Chain Reaction Artificial Intelligence Model

Jayant Dev Srivastava[1] and Param Shukla[2]

1.Amity School of Engineering and Technology, Amity University Noida, Sector-125, Uttar Pradesh, Noida, India.

2.Amity School of Engineering and Technology, Amity University Noida, Sector-125, Uttar Pradesh, Noida, India.

Abstract

As many of us play Zero Sum Board games like Chess,tic-tac-toe,Chain Reaction etc,so we note that between Human and AI,Generally Human wins over AI.The reason is because of the dumb and lame strategies or moves played by the AI of the game because of wrong decision making.So we believe that Board games are amazing examples of decision making under uncertainty. In specific, a few games have such an immense state space and high level of uncertainty that conventional algorithms and strategies struggle to play them effectively.The aim of this paper is to put forward Monte-Carlo Tree Search Algorithm as a unified framework to a Board game's AI. In the modeled framework, randomized explorations of the search space are utilized to anticipate the most promising game moves.The approach allows to exploit the strengths and overcome the weaknesses of the given AI and facilitates the creation of a superior AI in a service-oriented fashion.All these AIs aim to select the best possible move in a given situation, i.e. for a given board configuration by predicting the next moves of the human player and creating its hashmap. In Chain Reaction the most vital part of the game is to find the best position to make a corresponding move on the board at that position. Hence, Our artificial intelligence model for the game of a board game would first need to identify the best position, predict the opponent's future moves on the basis of previous moves select the optimal move and then make the move at that location.Particularly, we want to explore the scope of Monte Carlo Tree Search in Zero Sum Board Games like Chess,Chain Reaction,Tic-Tac Toe.We have used Chain Reaction as a basis for our Research. Based on the concepts of Model Based on the concepts of Monte Carlo Tree Search, a conceptual game model is developed in a Live Access Server extension of Visual Studio Code i.e a web browser extension to live reload and designed to provide access to the uploaded HTML/JavaScript/CSS/ ES6 files where Monte Carlo Tree Search Algorithm has been implemented for the game's Artificial Intelligence.This research concludes a proof of practical concept that is to believed to be very useful for Zero-Sum Board Games.

Keywords: Zero sum Board games - Chain Reaction - Artificial Intelligence - Monte Carlo Tree Search.

Introduction

Zero Sum Board game is one in which no wealth is made or pulverized. In this way, in a two-player Zero Sum Board game, whatever one player wins, the difference loses. Accordingly, the player shares no basic interests. There are two general kinds of Zero Sum Board games: those with flawless data and those without.

In a game with immaculate data, each player knows the aftereffects of every past move. Such games incorporate Chain-Reaction, Alpha Go, chess, tic-tac-toe, and Nim. In rounds of flawless data, there is at any rate one "best" approach to play for every player. This best procedure doesn't really permit him to win however will limit his misfortunes. For example, in tic-tac-toe, there is a methodology that will permit you to never lose, however there is no technique that will permit you to consistently win. Despite the fact that there is an ideal methodology, it isn't constantly feasible for players to discover it. For example, chess is a Zero Sum Board game with flawless data, yet the quantity of potential procedures is enormous to the point that it isn't workable for our PCs to decide the best system.

Chess

Chess is a Board game played by two players, who we'll call White and Black. It is played on a board of 64 squares. Each square can be empty or occupied by a piece. The initial position of the game consists of 16 white pieces and 16 black pieces.Players alternate making moves. White always goes first. In a typical move, White selects a white piece and moves it to another square. Chess is generally considered to be a zero sum game.This depends on how we assign a value to a draw. By convention, a draw is considered to be an intermediate outcome halfway between a win and a loss. This is how most tournament aggregation rules function. Nevertheless, you could have a particular game of chess in which one or both players subjectively did not value draws this way.This particular chess match would no longer be a zero-sum game.When one is discussing general chess strategy, or designing an AI for chess, it is entirely reasonable to treat chess as a zero sum game. The same applies for all two-player board games.

Tic-Tac-toe

Tic-tac-toe (also referred to as noughts and crosses or Xs and Os) is a paper-and-pencil game for two players, X and O, who take turns marking the areas in a three×three grid. The participant who succeeds in placing 3 of their marks in a horizontal, vertical, or diagonal row wins the game.

In order to remedy Tic Tac Toe, we need to move deeper than simply to think about it as a recreation where players region X's and O's at the board. Formally speaking, Tic Tac Toe is a zero-sum and perfect statistics gChain-Reaction game- is a Zero-Sum Board game.

Chain Reaction

ChainReaction is a tabletop game played by two players on a 6x5 field who, thusly, place purported iotas. Every area on the board, a cell, can hold all things considered 2, 3, or 4 molecules, contingent upon its area. At the point when the most extreme number of iotas is come to, the phone 'detonates' and appropriates its molecules to its neighbor cells. With enough iotas on a board such a response can without much of a stretch redistribute molecules over the whole board, thus the name, ChainReaction. A game where all successes by at least one player are coordinated by misfortunes of different players is known as a lose-lose situation.ChainReaction is a deterministic combinatorial dilemma game.The main body of a ChainReaction game has 2 noteworthy segments. The first are the cells. The blockade may be isolated into mxn cells where each cell speaks to a position where a move can be made. The accompanying noteworthy part of the board are the particles. Particles are used to check the region at which a move is made by a player on the board.

The rules of the games are according to the accompanying: The continuous cooperation occurs in a m×n board. The most normally used size of the board is 6x5.For each cell in the board, we describe a base sum. The base sum is comparable to the amount of evenly close cells. That would be 4 for standard cells, 3 for cells in the edge and 2 for cells in the corner. All cells are from the start unfilled.The Red and the blue players interchange to put "particles" of their related tones. The Red player can simply put a (red) molecule in an empty cell or a cell which starting at now contains at any rate one red particle.

Role of Artificial Intelligence in Zero-Sum Board games.

In the beyond decades, Artificial intelligence (AI) has been dominating the arena of two-player board games. In 1997, the very first laptop to overcome Garry Kasparov, the reigning global chess champion then, turned into IBM's Deep Blue. The victory bowled over the world; Kasparov even desired a rematch but IBM declined the idea.

Then, AlphaGo of DeepMind beat Lee Sedol, one of the global's top Go players, in March 2016 by means of a score of 4–1. Later that year, a mysterious on-line Go participant named "Master" stored on defeating the arena's first-class Go players in immediate weeks. The mystery player won 50 out of fifty one video games. And the 51st game wasn't even a loss however a draw due to connection problems. It was officially shown that the "Master" turned into, in fact, the brand new model of AlphaGo.

A sport is deterministic if there is no element of randomness. The next events have to be perfectly predictable given the understanding of the previous sport states and the player's actions. Zero-sum games contain pure competition. A player's loss is some other participant's benefit and vice versa. One participant tries to maximize one single fee, while the other

tries to decrease it. A sport offers ideal information if every player knows all of the occasions that formerly occurred. For example, every player in chess can see all the pieces at the board in any respect. The perfect examples of deterministic, -participant, zero-sum, and best-records games are chess, checkers, tic-tac-toe, and Go.The modern-day contemporary implementations of two-participant games are complex. But all involve simple blind computations. The manner of selection-making makes use of recreation bushes where every element represents something. Tree nodes represent game states (e.G. function of portions on the board for chess); root nodes are present day states, at the same time as leaf nodes are very last states (win, loss, or draw). Meanwhile, edges constitute the player's movements, and layers mirror the opportunity actions. To simulate the decision to locate the most effective flow for the player, we need an algorithm that could calculate all possible moves available for the laptop participant, and use metrics to decide the fine feasible choice. And for that, Minimax will become the correct solution. The algorithm has two actors: the maximizer and the minimizer. Maximizers pick out the node with the highest score, while minimizers choose the lowest rating. Below is a pattern example of the Minimax algorithm.

To visualize, we can use tic-tac-toe as an example. X-participant is the maximizer even as A-participant is the minimizer. The image underneath shows that A-participant has only viable moves left. If A-participant places the mark on the left container, O-participant wins and the evaluation rating could be -10. But if O-participant puts the mark at the proper box, gamers change will exchange turns, and X-player will mark the last area and win the sport. O-participant decided on the container on the left since it'll generate a -10 score. Choosing the container on the right will give it a +10 score.

It is interesting to realize how AI strategizes and makes decisions. We can say that AI's intellect is in its potential to foresee and consider multiple feasible outcomes and pick out the best amongst them all. This may be extremely helpful if one is trying to build an unbeatable AI player.

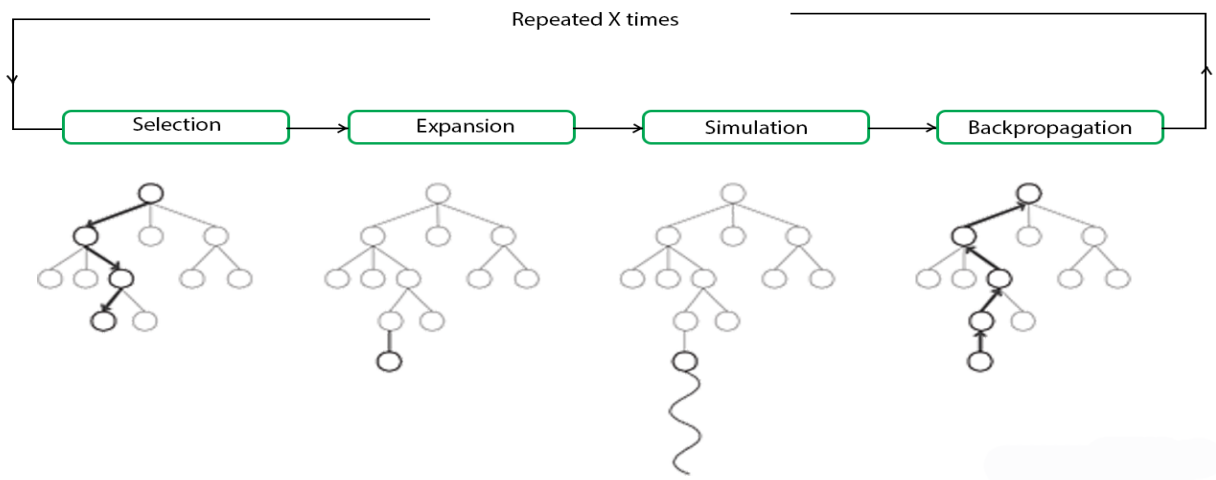
Monte Carlo Tree Search (MCTS) algorithm:

In MCTS, nodes are the building blocks of the search tree. These nodes are formed based on the outcome of a number of simulations. The process of Monte Carlo Tree Search can be broken down into four distinct steps, viz., selection, expansion, simulation and backpropagation. Each of these steps is explained in details below:

- **Selection:** In this process, the MCTS algorithm traverses the current tree from the root node using a specific strategy. The strategy uses an evaluation function to optimally select nodes with the highest estimated value. During tree traversal, a node is selected based on some parameters that return the maximum value. The parameters are characterized by the formula that is typically used for this purpose is given below.

$$S_i = x_i + C\sqrt{\frac{\ln(t)}{n_i}}$$

  When traversing a tree during the selection process, the child node that returns the greatest value from the above equation will be one that will get selected. During traversal, once a child node is found which is also a leaf node, the MCTS jumps into the expansion step.
- **Expansion:** In this process, a new child node is added to the tree to that node which was optimally reached during the selection process.
- **Simulation:** In this process, a simulation is performed by choosing moves or strategies until a result or predefined state is achieved.
- **Backpropagation:** After determining the value of the newly added node, the remaining tree must be updated. So, the backpropagation process is performed, where it backpropagates from the new node to the root node. During the process, the number of simulation stored in each node is incremented. Also, if the new node's simulation results in a win, then the number of wins is also incremented.The above steps can be visually understood by the diagram given below



These types of algorithms are particularly useful in turn based games where there is no element of chance in the game mechanics, such as Tic Tac Toe, Connect 4, Checkers, Chess, Go, etc. This has recently been used by Artificial Intelligence Programs like AlphaGo, to play against the world's top Go players. But, its application is not limited to games only.As we can see, the MCTS algorithm reduces to a very few set of functions which we can use in any choice of games or in any optimizing strategy.

Literature Review:

As we have mentioned before in abstract that we will use Chain Reaction as a basis for our Research.So let us have a look towards the previous work and methodology done and used to improvise the moves and strategies of the Artificial Intelligence of Chain Reaction game. ArtificiaL Intelligence and Chain Reaction is not about the predominance of Artificial Intelligence and its presence in almost all aspects of human life in catalyzing an explosion of happiness but is about a Mini-max algorithm(Game playing algorithm in AI) that plays the Chain reaction game to win against a human player.

The MINIMAX Algorithm

This approach is a look ahead decision-making strategy used by computers across a variety of strategic games, including chess, tic-tac-toe, and now Chain reaction and many others. It is a recursive algorithm where a player tries to maximize chances of his win by predicting multiple levels of possible reactions to his move (via the use of a search tree). It then chooses the play that optimizes future payoff accounting for these reactions.The Minimax Algorithm is defined with a tree data structure — nested data structure which starts with a single root node at Level 0, and branches to any number of child nodes from Level 1 to Level n.In board-games, the root node is considered the current state of the board, and the nodes at Level 1 as every possible next move. The algorithm makes two assumptions

The opponent should be playing optimally or trying to win.

The game should not incorporate any chance component or luck factor.

Terminology

MIN and MAX: The player MAX tries to get the highest possible score, and MIN tries to get the lowest possible score, i.e., MIN and MAX try to act opposite of each other.

Utility function: This function assigns a numeric value for the outcome of a game.

Alpha: Alpha evaluates to best choice so far for the player MAX. It represents the evaluation value on maximizing part.

Beta: Beta evaluates to is the best choice so far for MIN.It represents the evaluation value on minimizing part.

Working and Optimization

The basis of this algorithm is, the computer will calculate its next best move by evaluating the utility of the board at every turn down the road. In the process, the computer assumes that the opponent always selects the best move, which in turn will minimize the utility for the computer. Of course, this is a greedy assumption, but lo and behold, but it tends to work right. To increase the Optimality, a backward pruning strategy called alpha-beta pruning can be applied that reduces the number of tree branches to be explored by maintaining two thresholds called Alpha and Beta.

Pseudocode for MINIMAX with alpha-beta pruning :

The algorithm is recursive calls to evaluate based on specifics of tree depth, Alpha and Beta repeat accounting.

2.1. We reach a node in search space where someone has won the game

2.2. The board is full

2.3. We have reached the predetermined depth limit.

3.The condition to prune a node is when Alpha becomes greater than or equal to Beta.

How pruning works?

The idea of pruning is — no need to count the next branches if we know that the already founded branch will have a better result. It's just a waste of resources.We go to the end of our tree, and only then can we evaluate our results. So the first evaluation will be made on the leftmost vertex. After that, the evaluation process will start for its immediate parent vertex and will go on for all left vertices till one of the possible ones ends and so on. In maximizing turn, we calculate an alpha value, which is the maximum of all evaluation values of its offsprings.we have a Beta value which is the current Beta value of its parent. If Beta is lower or equal to Alpha, we prune, i.e., we stop valuation for that subtree. Because Beta is already lower than the calculated value(Alpha), we decide that the utility of subtree will not be chosen as its next possible step, because it will not change the final result.Returning to the scenario where it's the computer's turn, the computer would call the evaluate(Maximize) function on the current board. This would again call evaluate(Minimize) on each child of the board, which calls evaluate(Maximize) on each grandchild, and so on and so for visualizing the Algorithm:

Line of Thinking

This is the line of thinking behind this algorithm, applied over levels of our tree. We calculate the utilities of the possible moves in the future and decide whether or not these utilities should result in our current move options. If it is likely that the opponent, who is always trying to minimize utility will make all the moves that will lead us to said future point, we should pass that particular future utility value up the tree to represent our current decision node.

This general strategy works better than human brain power alone, provided the algorithm's utility rules are effective.

MINIMAX for Chain Reaction:

The bestn () calculates scores at all possible positions of placing orbs and returns the positions in sorted order of the scores to minimax().

Whenever it's the computer's turn, the computer would call the minimax() function on the current board. It is recursively evaluated on the alternate players best_moves up to a depth or until the conditions on values break us out of the recursive loop. And then chooses the play that optimizes future payoff accounting to all the considered reactions up to a depth.Chain Reaction is a game with not very complex utility measures but a vast search space(increases as board size increases). Applying the above logic to a chain reaction recursively might be enough to make your head spin, which is why the problem has taken to be solved more efficiently by computer. But the fundamental principle of what goes into computer playing programming is — look at possible moves in the future, decide how good these moves are, and predict whether the opponent will make it to explore more.

Working Methodology

Basically **we are putting forward Monte-Carlo Tree Search Algorithm as a unified framework to a Chain Reaction game's Artificial Intelligence. In the modeled framework, randomized explorations of the search space are utilized to anticipate the most promising game moves.The approach allows to exploit the strengths and overcome the weaknesses of the given AI and facilitates the creation of a superior AI in a service-oriented fashion.**All these AIs aim to select the best possible move in a given situation, i.e. for a given board configuration by predicting the next moves of the human player and creating its hashmap. In Chain Reaction the most vital part of the game is to find the best position to make a corresponding move on the board at that position. Hence, Our artificial intelligence model for the game of a board game would first need to identify the best position, predict the opponent's future moves on the basis of previous moves select the optimal move and then make the move at that location.Particularly, we want to explore the scope of Monte Carlo Tree Search in Zero Sum Board Games like Chess,Chain Reaction,Tic-Tac Toe.But as of Now we have used Chain Reaction game as the basis of this research.

Our working methodology is based on Implementation of Artificial Intelligence through a Model based on the concepts of Monte Carlo Tree Search Algorithm. A conceptual game model is developed in a Live Access Server extension of Visual Studio Code i.e. a web browser extension to live reload and designed to provide access to the uploaded HTML/JavaScript/CSS/ ES6 files .There Monte Carlo Tree Search has been implemented for theChain Reaction game's Artificial Intelligence.

The Uploaded JavaScript files have three main scripts i.e Index.js, Monaco.js and Worker.js as shown in the above figure.

As we know that Monte-Carlo Tree Search (MCTS) is a best-first search approach guided by the outcomes of Monte-Carlo simulations. It is primarily based on randomized exploration of the search space. Using the results of previous explorations, the method steadily builds up a game tree in reminiscence and successively will become better at as it should be estimating the values of the most promising moves. MCTS has substantially superior the country of the artwork in board games like Go, Amazons, Hex, Chinese Checkers, Kriegspiel, and Lines of Action.

But Now we are going to use the Monte Carlo Tree Search (MCTS) approach in Chain Reaction to improvise the moves and strategies of the Artificial Intelligence of Chain Reaction. And The process of Monte Carlo Tree Search can be broken down into four distinct steps, viz., selection, expansion, simulation and backpropagation.Each of these processes are explained in detail before in introduction.

Index.js:

This script is basically used for defining variables like atoms, boards, phase, players, split, animation,add and also defines some constants like canvas ,context ,worker,Monaco.

This is also responsible for adding audios and giving colours to the atoms like red and green and also for the background colour i.e black,this is also responsible for the size of the atom in terms of a mathematical formula and radius. As mentioned earlier that it is responsible for sounds in the game so when the atom splits it plays a different sound, if an atom is added it plays a different sound and also plays a victory sound when one of the players win.

Monaco.js

Monaco.js javascript is responsible for Monte Carlo Tree search approach to be implemented for strengthening the Artificial Intelligence of Chain Reaction Game. As all of us know, Monte Carlo Tree search has 5 phases i.e Selection, Expansion, simulation, backpropagation and updation. Basically this javascript flashes out these 5 phases of Monte carlo tree search.As it is divided into Exploring all the parts, Selecting the strategy in the selection process to traverse the tree. It balances the exploration-exploitation trade-off. During tree traversal, a node is selected based on some parameters that return the maximum value. , an effort to expand for a new child node added to the tree to that node which was optimally reached during the selection process i.e to get the child of Maven, operating in the Base tree (simulating the search tree and choosing moves or strategies until the result is achieved. and After figuring out the value of the newly added node, the remaining tree should be updated.

So, the backpropagation procedure is performed, wherein it propagates from the brand new node to the basis node. During the procedure, the number of simulations saved in each node is incremented. Also, if the brand new node's simulation outcomes in a win, then the number of wins is also incremented.)

Worker.js:

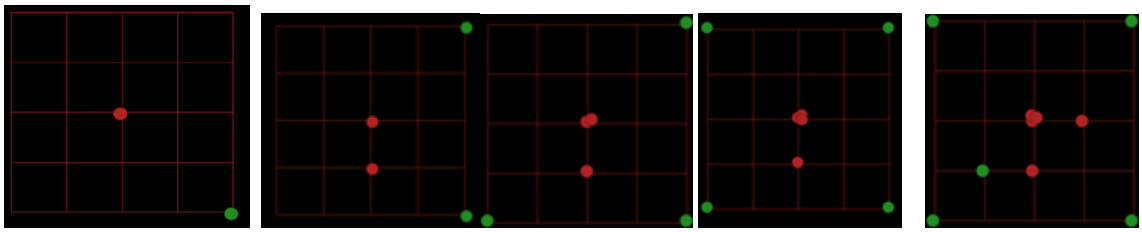
This javascript is responsible for operations happening within the game like switching players on the basis of chances ,splitting molecules or atoms as per the condition adding atoms and specifying timer for the keyframe and adding an atom.

Discussion of Result

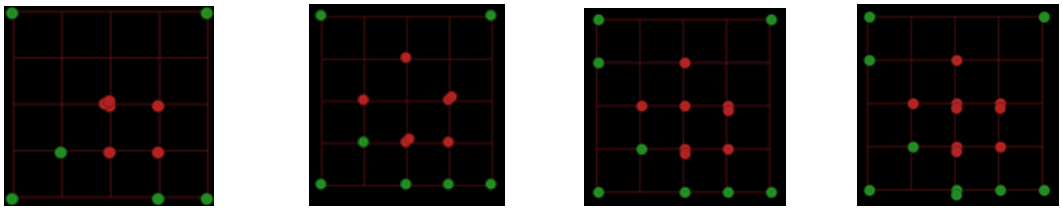
The purpose of this discussion is to interpret the results in the light of Chain Reaction game Artificial Intelligence model generated by the Monte Carlo Tree search Algorithm and to explain new understanding problem of Artificial Intelligence losing in Board games like Chain reaction, tic-tac-toe, GO etc after taking the results into consideration. It discusses the implications of the Game Model of Artificial Intelligence generated.The Discussion connects to the Monte Carlo Tree Search. Instead it tells how the study of Monte Carlo Tree search Algorithm has moved forward in solving problems for Artificial Intelligence in Board games.

Presentation of Results and their analysis

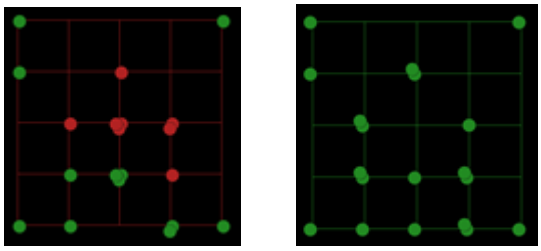
HUMAN PLAYER ARTIFICIAL INTELLIGENCE



STEP 1 STEP 2 STEP 3 STEP 4 STEP 5



STEP 6 STEP 7 STEP 8 STEP 9

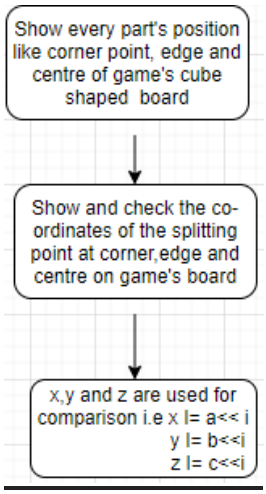


STEP 10 STEP 11•

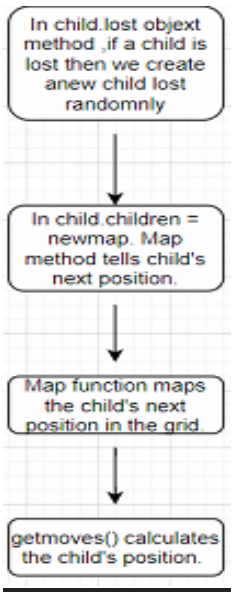
ARTIFICIAL INTELLIGENCE WINS

Working Model (Based on Monte Carlo Tree Search)

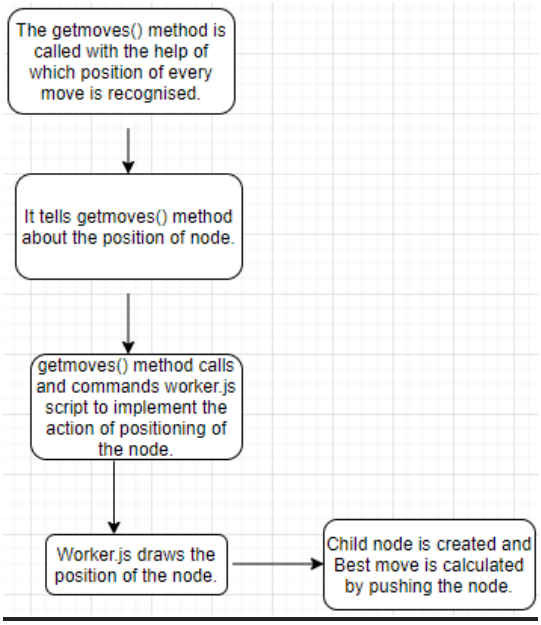
All the parts



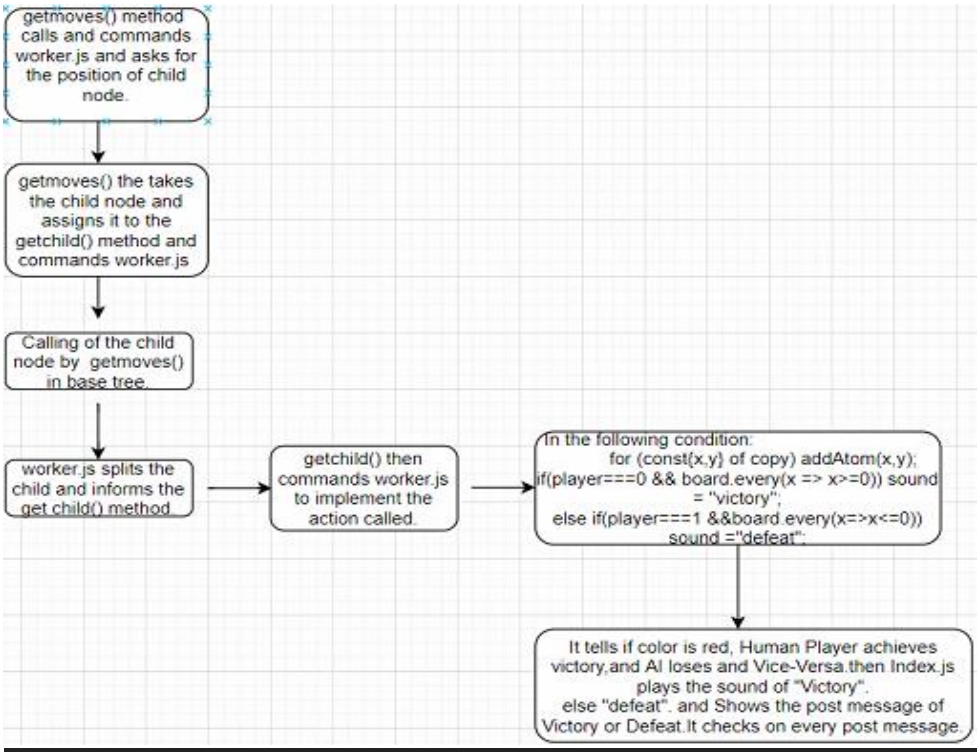
Get the best moves for players(Expansion)



Get child of Maven (Simulation)



Base Tree (Selection,Backpropagation,Update)



Conclusion

Our artificial intelligence model for the game of a generic board game(Chain Reaction or Tic-tac-toe) is able to identify the best position and then make a move at that location. The model is performing a tree search algorithm like Monte Carlo Tree search to exploit the strengths and overcome the weaknesses and creates a superior Artificial Intelligence game Model in a service-oriented fashion. We are successfully defining positions and next moves for a given scenario in generic board games.Our Artificial Intelligence of Chain Reaction game is very strategic and is able to beat human player through its amazing power of prior prediction of Human player's future moves and further make its move in a way that human player gets vulnerable and our AI wins.

Acknowledgement

We would like to acknowledge the efforts of all the individuals whose kind support and guidance made this Research Paper possible .I would like to extend our sincere thanks to all of them.

I am highly indebted to Prof.Dr.Nidhi Chandra for taking us under her guidance and supervision. We are grateful that we shared her vast ocean of knowledge and experience of many years with us .I would also like to extend my gratitude towards my family and friends who supported me throughout the Research Paper.

References

- https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/game-theory/zero.html
- https://medium.com/datadriveninvestor/how-ai-decides-in-a-two-player-game-a51bc21b7fe7

- https://support.chessclub.com/hc/en-us/articles/360006418153-What-is-Chess-
- https://www.quora.com/Is-chess-a-zero-sum-game-What-are-the-examples-of-some-of-the-famous-zero-sum-or-non-zero-sum-games
- https://towardsdatascience.com/tic-tac-toe-creating-unbeatable-ai-with-minimax-algorithm-8af9e52c1e7d?gi=b72bd3d9a840
- https://www.geeksforgeeks.org/ml-monte-carlo-tree-search-mcts/
- https://link.springer.com/referenceworkentry/10.1007%2F978-981-4560-50-4_27
- https://en.wikipedia.org/wiki/Monte_Carlo_tree_search