IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

Analysis of Algorithms for Implementing AI for Game "Tak"

Rajkumar pagey[1], Snehal Martiwar[2], Palak Khokle[3], Laxmi Nishad[4], Priya kakde[5], Rajesh Nasare[6]

Department of Computer Science, RGCER, Nagpur University, Maharashtra[1-6]

Abstract: "TAK" is a conceptual technique two player game, which has as of late picked up ubiquity in the gaming group because of its beginning in a Fantasy Novel and because of its Kickstarter crusade. It is a Zero-Sum, Perfect-Information game. A zero-whole game is a scientific portrayal of a circumstance in which every member's pick up (or misfortune) of utility is precisely adjusted by the misfortunes (or increases) of the utility of the other member. In the event that every player, when settling on any choice, is superbly educated of the considerable number of occasions that have beforehand happened, it is known as an immaculate data game. Because of these properties, we can actualize essential AI calculations, as Minimax and Alpha-beta look based calculations for "TAK".
"TAK" Community is consistently developing and dissimilar to chess there hasn't been a great deal of consideration given to "TAK" AI. A similar investigation of AI Algorithms will help future "TAK" AI designers to make an AI program all the more proficiently.

Keyword: Abstract strategy, board game, perfect information, TAK, zero-sum.

1.INTRODUCTION-

Since the advent of computers, researchers have tried to make artificial intelligence that can play games [3]. The initial research was focused on Chess. In 1996, Garry Kasparov, the best human player, of that time, lost one game to the computer Deep Blue but won the match, consisting of six games. Nowadays, chess programs are way stronger than the chess grandmasters.
The game-play of "TAK" involves Grid Movement and Route Building Mechanisms. It has some properties, most notably threat detection, which make "TAK" hard for computers [2]. We are implementing and performing a comparative study of various AI algorithms for "TAK". Our aim is to find the pros and cons of all the algorithms and derive an optimized algorithm designed specifically for "TAK". In this project we want to graphically map the performance of all algorithms and compare it with the performance of the optimized algorithm.

TAK is a relatively new game, published in 2016 by Patrick Rothfuss and James Ernest. TAK is an abstract strategy game inspired by Chess and Go.

Some programs for TAK are available but as far as we know, no one has done a research on the game properties of Tak and the algorithms appropriate for TAK. TAK has a complex design as it is a 3d game in contrast to the 2d design of Chess.

2. ARTIFICIAL INTELLIGENCE IN 2-AGENT GAME

Two Agent games are class of board games where two rivals play on the other hand. There is no real way to realize what the other player will do, so we approach the issue by picking a move, perceiving how the rival reacts, and utilizing that data to pick the following move.

2.1. MiniMax Algorithm

The goal of MiniMax Algorithm is to find best possible move by looking ahead and evaluating each of the possible moves. One player is designated MAX and other MIN [2]. We assume that every possible move can be evaluated and given a numerical value. If the move favours MAX, the value is positive while move favouring MIN have negative value. To find the best move, we create tree of all of the possible solutions according to the following structures [1]:

-The root is MAX node.

-All MAX nodes have MIN nodes as children and all MIN nodes have MAX children.

To make arrangement tree each piece is inspected and another hub is made for every conceivable move. At the point when the majority of the pieces have been analyzed, center movements to another hub that has not been finished. This procedure proceeds until the tree comes to a pre-indicated profundity. At the point when the tree is finished, we have to decide the most ideal move by utilizing assessment work. We accept that MIN will dependably make the most ideal move so we can't just discover the way from the leaf with most elevated an incentive back to root. To locate the best move, the qualities at the leaves are utilized reinforcement the tree.

Every min hub gets the estimation of its littlest tyke and every MAX hub gets the estimation of its biggest youngster. On the off chance that we proceed until we achieve the root, the offspring of root with most elevated esteem is the best move.

The MiniMax procedure is very straight-forward and correct, but it still has its drawbacks [4]. Depending on the type of game being solved, the branching factor may be very high, resulting in very large tree. In addition, the time to generate tree is O(n) and the time to backup tree is also O(n). Fortunately, there are ways to correct these limitations, like Alpha-Beta Pruning [3].

2.2. Alpha-Beta Pruning

Alpha-Beta Pruning is method of generating solution trees that "prunes" unnecessary branches from the tree[4]. In the MiniMax procedure, the tree generation and evaluation steps are separate. By combining these two steps the search can be stopped at nodes that cannot increase overall value of position. In order to terminate the search an ALPHA and a BETA values are maintained. ALPHA serves as an upper bound for the values of relevant MIN nodes and BETA serves as lower bound for relevant MAX nodes[1]. The search starts at the root and generates children that all get evaluated. The ALPHA and BETA values allow us to discontinue searching in the following cases [2]:

-A MIN node with a BETA value less than or equal to ALPHA value of any MAX ancestors. This node will pass its BETA value up as its backed-up value.

-A MAX node with an ALPHA value greater than or equal to BETA value of any of its MIN node ancestors. This node will contain ALPHA value up as its backed-up value.

The ALPHA and BETA qualities are refreshed amid the hunt, with every MAX hub having as APLHA the greatest current went down estimation of its MIN successors. The BETA estimation of every MIN hub is the base went down estimation of its Max successors.

The most ideal case for Alpha-Beta Pruning would have the successors of every MAX hub requested from most noteworthy to slightest, and the successors of every MIN hub requested from minimum to most noteworthy. In this about outlandish circumstance the requesting augments the quantity of remove that happens. It has been demonstrated that in the best case, the quantity of hubs at profundity D utilizing Alpha-Beta Pruning is about the same as the number at profundity D/2 without pruning.

Any technique for seeking that depends on a hard limit for looking has drawbacks. It is conceivable that Max could have a win that is characterized underneath the pursuit skyline. MIN might have the capacity to delay the win almost inconclusively. With a specific end goal to keep this alleged "skyline impact", many ventures will just end if the assessment estimation of the present position is very little not quite the same as estimation of past position.

2.3. Complexity Table

Sr. No	Name of Algorithm	Time Complexity	Space Complexity
1	Mini-Max	O(b^m)	O(m)
2	Alpha Beta pruning	O(n^m/2)	

3. SYSTEM ARCHITECTURE

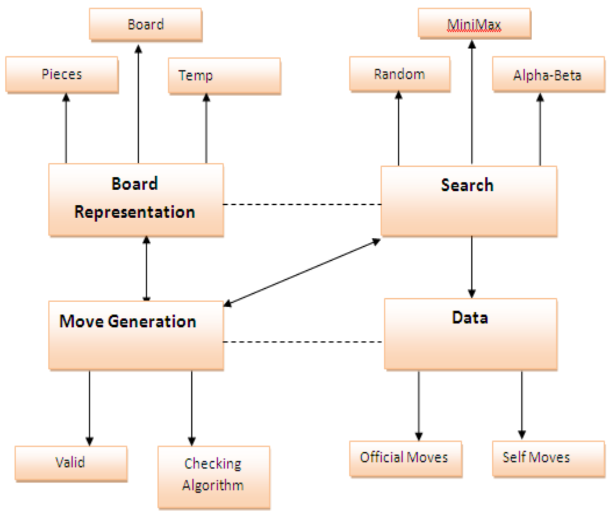

Figure1. System Architecture

Board representation- Board representation is a back end of "TAK" engine which controls how it keeps track of board and the rules of the game.

Move generation- It generates tree of all the possible moves .It checks that moves are valid or invalid and also check for winning move.

Data-It is Database that stores all moves that are generated by

1. Self Learning
2. Official Moves

Search- One among the various algorithms such as Minimax, Alpha-Beta pruning is used to select the move.

1.Minimax-

It is used to minimize the possible loss in a worst case (maximum loss) scenario [3].

2.Alpha-Beta Pruning -

It optimizes the Minimax algorithm and decreases the node of search tree generated by the Minimax algorithm [5].

4. MODULES

4.1. Module 1-Board Representation

This module is 5*5 dimension board, an array of stack internally represented from 0 to 24. Every single cell is a stack. This game goes 3-dimensional. There are 3 stones: flat stone, wall stone and cap stone. Stones can be laid flat or stood on end. When played flat, they are called “flat stones.” In this orientation, other stones can be stacked on them. If they are stood on end, they are called “standing stones” or “walls.” Nothing can be stacked atop a standing stone, but these do not count as part of a player’s road.

Contingent upon the measure of the amusement, players may likewise have capstones, which can come in numerous improving shapes. Capstones fill in as both a Flat stone and a wall, and can likewise straighten standing stones.

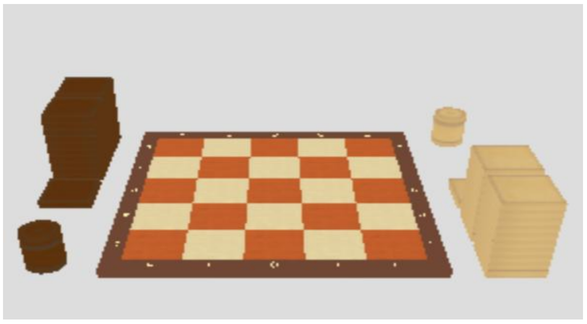

Figure2. Board Representation

- **Bitmap representation**

Move Notation

Take 5 stones from square b4 and move them right dropping 2 stones, 1 stone, then 2 stones as you come to each square.

(5b4>212)

At the beginning of a game of TAK, you start by placing pieces on the board. There are 3 piece types: Flat Stones, Standing Stones, and Capstones. These are called the *stone identifiers* and are represented by an F, S, and C throughout the notation. When a stone or stack of stones is being moved on the TAK board, the direction of the move is indicated by the *direction identifiers*. The four possible move directions are represented by the symbols < > + -.

The < and > identifiers represent movement from player 1's perspective, < moving to the *left*, and > moving to the *right*.

The + and - identifiers represent movement up and down ranks - moves toward the 1 rank, and + moves the opposite direction, away from the 1 rank. It helps to think of these as simple mathematical operators, so that + indicates "adding" or moving *up* in rank, and - indicates "subtraction" or moving *down* in rank.

- **Rules**

1. First turn

On every player's first turn, they should put one of their rival's pieces on any vacant space on the board. The piece must be a flat stone of their rival's colour. Play then continues ordinarily with players controlling their own particular pieces.

Players decide haphazardly who begins the main diversion, and substitute the primary move for future games. In aggressive play, white plays first.

2. Each turn

After the first turn, players may make the choice during their turn to either place a stone or move stones under their control. There is no option to pass a turn.

3. Placement

During their turn, players may place one stone from their reserve onto an empty spot on the board. There are three stone types that may be placed:

Flat stone: Normal stones played flat. Flat stones can be stacked upon, and they count as part of a road.

Standing stone: Normal stones played on their edge. Nothing can be stacked upon a standing stone, but they do not count as part of a road. Also it is commonly called a “wall”.

Capstone: The most powerful piece, as they count towards a road and cannot be stacked upon. The capstone has the ability to move by itself onto a standing stone and flatten the standing stone into a flat stone. An opponent’s standing stones and a player's own standing stones can be flattened in this manner.

An opponent’s standing stones and a player's own standing stones can be flattened in this manner.

4. Movement

A player may move a single piece or a stack of pieces they control. The stone on top of a stack determines which player has control of that entire stack. All stones move in a straight line on the board. There is no diagonal movement, and all stones must proceed forward across the board.

Moving stones is the only way to make stacks. As a stack moves, the player has the option of breaking the stack, covering any existing flat stones along the way. Each space must have one or more stones placed on each space as it moves, but a player has the option to leave zero or more pieces on the starting space. There is no height limit for stacks, but all stacks must be below the carry limit set by the board size in order to leave no stones on the starting space. For example, if the stack was on a 5x5, the carry limit of a stack is 5.

Standing stones and capstones cannot have any stone stack on top of it. Any move that would place a stone atop a standing stone or capstone is not legal. The only exception to this is when a capstone moves by itself onto a standing stone, flattening it. A capstone may make a longer move with a taller stack to flatten a standing stone, but it must be the only piece that moves onto the standing stone.

5. Additional Rules

Carry Limit: There is no upper limit on the height of the stack. However, there is a limit to the number of pieces you can move off that stack, also called “Carry limit” or “Hand Size” which is a number equal to the width of the board. So in 5*5 board the largest number of pieces that you can carry is 5.

Insurmountable Pieces: Neither a capstone nor a standing stone may have any piece stacked on top of it. This piece can be placed and moved normally, but can’t be stacked upon. Therefore it’s not legal to make a move that would place a flat stone a top either of this stone.

Flattening Stone: The capstone can move onto any standing stone, flattening it. A standing stone can only be flattened by the capstone by itself, not by the taller stack with the capstone on top.

6. End of Game

The primary goal of Tak is to build a road from one opposite end of the board to the other. Only flat stones and capstones can contribute to a road, while standing stones do not. As soon as the road is built, the player who built it wins. This is called a "road win". Roads do not have to be in a straight line, but stones can only connect when they are adjacent to one another. Stones cannot connect diagonally.

If a player makes a move that results in a winning road for both players, the active player wins.

If a road is not built by either player, a player can also win by controlling the most spaces with flat stones on the board. The game will end when a player places their last piece, or when all spaces on the board are covered. The player with the most flat stones wins. Standing stones and capstones do not count. Stones captured by other pieces also do not count, only the flat stone on top.

4.2 Module 2-Move Generation

This module contains two sub-modules: Valid moves and checking winning move.

1. Valid moves- This module checks the limitations and permits just substantial moves like placement and movement of pieces.

2. Checking winning move- Each time the piece is moved, it is checked whether it is a last winning move or not. In the event that it is not, the game will be proceeded with else the winning message with player name will be pronounced.

4.3 Module 3-Move Selection

This module includes selection of the move to be played after the opponent’s move. It has two sub-module viz. Random and through algorithms.

1. Random- Here moves are selected at random from DATA which does not assure winning the game.

2.Through algorithms- Here moves are selected by comparing moves generated by MiniMax and Alpha beta Algorithms on the basis of both time and space complexity.

5. CONCLUSION

Some programs for TAK are available but as far as we know, no one has done a research on the game properties of TAK and the algorithms appropriate for TAK so we have studied various algorithms and successfully implemented TAK using MiniMax algorithm. In future, modifications would be done using Alpha-Beta pruning.

6. REFERENCES

[1] Ahmed A. Elnaggar, Mostafa Abdel Aziem, Mahmoud Gadallah and Hesham El-Deeb :”A Comparative Study of Game Tree Searching Methods”. International journal of advanced computer science and applications, valume 5 no.5 (2014)

[2] Avneet Pannu: “Artificial Intelligence and its Application in Different Areas”. :Inernational journal of engineering and innovative technology(IJEIT) volume 4, issue 10(2015)

[3] Ronald L Rivest: “Game Tree Searching by Min/Max”, labouratry of computer science, MIT, Cambridge (1995)

[4] Jonathan Schaeffer and Aske Plaat: "An Analysis of Alpha-Beta Pruning", Carnegie Mellon University.

[5] Ashraf M.Abdelbar: "Alpha-Beta Pruning and Althofer's Pathology-Free Negamax Algorithm": Department of computer science and Engineering,American university in Cairo Egypt(2012)

[6] BarneyPell: "*METAGAME A new Challenge for Games and Learning*": The third computer Olympiad Ellias Horwood,UK(1992).

[7] Steven Balensiefer and Bill Grenzer: "*Artificial Intelligence for Checkers"*: University of Notre Dame (2003)

[8] Michiel van de Steeg, Madalina M. Drugan, Marco Wiering," Temporal Difference Learning for the Game Tic-Tac-Toe 3D: Applying Structure to Neural networks", 2015 IEEE Symposium Series on Computational Intelligence.

[9] Yue Fu and Tianyou Chai" Online Solution of Two-Player Zero-Sum Games
for Continuous-Time Nonlinear Systems With Completely Unknown Dynamics", IEEE Transactions On Neural Networks And Learning Systems, Vol. 27, No. 12, December 2016.

[10] Tzung-Pei Hong Ke-Yuan Huang Wen-Yang Lin, Department of Information Management I-Shou University Kaohsiung, 84008, Taiwan, R.O.C. "A Genetic Minimax Game-Playing Strategy".