AD work

first designs for process model + game theoretic model

Ostap S

May 15, 2025

Contents

1 Introduction

This doc contains some work from May 15 where I play around with one component of a possible process model and one type of game theoretic model of AD placement.

The process model is of the movement of interceptor groups, from the time they receive information on a drone heading their way to launching the drone.

The game-theoretic model explores high-level AD placement, given a defender with limited resources, and an attacker which learns over time.

Code for everything here is in a github repo: *https://github.com/ostapstefak99/AD_robota* (private).

2 Game Theoretic Model

Two cases here.

First is 'Simple Model'. Its the simplest case I could think of that retains the structure of the problem. I think its useful because you clearly see the min-max structure of the objective, plus the reason for randomization.

The second is a slightly more complex formulation, since we increase the number of targets to be defended + number of interceptors to defend them. It cannot be solved in the algebraic way we used for the simple model, and so uses a linear program. I (TODO) simulate the performance of this model using a simple model of an attacker who is learning over time.

2.1 Model class in words + justification

The model approach here is a security game. Security games are a type of Stackelberg game (leader-follower structure) where what is good for the attacker is bad for the defender.

This type of game also usually has these features:

1. Defender is capacity constrained; cannot fully defend everything
2. Attacker learns over time

In our AD case, we face just this problem structure. We are limited by the number of interceptor teams and interceptors at our disposal. Also the attacker is learning about interceptor placement, and will adapt behavior over time if there are any patterns.

A third aspect of the situation is that (at least if we are solving this problem at the level of the entire country), we also have a large number of targets to protect, each of which incurs a *different* cost if hit.

Together, these conditions lead us to a solution approach involving randomization.

2.2 Literature

There's not that much publicly accessible stuff about AD modelling in particular. Luckily, security games have a lot of the same structure. So we draw on this literature. Esp Milind Tambe's book 'Security and Game Theory (2012)'. This is not just an academic approach to security. There have been a bunch of super successful real life applications in the past decade, from maritime protection to airport protection (both USA) to assignment of patrols to street intersections (Mumbai, India).

2.3 Objective function

The objective is to minimize cost of the targets getting hit.

2.4 Simple model - theory

This is the simplest non-trivial Stackelberg security game: one defender w one interceptor, one attacker with one missile, two targets of different values.

Setup.

- Targets 1 and 2 have values v_1, v_2. If a target is destroyed, cost to the defender is value of target.
- Defender commits to defense of target 1 with probability $p \in [0, 1]$ and so defends target 2 with probability $1 - p$.
- Attacker observes p, then chooses one target to strike.

If a defended target is attacked, missile is perfectly intercepted (no damage). If an undefended target is attacked, missile succeeds with probability 1. Payoffs zero-sum: attacker's gain = defender's loss.

1. Attacker's best response. Given defender's p:

$$A_1(p) \;=\; (1 - p)\, v_1 \quad \text{if attacker strikes target 1,}$$

$$A_2(p) \;=\; p\, v_2 \quad \text{if attacker strikes target 2.}$$

The attacker picks whichever of $A_1(p)$ or $A_2(p)$ is larger.

2. Defender's problem. Defender wants to choose p to

$$\boxed{\min_{0 \le p \le 1} \; \max\bigl\{(1 - p)\, v_1,\; p\, v_2\bigr\}}$$

3. Equilibrium mixing probability. At optimum the two inside terms are equal:

$$(1 - p^*)\, v_1 \;=\; p^*\, v_2.$$

Solving:

$$v_1 - v_1\, p^* = v_2\, p^* \quad \Longrightarrow \quad v_1 = p^*(v_1 + v_2) \quad \Longrightarrow \quad p^* = \frac{v_1}{v_1 + v_2}.$$

Hence the defender randomizes by defending target 1 with probability

$$p^* = \frac{v_1}{v_1 + v_2}, \quad 1 - p^* = \frac{v_2}{v_1 + v_2}.$$

4. Game value. Plug p^* back into, say, $A_1(p)$:

$$A_1(p^*) = (1 - p^*)\, v_1 = \frac{v_2}{v_1 + v_2}\, v_1 = \frac{v_1\, v_2}{v_1 + v_2}.$$

By symmetry this equals $A_2(p^*)$, so the attacker's expected payoff (and defender's expected loss) is

$$L^* = \frac{v_1\, v_2}{v_1 + v_2}.$$

Intuition.

- If $v_1 = v_2$, then $p^* = 0.5$. You flip a fair coin, and $L^* = v_1/2$.
- If $v_1 \gg v_2$, then $p^* \to 1$: almost always defend high-value target. Necessary to accept loss of less valuable one.

In any case, the mixing weights balance so that each target, when attacked, yields the same expected damage.

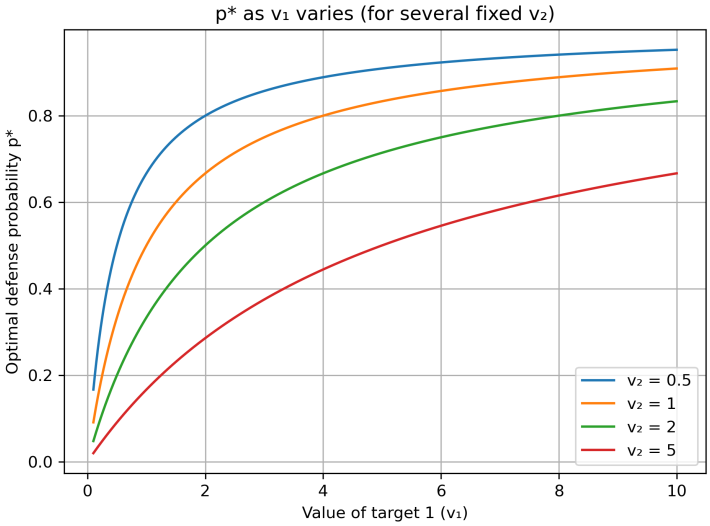

Figure 1: p* over range of v_1, v_2 values

2.5 More complex model

Here we make the model more complex by introducing more targets and more interceptors. But still just one attacking missile for now.

We need a linear program to solve this now.

2.5.1 LP

- $N = \binom{k}{r}$ – count of possible interceptor placements, labelled C_i.
- M – number of ways attacker can attack. In this case, since still one missile, its just the number of targets.
- D_{ij} – damage if attacker shoots at target j, and faces interceptor configuration C_i.
- Variables
  - x_i: probability of choosing placement C_i
  - U: cap on worst-case expected damage

Solve:

$$\boxed{\begin{array}{ll} \text{minimize} & U \\ \text{subject to} & \displaystyle\sum_{i=1}^{N} D_{ij}\, x_i \;\le\; U, \qquad j = 1, \ldots, M \\ & \displaystyle\sum_{i=1}^{N} x_i = 1 \\ & x_i \ge 0 \quad \text{for each } i. \end{array}}$$

Meaning: pick a probability mix $\{x_i\}$ so every attacker choice deals at most U damage, then push U as low as possible. Result gives a randomized interceptor positioning plan and a worst-case damage number to expect.

2.5.2 Downsides of this LP formulation

The main downside is the exploding number of terms in the summation, due to the combinatorial quantity. There are however ways to solve these faster written up in the literature. I have not looked into it very deeply, but I know they exist :)

2.5.3 Attacker model - for future simulations

Assume attacker treats each target t_i as an “arm” in an adversarial multi-armed bandit, using the EXP3 bandit algorithm to balance exploration and exploitation without assuming a fixed payoff distribution. At each round it samples a target according to a probability vector $\mathbf{p}$, observes the “reward” (successful hit or failure), and then updates each arm’s weight via an exponential rule

$$w_i \leftarrow w_i \, \exp\Bigl(\eta \, \hat{r}_i\Bigr),$$

where $\hat{r}_i$ is the importance-weighted reward estimate and η the learning rate. This ensures even poorly performing targets continue to be explored with nonzero probability, preventing the defender from exploiting predictable gaps.

I think this is a reasonable model of a ‘learning’ attacker to use in future simulations.

3 Process Model

There are multiple elements to a good process model of this situation. I'm only looking at one of them for now.

3.1 Interceptor Group Movement

An important part of the overall process model is the behavior of the interceptor teams on the ground. The assumption is we must station there somewhere, and they are limited by that starting point in terms of where they can travel to. They also only have a finite time in which to travel between getting information on a target approaching, and the time by which they must fire the interceptor.

To model this we can use data on the actual road network available from OpenStreetMap (OSM). We can treat this as a graph along which the interceptor group can travel.

3.1.1 Road-network reachability model

We want to know how far an interceptor unit can go from a start point s in T minutes, if it only travels along roads (up to speed limits) plus a final 200 m off-road hop. To do this, we:

1. **Build graph**: Download a driving network $G = (V, E)$ around s via osm (osmnx). Each edge e has length ℓ_e and speed limit v_e. Project G to a metric crs so units are in meters.

2. **Find reachable subgraph**: assign each edge a travel-time weight $\tau_e \approx \ell_e / v_e$, run Dijkstra from s to compute

$$d(s, v) = \min_{\pi :\ s \to v} \sum_{e \in \pi} \tau_e,$$

then keep $V_R = \{v : d(s, v) \leq T\}$, inducing subgraph $G_R = G[V_R]$.

3. **Add off-road buffer**: let

$$U = \Big(\bigcup_{v \in V_R} \{v\} \Big) \ \cup \ \Big(\bigcup_{e \in E_R} e \Big)$$

be the union of all reachable node-points and edge-lines. then form

$$A_R \ = \ U \ \oplus \ B(0, 200),$$

where $\oplus$ denotes the minkowski sum (a 200 m disk around every point of U). The resulting polygon A_R is the continuous area the interceptor can access within T minutes under the road-only+last-mile assumption.

3.1.2 Visual demo

Here we choose the start coordinate as the village of Pikovets (Cherkasy), and run the model described above. The chosen params here are 10 mins allowed of movement from the time of warning until launch required, and a 200m off-roading buffer. Also we are assuming speed limits are being followed...

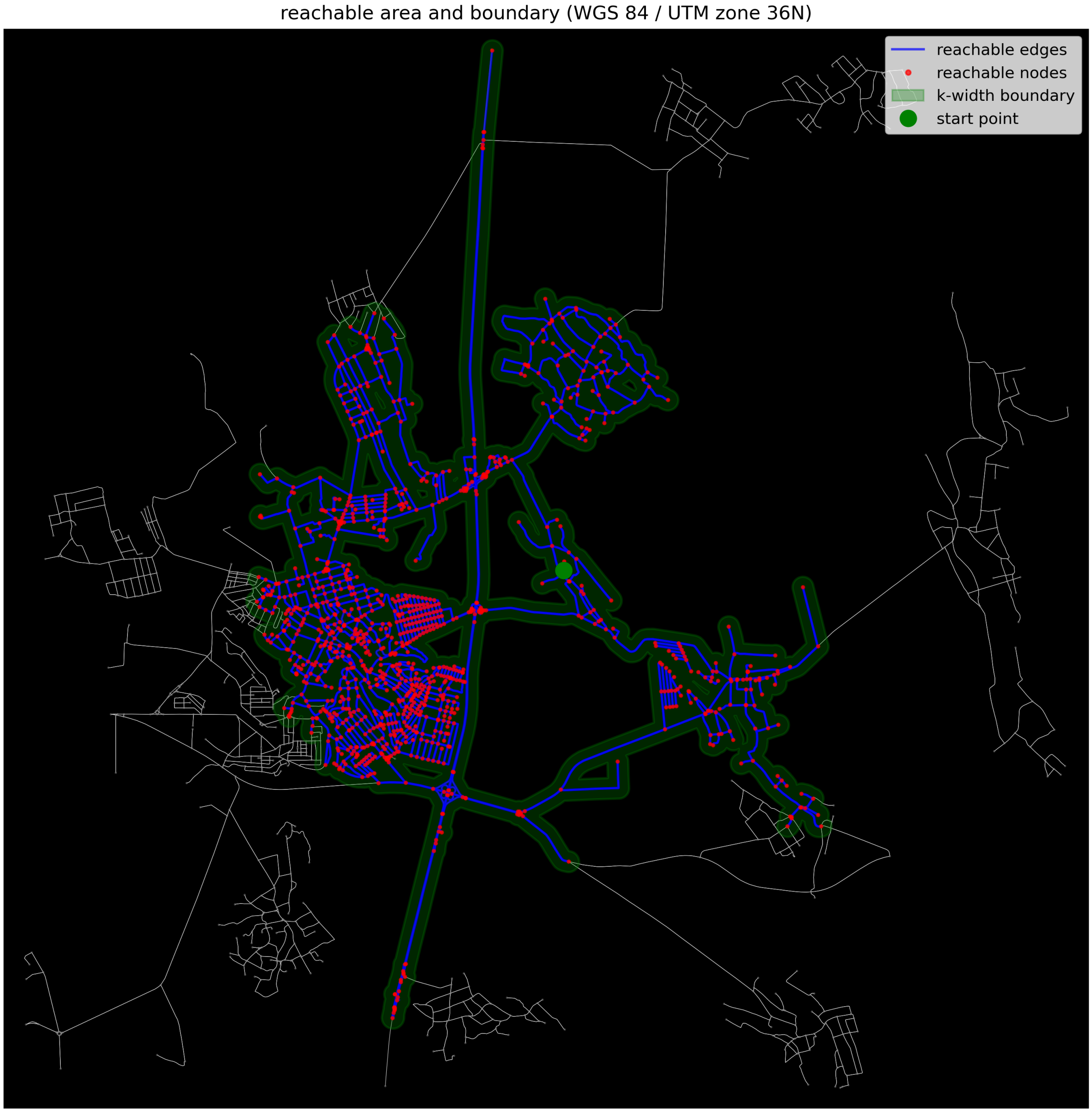

Figure 2: Possible interceptor team movement (within 10 mins, 200m buffer, initial station point @ Pikovets, Cherkasy)