Modeling an Adversarial Poacher-Ranger Hybrid Game

Maia Iyer and Benjamin Gilby
maiai@andrew.cmu.edu bgilby@andrew.cmu.edu
Introduction

Pursuit-evasion games:

● Two-player
● Real-dimensional space
● Win-condition defined by positions in space
Introduction

Active-Target Defense Differential Games:

- Two-player, three entities
Previous Work

**Perfect-information** (in the literature)

**Nash Equilibrium**

- Optimal play
- Probabilistic strategies
Introduction

**Partially-Observable** Active-Target Defense Differential Games (**POATDDG**)
Basic Poacher-Ranger Game

Partially-Observable Active-Target Defense Differential Games (POATDDG)

- 3 entities: poacher (attacker), ranger (defender), prey (target)
- Ranger/Prey win if:
  - ranger catches poacher or if poacher is never able to kill prey
- Poacher wins if it reaches prey position and not in ranger’s range
Introduction

Hybrid systems:
- Added *discrete control* dynamic
- Can augment motion fidelity

Hybrid games:
- Can model the *adversarial aspect*
- Can directly capture *imperfect information*
How can we incorporate asymmetric partial information into the game using the hybrid game framework to reason about winning ranger strategies?

Contributions
Incorporating Partial Information

- **State Observability** is ability to **observe the environment**
- **Structural Observability** is ability to understand win conditions and **play optimally**
Model Description - Constants

Definitions /* CONSTANTS */
Real maxVp; /* max poacher velocity */
Real maxVr; /* max ranger velocity */
Real maxVt; /* max target velocity */
Real killr; /* how close poacher should be to kill target */
Real capr; /* how close ranger should be to catch poacher */
End.
Model Description - Players

We focus on ranger strategy with imperfect cooperation by prey

- Prey and Poacher are on the same team
- Ranger is demonic (needs existence of one path of strategies)
- Prey/Poacher are angelic (ranger needs to work on all possible prey/poacher paths)
Model Description - Hybrid Game Structure

Problem

```
/* CONSTANT assumptions */
-
[
  /* Initial positions chosen by all entities */
  {
    /* Control */
    /* Dynamics */
    /* poacher responsible to ensure not caught */
    ?(distancesq(xP, xR, yP, yR) > capr^2);
  } /* poacher can choose to continue the game */
]( /* Safety condition */
  distancesq(xP, xT, yP, yT) > killr^2 /* target never caught */
)
End.
```
Model Description - Dynamics

```
/* Dynamics */
{pos’ = v &
  /* Poacher caught -> ranger win */
  distancesq(xP, xR, yP, yR) >= capr^2 &
  /* Target caught -> poacher win */
  distancesq(xP, xT, yP, yT) >= killr^2 &
};
/* poacher responsible to ensure not caught */
?(distancesq(xP, xR, yP, yR)> capr^2);
```
Model Description - Control

/* Target Control */
dxT1 := *; dyT1 := *; (dxT1^2 + dyT1^2 <= maxVt^2);

/* Ranger Control, generally predefined */
{dxR1 := *; dyR1 := *; (dxR1^2 + dyR1^2 <= maxVr^2);}^@;

/* Poacher Control */
dxP1 := *; dyP1 := *; (dxP1^2 + dyP1^2 <= maxVp^2);
Model Description - Partial State Observability

```plaintext
Definitions

/* Hybrid Programs */
/* this strategy keeps ranger at rest */
HP rangerStay ::= {dxR1:=0; dyR1 :=0;};
/* this strategy assumes ranger observes prey velocity and copies*/
HP rangerGo ::= {dxR1:=dxT1; dyR1:=dyT1};

End.

Problem

/* Initial conditions & constant assumptions */ ()
->
[

/* Initial positions */
{
/* Target Control */
dxT1 := *; dyT1 := *; ?(dxT1^2 + dyT1^2 <= maxVt^2);
/* Ranger Control, generally predefined */
{rangerGo; ?(dxR1^2 + dyR1^2 <= maxVr^2);}^0;
/* Poacher Control */
dxP1 := *; dyP1 := *; ?(dxP1^2 + dyP1^2 <= maxVp^2);
/* Dynamics */
}

/* Safety condition */

End.
```
Model Description - Assumptions

- Bounded space: we focused on making prey faster
- Players can choose initial state: we focused on existence of mid-game strat
- killr=0
Three POATDDGs - Extending the Poacher-Ranger Game

- **Panda Game**
  - Stationary prey, **unknown poacher parameters**

- **Elephant Game**
  - Relatively slow moving prey, **unknown poacher parameters**

- **Ostrich Game**
  - Relatively fast moving prey, **unknown poacher parameters**
  - Prey freezes when poacher is near (within “preyr” distance)
  - Ranger knows when prey sees poacher
Panda Game
Three POATDDGs - Extending the Poacher-Ranger Game

- **Panda Game**
  - Stationary prey, **unknown poacher parameters**

- **Elephant Game**
  - Relatively slow moving prey, **unknown poacher parameters**

- **Ostrich Game**
  - Relatively fast moving prey, **unknown poacher parameters**
  - Prey freezes when poacher is near (within “preyr” distance)
  - Ranger knows when prey sees poacher
Elephant Game

- Green rectangle = subspace
- Red diagonal = spaceDiagonal
- Green diagonal = rangerDiagonal

```c
/*! Required Assumption */
rangerDiagonal() / maxVr <= spaceDiagonal() / maxVt
```
Three POATDDGs - Extending the Poacher-Ranger Game

- **Panda Game**
  - Stationary prey, unknown poacher parameters

- **Elephant Game**
  - Relatively slow moving prey, unknown poacher parameters

- **Ostrich Game**
  - Relatively fast moving prey, unknown poacher parameters
  - Prey freezes when poacher is near (within “preyr” distance)
  - Ranger knows when prey sees poacher
Ostrich Game

- Green rectangle = subspace
- Yellow rectangle = miniSubspace
- Red diagonal = spaceDiagonal
- Green diagonal = rangerDiagonal
- Yellow diagonal = rangerMiniDiagonal

/* Required Assumptions */
(rangerDiagonal() - rangerMiniDiagonal()) / 2 <= preyr / maxVp * maxVr &
(rangerMiniDiagonal()) / maxVr <= spaceDiagonal() / maxVt
Ostrich Game

- Green rectangle = subspace
- Yellow rectangle = miniSubspace
- Red diagonal = spaceDiagonal
- Green diagonal = rangerDiagonal
- Yellow diagonal = rangerMiniDiagonal

```c
/* Required Assumptions */
(rangerDiagonal() - rangerMiniDiagonal()) / 2 <= preyr/maxVp*maxVr &
(rangerMiniDiagonal()) / maxVr <= spaceDiagonal() / maxVt
```
Ostrich Game

- Green rectangle = subspace
- Yellow rectangle = miniSubspace
- Red diagonal = spaceDiagonal
- Green diagonal = rangerDiagonal
- Yellow diagonal = rangerMiniDiagonal

```c
/* Required Assumptions */
(rangerDiagonal() - rangerMiniDiagonal())/2 <= preyr/maxVp*maxVr &
(rangerMiniDiagonal()/maxVr <= spaceDiagonal()/maxVt)
```
Future Work

- Extensions to Multiple Entities
  - Double the rangers -> double the playing space

- New Environmental Factors
  - Terrain
  - Acceleration
Challenges

● Formulating and balancing the game

● Strategy Complexity
  ○ Translating a geometric strategy in a dGL proof

● Gazelle Game
  ○ Prey flees from poacher
  ○ Complex strategies
  ○ Falling behind
Challenges

● Formulating and balancing the game

● Strategy Complexity
  ○ Translating a geometric strategy in a dGL proof

● Gazelle Game
  ○ Prey flees from poacher
  ○ Complex strategies
  ○ Falling behind
References

] (distancesq(xP1, xT1, yP1, yT1) > killr^2 &
(tSinceTargetSeenPoacher >= preyR/maxVp -> (xR1 = spaceToMiniRanger(xT1, maxX) & yR1 = spaceToMiniRanger(yT1, maxY))) &
(xR1 = ((tSeeingPoacher/(preyR/maxVp))*spaceToRanger(xT1, maxX) + (1-(tSeeingPoacher/(preyR/maxVp)))*spaceToMiniRanger(xT1, maxX)) &
(yR1 = ((tSeeingPoacher/(preyR/maxVp))*spaceToRanger(yT1, maxY) + (1-(tSeeingPoacher/(preyR/maxVp)))*spaceToMiniRanger(yT1, maxY)) &
(tSeeingPoacher <= preyR/maxVp - tSinceTargetSeenPoacher) &
(0 <= tSeeingPoacher & tSeeingPoacher <= preyR/maxVp) &
(0 <= tSinceTargetSeenPoacher & tSinceTargetSeenPoacher <= preyR/maxVp) &
(tSinceTargetSeenPoacher >= preyR/maxVp -> !targetSeesPoacher(xP1, yP1, xT1, yT1)) &
inBorder(xR1, yR1) & inBorder(xP1, yP1) & inBorder(xT1, yT1)
& (targetSeesPoacher(xP1, yP1, xT1, yT1) -> tSinceTargetSeenPoacher = 0))
End.