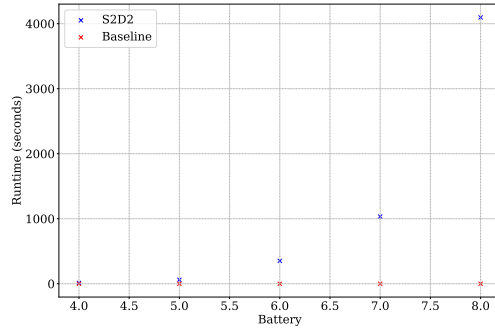
Figure 9 shows the runtimes and utility ratios we obtained when varying the number of defender drones, the payload, and the battery capacity. Not surprisingly, our very simple baseline is significantly faster than the S2D2



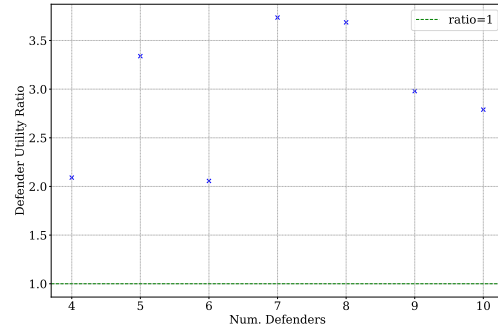
Runtime (s)
vs. number of defender drones



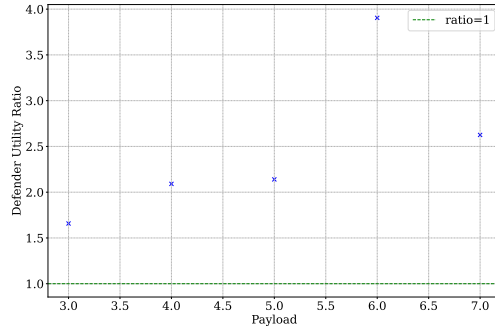
Runtime (s)
vs. payload



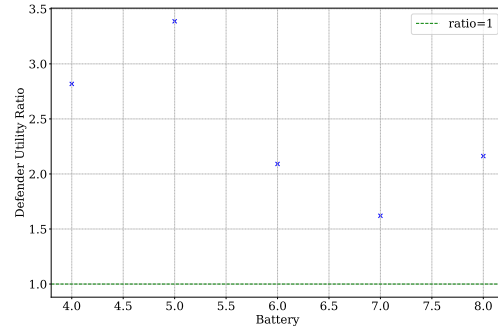
Runtime (s)
vs. battery capacity



Defender utility ratio
vs. number of defender drones



Defender utility ratio
vs. payload



Defender utility ratio
vs. battery capacity

Figure 9: Results of a case study for one major city.

algorithm. Nevertheless, Figure 9(a) shows that S2D2 takes a reasonable amount of runtime (about 3–4 minutes) when the number of defensive drones is varied from 4 to 12 — moreover, the time seems more or less constant. When the payload is varied (see Figure 9(b)), we see that S2D2’s runtime increases linearly — but still stays in a matter of minutes (13 minutes when payload is 7). When the battery capacity is varied (see Figure 9(c)), S2D2’s runtime increases exponentially — when battery capacity is 7 or less, it takes about 16-17 minutes, but once it goes up to 8, the runtime increases significantly to about 66 minutes. This is not surprisingly because the number of possible paths grows exponentially with the battery capacity — as battery capacity increases, the attacker can travel further.

Figure 9(d) shows that when the number of defensive drones is varied from 4 to 12, S2D2 delivers 2 to 4 times the utility provided by the baseline. However, there is no consistent increase in this ratio. While both S2D2 and baseline defender utilities are expected to increase when the number of defenders goes up, we do not see a reason for the ratio to increase nor decrease. Figure 9(e) shows that when the payload is varied from 3 to 7, S2D2 delivers 1.7 to 2.7 times the utility provided by the baseline. Again, there is some fluctuation in the ratio. Finally, Figure 9(f) shows that when the battery capacity of the drones is varied from 4 to 8, S2D2 delivers 1.7 to 4 times the utility provided by the baseline. All of these numbers suggest that S2D2 is a definitive improvement over the baseline as far as defender utility is concerned — this comes at the cost of runtime, even though the latter is still reasonable.

Figure 10 shows what percentage of the nodes were destroyed for each of the 5 utility values, with $B = 6$, $P = 4$, $ADR = 1$, and $D = 4$. Specifically,

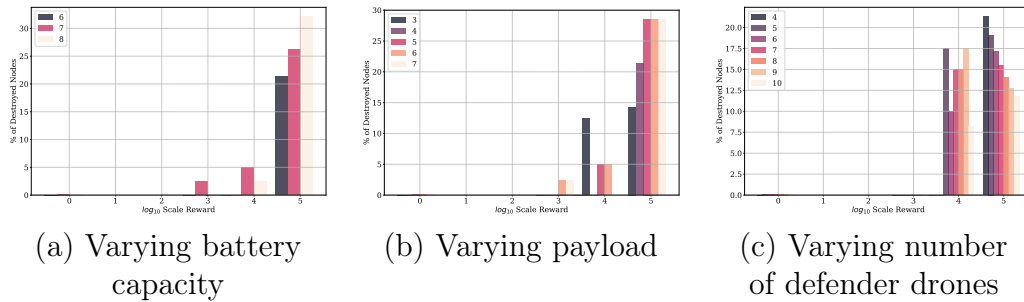


Figure 10: Percentage of destroyed nodes for each utility value.

Figure 10(a) looks at what happens when varying battery capacity. We

observe that as battery capacity increases, the percentage of destroyed nodes having utility 5 nodes increases — this is probably because the attacker has enough battery to attack multiple 5-ranked nodes with a single drone. Figure 10(b) shows what happens when we vary the drones’ payload. Again, we see that as payload increases, the percentage of destroyed nodes with utility 5 increases — as the attacker has more payload, it may prefer to attack more of those nodes. The saturation at payload of 5 is probably because there are no more 5-ranked nodes that a single drone can reach with its battery capacity. Figure 10(c) shows the situation when we vary the number of defender drones. As expected, we see that as the number increases, the percentage of nodes with utility 5 being destroyed decreases. This demonstrates that S2D2 utilizes each added defender drone to cover more of the high ranked nodes. We see a similar correlation for utility 4 nodes, but it is weaker. This can be explained as a side effect where the attacker, taking into consideration that the utility 5 nodes are more protected, now prefers to strike against utility 4 nodes where it is less likely to get caught.

7. Discussion

Coarsening. The first step of S2D2 is to coarsen the input graph. Specifically, the neighborhoods should be densely connected inside and relatively isolated from one another. However, in urban environments, target locations may not exhibit this kind of clean structure. First, we emphasize that the output coarsening is not restricted to respect jurisdictional divisions or any type of man-made partition of the city. Second, our intuition was that important areas of interest typically come in clusters, e.g., a dense neighborhood, an industrial area, or a group of government buildings, and so, we expect a good coarsening to exist. For instance, Wall Street in New York City is densely clustered. Likewise, the major government buildings in Washington DC are also densely clustered. We then tested our intuition via two approaches. First, we analyzed real-world large-scale cities, annotated according to the knowledge of security experts. Second, we synthetically annotated nodes with respect to two distributions. In both cases, we observed that the output coarsening was not always ideal. Nevertheless, it did effectively separate the city into neighborhoods and in most of the cities, our experimental results were reasonable even when the coarsening was not ideal. From both theoretical and experimental perspectives, we identify the coarsening approach to be promising, and believe that improvements in coarsening have the potential

to significantly improve defense strategies.

Graph Structure. Aerial drones need not be subject to any ground-based constraints of the underlying city, and can seamlessly reach any location through direct aerial traversal. However, we note that this does not suggest that all targets are fully connected. Cities can be quite large, and it takes time to go from one point to another. Also, the defender is also using drones and is therefore moving at a comparable speed as the attacker.¹³ We also emphasize that our choice of modeling a city as a general graph, enables applicability beyond drone swarm defense. Land-based vehicles are more subject to the topographical structure of the city, and so our approach can be even more effective in this setting. Also, not using a perfect grid allows our model to capture various obstacles, e.g., a skyscraper cluster, or an area that is protected with GPS jamming devices.

Attacker adaptivity. While our model allows the defender to apply an adaptive strategy that changes as a function of the state, the attacker is modeled to be static. This simplification seems to contrast with some pursuit-evasion games of similar type, where both players are adaptive. In real-world scenarios, attackers are likely to observe the state, at least partially, and adapt their strategies in accordance. This may raise concerns about whether the model’s validity is compromised by this simplification. To this end, we emphasize that while acquiring aerial drones is relatively easy, tracking drones is still imperfect, and as we cover in the related work, is an independent area of interest. For instance, [23] analyzes 8 months of drone flights over The Hague, but it is clear that some drone flights may have been missed due to imperfections in tracking. Therefore, we assume that both the attacker and the defender are not aware of the locations of the opposing drones at the beginning. The asymmetry stems from the assumption that after significant damage is caused by an attacker drone, it can be tracked. We believe this assumption to be more realistic compared to prior works, at least in the context of drones that can be small, silent, fast, and may not communicate.

In the case the attacker is aware of the defensive drone’s location, it could utilize this additional information to evade the defender more effectively, and

¹³We believe that in cases when the drones move so fast that the targets are fully connected, the problem should be modeled as a non-sequential SSG where the attacker has $A \times P$ resources, and the defender has D resources. For that matter, one may use [20] to get an efficient exact solution.

therefore is expected to cause more damage. This framework could be an interesting future work. However, if there is concern that the defender utility is significantly smaller, a conclusion could be that more research and effort should be put into preventing the attacker from gaining this information to begin with.

Knowledge of \mathbf{x}_d . In Stackelberg games, the attacker knows the defender’s committed mixed strategy and best responds to it. This is often justified by the claim that the attacker can survey the defender’s strategies before launching an attack. In this paper, we follow this approach. However, this assumption may seem less feasible in the context of state-dependent strategies, as the defender’s strategy space is overwhelmingly large. Without sufficient real attacks, many states may not even occur, making it harder for an attacker to accurately observe and infer the defender’s strategies. While we acknowledge that surveillance in our case is probably not sufficient for the attacker to unravel the defender’s mixed strategy, assuming a stronger attacker may result in a more robust system. The alternative approach, of attempting to model the limitation of an attacker, come at a high risk. If the defense relies on an attacker with certain capabilities which do not hold in reality, it can lead to severe consequences. In contrast, overestimating the attacker capabilities could result in a less effective defense overall, but the caused damage will not exceed model expectations. Indeed, we point out that at least the division of the defender into neighborhoods, and the allocation strategy into neighborhoods, can be observed. The single-drone defensive strategy within a neighborhood is then already more compact. If S2D2 is the chosen implementation, the attacker may be able to compute it by itself. Additionally, the attacker can gain information in other manners. For instance, it may conduct a cyber attack or procure a captured defense drone to get the defender’s code. It may also get human intelligence from people who work at the companies that manufactures the drones, or from an employee who developed the defense mechanism for the drones. With all of these considerations in mind, we opted to assume full knowledge of the adversary with regards to the mixed defender strategy.

Zero-sum games. Another alternative approach would be to assume the game is zero sum. In this case, Nash equilibrium is sufficient, and one need not assume knowledge of the defender’s mixed strategy. On the other hand, this reduces the model’s generality compared to our proposed general-sum

setup. This is crucial, as in many cases, the objectives of the attacker and defender may not be aligned. For instance, the attacker may care more about damaging infrastructure, while the defender may care more about civilian casualties, or vice versa. In such scenarios, either a zero-sum based model will not be deployed at all, or alternatively, one would approximate it to be such. Similarly to the previous point, this may result in an unrealistic model of the adversary, and in turn could cause damage that exceeds model expectations.

Scalability. Since the defender’s strategy space appears extremely large, questions about scalability may arise. While we claim the runtime to be polynomial with $|\mathcal{S}^d|$, the strategy space itself is exponential with respect to the input problem size. In this paper, we took a three step approach to address the problem at hand. (i) First, we provided theoretical analysis and explored a theoretically proven algorithm which is inevitably impractical. (ii) After identifying the bottlenecks, we introduced heuristics to make the algorithm practical. These include narrowing down the strategy space $|\mathcal{S}^d|$ of the defender, as well as relaxing the conditions for our coarsening. (iii) Finally, we extensively tested the performance of the heuristic algorithm. We acknowledge that S2D2 can be improved both in scale and performance in future works. However, we believe that following the blueprint outlined above is a vital cornerstone. Therefore, down the line, our theoretical results may turn out to be of greater importance than any of the three building blocks of S2D2(coarsening, single drone sequential sub-games, multi-drone meta-game) as instantiated in this work.

8. Conclusions

Multi-drone strikes are increasingly likely to be used to target cities. The threat actors using such techniques will include both nation state actors and terrorist groups.

In this paper, we have developed a realistic model to defend cities against multi-drone attacks via 4 contributions: (i) We extend sequential SSGs involving multiple attack/defense drones with payload and battery constraints. (ii) We propose the Sequential Stackelberg Drone Defense (S2D2) paradigm to solve the problem of minimizing damage to the city by the attacker and show detailed theoretical results that show that under some conditions related to a novel concept called δ -coarsening, S2D2 provides a strong approximation algorithm for a computationally difficult problem. We prove that

S2D2 outputs an approximate SSE and an upper bound on the error, under such conditions. (iii) Experiments on a dataset of 80 famous cities compare S2D2 with a heuristic swarm-defense algorithm, demonstrating a trade-off between runtime and defender utility. Importantly, the experiments show that even when the δ -coarsening conditions do not hold, S2D2 still works effectively. (iv) Our experiments were run with real data on 80 cities using randomly assigned utilities. But in addition, we assigned utilities to locations in 6 cities using general guidelines provided by security experts from the US, EU, Asia, and Middle East. Our experiments also included these 6 cities. Finally, we did a detailed case study of one large North American city with expert input. To the best of our knowledge, past work on game-theoretic defenses of cities against multi-drone attacks have not done that.

References

- [1] M. R. Brust, G. Danoy, P. Bouvry, D. Gashi, H. Pathak, M. P. Gonçalves, Defending against intrusion of malicious UAVs with networked uav defense swarms, in: 2017 IEEE 42nd conference on local computer networks workshops (LCN workshops), IEEE, 2017, pp. 103–111.
- [2] D. He, G. Yang, H. Li, S. Chan, Y. Cheng, N. Guizani, An effective countermeasure against uav swarm attack, *IEEE Network* 35 (1) (2020) 380–385.
- [3] Y. N. Jurn, S. A. Mahmood, J. A. Aldhaibani, Anti-drone system based different technologies: architecture, threats and challenges, in: 2021 11th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), IEEE, 2021, pp. 114–119.
- [4] M. J. Guitton, Fighting the locusts: Implementing military countermeasures against drones and drone swarms, *Scandinavian Journal of Military Studies* 4 (1) (2021) 26–36.
- [5] M. R. Brust, G. Danoy, D. H. Stolfi, P. Bouvry, Swarm-based counter uav defense system, *Discover Internet of Things* 1 (2021) 1–19.
- [6] W. Chen, X. Meng, J. Liu, H. Guo, B. Mao, Countering large-scale drone swarm attack by efficient splitting, *IEEE Transactions on Vehicular Technology* 71 (9) (2022) 9967–9979.

- [7] V. U. Castrillo, A. Manco, D. Pascarella, G. Gigante, A review of counter-uas technologies for cooperative defensive teams of drones, *Drones* 6 (3) (2022) 65.
- [8] N. Li, Z. Su, H. Ling, M. Karatas, Y. Zheng, Optimization of air defense system deployment against reconnaissance drone swarms, *Complex System Modeling and Simulation* 3 (2) (2023) 102–117.
- [9] D. Kar, T. H. Nguyen, F. Fang, M. Brown, A. Sinha, M. Tambe, A. X. Jiang, Trends and applications in stackelberg security games, *Handbook of dynamic game theory* (2017) 1–47.
- [10] A. Sinha, F. Fang, B. An, C. Kiekintveld, M. Tambe, Stackelberg security games: Looking beyond a decade of success, in: *IJCAI Conference*, 2018.
- [11] P. Paruchuri, M. Tambe, F. Ordóñez, S. Kraus, Security in multiagent systems by policy randomization, in: *AAMAS Conference*, 2006.
- [12] J. Pita, R. John, R. Maheswaran, M. Tambe, S. Kraus, A robust approach to addressing human adversaries in security games, in: *ECAI Conference*, IOS Press, 2012.
- [13] J. Pita, M. Jain, M. Tambe, F. Ordóñez, S. Kraus, Robust solutions to stackelberg games: Addressing bounded rationality and limited observations in human cognition, *Artif. Intell.* 174 (15) (2010) 1142–1171.
- [14] D. Mutzari, Y. Aumann, S. Kraus, Robust solutions for multi-defender Stackelberg security games, *IJCAI* (2022).
- [15] C. Kiekintveld, M. Tambe, J. Marecki, Robust bayesian methods for stackelberg security games, in: *AAMAS Conference*, 2010.
- [16] D. Korzhyk, V. Conitzer, R. Parr, Complexity of computing optimal stackelberg strategies in security resource allocation games, in: *AAAI Conference*, 2010.
- [17] J. Gan, E. Elkind, M. J. Wooldridge, Stackelberg security games with multiple uncoordinated defenders, in: *AAMAS Conference*, 2018.
- [18] D. Mutzari, J. Gan, S. Kraus, Coalition formation in multi-defender security games, in: *AAAI Conference*, 2021.

- [19] C. U. Solis, A. S. Poznyak, J. B. Clempner, Solving stackelberg security games for multiple defenders and multiple attackers, in: Stony Brook International Conference on Game Theory, 2015.
- [20] D. Korzhyk, V. Conitzer, R. Parr, Security games with multiple attacker resources, in: IJCAI Conference, 2011.
- [21] D. Kar, F. Fang, F. Delle Fave, N. Sintov, A. Sinha, A. Galstyan, B. An, M. Tambe, Learning bounded rationality models of the adversary in repeated stackelberg security games, Retrieved from Nanyang Technological University: http://www3.ntu.edu.sg/home/boan/papers/ALA15_Debarun.pdf (2015).
- [22] F.-L. Chiper, A. Martian, C. Vladeanu, I. Marghescu, R. Craciunescu, O. Fratu, Drone detection and defense systems: Survey and a software-defined radio-based solution, *Sensors* 22 (4) (2022) 1453.
- [23] T. Deb, S. de Laaf, V. La Gatta, O. Lemmens, R. Lindelauf, M. van Meerten, H. Meerveld, A. Neeleman, M. Postiglione, V. Subrahmanian, A drone early warning system (dews) for predicting threatening trajectories, *IEEE Intelligent Systems* (2025).
- [24] L. Han, W. Song, T. Yang, Z. Tian, X. Yu, X. An, Cooperative decisions of a multi-agent system for the target-pursuit problem in manned–unmanned environment, *Electronics* 12 (17) (2023) 3630.
- [25] C. De Souza, R. Newbury, A. Cosgun, P. Castillo, B. Vidolov, D. Kulić, Decentralized multi-agent pursuit using deep reinforcement learning, *IEEE Robotics and Automation Letters* 6 (3) (2021) 4552–4559.
- [26] A. Manoharan, P. Thakur, A. K. Singh, Multi-agent target defense game with learned defender to attacker assignment, in: 2023 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, 2023, pp. 297–304.
- [27] L. S. Pontryagin, On the theory of differential games, *Russian Mathematical Surveys* 21 (4) (1966) 193.
- [28] M. Chen, Z. Zhou, C. J. Tomlin, Multiplayer reach-avoid games via pairwise outcomes, *IEEE Transactions on Automatic Control* 62 (3) (2016) 1451–1457.

- [29] A. Bonato, The game of cops and robbers on graphs, American Mathematical Soc., 2011.
- [30] A. Kehagias, D. Mitsche, P. Prałat, Cops and invisible robbers: The cost of drunkenness, *Theoretical Computer Science* 481 (2013) 100–120.
- [31] A. Kehagias, D. Mitsche, P. Prałat, The role of visibility in pursuit/evasion games, *Robotics* 3 (4) (2014) 371–399.
- [32] D. Dereniowski, D. Dyer, R. M. Tifembach, B. Yang, Zero-visibility cops and robber and the pathwidth of a graph, *Journal of Combinatorial Optimization* 29 (2015) 541–564.
- [33] T. H. Nguyen, A. Butler, H. Xu, Tackling imitative attacker deception in repeated bayesian stackelberg security games, in: *ECAI Conference*, 2020.
- [34] T. H. Nguyen, A. Yadav, B. Bosanský, Y. Liang, Tackling sequential attacks in security games, in: *Decision and Game Theory for Security Conference*, 2019.
- [35] T. Petr, B. Bosansky, T. H. Nguyen, Using one-sided partially observable stochastic games for solving zero-sum security games with sequential attacks, in: *Decision and Game Theory for Security: 11th International Conference, GameSec 2020, College Park, MD, USA, October 28–30, 2020, Proceedings 11*, Springer, 2020, pp. 385–404.
- [36] D. Korzhuk, Z. Yin, C. Kiekintveld, V. Conitzer, M. Tambe, Stackelberg vs. Nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness, *Journal of Artificial Intelligence Research* 41 (2011) 297–327.
- [37] M. Breton, A. Alj, A. Haurie, Sequential stackelberg equilibria in two-person games, *Journal of Optimization Theory and Applications* 59 (1988) 71–97.
- [38] B. Bosanský, J. Cermak, Sequence-form algorithm for computing stackelberg equilibria in extensive-form games, in: *AAAI Conference*, 2015.
- [39] J. Cermak, B. Bosanský, K. Durkota, V. Lisý, C. Kiekintveld, Using correlated strategies for computing stackelberg equilibria in extensive-form games, in: *AAAI Conference*, 2016.

- [40] J. Cerný, B. Bosanský, C. Kiekintveld, Incremental strategy generation for stackelberg equilibria in extensive-form games, in: ACM Conference on Economics and Computation, 2018.
- [41] J. Karwowski, J. Mandziuk, A new approach to security games, in: ICAISC Conference, 2015.
- [42] D. Vasal, Sequential decomposition of stochastic Stackelberg games, in: 2022 American Control Conference (ACC), IEEE, 2022, pp. 1266–1271.
- [43] G. Leitmann, On generalized stackelberg strategies, *Journal of optimization theory and applications* 26 (4) (1978) 637–643.
- [44] M. Jünger, G. Reinelt, G. Rinaldi, The traveling salesman problem, *Handbooks in operations research and management science* 7 (1995) 225–330.
- [45] K. Kerdprasop, N. Kerdprasop, P. Sattayatham, Weighted k-means for density-biased clustering, in: International conference on data warehousing and knowledge discovery, Springer, 2005, pp. 488–497.
- [46] V. D. Angelis, Minimization of a separable function subject to linear constraints, Princeton University Press, 1971, pp. 503–510.
- [47] M. Oral, O. Kettani, A linearization procedure for quadratic and cubic mixed-integer problems, *Operations Research* 40 (1-supplement-1) (1992) S109–S116.
- [48] G. Boeing, Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks, *Computers, Environment and Urban Systems* 65 (2017) 126–139.