Multi-Robot Visibility-Based Pursuit-Evasion with Probabilistic Evader Models

Mukilan A.R. — **University of Dayton** — ashokrajrajapriyam1@udayton.edu
Nicholas M. Stiffler — University of Dayton — nstiffler1@udayton.edu

University *of* Dayton

Contribution

A formulation of the visibility-based pursuit-evasion problem that incorporates information about how the target agents (evaders) are "likely" to move in a given scenario to produce a more robust search strategy for our searchers (pursuers).

Existing Techniques

- Can not readily incorporate any information about the targets behavior/movement and default to an uninformed search.
- These techniques are designed to be "complete" (they will find everyone ... eventually). However, there is no way to modify the search strategy in any meaningful way to search known high density areas first.

Our Approach

- *Predicts* what the targets are likely to do.
- Design a two-phase algorithm that
  1. **Exploits** the information from our prediction.
  2. **Explores** the remainder of the environment to find any remaining targets.

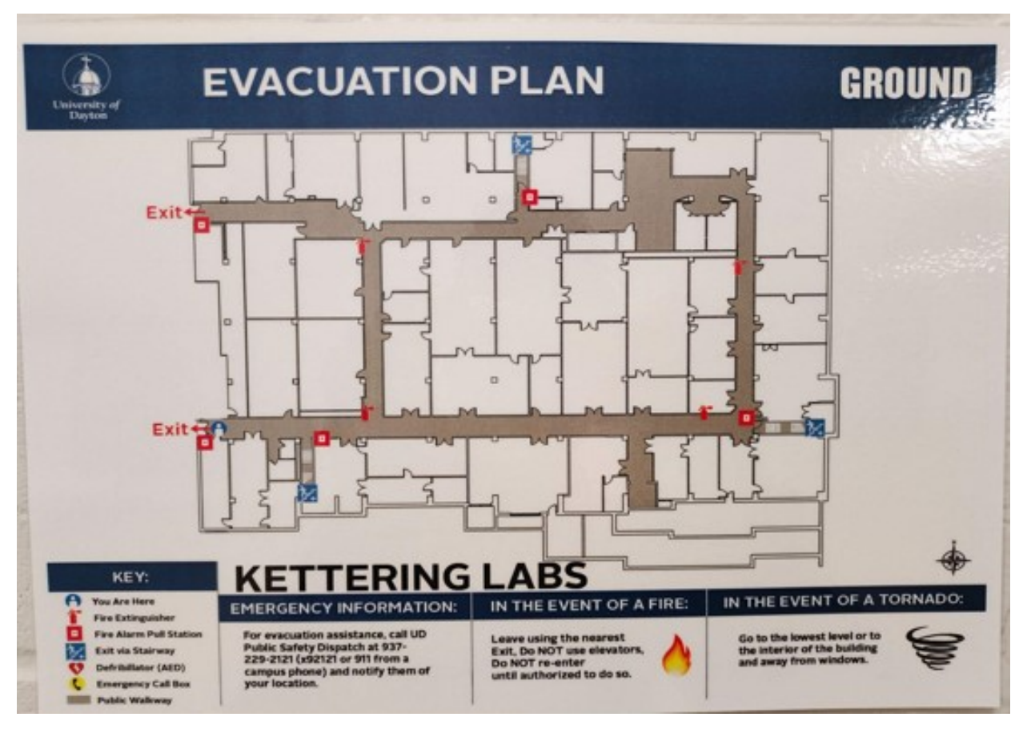
What is Pursuit-Evasion?

A two-player game where one group of agents, called the **pursuers**, actively seek out members of a second group, the **evaders**.

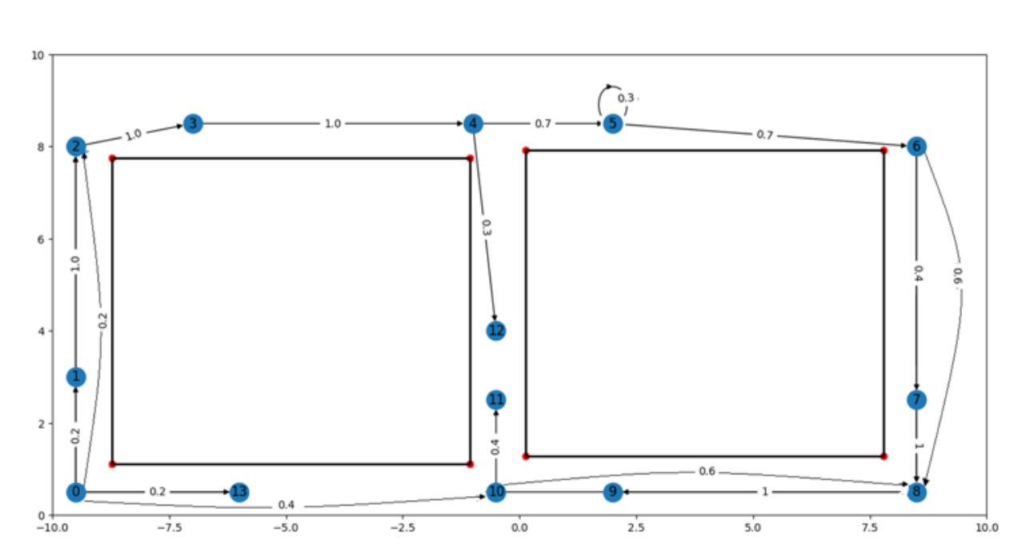
- The pursuers' goal is to capture the evaders.
- The evaders wish to evade capture.

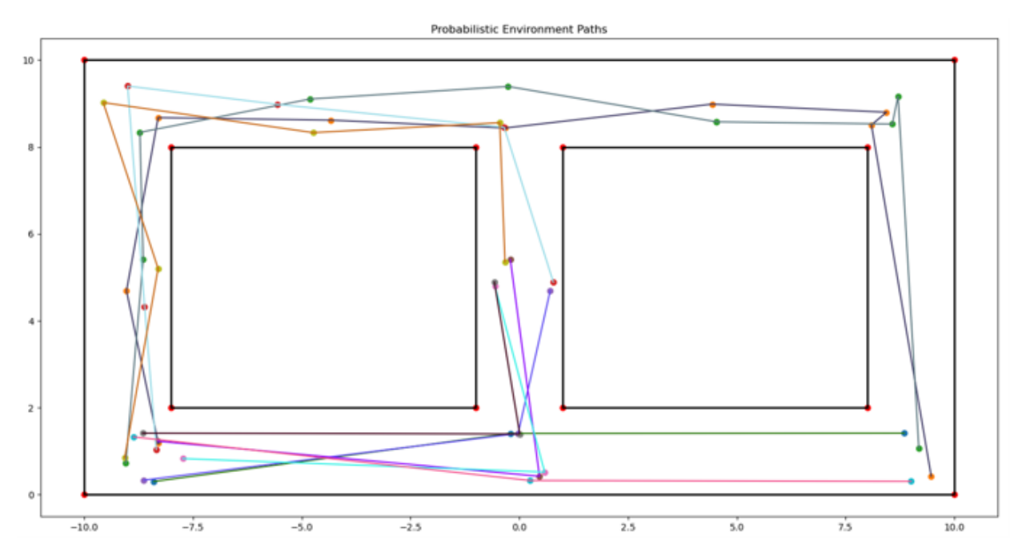
Modeling Evaders' behavior

- **Emergency Egress** - a method of exit which people can access safely in an emergency.
- **Informed Evacuation Model** - A predictive model that describes how an agent might behave in an emergency situation.



Probabilistic Evader Models



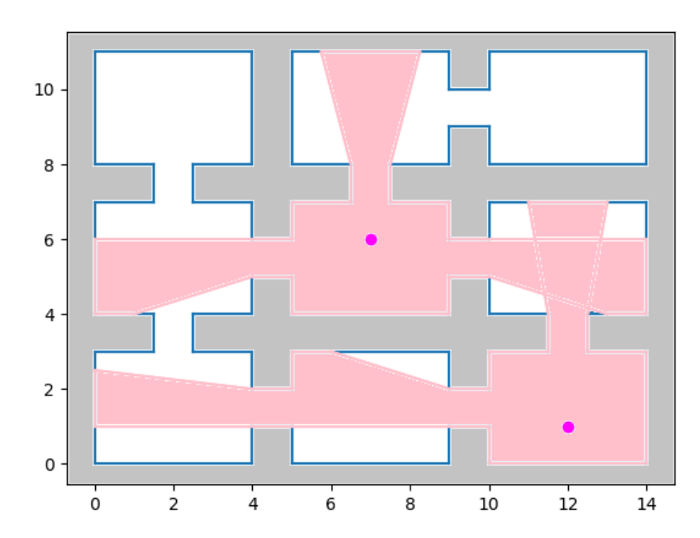


The pursuers can use the Probabilistic Evader Model to *generate/sample* a path representative of an evader that decides to follow a given model.

Multi-Robot Visibility-Based Pursuit-Evasion

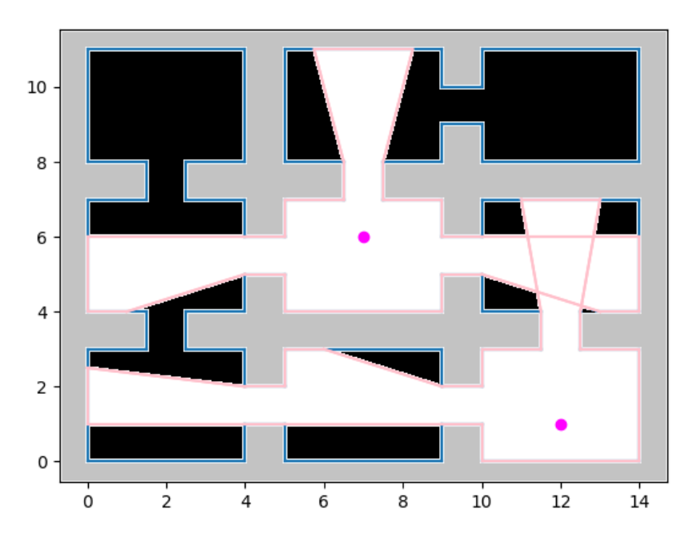
Rather than try to reason about where any specific evader "could be" we instead worry about the currently *seen* and *unseen* areas of the environment.

Visibility Polygons (Seen)



All points visible from a given point $p = (x, y)$ contribute to the Visibility Polygon $\mathcal{VP}(p)$.

Shadows (unseen)



Shadows are the unseen area of the environment, $\mathcal{S} = Env \setminus \mathcal{VP}(p)$. The key idea is that we maintain the *status* of each shadow.

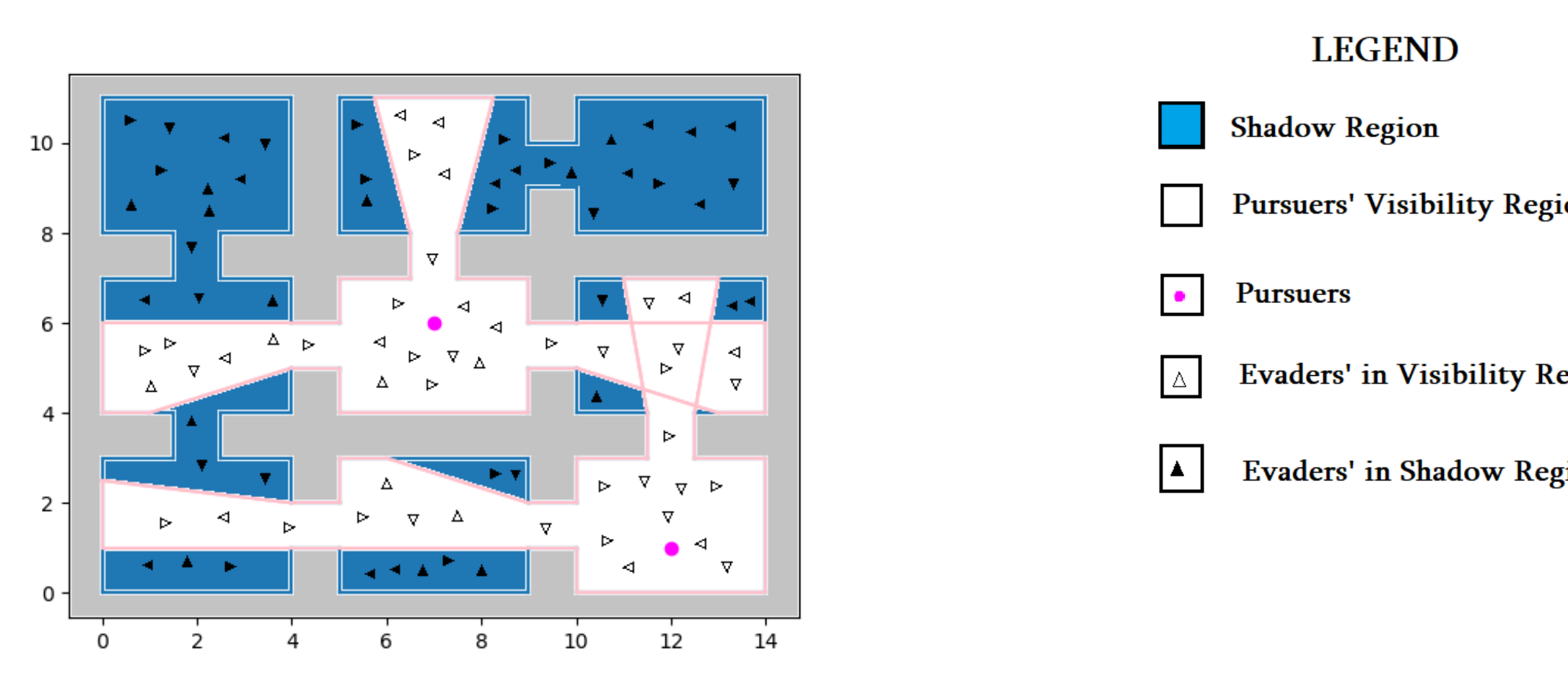
A **shadow** can be either:

- clear - we can guarantee it does not contain an evader
- contaminated - it "may" contain an evader

Objective

Compute a strategy where all shadows are simultaneously clear.

Problem Setup



Problem Formulation and Algorithm

Problem Definition - *Given an environment and a team of pursuers, compute a joint strategy for the pursuers that locates all of the evaders in the environment.*

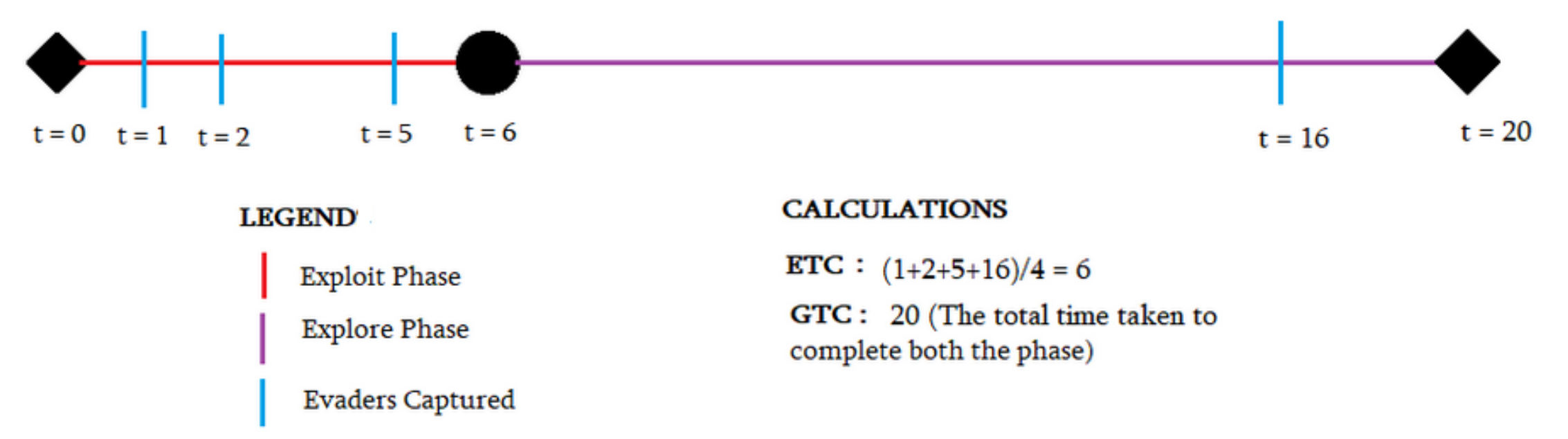
Our algorithm proceeds in TWO PHASES

1. Exploit Phase
2. Explore Phase

Metrics

Expected Time to Capture (ETC): *Expected time to locate each and every evader in the environment. ETC is the average time taken by the pursuers to capture each evader.*

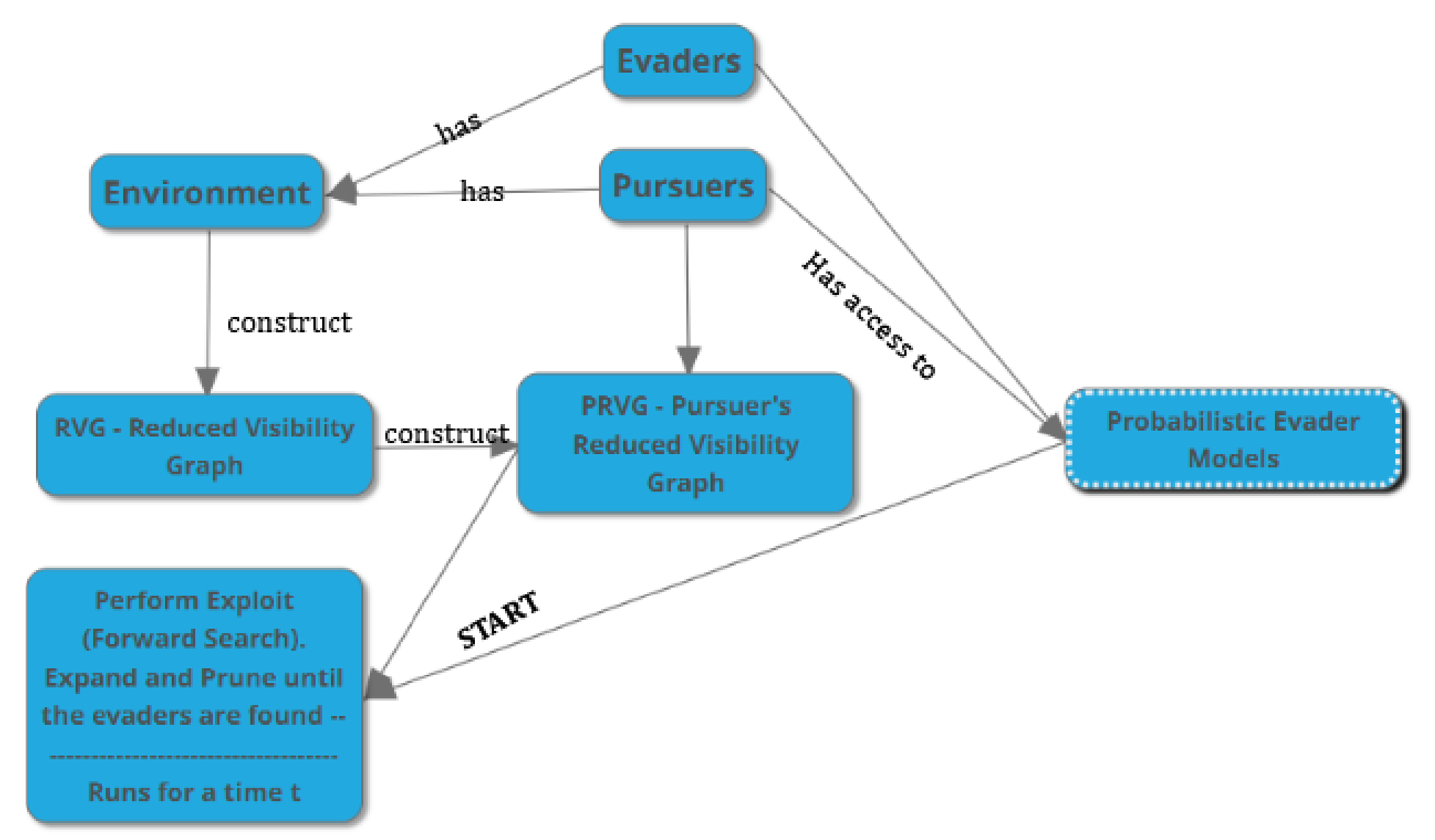
Guaranteed Time to Capture (GTC): *Time required for the pursuer to complete its solution strategy. GTC is the total time taken to complete both the phases.*



Exploit Phase

Generate representative paths from a probabilistic model of evader behavior and compute a joint plan to capture the evaders based on the model.

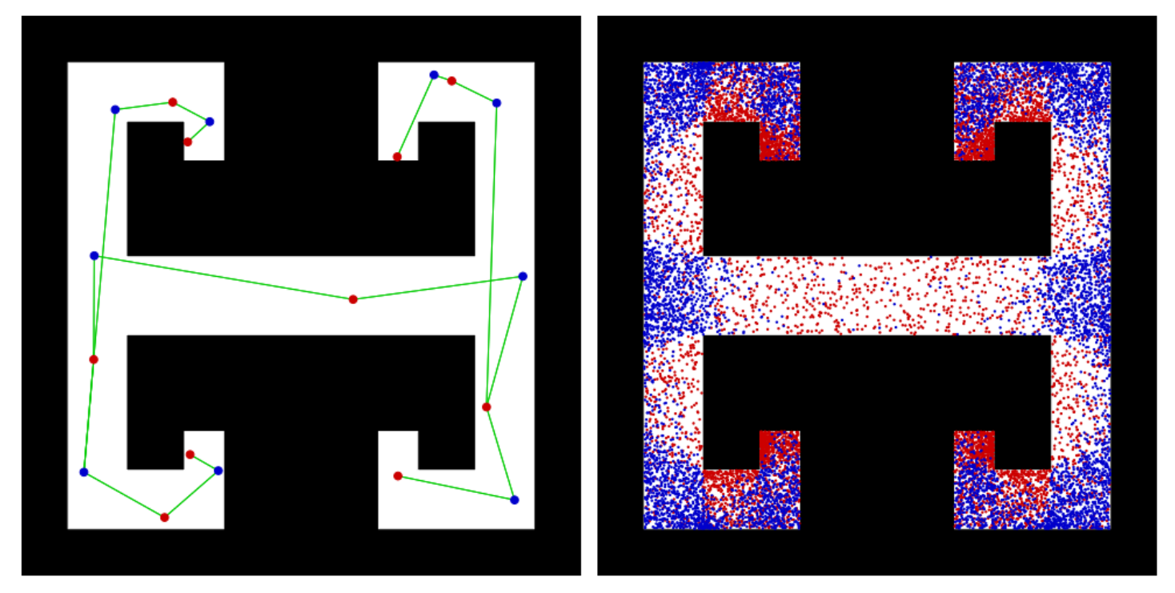
- **INPUT:** Environment, Probabilistic Evader Model(s), # of Pursuers
- **OUTPUT:** A joint parameterized Pursuer path


Evaders
has
Environment
has
Pursuers
construct
Has access to
RVG - Reduced Visibility Graph
construct
PRVG - Pursuer's Reduced Visibility Graph
Probabilistic Evader Models
Perform Exploit (Forward Search). Expand and Prune until the evaders are found --
Runs for a time t
START

Explore Phase

Although the *exploit* phase guarantees to find all the evader that adhere to the "sampled" trajectories, it does not provide any guarantees regarding evaders who do not adhere to the model. Thus, we will Append additional pursuer motions to the plan to ensure all evaders are found, regardless of their behavior.

- We use a data structure called the Sample-Generated Pursuit-Evasion Graph (SG-PEG) to reason about reachable joint pursuer configurations.
- A search through this graph yields a feasible motion strategy for the pusuers to ensure the detection of all evaders.



Implementation/Simulation

The algorithm was implemented in Python. Pertinent packages include:

- Graphs and other Data Structures : NetworkX Package
- Computational Geometry : Scikit-Geometry Package
- Plots and Figures : Matplotlib Package
- Optimized representations : BitVector (For Bit values)

We are in the process of building the simulator for the purpose of demonstration, and it is expected that the simulation would be ready by June.

Acknowledgement

University *of* Dayton
Department of Computer Science

This material is supported by the University of Dayton, College of Arts and Sciences, Graduate Student Summer Fellowship.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors.