Exploring Multiplayer Algorithm through Chopsticks

Student Name : Dave Pierre

Student Number : 101014409

Course Code : COMP 4106

Date : Monday April 20 2020

The description of the problem domain:

The game of Chopsticks is a strategic multiplayer game that can be played with 2 or more players. They are many different variations to this game but the games details will be described in the following sentences. The game starts with each player is holding out two hands with one finger extended. The game begins when a random person performs a "hit move" where the person taps one of their opponent's hands with at least one finger that is extended. The opponent who got their hand tapped will increase the number of fingers extended by the number of fingers that the hand was tapped with. For example, player B was tapped by a hand with one finger extended then player B will add one to the number of fingers extended before the tapped. If a hand is tapped and it has more then 10 fingers then subtract 10 from that hand (roll over rule). For example, if a hand has 6 fingers extended and is tapped by a hand with 8 then that hand will have 4 fingers left .Once a hand has exactly 10 fingers extended on one hand then that hand is considered to be "dead" and that hand will have zero finger extended. A player during their turn can perform a "split move" by tapping their hands together and can rearrange the number of fingers extended for both hands. For example, player A has only one hand with 4 fingers can tap their hands together and can have 2 fingers extended on both hands. A split move can only happen if there is one hand that's alive, the number of fingers on that is an even number and the number of fingers on each hand is the same (Halvesies rule). A player is consider to be knocked out and is unable to play once both of their hand are dead. The game continues until there is one player with one hand left with at least one finger extended.

The motivation for the problem:

The state of a Chopstick game with a roll over amount of five between two people can be represented as a four digit number. The number of fingers extended on one hand is represented by a digit. There is a maximum of 625 states since each digit can be any number from 0 to 4 which are five choices and since they are four digits then by product rule 5 *4 = 625 states. However filtering out the states that occurs more then once and states that are unreachable then they are 204 states. The game of Chopsticks becomes more complex if we add more then one player to the game. They are 3337 states in a 3 player game of a rollout of 5. They are even more states in a game of 4 players of a rollout of 5 with 25000 states. Unfortunately, we cannot use the Mini-Max algorithm because that can be only used for two player games. This creates an opportunity to investigate and evaluate different multiple algorithms that are currently referenced. We would like to create a high-performance artificial intelligence player that performs as well or better than a human. The AI player should use search techniques that are fast and leads them to win most of the time.

AI techniques to be used:

This project will explore and evaluate the following search techniques Paranoid Algorithm, Max-N algorithm, Best-Reply search, and UCT Algorithm.

Design choices made:

The rules of the Chopstick game that was implemented for the project was describe in the description of problem domain section above. The game also included an addition rule of once they are 2 players remaining then the winner will be the person with the most finger extended after 200 turns have been made between them. The reason for this rule was because in some occasions when stimulating a game the AI players would play a series of moves that causes a loop and would never end the game. The state of the game is represented as a dictionary where the key is the player id number and the value is a fixed array length of 2 numbers. The first index of the fixed array length of 2 numbers represent the number of fingers extended on the left hand and the second index represent the number of fingers extended on the right hand. A terminal state is when exactly one of fixed array length of 2 numbers has at least one number that greater then 0 and all the other are zeros. The heuristic that was used for Best-Reply, Max-n and Paranoid Algorithm returns the total number of fingers extended in their hands for a particular player. The UCT, Best- Reply, Paranoid and Max-n algorithms were implemented to follow the pseudo code found in AICh04IntelGamePlaying.pdf.

Experiment

The experiment was that we generate a Chopstick game where they are four players that uses one of the multiplayer algorithm describe above. The order of the players are randomly assigned at the beginning of every game. Best-Reply, Paranoid and Max-n had a depth limit of 3. The reason we limited the search depth to be 3 is because a depth limit of 4 took to long to finish and we like the algorithm to be fast. UCT algorithm had one sec to run its algorithm. The reason for this was because we wanted UCT not to take so much computational power. The game is run until there is only on person left. To avoid players performing the same order of moves each time, whenever there was a tie it was broken randomly. After running the game for 15 iterations, the average number of nodes, the search limit and the percentage of wins were recorded into the table below.

Results of a four Chopsticks player game			
Algorithm Name	Average Node Count	Search limit	Percentage of wins out of 15
Best- Reply	2615	Depth of 3	26%
Paranoid	993	Depth of 3	13%
Max -n	1100	Depth of 3	33%
UCT- algorithm	97	1 second	26%

Discussion of Experiment :

The results of UCT, Best Reply and Max-n all perform relatively similar. The Paranoid algorithm performed the worst. A potential reason for the Paranoid algorithm performing poorly because in the game of Chopstick, a player is trying to increase their odds of winning with each turn instead of trying to minimize a particular player winning chance. This could be the reason why Max-n was the best performer. The Best Reply algorithm also performed well maybe because the maximising player will be trying to counter the worst move out of all possible move that an opponent can do. The UCT algorithm also performed well maybe because it learns the best move through random stimulation. In summary, from the result Best-Reply, Max-n, UCT algorithm would be excellent multiplayer algorithm to use for multiplayer game in chopsticks.

The possible enhancements to the study :

Some future enhancements for this study would be to try different and appropriate heuristic in order to maximize each algorithm to their full potential. When performing one of theses algorithm, a dominant task is exploring through a large number of nodes. Another enhancement to this study is to include Alpha Beta pruning technique with Best-reply and Paranoid which increase it speed of search for larger depth limits. An additional enhancement is to implementing search heuristics such as killer moves, history heuristic and move ordering.

References:

https://en.wikipedia.org/wiki/Chopsticks_(hand_game)

https://people.scs.carleton.ca/~oommen/Courses/COMP4106Winter20/AICh04IntelGamePlaying.pdf

https://webdocs.cs.ualberta.ca/~nathanst/papers/comparison_algorithms.pdf

https://dke.maastrichtuniversity.nl/m.winands/documents/BestReplySearch.pdf

Appendix of Technological

The project was coded in and will require Python 3.8 to run. The project will also need to import SQLite to be able to save the result of each search.