Monte Carlo Tree Search for Love Letter

Conference Paper · September 2018

CITATIONS
0

READS
250

4 authors, including:

Joseph Alexander Brown
Innopolis University
94 PUBLICATIONS **227** CITATIONS

SEE PROFILE

Monte Carlo Tree Search for Love Letter

Tamirlan Omarov, Hamna Aslam, Joseph Alexander Brown, Elizabeth Reading
Artificial Intelligence in Games Development Lab, Innopolis University,
Innopolis, Republic of Tatarstan, Russia, 420500
email: t.omarov@innopolis.ru, h.aslam@innopolis.ru, j.brown@innopolis.ru, beth_reading@hotmail.com

KEYWORDS
monte carlo tree search, imperfect information games, determinization, minimax

ABSTRACT

Love Letter is a card game for two to four players. Players maintain one card in their hands at all times. For a series of rounds, a player draws a card and then plays one of the two cards in their hand with a goal of knocking out other players from the game by a card effect or by having the highest card at the end of the deck when all the cards have been played. We examined the use of Single Observer Information Set Monte Carlo Tree Search (SO-ISMCTS) in a player agent for a two-player Love Letter, compared against a knowledge based agent, Perfect Information Monte Carlo (PIMC) and Determinized Minimax and found that none of the algorithm could show the stable results due to game randomness.

Introduction

The research work presented in this paper, compares some of the existing MCTS methods (Browne et al. 2012) for imperfect information games(such as games with hidden information and uncertainty property) and traditional approaches such as knowledge based agent, Perfect Information Monte Carlo (PIMC) algorithm, Single Observer-Information Set Monte Carlo Tree Search (SO-ISMCTS) and Determinized Minimax. These algorithms have been applied to develop an AI player for the Love Letter game. Love Letter is an *imperfect information game* and players can not observe their opponents' cards and do not know the particular state of other players. The SO-ISMCTS algorithm is the main AI agent for which the rest of the three algorithms are playing as opponents. We have selected SO-ISMCTS as the main algorithm for comparison for its ability to search a single tree for games of imperfect information, it can be adopted for multi-player games, it has shown good results for more complex card games, it makes 4000 iterations during 2 seconds and can be used in games against human opponent. Also, SO-ISMCTS searches more deeply than PIMC with the same computational budget (Cowling et al. 2012), as Love Letter game has $\frac{16!}{5!2!2!2!2!} = 21794572800$ different initial states. These properties are essential to implement AI agent for the Love Letter game.

Love Letter

Love Letter (Kanai 2012) is a card game for two to four players. The game includes sixteen cards and thirteen red cube tokens. Each of the cards have a rank and text which explains its effect, see table 1. The game play starts by shuffling the cards and forming a face down draw deck. Players have a hand size of one and draw one card during their turn. The player must play one of his two cards and follow the effect of the card which is written over the card. Some of the cards have effects which remove a player from the round.
There are four win conditions in two player Love Letter. A player discards the Princess, so the other player wins. A player correctly guesses their opponent's hand using the Guard. A player has the higher card when the Baron is played, or when there is no deck remaining. In each round of the game, winner gets a token. In the event of a tie, the winner is the one who played the highest value of cards. The player with the most tokens of the thirteen rounds wins the game.
In the beginning of every round one card is removed from a deck that can be used if players run out of cards in the deck. We refer to it as an *out card* in this paper. In 2 player games, three more cards are removed from the deck and placed all face up.
In this paper we assume that playing the first move does not give any advantages. The maximum length of the game for 2 player version consists of 10 moves. So the depth of the game tree for all search tree based algorithms used in this work is 10 too.

Implementation

We have implemented four AI agents for the game Love Letter based on Single Observer Information Set Monte Carlo Tree Search (SO-ISMCTS) Algorithm, Perfect Information Monte Carlo (PIMC), Knowledge based agent and Determinized Minimax. The vertex of the game tree built, corresponds to a player's current hand and edges correspond to moves. Players are assumed to play rationally, i.e if a player can make a move and win the game, he has to make this move. For SO-ISMCTS and

Table 1: List of Cards - with some edits of rules text for clarity based off Kanai (2012)

Value	Name	Amount	Text
8	Princess	1	Player loses if this card is discarded or played
7	Countess	1	Player Must Discard this Card if the other card in hand is a King or Prince
6	King	1	Players trade hands
5	Prince	2	One player at the table *must* discard his hand
4	Handmaid	2	Player is protected from all card text until next turn
3	Baron	2	Player compares hands with another player; The player with the card of lower value loses
2	Priest	2	Player must look at another players hand
1	Guard	5	Player guesses another player's hand, if that guess is correct then that player loses

Player A cards: Guard, Countess
Player B card: Priest
Non played deck: Handmaid, Prince
Out card: Princess

Figure 1: Original Game State (assuming that players have already made some turns)

Player A cards: Guard, Countess
Player B card: Priest
Non played deck: Handmaid, Princess
Out card: Prince

Figure 2: Game State after Determinization

PIMC algorithms, the first step is the same. All of them are sampling the *determinization* randomly from *information set*. Where ***informations sets*** are collections of states, which appear in the game when some players have information about the state that others do not. For example, in Poker each player hides card from his opponents. Here information set consists of all possible permutations of states which corresponds to opponent's cards. A player knows his own information set, but he does not know particular state within this set. The ***determinization*** involves selecting the current state from player's information set and then all future events are fixed and known. In other words, it is the process of converting an imperfect information game to an instance of perfect information game(such as, a game in which all information about the state of the game is observable). For example, in case of a card game, determinization is the instance when all players' cards and deck are visible to all players.
In the next paragraphs algorithms for two player version will be described. The Player A has two cards in hand, one is a Guard and the other is the Countess. The player B has Priest card. The non-played deck consists of two cards, Handmaid and the Prince card. The out card is the Princess. This example will be used to describe steps of the algorithms. The state is shown in the figure 1.

Knowledge based agent

Knowledge based agent enforces game moves in accordance to the actual game rules. Secondly the algorithm checks for the game move that can lead to an immediate win in single turn. For instance, in our example (see Fig.2), player 'A' could win the round using the Guard's ability, provided player 'A' can correctly guess their opponent's card (see Table.1 for card's abilities). The knowledge based agent handles the selection of the guess card as: the card which is encountered the minimum number of times in the played deck is chosen. In Love Letter, there are predefined number of each type of card therefore the algorithm can deduce the number of non-played cards for each character in the game. The rule based approach prefers to make a move with the smallest valued card in the hand in order to save the highest one (each character card has a unique value). These set of rules and checks is called *domain knowledge*. The domain knowledge is reused in other three algorithms described further in the paper. For our example, the Guard has been selected by the knowledge based agent. The knowledge based agent algorithm does not need any iterations and it runs faster than the rest of the algorithms with the time complexity O(1). The full set of checks and conditions can be found in the code repository [1].

Perfect Information Monte Carlo (PIMC) algorithm

The first step in PIMC involve finding moves that can lead to an immediate win (as mentioned in the knowledge based algorithm). Then determinization (defined in Implementation section) is done and algorithm removes played deck from the full deck. After that, it deals card to the opponent randomly and shuffle the remaining cards. Thus, it has sampled one state from information set. One of the possible new states after determinization is shown in figure 2. Currently, the game tree consists of single vertex which is a root node only. Then Monte Carlo Tree Search (MCTS) steps are

[1] https://github.com/tamirOK/ISMCTS-for-love-letter-game

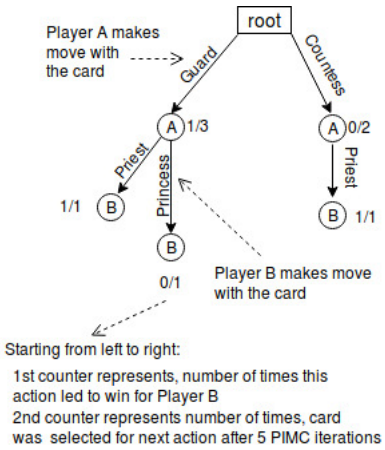

Figure 3: Game Tree after 5 MCTS iterations on Determinized game state

applied on this tree.

The first step is selection in which the algorithm chooses move according to UCB1 formula (Auer et al. 2002):

$$\bar{x}_i + c\sqrt{\frac{\ln n}{n_i}} \tag{1}$$

where $\bar{\boldsymbol{x}}_{\boldsymbol{i}}$ is average reward passing through the node $\boldsymbol{i}$. It is calculated as ratio of number of won simulations passed through the node to number of all simulations passed through the node. $\boldsymbol{n_i}$ is a number of times the node $\boldsymbol{i}$ was selected from its parent, $\boldsymbol{c}$ is a constant, that controls the rate of exploitation and exploration while traversing the tree nodes and $\boldsymbol{n}$ is number of times parent node was visited. When using UCB1, it is important to select optimal $\boldsymbol{c}$ because it affects playing strength. The value depends on the applied domain and the MCTS algorithm used. (Cowling et al. 2012) conducted an experiment where the PIMC and SO-ISMCTS players with exploration constant {0.25, 0.5, 0.75, ..., 1.75, 2} played repeatedly against PIMC with exploration constant 0,7. Performance of the algorithms decreased outside the range [0.5, 1], but inside the range none of the algorithms were particularly sensitive to the coefficient value. Thus, value of 0.7 has been used for all the algorithms in this work.
After **five** MCTS iterations on determinized game state, the built game tree is shown in the figure 3. MCTS steps will be explained using this game tree.
The MCTS consists of 4 stages, selection, expansion, simulation, backpropagation. The selection happens until algorithm encounters a node in the game tree which has unexplored moves. In our example algorithm will select a move with the Guard for the player A, then a move with the Priest for the player B and stop in that state. At the expansion state a child node which is not a part of the game tree yet is added to the tree. After that, at simulation step, game is played till the end and no new node is appended to the game tree. In our example player B won the simulated game. Then, at backpropagation step, starting from the leaf node of the game tree, win counter is updated for every node on the path to root where player B made move. Also, the visit count for every node in the path is updated.
At this point single MCTS iteration is completed. After all MCTS iterations, a counter for the move which was visited most is increased. Then determinization step is completed.
After all determinizations are completed, the root's child which was selected highest number of times is returned by the PIMC algorithm. If this move is Guard, algorithm also selects the guess card as in knowledge based agent (explained in previous section) .
It is important to select balanced number of determinization and MCTS iterations for single game tree. (Cowling et al. 2012) and (Powley et al. 2011) applied the PIMC for the Dou Di Zhu card game and found that as long as both parameters are sufficiently large, their precise values do not have a significant impact on playing strength. They used 40 determinizations with 250 MCTS iterations. In our implementation we have used 50 determinizations with 160 MCTS iterations.
This method has several disadvantages. Russel and Norvig (Russell et al. 2003) point out that determinization will never decide to make information gathering play (i.e. move which causes opponent to reveal his cards) or information hiding play (i.e. move that avoids revealing current player's hidden information). (Frank and Basin 1998) found two main problems of determinization:

- *Strategy fusion*: An AI agent incorrectly assumes that it can make different decisions from different determinizations in the same information set. For example, consider a game where card is taken from the Love Letter deck and put face down on the table. The player has two options: get 0.4 points without guessing the taken card or get 1 point if his guess was correct or 0 otherwise. The first choice's expected reward is 0.4. The second choice's reward is 0.125. So the first option is better. However, if PIMC is applied to this game, then in each determinization the card will be known and expected reward will be 1 point.

- *Non-locality*: Some states are very unlikely because other players will play away from the corresponding states. However, PIMC considers them too. For example, if player plays rationally, knows the opponent's card and could have played the Guard in order to win, he will play the Guard. However, in PIMC determinizations in which player holds the Guard will be considered. This case contradicts assumption of rationality.

Despite these problems many researchers successfully

applied it in different domains. Ginsberg in (Ginsberg 2001) applied determinization to create AI for Bridge game which plays at human expert level. Bjarnson (Bjarnason et al. 2009) used this approach for the single-player card game Klondike Solitaire.

Single Observer-Information Set Monte Carlo Tree Search (SO-ISMCTS) algorithm

In SO-ISMCTS set of available actions from some node varies between visits to that node. For example, in our implementation a new deck is generated on every determinization and set of the available moves depends on the deck. Thus, a node in the game tree has branches to every legal move that was available from this state at some moment, but availability of a branch depends on current determinization. It is a multiarmed bandit (Auer et al. 2002), in which only subset of the arms are available on each trial. In the work of Cowling et al. (2012), it is called a *subset-armed bandit*. They replace n in formula 1 with the number of iterations in which the parent was visited and *node i was available*. This modification prevents over exploration of rare actions, i.e. actions which are available in few states of the information set. If every state in an information set has rare action, search will perform almost no exploitation and almost all exploration.
In SO-ISMCTS states are grouped together into information sets. It is improvement upon previous approach, designed not to suffer from strategy fusion (Whitehouse 2014). To solve the problem of learning in the context of specific states, SO-ISMTCS learns values in the context of information set instead. Thus, it is a tree where every node is information set and knowledge is shared during the learning between different determinizations. This approach does not need to find correct values for determinizations and simulations as in PIMC. The algorithm uses a new determinization on every iteration in order to take average value in final decision step. However, usually there is large number of states in the information set. For example, on the first move of the game there are 14! determinizations in the information set. Thus, it is infeasible to iterate over all determinizations in the information sets.
This algorithm is very similar to the PIMC approach except that is uses modified UCB1 formula in selection step and shares learned information during iterations. Therefore, SO-ISMCTS algorithm performs more iterations and greater playing strength is expected from it.
We will consider the determinization on the figure 2 example. After ten SO-ISMCTS iterations, Guard card has been selected nine times to be played by player A and it led to win six times. The Countess was played one time and Player A lost. This game tree is shown on the figure 4.

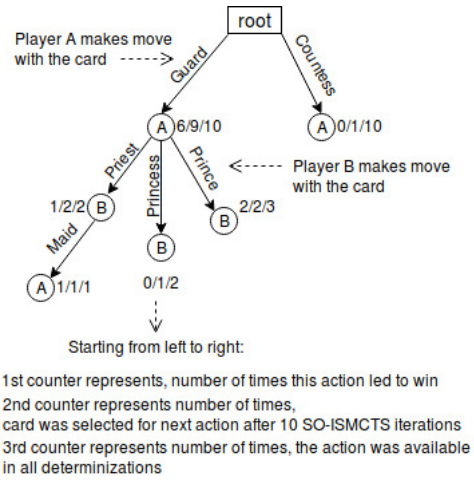

Figure 4: SO-ISMCTS Game Tree after 10 Iterations

Determinized Minimax

We have applied the Determinized Minimax approach as in Ginsberg (2001). Particularly, after each determinization, the vanilla minimax algorithm (Russell et al. 2003) is applied. After all determinizations, the move which was returned the most number of times by minimax is selected. There are 100 determinizations applied in author's implementation.
After determinization, the game state is fixed. That is, the order of cards in the deck and opponent's card will be the same. It means that algorithm can build the whole game tree. Because at each move, there are only 2 possible moves at most and 10 turns in the worst cases, the game tree will contain $O(2^{10})$ nodes and it is feasible to use full search here.

Minimax

Considering game state after determinization in the figure 2. In the minimax algorithm, there are two players, Max and Min, who make alternating moves. The Max players starts the game and authors assume that both players play optimally, that is they prefer the best possible result. The Max player tries to maximize own reward and Min player tries to minimize the Max player's result, that is to maximize his own gain. In other words, the Max player tries both moves and selects one with maximum value and Min player selects one with minimal value.
The algorithm starts from the terminal nodes where minimax values are known and then goes upwards and calculates a node's minimax value depending on the current player. After the whole tree is built, the algorithm selects the move which corresponds to the root's child node with maximum minimax value. In our case both moves have the same value, so algorithm will return any of them. The minimax algorithm performs complete ex-

ploration of the game tree. Let m be the maximal depth of the tree and b be a branching factor of each node, then the time complexity of the algorithm is $O(b^m)$ and space complexity is $O(bm)$. As it was mentioned above, the time complexity is reasonable in our case and a version of the algorithm without any optimizations can be applied here.

Experiments

In this section we will compare SO-ISMCTS with other techniques. Sometimes random deals have a significant effect on the outcome of the game. For example, at dealing stage, a player can get Princess and Baron cards and win immediately in 2-player game. To make a fair comparison between algorithms and reduce the variance of the results, we have extracted a set of 36 decks which does not give any advantages in the beginning of the round and used them in all experiments. The practice of specifying deck ordering in advance is common in Bridge tournaments between human players, to minimize the effect of luck when comparing players (Powley et al. 2011).
In comparison of two player version binomial distribution is assumed and error bars show 95% binomial proportion confidence interval. In all experiment there were 200 games played for each number of iterations. The first move is done by SO-ISMCTS algorithm. First moves on the successive rounds are done by the algorithm which won the previous one. All algorithms are implemented using Python 2 programming language. All implementations can be found in code repository [2]. First of all, we need to determine a number of iterations for SO-ISMCTS. To do this we have run a series of games against the same technique where first version played using 1000 iterations and number of iterations for the second version is varied. The results of the experiment is shown in the figure 5.

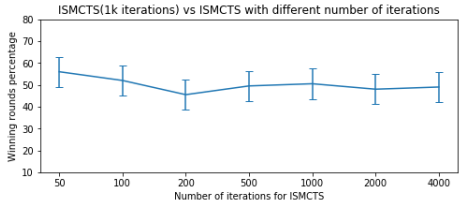

Figure 5: Winning Percentage for SO-ISMCTS with Varied Number of Iterations

It is shown that SO-ISMCTS with 1000 iterations shows almost the same performance as SO-ISMCTS with any other number of iterations . We will use 1000 iterations for SO-ISMCTS in all experiments for convenience.

[2]https://github.com/tamirOK/ISMCTS-for-love-letter-game

SO-ISMCTS vs PIMC

This section presents a comparison of SO-ISMCTS and PIMC. The number of iterations for PIMC will be varied. As it was mentioned in the PIMC section, we will use 50 determinizations. It means that if there are 1000 iterations, there will be 50 game trees built and for each tree there will be 20 MCTS iterations. Results of the experiment are shown in the figure 6.

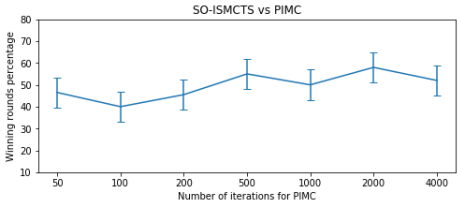

Figure 6: Winning Percentage for PIMC with Varied Number of Iterations

It is shown that initially SO-ISMCTS performs better. However, at 500 iterations PIMC wins approximately in 55% of cases, continues to win till the end of experiment and shows maximal performance of 58% with 2000 iterations.

SO-ISMCTS vs Determinized Minimax

This section presents a comparison of SO-ISMCTS and Determinized Minimax. It was mentioned that for each determinization, minimax builds a whole game tree and then traverse it. For this reason its performance is slower compared to the previous algorithms, so we decreased the number of iterations for the Determinized Minimax in our experiments. The results are shown below in figure 7.

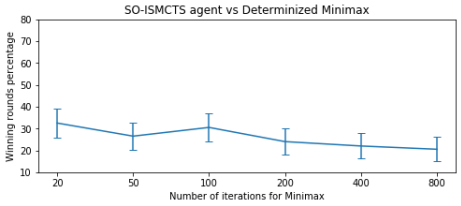

Figure 7: Winning Percentage for Determinized Minimax

It is observed that Determinized Minimax performance is low. The results provide interesting evidence that the classical approach for the perfect information game does not show good results for the Love Letter game.

SO-ISMCTS vs Knowledge based agent

In this section SO-ISMCTS is compared to knowledge based agent. Knowledge based agent does not have any

simulations and always plays the same move for the fixed game state. For this reason we changed number of iterations for the SO-ISMCTS approach. Results of the experiment are shown in the figure 8.

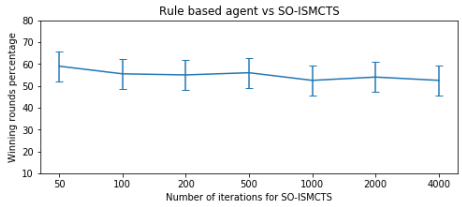

Figure 8: Winning Percentage for SO-ISMCTS with Varied Number of Iterations

It is seen that SO-ISMCTS shows better performance than knowledge based agent. However, its playing strength does not significantly outperform the knowledge based agent. The experiment shows that the game is stochastic and even simple deterministic approaches like knowledge based can win in half of simulations.

Conclusions

This paper presents an AI player for the game Love Letter. The AI player has been implemented using four algorithms among which SO-ISMCTS AI player's performance has been compared to knowledge based, PIMC and Determinized Minimax player. We have found that the playing strength is not statistically effected by the number of iterations played in Love Letter game. The results of the experiments have revealed SO-ISMCTS played much better than Determinized Minimax, slightly outperformed the knowledge based agent and plays a little weaker than PIMC. We can also conclude that the game is random and none of the algorithms could manage to take advantage of the randomness for the improved performance. We found out that PIMC and SO-ISMCTS agents cannot find the best move for every possible determinizations in the information set. This problem can possibly be minimized by increasing number of iterations for SO-ISMCTS, however it will increase running time of the algorithm.
Implementing the algorithm which will handle the specifics of the game, applying classical AI approaches like expectimax and use inferencing technique is a subject for future work. After solving this issue, the next step is to extend the algorithm to the multiplayer version and compare its performance to human players via human competitive testing. The implementation of the algorithm can support multi player version, the issue is opponent selection. Player selection can be done randomly or selecting the player who won maximum rounds so far or choosing the immediate player on your right or left etc. Another approach to improve the playing strength is to vary exploration-exploitation constant, number of iterations and adding more rules to the knowledge based agent.

REFERENCES

Auer P.; Cesa-Bianchi N.; and Fischer P., 2002. *Finite-time analysis of the multiarmed bandit problem. Machine learning*, 47, no. 2-3, 235–256.

Bjarnason R.; Fern A.; and Tadepalli P., 2009. *Lower Bounding Klondike Solitaire with Monte-Carlo Planning.* In *ICAPS*.

Browne C.B.; Powley E.; Whitehouse D.; Lucas S.M.; Cowling P.I.; Rohlfshagen P.; Tavener S.; Perez D.; Samothrakis S.; and Colton S., 2012. *A survey of monte carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in games*, 4, no. 1, 1–43.

Cowling P.I.; Powley E.J.; and Whitehouse D., 2012. *Information set monte carlo tree search. IEEE Transactions on Computational Intelligence and AI in Games*, 4, no. 2, 120–143.

Frank I. and Basin D., 1998. *Search in games with incomplete information: A case study using bridge card play. Artificial Intelligence*, 100, no. 1-2, 87–123.

Ginsberg M.L., 2001. *GIB: Imperfect information in a computationally challenging game. Journal of Artificial Intelligence Research*, 14, 303–358.

Kanai S., 2012. *Love Letter : Tempest Edition.* Alderac Entertainment Group.

Powley E.J.; Whitehouse D.; and Cowling P.I., 2011. *Determinization in Monte-Carlo tree search for the card game Dou Di Zhu. Proc Artif Intell Simul Behav*, 17–24.

Russell S.J.; Norvig P.; Canny J.F.; Malik J.M.; and Edwards D.D., 2003. *Artificial intelligence: a modern approach*, vol. 2. Prentice hall Upper Saddle River.

Whitehouse D., 2014. *Monte Carlo Tree Search for games with Hidden Information and Uncertainty.* Ph.D. thesis, University of York.