Kabaddi Game — AI Simulation

Comparison of AI Agents: Random, Greedy, Alpha-Beta, MCTS

Gour Krishna Dey | Roll - MT24035

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Agenda

1. Motivation & Problem Statement
2. Game Environment & Rules
3. Agents & Strategies
4. System Architecture & Design
5. Code Walkthrough
6. Experiments & Results
7. Final Conclusions & Future Work

Motivation & Problem Statement

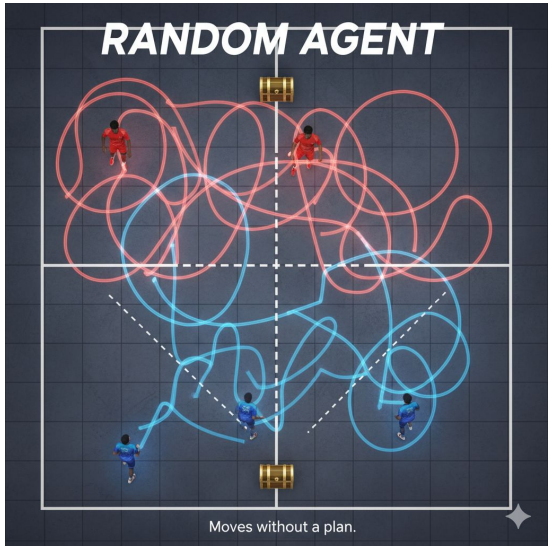
- Why Kabaddi? → adversarial, multi-agent, strategic
- **Goal**: simulate simplified Kabaddi on a grid
- **Objective**: compare agent strategies on win rate, steps, captures
- **Deliverable**: a single environment with multiple agents

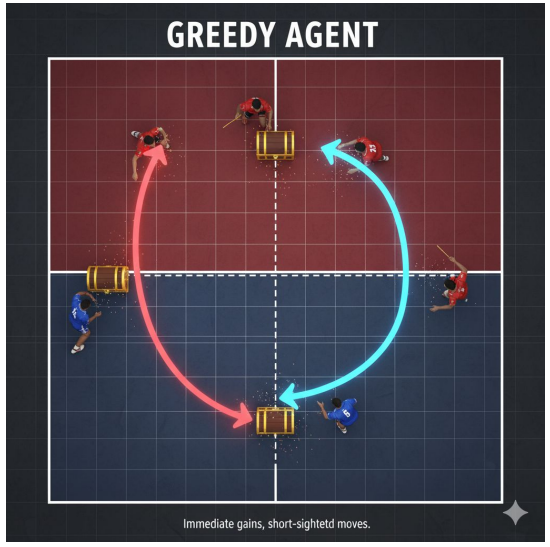
Game Environment & Rules

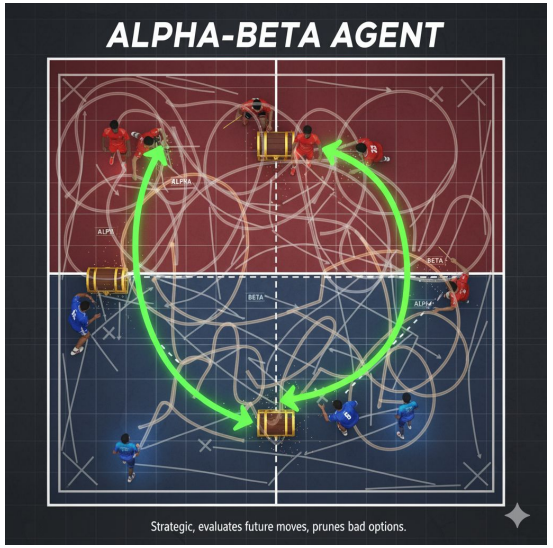
- Grid: default 5×6 (split into halves)
- Each team: 2 players + 1 treasure
- Allowed moves: Stay, Up, Down, Left, Right
- Capture rule: intruder in enemy half eliminated if collision
- **Winning**: treasure stolen and returned to home half
- **Draw**: simultaneous success or step limit exceeded

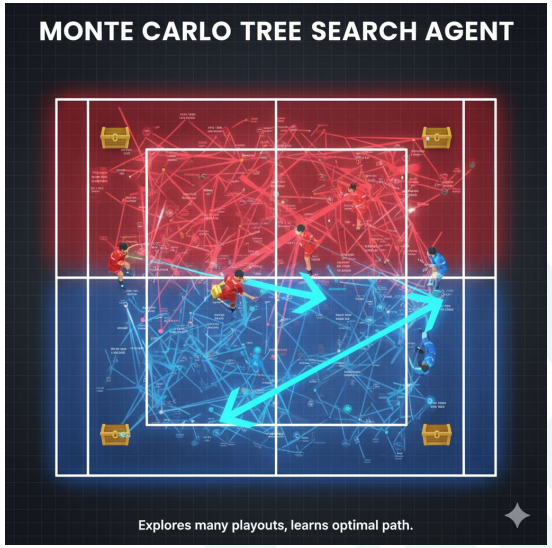
Agents & Strategies

- **Random Agent** → non-intelligent baseline
- **Greedy Agent** → follows Manhattan distance heuristic
- **Alpha-Beta Agent** → minimax search with alpha-beta pruning
- **MCTS Agent** → Monte Carlo rollouts and statistical reasoning
- Key comparison: chance vs heuristic vs search vs simulation









System Architecture & Key Classes

- **Position**: grid coordinates
- **Player**: state (pos, alive/dead, treasure flag)
- **GameState**: rules & environment handling
- **Agent interface**: getMoves() function
- Simulation functions: playGame(), simulateTournament()

Code Walkthrough

Results & Analysis

Tournament outcomes:

- Alpha-Beta & MCTS → strongest performers
- Greedy → weakest
- Random → inconsistent, sometimes effective

Mode effect:

- Turn-based → longer, strategic games
- Simultaneous → more dynamic, uncertain

```
===== Tournament: 100 games per ordered pair =====
Random (A) vs Greedy (B) --> A_wins=99, B_wins=0, Draws=1, avgStepsA=75.60
Random (A) vs AlphaBeta (B) --> A_wins=42, B_wins=51, Draws=7, avgStepsA=88.48, avgStepsB=81.92
Random (A) vs MCTS (B) --> A_wins=44, B_wins=49, Draws=7, avgStepsA=85.45, avgStepsB=97.61
Greedy (A) vs Random (B) --> A_wins=0, B_wins=98, Draws=2, avgStepsB=68.78
Greedy (A) vs AlphaBeta (B) --> A_wins=0, B_wins=100, Draws=0, avgStepsB=72.45
Greedy (A) vs MCTS (B) --> A_wins=0, B_wins=99, Draws=1, avgStepsB=68.19
AlphaBeta (A) vs Random (B) --> A_wins=47, B_wins=37, Draws=16, avgStepsA=91.57, avgStepsB=94.03
AlphaBeta (A) vs Greedy (B) --> A_wins=100, B_wins=0, Draws=0, avgStepsA=73.08
AlphaBeta (A) vs MCTS (B) --> A_wins=50, B_wins=47, Draws=3, avgStepsA=94.58, avgStepsB=90.30
MCTS (A) vs Random (B) --> A_wins=44, B_wins=47, Draws=9, avgStepsA=90.30, avgStepsB=79.49
MCTS (A) vs Greedy (B) --> A_wins=99, B_wins=0, Draws=1, avgStepsA=77.87
MCTS (A) vs AlphaBeta (B) --> A_wins=51, B_wins=41, Draws=8, avgStepsA=87.94, avgStepsB=90.44
```

```
PS E:\MTechCSE\Study\Sem3\AI\Assignment\Assignment_2> ./a
Do you want default setup? (y/n): y
Using default setup (5x6).

Choose mode: 1 = Single Game, 2 = Tournament: 1
Choose Agent for Team A (1=Random,2=Greedy,3=AlphaBeta,4=MCTS): 1
Choose Agent for Team B (1=Random,2=Greedy,3=AlphaBeta,4=MCTS): 2
Choose gameplay type: 1=Turn-based,2=Simultaneous: 1
Game Start: Team A(Random) vs Team B(Greedy)
A . . | . . .
A . . | . . .
. T . | . t .
. . . | . . B
. . . | . . B
----------------------
Step 1:
Team A moves: Up Stay
Team B moves: Left Left
A . . | . . .
A . . | . . .
. T . | . t .
. . . | . B .
. . . | . B .
----------------------
Step 2:
Team A moves: Down Down
Team B moves: Left Left
. . . | . . .
A . . | . . .
A T . | . t .
. . . | B . .
. . . | B . .
----------------------
Step 3:
Team A moves: Up Stay
Team B moves: Left Up
A . . | . . .
. . . | . . .
```

Conclusion

- **Alpha-Beta** is the most consistent and effective strategy.
- **MCTS** performs strongly but shows variability due to randomness.
- **Greedy** consistently fails in adversarial setups.
- **Random** can occasionally succeed through unpredictability.
- **Final Takeaway:**
  Success in multi-agent games requires a balance between strategic depth and adaptability.

Links

Github Repo for Code , Reports & PPT - https://github.com/gkdey17cse/AI_Assignment_2025/tree/main/Assignment_2

Submission Drive Link - https://drive.google.com/drive/u/0/folders/1BcZOZ_CILh2-h756Sk1J3HYpdW4VmWoN

! Thank You !