**CITS3001 Algorithms, Agents & Artificial Intelligence**

**Project – Love Letter**

Jaydeep Gajera – 22228003

Parthvi Sheladia – 22241757

**Literature Review**

Love Letter is a game of risk, deduction, and luck and can be played between 2–4 players. A player’s goal is to get their love letter to the Princess. Two ways through which the letter can be delivered are: giving the letter to an intermediary or directly to the princess. From a deck of sixteen cards, each player starts with only one card in hand; with each card representing an intermediary. Each intermediary corresponds to an action that can affect all players in the game in different ways. Actions consists of swapping cards with another player to eliminating a player completely from the round. The game comprises of multiple rounds and each player draws a card on their turn leading to make a choice between two cards depending on what action increases their chance of winning the round and essentially the game. Powerful cards lead to early gains, but make you a target (Brown, 2018).The game is full of uncertainty and no player have a clue of other player’s cards, hence can be referred to as a imperfect information game.

Monte Carlo Tree Search(MCTS) is a decision making algorithm that helps deal with such games. It is probabilistic and heuristic driven search which works even better when combined with other useful implementation like the classic tree search implementation. It uses the idea of building a game tree with each node representing a game state that potentially helps with making optimal decisions. The algorithm periodically exploits the best action and strategy found while also continuing to explore other alternative to see if they could replace the current best. The nodes as previously mentioned are a result of a number of simulations. The whole process can be broken down into four distinct steps: selection, expansion, simulation and backpropagation. The MCTS algorithm relies on two fundamental concepts: 1)The expected reward of an action can be estimated doing many random simulations. 2) These rewards can be used to adjust the search toward a best-first strategy (Roy, 2019).

Perfect Information Monte Carlo Search with the Determinized Upper Confidence Bounds Applied to Trees is based on the idea to build a tree in an incremental and asymmetric manner by doing many random simulated games. Mańdziuk (2018) highlights the main purpose of this algorithm is to find a balance between the exploration of the less frequently simulated nodes and the exploitation of the already chosen best ones. To explain the whole process in terms of the four steps of MCTS, once a node is selected, expansion occurs by choosing an available move and adding it to the tree. Once that is done, by running a simulation on the child node and backpropagating, a result is achieved. A simulation is based on a default policy in the algorithm and the result is calculated using the UCB formula. By continuously repeating these steps, the result values are updated. This algorithm is beneficial as no prior knowledge is required for the agent(knowledge of game rules is enough). Along with this, a determinization is a conversion of a stochastic game with imperfect information to a deterministic game with perfect information, in which the hidden information and the outcomes of all future chance events are fixed and known (Lanzi and Palma, 2014).

Another useful search based algorithm that can deal with imperfect information games is an extension of MCTS called the Information Set MCTS(ISMCTS) (Mańdziuk, 2018). In this algorithm, rather than the node being represented by the state of the game/player, it is now represented by information sets. The information set is from the root player’s point of view and corresponds to moves player by the corresponding player.

**Selected Techniques**

* **Knowledge Based Algorithm**
* **Monte Carlo Tree Search (MCTS) Algorithm**

As previously mentioned, MCTS is quite an advantageous algorithm to implement for a game with hidden information and uncertainty. Baeldung (2019) highlights the fact that MCTS is efficient in open-ended environments with an enormous amount of possibilities. We chose to work with this algorithm as it is a feasible algorithm with no prior requirements or brute forcing of every possible action. The four steps of the algorithm make the algorithm work in a unique way. The usual sequence is selection, expansion, simulation and backpropagation. Although in certain situations, the sequence is subject to change. In comparison to the knowledge based approach, this algorithm does not based it’s decision just on the game rules and random guesses. In fact, it simulates random plays(based on the number of iteration given) and calculates and updates the winning likelihood of any given state which then helps choose the optimal action.

MCTS with determinization is able to handle imperfect information games much better than the basic MCTS algorithm. With determinization, probabilities of different possible states based on the information about the current game state of the player(also known as the player’s information set) is calculated. Then next steps are the same as the basic MCTS. the best moves in each state are selected assuming that players can see the entire state for the rest of the game, and it therefore it doesn’t account well for the effects of information being hidden(GitHub, 2017). And as cited in the Monte Carlo Tree Search for Love Letter paper, Frank and Basin (1998) discovered two problems of determinization, which are strategy fusion and non-locality. Strategy fusion is where the agent assumes that different decision can be made for different determinizations on the same information set and Non-locality is where certain states are consider even though they are very unlikely to be chosen. These don’t limit the results of the algorithm. The algorithm will still perform well compared to a basic MCTS.

Information Set MCTS is also a useful approach and the only difference is that it accounts for information sets as the node values instead of the states.

**Implementation**

* **Knowledge Based Algorithm**
* **Monte Carlo Tree Search (Basic MCTS) Algorithm**

Baeldung (2019) explains the basic implementation of the MCTS algorithm. Firstly, selection helps with finding the optimal child nodes (the current state is the root node) until there can be no further exploration. Using the Upper Bound Confidence applied to trees formula make the selection easier. Once UCT cannot be applied, expansion of the game tree occurs and all the possible states are added to the leaf node. And as soon as that is done, simulation of random plays on the chosen child node occurs. Simulation continues until the desired game state is achieved. Once the previous three steps are completed, the last steps comes in and updates the state values by traversing back to the root node. These steps are repeated until the given number of iterations.

**Validation**

Firstly, to test the performance of our knowledge based agent, we ran a tournament between three RandomAgents and our knowledge based agent. The tournament included 100,000 games of Love Letter with knowledge based agent playing from different position in each tournament.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Pos 1 Wins | Pos 1 Win Rate | Pos 2 Wins | Pos 2 Win Rate | Pos 3 Wins | Pos 3 Win Rate(%) | Pos 4 Wins | Pos 4 Win rate(%) |
| KBAgent | 50660 | 50.660004 | 49260 | 49.26 | 57159 | 57.159 | 51182 | 51.182003 |

**References**

* Brown, J. (2018). *Monte Carlo Tree Search for Love Letter*. [online] ResearchGate. Available at: <https://www.researchgate.net/publication/327679828_Monte_Carlo_Tree_Search_for_Love_Letter> [Accessed 16 October 2019].
* GitHub. (2018). *eugenp/tutorials*. [online] Available at: <https://github.com/eugenp/tutorials/tree/master/algorithms-miscellaneous-1/src/main/java/com/baeldung/algorithms/mcts/montecarlo> [Accessed 18 October 2019].
* GitHub. (2019). *eugenp/tutorials*. [online] Available at: <https://github.com/eugenp/tutorials/tree/master/algorithms-miscellaneous-1/src/main/java/com/baeldung/algorithms/mcts/tree> [Accessed 18 October 2019].
* GitHub. (2018). *Raptortech-js/love-letter-AI*. [online] Available at:

<https://github.com/raptortech-js/love-letter-AI> [Accessed 18 October 2019]

* Lanzi, P. and Palma, S. (2014). *Monte Carlo Tree Search algorithms applied to the card game Scopone*. [ebook] The University of Western Australia. Available at: <http://teaching.csse.uwa.edu.au/units/CITS3001/project/2017/paper1.pdf> [Accessed 15 October 2019].
* Mańdziuk, J. (2018). *MCTS/UCT in solving real-life problems*. [online] ResearchGate. Available at: <https://www.researchgate.net/publication/320003615_MCTSUCT_in_solving_real-life_problems> [Accessed 12 October 2019].
* Monte Carlo Tree Search. (2016). [ebook] Swarthmore College. Available at: <https://www.cs.swarthmore.edu/~bryce/cs63/s16/slides/2-15_MCTS.pdf> [Accessed 17 October 2019].
* Baeldung. (2019). *Monte Carlo Tree Search for Tic-Tac-Toe Game | Baeldung*. [online] Available at: <https://www.baeldung.com/java-monte-carlo-tree-search> [Accessed 18 October 2019].
* Roy, R. (2019). *ML | Monte Carlo Tree Search (MCTS) - GeeksforGeeks*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/ml-monte-carlo-tree-search-mcts/> [Accessed 16 Oct. 2019].