CSCI-8920
WarLight Bot : A Gaming Problem in a Unique Environment

Maulik Shah
Anubhav Bharadwaj

Agenda

Introduction and Goals

- Rules
- Goals

Analysis and Methodology

- Analysis
- Framework
- POMDP – Search Tree
- Bayesian Games

Performance

Challenges and Solutions

- Memory issues
- Time issues

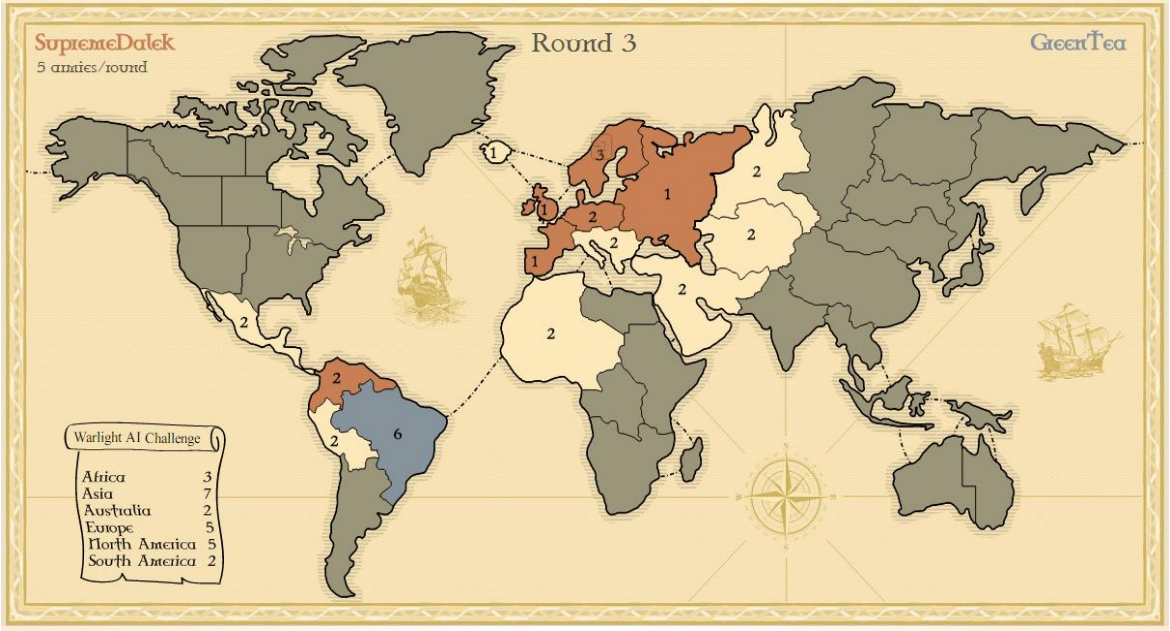
Limitations and Future Improvements

Questions

Introduction and Goals

Introduction and Project Goals

Introduction



Project Goals

- Analysis of the overall game and finding the impacting parameters
- Come up with best suitable architecture for game playing
- Create a BOT, which efficiently and effectively work in the given environment.

Analysis and Methodology

Analysis

Input Parameters :

- Neighborhood Regions
- Actions in Neighborhood Regions
- Number of Rounds

Output Parameters:

- Overall Map
- Army of the Opponent per turn

Methodology

01

POMDP

- Find output parameters for fog region is the only uncertain

02

Solution Tree

- Explore all possible options – Depth First and choose the best.

03

Bayesian Games

- Explorer – Warrior
- Reward Functions

04

Implementation

- MDP ➔ POMDP
- Apply Bayesian Games

05

Technical Implementation

- From scratch in Java
- Sample BOT available for reading the Map

POMDP

POMDP := < State Space(S), Set of Actions(A), T(Transition Function), Observation(Ω), Reward Function (R) , OC (Optimality Criteria)>

S : Overall Map with information of each region and its properties (Army on the Region, Neighbors, Owner)
Fogg region is uncertain.

A : Set of actions in A’ (Set should be considered sequentially)
Where, A’ = { Deploy, Attack, Transfer}

POMDP

T : (3 different functions for each action)
Deployment : Add the armies to that of the region
Attack : Army after conflict in Destination =
(Destination Region Army – 60% of the Attacking Army) +
(Attacking Army – 70% of the Source Region Army)
Source Army After Conflict = Source Army – Attacking Army
Transfer : Move from one region to another.

Ω : Input Variables

- Number of Rounds
- Neighborhood Region
- Actions of the opponents in the Neighborhood

POMDP

R : Bayesian Games Impact

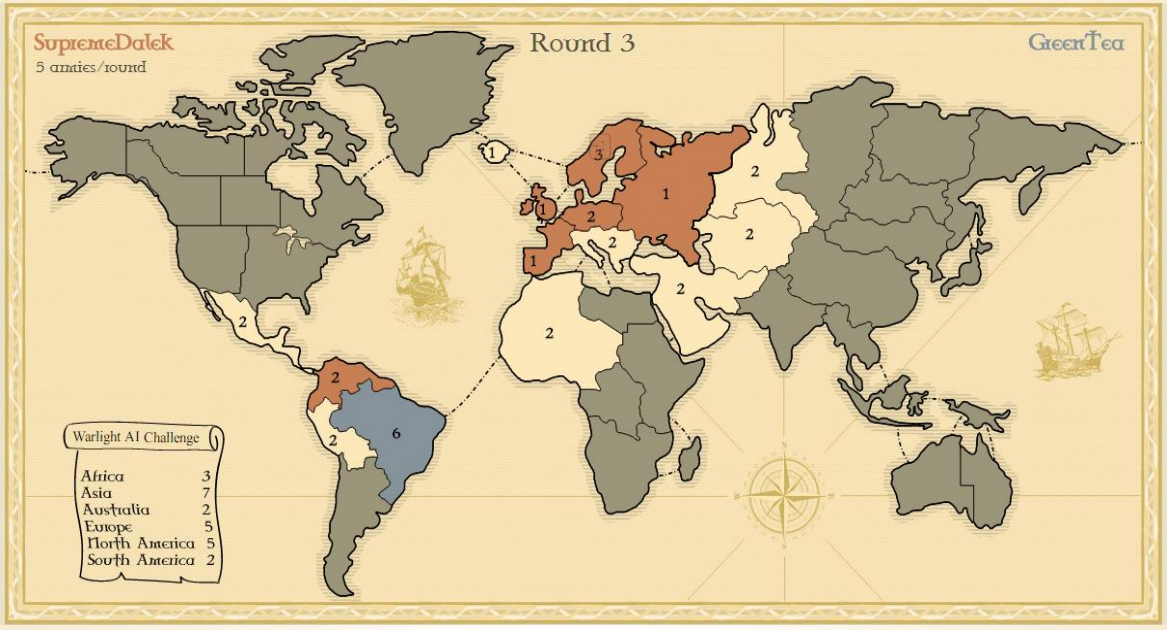
Explorer = (Army Strength – Army In Mid Regions)
* (Super Region size to reward ratio * No. of regions) ^ 2
* Armies Per Turn / 100

Warrior = (Army Strength – Army In Mid Regions)
* (Sum of Superiority with opponent's army per region)
* (Super Region size to reward ratio * No. of regions) ^ 2
* Armies Per Turn / 100

OC : Capture all the territories.

Solution Tree in POMDP

- Solution Tree instead of Value Iteration – More suitable.
- Each node is the State of the Map.
- Iterate all possible combinations. Depth First Tree Exploration.
- Best Reward is the Solution.



Performance

Performance

- Let’s have a demo !
- Working well in initial states.
- Glitches start after some rounds.
- Completely stops after some more !

Challenges and Solutions

Challenges And Solutions

Error dump

```
Java HotSpot(TM) 64-Bit Server VM warning: No monotonic clock was available - timed services may be
adversely affected if the time-of-day clock changes

Exception in thread "main" java.lang.OutOfMemoryError: Java heap space

        at java.util.LinkedList.linkLast(LinkedList.java:142)

        at java.util.LinkedList.add(LinkedList.java:338)

        at main.Region.addNeighbor(Region.java:52)

        at main.Map.getMapCopy(Map.java:81)

        at bot.BotState.(BotState.java:57)

        at mdp.beans.WarlightMDPSkeleton.(WarlightMDPSkeleton.java:47)

        at mdp.WarlightMDPExecution.createTree(WarlightMDPExecution.java:274)

        at mdp.WarlightMDPExecution.createTree(WarlightMDPExecution.java:312)

        at mdp.WarlightMDPExecution.createTree(WarlightMDPExecution.java:312)

        at mdp.WarlightMDPExecution.createTree(WarlightMDPExecution.java:312)
```

01 **Out Of Memory** Issues due to huge search space :

Removed each state after successor is created.

Just keep leaf nodes with through actions.

02 **Timeout :** 2 seconds limit for a move.

Prune nodes after deployment.

Choose the current best, when near to timeout.

03 **Tuning Reward Function:**

Updating to match the strategies.

Limitations and Future Work

Limitations and Future Improvements

- Time out issues : Not able to complete game due to huge time complexity into the given environment.
- Need to drop some important features.
- Parallel exploration of each node would help the performance. - Multi threading approach in Java.
- Centralized I-POMDP can be a better approach then the given solution.

Thank you!
Questions?