AI Agents for Planet Wars

Aman Choudhary , Drumil Vasani
High Integrity System, Frankfurt University of Applied Sciences,
Frankfurt am Main, Germany

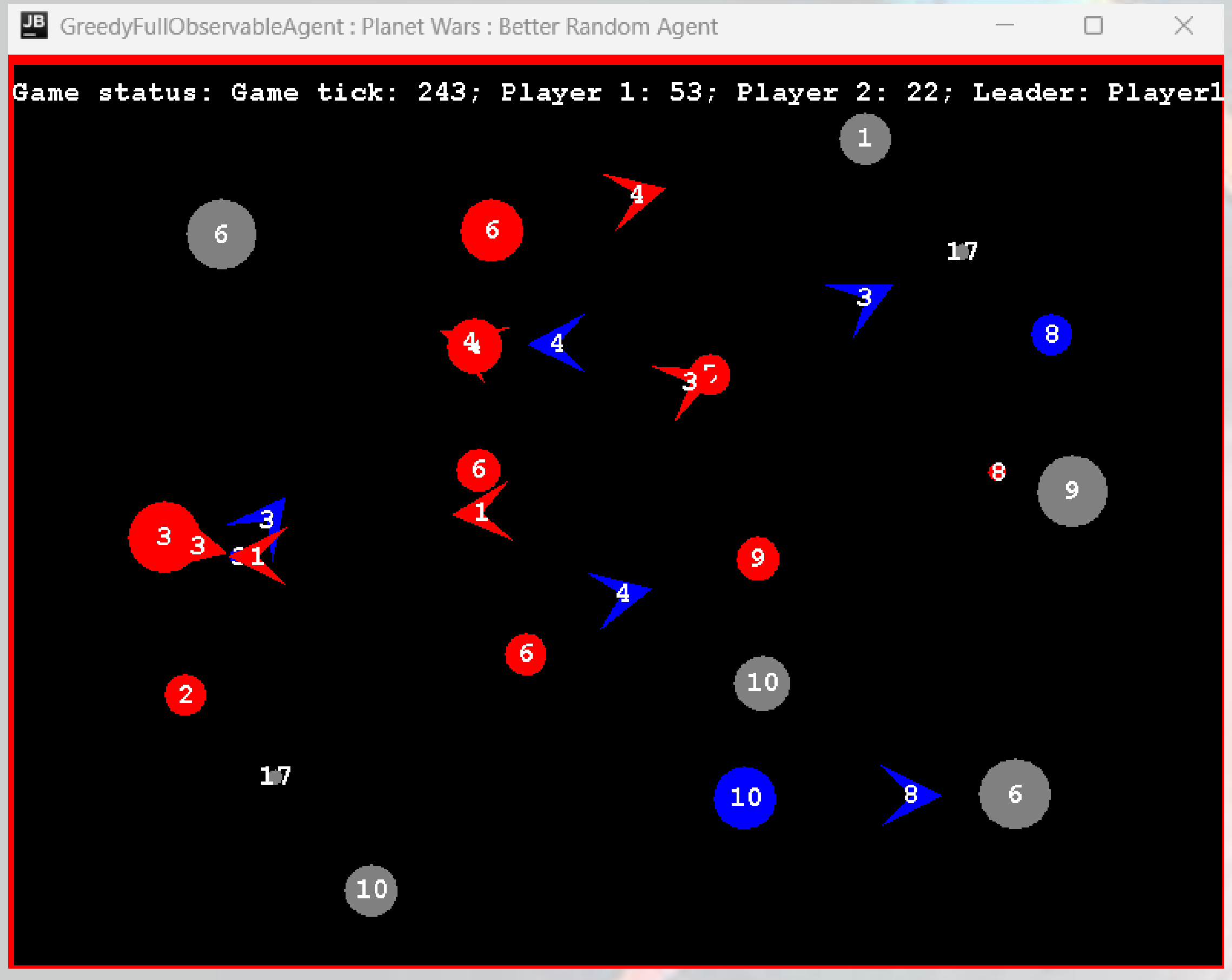
Summary

We designed a heuristic-based greedy agent for *Planet Wars*, a two-player RTS game. Each turn the agent scores every possible fleet transfer using a weighted function of enemy strength, planet growth, and distance, then applies ε-greedy randomness to avoid deterministic traps. A reinforcement fallback kicks in when direct attacks look poor. Across 50 matches versus a random agent, a greedy baseline, and an evolutionary agent (EvoAgent), our method won **84 %** of games while keeping computation minimal—showing that lightweight heuristics can compete with more complex strategies in dynamic RTS settings.

Agent Overview

- Core concept of our agent
  - A fully observable greedy agent that evaluates every possible fleet transfer using a weighted cost-benefit heuristic.
- High-level strategy: heuristic
  - Scores each source → target pair based on:

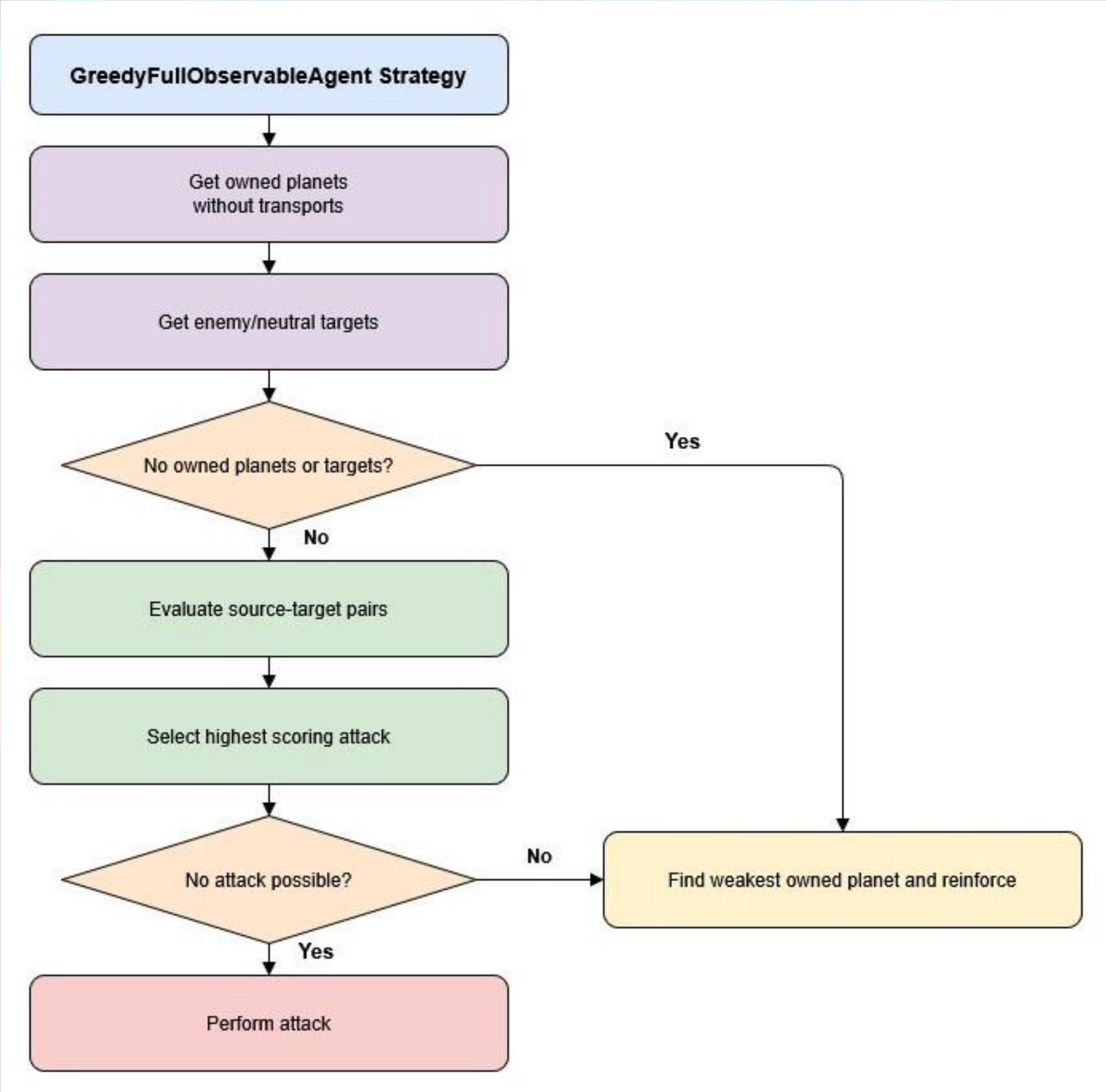
    score = −ships × w1 + growth × w2 − distance × w3 + ε
  - Selects the highest-scoring valid move per turn.
  - Reinforces weak owned planets if no attacks are viable.
  - Random tie-breaking ensures strategy variety.


GreedyFullObservableAgent : Planet Wars : Better Random Agent

```
Game status: Game tick: 243; Player 1: 53; Player 2: 22; Leader: Player1
```

- Key novelty or approach
  - All-path scoring ensures maximum tactical coverage.
  - ε-noise in scoring avoids deterministic traps.
  - Reinforcement fallback prevents stagnation.

System Design


GreedyFullObservableAgent Strategy

Get owned planets without transports

Get enemy/neutral targets

No owned planets or targets?

Yes

No

Evaluate source-target pairs

Select highest scoring attack

No attack possible?

No

Find weakest owned planet and reinforce

Yes

Perform attack

Agent Logic Flow

- The agent first identifies owned planets without active fleets.
- It then filters viable targets (enemy or neutral).
- All possible source-target actions are evaluated using a weighted score:

Score = − (enemy ships) × w1 + (growth rate) × w2 − (distance) × w3 + ε

- If no attacks are safe (based on a safetyBuffer), it reinforces the weakest owned planet.
- Otherwise, it dispatches 50% of ships from the best source planet.

Component Interaction

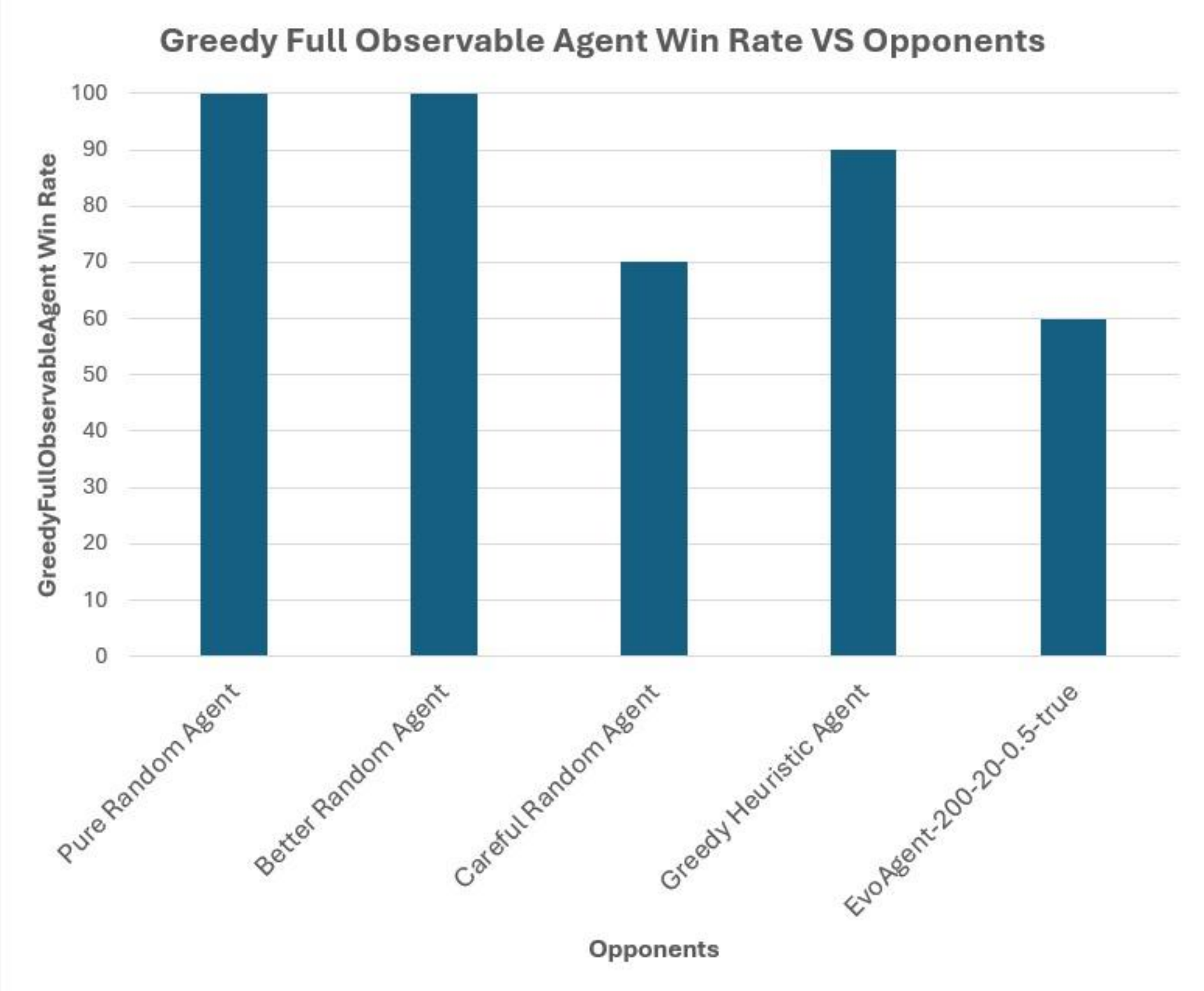
1. **GameState → State Parser:** Extracts planet details (position, ownership, ships, growth rate).
2. **State Parser → Scoring Engine:** Calculates heuristic values for each source-target planet combination (growth, distance, ship cost), adding random noise (ε) for diversity.
3. **Scoring Engine → Decision Logic:** Chooses the best move if valid; considers reinforcements if no attacks are viable.
4. **Decision Logic → Action Output:** Converts the selected move into a fleet command (Action) or "DoNothing" if no useful move is available.

Heuristic Function

Score = − (enemy ships) × w1 + (growth rate) × w2 − (distance) × w3 + ε

- w_1: 1.0, w_2: 2.0, w_3: 0.5
- ε: Random [0.0, 0.1] for strategy variation
- **Safety buffer**: 1.2× enemy ships

Result



- Evaluation Setup
  - 50 games total (10 per opponent).
  - Fully observable mode, remote Docker execution.
  - Opponents: Randoms, Heuristic, EvoAgent.
- Notable matchups
  - Dominated all random agents (100%).
  - Outperformed Greedy Heuristic (90%).
  - Held ground vs. EvoAgent (60%).

Conclusion

- Our greedy agent uses a weighted heuristic to evaluate all source-target fleet transfers in fully observable RTS environments.
- Enemy strength, planet growth rate, and distance are balanced to select high-value actions; ε-greedy randomness avoids deterministic traps.
- A reinforcement-based fallback adds robustness when direct attacks are suboptimal.
- Achieved **84% win rate** over 50 games, outperforming random and heuristic agents, and competing strongly with evolutionary agents.
- Demonstrates that lightweight, heuristic-driven strategies can achieve high performance with minimal computational cost.

Github Page

The source code for the AI agent can be found here: https://github.com/drumilvasani/Planet-Wars-AI-Challenge-2025.git

Scan Here