Strategy Game Programming

* Summary of the first lecture

Aydan Namdar Ghazani, 11709245

I. Frontmatter

The educational goals of this course are summarized as following:

1) understanding the Minimax algorithm and beeing able to implement it
2) understanding the Monte Carlo Tree Search and beeing able to implement it
3) have experience in implementing a game AI in a Java environment.

History:
The first occurence of a machine doing strategic plays was in the 18th century. There was a fake chess machine called "The Turk", because of a small guy sitting inside the machine and playing it. 1970s the first chess programms emerged and since 1997 many neural networks have come up with all sorts of heuristics and search strategys that started to beat experts and Grandmasters. Many of them were not based on intelligence but on number crunching.
What makes a Game?:
A game can either be a zero sum game, meaning you either win or lose or a cooperative game where you can play with multiple players together. The level of information is an important aspect of Games. It can either be fully observable or partial. Games can also include randomness or not. Move orders can be sequential or simultaneous and the time structure is either discreet or real-time. Sequential games can hardly be in real-time and simultaneous games are often discrete.

II. Algorithms

The essential problem is the branching factor which is at least exponential. Even with modern hardware there is no chance to go through all branches. There are multiple different Algorithms which find a good balance between exploration and exploitation. Exploit means we are going for options which tend to look better then others. If we do so we end up with a dilemma, because we dont know if it will pay off, since its hidden in the future. Beeing smart is not always the best choice, sometimes it is better to act stupid continously.

1) minimax:
   A dumb fast fail hard algorithm. Goes as deep as the computation allows and evaluates all nodes. If you manage to do so, you get the best path(game moves).
2) Alpha Beta pruning:
   prunes a minmax search tree for example based on alpha(max), beta(min) bounds. Reducing the tree to a more promising one. This falls in the category of Branch and Bound algorithms.
3) Smart Slow: Bandit-Based Methods
   Choose a policy and utilize Upper confidence Bounds to determine your actions.
4) Monte Carlo Tree Search:
   This algorithm is State of the Art. Monte Carlo means adding some factor of randomness to your programm. The intelligence lies in the picking of the node. In this case it means we are not uniformly exploring the branches but rather cover some more then others. It is not stable because we are not looking at the whole tree so it fails soft. Today this is combined with neural networks.
5) A* Algorithm(Dijkstra):
   Is about finding the shortest path from A to B. There are 2 known optimizations. Usually you go from one node to another. If you find along your path a cheaper one, you switch your old solution with the new one.

Usually in low branching factor trees it is fine to use Alpha Beta pruning but in high branching factors it is better to utilize MCTS

III. Heuristics

Endgame databases are more important then early game databases. Since 2020 this course is also dealing with uncerrtainty, which means limited information and elements of chance. The most common strategy is to guess what you dont know(determinization) and define all unknown parameters. But usually you dont know the outcome, since the reward is delayed.
There are multiple Heuristics which can help us to improve our possibility selection by either using a neural network, pruning the tree or choosing specific paths which have higher weights.

1) alpha go
2) selection by thompson sampling
3) selection by policy and value networks
4) ave heuristic

Databases are still very common and powerfull tool. They are state of the art but very expensive. In general we can say that it is all a ressource battle. The bigger your database the stronger your AI. Which can sometimes take years to accomplish and damages the environment.

IV. Programming

In Tuwel we can find sample projects and all informations we need to setup our project environment. We should make

our project addable to the runtime environment of the User Interface. Both engines have loading functions for new agents. We should not implement depth first search algorithms and come up with clever heuristic since its not that straight forward and boring. The Evaluation is based on a tournament of all projects. There is a short term and a long term evaluation. There will be an ELO ranking meaning every new score is based on the old score.

V. Related Topics

1) Reinforcment learning:
   Delayed rewards which are not known until the playout stage. We end up with the Exploitation-Exploration problem by sampling experiencdes states and summarizin them in one category
2) Deep learning:
   A very old idea which goes back to the 1950s. Today exepriencing a rennaissance because of the cheap hardware realizing all those expensive calculations
3) Decision theory:
   maximize my outcome in a static environment and find out what i need by sampling.
4) Game theory:
   Optimal decision making for N players represented by a strategy matrix: one axis per player
5) Simulation:
   Different agents/actors represent a player. By simulating them we can see how they do with a certain heuristic. They act independent of each other

Sources: Lecture Video https://www.youtube.com/watch?v=LZqfBmbXDcI