Adversarial Search Methods In Two-Player Games

Milan Wittpohl, December 2019

Abstract

Adversarial search methods are used to find the best next move in games. However, as the depth of the search tree increases the performance of classic adversarial search methods suffers. The genetic minimax algorithm seems to be a promising alternative. In this research the minimax, the alpha-beta pruning and the genetic minimax search algorithm are used in the two-player games Tic Tac Toe and Connect Four to gain a better understanding of the algorithm's strengths and weaknesses. In conclusion the genetic minimax algorithm performs better as the depth increases, however the need for a high depth should be considered carefully. If not needed, alpha-beta pruning method should be preferred.

Key words: Adversarial Search Methods, Minimax, Alpha-Beta Pruning, Genetic Minimax Algorithm, Two-Player Games, Evolutionary Algorithm

1 Introduction

In computing, it is often required to find an optimal solution to a given problem. Search methods are a common approach to evaluate possible solutions. This is done by defining different states of the problem and then transitioning through them. As search methods mimic human behaviour (e.g. decision making), they are part of cognitive simulation approach of artificial intelligence. [1]

Search methods transition thorough the search space to find the optimal solution and use an evaluation function to define their quality. Usually, the quality can be calculated without having to make assumptions about other external factors that might influence the quality. However, for example, in games or negotiations this changes, as the quality of the next problem state has to take the actions of the opponent(s) into account.

Adversarial search methods are used for problems where two or more parties influence the current problem state. [1] This makes them suitable to evaluate the optimal next move in games.

The most common adversarial search methods are the minimax and alpha-beta pruning method. [1] However, as the number of possible next moves increases, these algorithms need more time, as there are more problem states to evaluate. In games, time is often a constraining factor. In 2001, Tzung-Pei Hong, Ke-Yuan Huang and Wen-Yang Lin, introduced the genetic minimax algorithm, which combines evolutionary algorithms with the central idea of the minimax method. [2] Even though the algorithm does not look at all possible solutions, the researchers concluded that it could find a (near) optimal solution in significantly less time.

This paper looks at adversarial search methods for two-player games to gain a better understanding of their strength and weaknesses. This was achieved by implementing these algorithms and conducting experiments for two two-player games.

2 Materials and Methods

Hong, Huang and Lin already conducted experiments in their research and found that the genetic minimax algorithm was almost as good in terms of accuracy and significantly better in terms of performance, than the classic adversarial search methods. [2] In this research, the algorithms are compared in real-world examples for two two-player games. The two games were chosen based on their number of possible next moves. The first game, **Tic Tac Toe**, has relatively few possible next moves, with a maximum of 9! # 363.880 possible game states. **Connect Four**, serves as a more complex two-player game with approximately 4,5 x 10^{10} game states. An even more complex game, which is not considered here, would be a game of chess with an average of approximately $3{,}35 \times 10^{123}$ game states. [3]

For this research, the next optimal move in a two-player game should not take more than two seconds.

2.1 General Approach

In the researched mentioned by Hong, Huang and Lin the accuracy was measured by the number of times the genetic minimax algorithm could find the same solution as the minimax solution. In this research a different approach is chosen to validate the algorithms accuracy. It is not necessary to find the optimal next move every time to win the game. For this research, if the algorithm manages to take less time per move but still wins the game it is considered better.

In order to gain a better understanding of the strength and weaknesses, the following requirements for the software were defined:

1. In addition to the minimax, alpha-beta pruning and genetic minimax algorithm, a random selection algorithm should be implemented, that every other algorithm should beat comfortably.
2. The depth of the search tree and the number of moves to take into account, has to be variable to the board.

3. The software needs to be able to simulate n games automatically with a fixed board, fixed algorithm for player A and fixed algorithm for player B.
4. The software needs to document:
   a. The number of wins per player
   b. The average number of moves per play – few moves indicate a one-sided game
   c. The average time needed to calculate the next move per player
5. To gain a more transparent understanding of the algorithms, it has to be possible to simulate a single game and view each move per player.

Based on the requirements, the author chose to implement the software as a web application, using typescript, which was compiled to javascript.
To be more flexible during the implementation and to gain a deeper understanding, **no** existing frameworks for the algorithms were used.

2.1.1 Implementation – Tic Tac Toe Board

The board of Tic Tac Toe has 3x3 fields, and any player can access every empty field at any time. The game is won by the player who has three connected fields vertically, horizontally or diagonally.
The depth of the search tree is set to the number of remaining fields, which is also the maximum depth. The factorial of the depth is the number of possible game states, meaning that the search tree dramatically shrinks after every move. This is illustrated in Table 1.

Move	**Depth**	**# possible game states**
1	9	362.880
2	8	40.320
3	7	5.040
4	6	720
5	5	120
6	4	24
7	3	6
8	2	2
9	1	1
Avg.		*45.459*

Table 1 - Tic Tac Toe - Depth & Game States

2.1.2 Implementation – Connect Four

A board with 7x6 fields, where players can only access fields if the field below is already occupied by any other player (this excludes the first row). The game is won by the player who has four connected fields vertically, horizontally or diagonally.
The depth of the search tree is not set the maximum, as it could result in enormous search trees. As the number of possible game states per move is *mostly* seven (number of fields per row), Connect Four presents a more complex game at the same depth (as illustrated in Table 2).

Depth	**# possible game states**
10	$2{,}82 \times 10^8$
9	40.353.607
8	5.764.801
7	823.543
6	117.649
5	16.807
4	2.401
3	343
2	49
1	7

Table 2 - Connect Four - Possible Game States

For this research, the depth for a game of Connect Four is limited to 10.

2.1.3 Implementation – Genetic Algorithm

Genetic algorithms can be optimized to a high degree to serve a particular purpose. For instance, the population size, the selection process of parents, crossover- and mutation-operator influence the outcome of the algorithm heavily. It is **not** the focus of this research to optimize the genetic minimax algorithm for one specific game. Therefore, the results conducted should be considered as the minimal performance as they should improve significantly when tuning them to a specific game. Only the population size and the time limit were customized for each experiment. Even though these parameters were customized, this was not the focus of the research, and it is in the authors' opinion that more significant optimizations could be achieved.
In all experiments, the parents were chosen based on their fitness from a random subpopulation, of one-tenth of the population size. The children were produced

using a one-point crossover and were always mutated by randomly flipping two values of the chromosome.

Genetic algorithms improve the current population until a particular stopping criterion occurs. In the case of this research, the stopping criteria is a time limit. On an abstract level, the genetic minimax algorithm creates a random population and then optimizes that population over and over again. The creation of the random population is mandatory and is limited in time by the number of chromosomes to generate. The second process is then repeated until the time is up.

3 Experiments

The experiments generally follow the same schema for each of the two games. Additionally, there are a few differences which are described afterwards.

3.1 Experiments – General Schema

Firstly, it is verified that the classic adversarial search methods dominate against the random selection algorithm.

Secondly, it is verified that the alpha-beta pruning finds the optimal solution significantly faster.

Thirdly, the genetic minimax algorithm is only better if, it can get similar results as the alpha-beta pruning algorithm in significantly (at least 20%) less time. The population size and time limit are set accordingly before experimenting.

Finally, a summary of the results is presented.

3.2 Experiments – Tic Tac Toe

All simulations regarding Tic Tac Toe are simulated over 100 games. Additionally, the order of which player starts first has to be taken into account. As stated before, the number of possible game states is reduced dramatically after each move. Therefore, player B (every second move) has an advantage, as fewer possibilities need to be evaluated. When simulating algorithm A vs algorithm B, algorithm B vs algorithm A also has to be simulated.

3.3 Experiments – Connect Four

As the classic adversarial search methods performance decreases massively, as the depth increases, not all simulations are run over the same number of games.
In contrast to Tic Tac Toe, the order of the algorithms does not significantly affect the outcome, as the number of available fields is the same after almost every move. Therefore, the reversed order of the two algorithms does not have to be simulated.

4 Results

The results are first presented for Tic Tac Toe and then for Connect Four.

4.1 Tic Tac Toe

The results are presented according to the general schema stated in the previous section.

4.1.1 Verification of classic search methods

Both the minimax and the alpha-beta pruning defeated the random selection algorithm in most cases. The random algorithm never won but managed to get a significant number of tied games.

Minimax v. Random Selection	
Wins by Minimax	99
Wins by Random Selection	0
Number of Ties	1
Random Selection vs Minimax	
Wins by Minimax	74
Wins by Random Selection	0
Number of Ties	26
Alpha-Beta Pruning vs Random Selection	
Wins by Alpha-Beta Pruning	100
Wins by Random Selection	0
Number of Ties	0
Random Selection vs Alpha-Beta Pruning	
Wins by Alpha-Beta Pruning	80
Wins by Random Selection	0
Number of Ties	20

Table 3 - Verification of classic search methods

4.1.2 Verification of performance increase

The alpha-beta pruning algorithm found the solution significantly quicker than the minimax algorithm. Additionally, it became clear that moving first results in highly more possible game states, that need to be evaluated, and therefore results in longer run times.

Minimax vs Random Selection	
Avg. number of moves per Player	2,78
Avg. time per move of Minimax in seconds	6,78
Alpha-Beta Pruning vs Random Selection	
Avg. number of moves per Player	2,89
Avg. time per move of Alpha-Beta Pruning in seconds	0,27
Random Selection vs Minimax	
Avg. number of moves per Player	3,79
Avg. time per move of Minimax in seconds	0,77
Random Selection vs Alpha-Beta Pruning	
Avg. number of moves per Player	3,72
Avg. time per move of Alpha-Beta Pruning in seconds	0,06

Table 4 - Verification of performance increase

4.1.3 Genetic Minimax Algorithm

For games against the random selection algorithm, the alpha-beta method only took about 60 milliseconds when moving second and about 270 milliseconds when moving first. To beat the alpha-beta algorithm, the genetic minimax algorithm had to get similar results significantly less time. Therefore, the population size was reduced dramatically and no to little time for progression the population was given. The results show that even though the accuracy suffered, the genetic algorithm was able to win most games in less time.

Genetic Minimax vs Random Selection	
Wins by Genetic Minimax	95
Wins by Random Selection	1
Number of Ties	4
Avg. number of moves per Player	3,3
Avg. time per move of Genetic Minimax in seconds	0,12
Random Selection vs Genetic Minimax	
Wins by Genetic Minimax	22
Wins by Random Selection	67
Number of Ties	11
Avg. number of moves per Player	3,65
Avg. time per move of Genetic Minimax in seconds	0,04

Table 5 - Genetic Minimax vs Random Selection

As the genetic minimax algorithm does not evaluate all relevant possibilities, in contrast to the alpha-beta pruning method, it was also conducted how the two algorithms play against each other.

Genetic Minimax vs Alpha-Beta Pruning	
Wins by Genetic Minimax	0
Wins by Alpha-Beta Pruning	7
Number of Ties	93
Alpha-Beta Pruning vs Genetic Minimax	
Wins by Genetic Minimax	9
Wins by Alpha-Beta Pruning	97
Number of Ties	3

Table 6 - Genetic Minimax vs Alpha-Beta Pruning

4.1.4 Summary

The simulations show that the classic adversarial search methods found the optimal next move reliably and therefore win comfortably against the random selection algorithm. In the next section the following points are discussed:

- The uneven results of the simulations of the genetic minimax algorithm against the alpha-beta pruning algorithm
- The value of the genetic minimax algorithm for Tic Tac Toe

4.2 Connect Four

Connect Four presents the challenge of long simulation times, as it took the alpha-beta pruning method at a depth of *6* already about 10 seconds per move.

4.2.1 Verification of classic search methods

To verify that the classic algorithms comfortably defeat the random selection, only ten simulations were conducted with a depth of four. At depth four, it took the alpha-beta pruning method about one second per move. As the minimax methods took more than five seconds for one move in Connect Four, the method was not considered.

Alpha-Beta Pruning vs Random Selection	
Wins by Alpha-Beta Pruning	10
Wins by Random Selection	0
Number of Ties	0

Table 7 - Alpha-Beta Pruning vs Random Selection - Depth: 4

With a depth of just two, it was possible to run 100 simulations of the alpha-beta pruning algorithm against the random selection algorithm.

Alpha-Beta Pruning vs Random Selection	
Wins by Alpha-Beta Pruning	94
Wins by Random Selection	6
Number of Ties	0
Avg. number of moves per Player	9,72
Avg. time per move of Alpha-Beta Pruning in seconds	0,57

Table 8 - Alpha-Beta Pruning vs Random Selection - Depth: 2

4.2.2 Verification of performance increase

As the minimax method is not even considered due to poor performance, no verification was done.

4.2.3 Genetic Minimax Algorithm

The previous tests showed that the classic adversarial search methods are useless with a depth of six or more. To prove the performance of the genetic minimax algorithm, it was run 100 times with a depth of 10 against the random selection algorithm. As

Table 9 shows, it only took the algorithm about 1,3 seconds to calculate the next move and it won comfortably.

Genetic Minimax vs Random Selection	
Wins by Genetic Minimax	94
Wins by Random Selection	6
Number of Ties	0
Avg. number of moves per Player	8,68
Avg. time per move of Genetic Minimax in seconds	1,29

Table 9 - Genetic Minimax vs Random Selection - Depth: 10

4.2.4 Summary

As the depth increased the classic adversarial search methods failed to find the next best move in considerable time. However, the results raise the question, how important a high depth for the simulation is.

5 Discussion

After discussing the main points of the Tic Tac Toe and Connect Four simulations, a conclusion is drawn.

5.1 Discussion – Tic Tac Toe

When simulating the alpha-beta pruning algorithm against the genetic minimax algorithm, the latter performed well when moving first, but poor when moving second. This can easily be explained by the fact that the first player has an advantage in games like Tic Tac Toe, where both players have the same options. [4]

In a game of Tic Tac Toe or other games with similar amounts of possible game states, the genetic minimax algorithm should not be used. As it takes less than half a second for the alpha-beta pruning method to find the optimal solution, it should be preferred over the genetic algorithm. With genetic algorithms, it is always possible to not find the optimal solution, whereas the alpha-beta pruning method will always find it.

5.2 Discussion – Connect Four

Connect Four shows that classic adversarial search methods need too long as depth increases. The genetic minimax algorithm was able to perform significantly better as the depth increased. However, even with a depth of two, the alpha-beta pruning method could win reliably. As it takes more time to implement and tune a genetic algorithm, the alpha-beta pruning method should be chosen over the genetic minimax method as long as a small level of depth is sufficient.

5.3 Final Conclusion

This research aimed to gain a better understanding of the considered adversarial search methods. In conclusion, the classic adversarial search methods are a good fit for less complex games (in terms of possible game states). For more complex games, the genetic minimax algorithms is a good alternative. However, the performance decrease for classic methods is mostly due to a higher depth. It should be carefully considered if a higher depth is in fact, needed. If not, the alpha-beta pruning method might be a better fit, as it is easier to implement and more reliable.

6 References

[1] M. Flasiński, "Introduction to Artificial Intelligence," in *Introduction to Artificial Intelligence*, Springer International Publishing Switzerland, 2016, pp. 31, 41-44.

[2] T.-P. Hong, K.-Y. Huang and W.-Y. Lin, "Adversarial Search by Evolutionary Computation," *Evolutionary Computation,* vol. 9, no. 3, pp. 371-385, 2001.

[3] M. J, Knowledge-free and learning-based methods in intelligent game playing, Berlin: Springer, 2010.

[4] A. Levine, "Exploring Tic-Tac-Toe Variants," Mathematics and Computer Science, Stetson University, 2017.