RISK GAME

Game Structure

The game is partitioned into objects with the main one being the game board which has all the data of a game and controls the turns, it includes all the territories of the map, all players' data, and the current game state.

On starting the game, the whole board is the generated and each player is assigned $\left\lceil \frac{2*Territories\ Count}{Player\ Count} \right\rceil$ troops randomly, if the player's type is an AI, an agent in initialized for it and then plys can begin.

On the start of a players turn they are assigned new bonus troops which can be assigned to any of their territories, then they can declare attacks or end their turn.

Attacks are declared by the agent by choosing a territory of their own, an adjacent territory to attack and the number of troops to attack with.

On attack an array of three random numbers(dice) is generated for the attacking player, and same for the defending, if the attacker is attacking with more than three troops the process is repeated in 3s until finished.

The game state is a dictionary of player ids each containing dictionaries of territories they occupy and the number of troops in it, which is updated after every attack depending on its outcome.

Non-AI Agents

HUMAN:
A human agent is available using a web GUI, where you assign your troops at the beginning of your turn and then declare as many attacks as you want.

PASSIVE:
Passive agent finds the territory with the minimum number of troops and assigns all bonus troops to it and then ends its turn.

AGGRESSIVE:
Aggressive agent just goes all the way with brute force, first placing all troops in the territory with the most troops, then looping on each territory and then it's corresponding adjacent territories and tries to attack with all the troops in it if possible.

PACIFIST:
Pacifist agent places troops in its least troops territory and searches for the territory that's attackable and has the least number of troops.

AI Agents

Border Security Ratio:

Taking the summation of all troops in enemy territories y adjacent to territory x will give a measure which we call Border Security Threat (BST) in x.

$$BST_x = \sum_{y=1}^{n} AmountOfTroops_y$$

Dividing this BST by the troops situated in x gives a Border Security Ratio (BSR) which can be compared among all border territories.

$$BSR_x = \frac{BST_x}{AmountOfTroops_x}$$

Heuristic function:

h(n) = sum of BSR value of each territory and adding to it number of territories occupied by other players.

Where the BSR value is the border security ratio. By adding these values, they heuristic function underestimates the goal state of the player, by minimizing the number of territories occupied by other players and minimizing their troops while increasing his.

g(n) = 1 for reinforce phase or attack phase, 3 for end turn to give it more weight to favor attacking if 2 nodes have the same cost but one is attack and the other is end turn.

Utility function:

u(n) is given as the sum of reciprocal of the BSR value for each territory adding to it then number of territories occupied by the agent. Which gives a generic view of the state, favoring states where agent occupies more territories and have more troops than other player to defend against his attacks.

Assumptions:

- Search Tree Nodes:
    - Player: Current player the game tree is generated to reach its goal node.
    - State: Current state game when an action is taken.
    - Phase: 0 for reinforce node, 1 for attack and 2 for end turn.
    - Previous Action: Action taken to reach this node and the state is updated according to it.
    - Stochastic: Boolean value to provide whether the tree is created according to win-lose probability or just simulates the game attack by throwing dice.
    - Heuristic, Cost and Utility measures.

- Possible moves:
    - If node phase is None (root node) or end turn, reinforce children are generated by placing ALL bonus troops in each territory that has a BSR value higher than a given threshold which eliminates the generation of nodes without the need of exploring it.
    - If node phase is reinforce, attack children are generated by trying to attack all adjacent territories of agent's territories if it has a BSR value lower than a given threshold same as the reinforce phase, an end turn node is also appended to the children of an attack.
    - If there are no possible attacks after an attack node, and end turn node is returned as the only child of this node and a new turn begins with a new reinforce phase and the step is repeated until goal is reached.
    - Agents attack with all troops in a territory leaving one behind.
    - After the children are generated the attack is simulated and the state is updated accordingly.
- Minimax tree:
    - Since a turn is more than just one ply we need to maximize the utility function until the turn is ended and then minimize the following nodes until the agent turn comes again.
    - If the node's action is not end turn, we continue to create max nodes until the agents ends his turn.
- Minimax creates a new tree for each move unlike the Informed search methods which generates tree every turn which results in some inconsistency but generating the tree every move would take forever to end a single turn.
- Non-stochastic agents just simulate the attack of the real game by throwing dice and updating the state according to the outcome of the roll, while stochastic agents for every attack generate a child of each possible outcome (win the round or lose your troop) so, for example if a node is an attack node where agent attacks territory X with territory Y with N troops, N+1 nodes are generated(win 0 rounds, win 1 round,…,win N rounds) and the probability of the outcome is generated and used in the heuristic function.

GREEDY AND A*:

- Greedy and A* agents have the same structure but with different cost functions, they are implemented using the pseudo code given in the lecture.
- Greedy agent is just terribad.

- A* uses non-stochastic trees because the branching factor would be too large if stochastic is used.

REAL TIME A*:

- Real time A* is implemented using iterative deepening A* (IDA*)
- IDA* uses non-stochastic trees because the branching factor would be too large if stochastic is used.

MINIMAX:

- Minimax is implemented using the pseudo code given in the lecture with alpha beta pruning.
- Stochastic trees are generated as they perform much better, and won't slow the process down.

Performance Analysis and Evaluation

Tests were run on a 2-player game where an AI agent is playing against pacifist, aggressive and passive agents with the AI agents getting the first attack, the tables are a result of the average of 3-6 games with randomly assigned troops at the start of the game.

Greedy:

Playing Against	*Map*	*Number of Turns*
Passive	Egypt	Won in 71
Aggressive	Egypt	Won in 47
Pacifist	Egypt	Won in 52

A*:

Playing Against	*Map*	*Number of Turns*
Passive	USA/Egypt	Won in 16/Won in 22
Aggressive	USA/Egypt	Won in 15/Won in 17
Pacifist	USA/Egypt	Won in 26/Won in 16

IDA*:

Playing Against	*Map*	*Number of Turns*
Passive	USA/Egypt	Won in 30/Won in 39
Aggressive	USA/Egypt	Won in 22/Won in 13
Pacifist	USA/Egypt	Won in 25/Won in 20

Minimax:

Playing Against	*Map*	*Number of Turns*
Passive	USA/Egypt	Won in 16/Won in 25
Aggressive	USA/Egypt	Won in 18/Won in 15
Pacifist	USA/Egypt	Won in 19/ Won in 13
*A**	USA/Egypt	Won in 28/Won in 18
*IDA**	USA/Egypt	Won in 26/ Won in 14

Conclusion

Best performer is minimax as expected. Greedy is just awful with moves and time, A* and IDA* are good event excelling at sometimes and beating minimax in 20~30 turns, but overall performance is given to minimax.
For the non-AI agents, the easiest one to beat was the aggressive, the toughest was the passive and the best player was the pacifist.

REFRENCES

(1) https://project.dke.maastrichtuniversity.nl/games/files/bsc/Hahn_Bsc-paper.pdf
(2) http://cs229.stanford.edu/proj2012/LozanoBratzA-RiskyProposalDesigningARiskGamePlayingAgent.pdf
(3) https://www4.stat.ncsu.edu/~jaosborn/research/RISK.pdf
(4) https://www.cs.bgu.ac.il/~shimony/AI2004/AIass1.html