Project II Report for COM S 4/5720 Spring 2025: Multi-Agent Pursuit and Evasion Planner

Nicholas Morrow[1]

I. PLANNER OVERVIEW

The planner is designed for the three-agent pursuit-evasion environment of Project II. Each agent receives:

- Its own position,
- The position of its target,
- The position of its pursuer,
- The grid (2D numpy array) representing the environment.

The agent must choose one of 8-connected moves (including diagonals) to pursue the target while evading the pursuer. The ‘PlannerAgent‘ class implements a role-agnostic, greedy heuristic-based planner that balances:

- Shortest path to the target,
- Distance from the pursuer,
- Escape potential (number of valid neighboring tiles),
- Trap avoidance,
- Loop detection and breaking.

II. DETAILED PLANNER LOGIC

The planner operates as a role-agnostic agent that must simultaneously:

- Pursue its designated target,
- Avoid its own pursuer,
- Maintain mobility and avoid traps,
- Break out of cycles (e.g., oscillating between two tiles).

A. Heuristic Action Scoring

At each step, the agent evaluates all valid 8-connected moves. Each move is scored using the following function:

$$\text{score} = \frac{w_t}{d_t + 1} - \frac{w_p}{d_p + 1} + w_e \cdot E - \text{TrapPenalty}$$

- d_t: Manhattan distance to the predicted position of the target.
- d_p: Manhattan distance to the predicted position of the pursuer.
- E: Number of valid neighboring cells (escape options).
- TrapPenalty: Applied if $E \leq 2$ and $d_p < 4$.

This scoring function enables the agent to balance aggression (catching the target) with caution (avoiding being caught or trapped).

[1]Nicholas Morrow is with the Department of Computer Science, Iowa State University, Ames, IA 50011, USA `nmorrow@iastate.edu`

B. Prediction Mechanism

The planner predicts where the target and pursuer will move using:

- Greedy estimation (toward or away from agent),
- Velocity extrapolation based on the last 2–3 positions.

This allows the agent to anticipate motion trends without needing to simulate the opponent's internal logic.

C. Trap and Loop Avoidance

Trap Detection: The agent uses the number of escape routes to detect tight spaces. If it has ≤ 2 valid neighbors and the pursuer is nearby, a penalty is applied to discourage entering the area.

Loop Detection: The agent tracks its recent positions. If the same position is visited multiple times (based on a tunable threshold), a loop-breaking mode is triggered. In this mode, the agent chooses the move with the most escape routes and farthest from the pursuer.

D. Design Rationale and Theoretical Justification

Each component of the planner was designed to operate under the constraints of the Project II environment: real-time decision-making, local observations only, and adversarial dynamics. Below I explain the rationale for key decisions and relate them to established techniques in AI and robotics.

Use of Manhattan Distance: Manhattan distance was used to estimate proximity to both the target and the pursuer. This choice was made because:

- It is computationally efficient compared to Euclidean distance,
- It aligns with grid-based 8-connected movement,
- It provides a consistent gradient for scoring positions.

This is a common heuristic in grid pathfinding and real-time agent movement [1].

Scoring Function Formulation: The scoring function is a weighted sum of pursuit, evasion, mobility, and safety terms:

$$\text{score} = \frac{w_t}{d_t + 1} - \frac{w_p}{d_p + 1} + w_e \cdot E - \text{TrapPenalty}$$

This structure is inspired by utility-based agents [1], where multiple competing goals are balanced using tunable weights. The use of inverse distance allows for diminishing returns as the agent gets closer to the target and increases responsiveness to nearby threats.

Escape Route Encouragement: The number of valid neighboring tiles (E) is used to estimate local freedom of movement. This is similar to mobility-based scoring in tactical planning and helps avoid narrow paths, dead ends, and early cornering. It encourages the agent to stay in open areas where more evasive options are available.

Trap Avoidance: Trap detection is based on both local geometry (low escape routes) and proximity to the pursuer. Without this term, the agent frequently entered cul-de-sacs and was captured. By penalizing such positions, the agent exhibits behavior similar to safe-zone estimation in adversarial robotics [2].

Loop Detection and Cycle Breaking: The planner tracks recent positions and detects loops by checking for repeated states. If a loop is detected, the agent breaks out by selecting moves that maximize mobility and distance from the pursuer. This technique is similar to the cycle detection strategies used in real-time search algorithms [3].

Prediction Using Velocity Estimation: To anticipate the motion of other agents, the planner uses a simple velocity-based model based on the last 2–3 observed positions. This is inspired by behavioral steering models used in navigation and obstacle avoidance [4], where recent movement is extrapolated to predict future locations.

III. Algorithm Selection Rationale

Several classical and modern planning approaches were considered and tested during development, including A* and Monte Carlo Tree Search (MCTS). Ultimately, we chose a lightweight, greedy heuristic planner with prediction and trap-avoidance for the following reasons:

A. A* Pathfinding

While A* is optimal for static pathfinding, it does not account for dynamic threats or adversarial agents. It will lead the agent blindly into traps or into the path of its pursuer. Since the agent must simultaneously pursue and evade, A* was not suitable.

B. Monte Carlo Tree Search (MCTS)

MCTS was tested and performed well in some cases, but required significantly more computation per step. Due to Project II's real-time constraints and the small action space, MCTS was unnecessarily expensive. Additionally, it did not outperform simpler heuristics in many maps.

IV. Hyperparameter Tuning and Evaluation

To optimize performance, we implemented a fully parallelized tuning framework using Python's concurrent.futures. The framework tested thousands of configurations across multiple maps, roles, and random seeds.

A. Parameters Tuned

- history_length: Number of past positions to track (for velocity prediction).
- cycle_buffer / cycle_trigger: Loop detection sensitivity.
- w_target: Weight for approaching the target.
- w_pursuer: Weight for avoiding the pursuer.
- w_escape: Bonus for high escape mobility.
- w_trap: Penalty for being in risky areas.

Each configuration was scored using:

$$\text{score} = 3 \cdot \text{wins} + \text{ties} - \text{losses}$$

The search space included over 10,000 combinations, tested efficiently in under 30 minutes on a multi-core CPU.

B. Final Showdown Evaluation

After identifying top candidates, we performed a direct head-to-head evaluation between the two best configurations:

Config A:

- history_length = 3, buffer = 10, trigger = 4
- w_target = 12.5, w_pursuer = 6.5, w_escape = 0.35, w_trap = 2.0

Config B (Final Selection):

- history_length = 3, buffer = 10, trigger = 4
- w_target = 13.0, w_pursuer = 7.5, w_escape = 0.3, w_trap = 1.5

Each configuration was run as the Jerry agent for:

- 100 maps × 5 seeds = 500 matches
- Repeated across 5 full trials to account for randomness

C. Results (Averaged Over 5 Trials)

Config	**Wins**	**Ties**	**Losses**	**Avg Score**
B (winner)	155.8	54.4	289.8	**232.0**
A	147.6	52.0	300.4	194.4

Config B demonstrated consistent superiority, particularly in both win rate and overall score. It was selected as my final planner configuration due to its aggressive pursuit behavior balanced with effective survival logic.

References

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach.* Pearson, 2016.

[2] N. Iskander and S. M. LaValle, "Pursuit and evasion in a polygonal environment using a visibility graph," *IEEE International Conference on Robotics and Automation*, 2005.

[3] S. Koenig and M. Likhachev, "Real-time search for robot path planning," in *IJCAI*, 2005.

[4] B. R. Fajen and W. H. Warren, "Behavioral dynamics of steering, obstacle avoidance, and route selection," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 29, no. 2, p. 343, 2003.