Three Player Chess: Algorithms

Kane Alexander (22710428), Andrey Sabile (22226065)

The University of Western Australia

Literature Review

Chess is a common strategy board game with deep historical roots (John, 2017). Played on a checked board each player uses their pieces to capture those of their opponents, with the goal of checkmating the opponents 'King' piece. This traditional two-player game can be adapted to suit three-players; the winner being the player who captured any opposing 'King', the loser whose 'King' was captured and a third impartial player. This adaptation follows the same strict rules as standard chess, with each player taking a move in turn. The finite number of possible moves allows the game to be broken down into several finite states with distinct causation paths. The expansive nature of the game, coupled with the number of possible moves each turn, leads to an impossibly large perfect outcome tree and therefore more localized algorithms are more viable (Atashpendar, Schilling, & Voigtmann, 2016).

First published in 1928 by John von Neumann, the minimax algorithm is a popular mathematical approach to game theory (Krantz, 2015). The algorithm treats game outcomes under a zero-sum gain proposition, the gain of one player is the loss of another, and thus looks ahead to maximise its own return. Despite being a very effective and decisive algorithm in traditional chess, when expanded to multiplayer games its desired efficacy decreases drastically as it exposes a flaw in the zero-sum gain proposition. The underlying assumption of zero-sum gain is that all players target the AI; an advantageous assumption in two player games. When expanded to 3 player games this forms a fallacy as individual players don't necessarily hold biases and instead generally play for their own self-interest. An adaptation for this algorithm to suit an arbitrary number of players was first published in 1986 by Carol A. Luckhardt et al, in which the zero-sum gain assumption was modified. Commonly referred to as MAXN, this algorithm assumes each player makes the move that maximises their own individual return and acts accordingly (Luckhardt & Irani, 1986). It is proven to provide a statistical advantage over its minimax predecessor for 3 player games. Further adaptations of both minimax and MAXN were explored in Zuckerman et al, altering the playstyles of the algorithm depending on the current game state. These modifications of which were shown to have up to a 60% improvement for the test game of 'Hearts' (Zuckerman, Felner, & Kraus, 2009).

A completely alternative algorithm employed in game theory is the Monte Carlo Tree Search. MCTS focuses on analysing the most promising moves, iteratively expanding the current search tree from a random sampling of the search space. Each time the algorithm is applied, a 'playout' occurs in which the game tree is played to the end by selecting moves at random. The result of each playout is then used as a weighting system for the likely hood of these nodes being chosen again in future playouts. As explored in the 2012 paper by Browne et al, the Monte Carlo Tree Search has favourable applications in game theory, most notably in games with extremely high branching factors such as 'Go'. This study goes on to state that; 'MCTS approaches to games such as Chess are not as successful as for games such as Go' and instead algorithms such as 'minimax perform admirably well' (Zuckerman et al., 2009).

A new algorithm for game playing proposed by Schadd et all in 2012 is Best Reply Search (Schadd & Winands, 2012). BRS is game tree search technique that flattens its opponents' players moves such that only a single move, the one that has highest utility among all opponents, is chosen when building the game tree. This allows a significant lookahead to be achieved and enables further future planning then standard MAXN. This research paper found significant advantages over MAXN and minimax algorithms in the games Chinese Checkers and Focus while it performed roughly the same as a minimax algorithm in Rolit.

Game theory and game playing strategies are an area of computer science that have been explored extensively and numerous algorithms have been constructed ("International journal of game theory (Online)," 1971). Applications for standard chess have been widely explored and tested with such algorithms, but very little expansive work has been done on three or more player chess games. As such, this paper will explore the application of such algorithms on three chess games.

Selected Technique

The deciding factors on which algorithm will be the most effective for a given game, can be largely domain dependant. Unique strategies, random number generation and even accidental or advantageous collusion are ever-present in multiplayer games. Creating an algorithm to counteract these possibilities, especially in a multiplayer setting where the relative impact an individual player holds is diminished, requires an extremely adaptable algorithm. A possible solution to this is to employ different strategies and algorithms depending on the current domain or game state. Instead of relying on a singular overarching algorithm, this AI employs separate offensive, defensive and standard algorithms contingent on its current position.

The algorithm the AI employs throughout the game states in which no one individual player has a substantial lead is the MAXN algorithm. Working on the underlying assumption that every player is acting for their own self-interest, it presumes every player will make the move that will increase their own utility and thus acts accordingly. The MAXN algorithm is a derivation of the minimax algorithm with a resolve for the inclusion of a third player. The standard minimizing function is replaced with a maximising one; instead of assuming the players are trying to minimize the AI's utility, it instead assumes they're trying to maximise their own. As shown in figure(1) the algorithm recursively creates a tree of a set depth then evaluates the utility of each player in those game states. MAXN iterates its way up through the tree carrying the nodes that maximise the utility of the player at that level whose theoretical turn it is. In the early or even game states this algorithm provides the logical advantage, as it is probable that no one player is irrationally targeting another but instead each player is acting for their own self-interest.

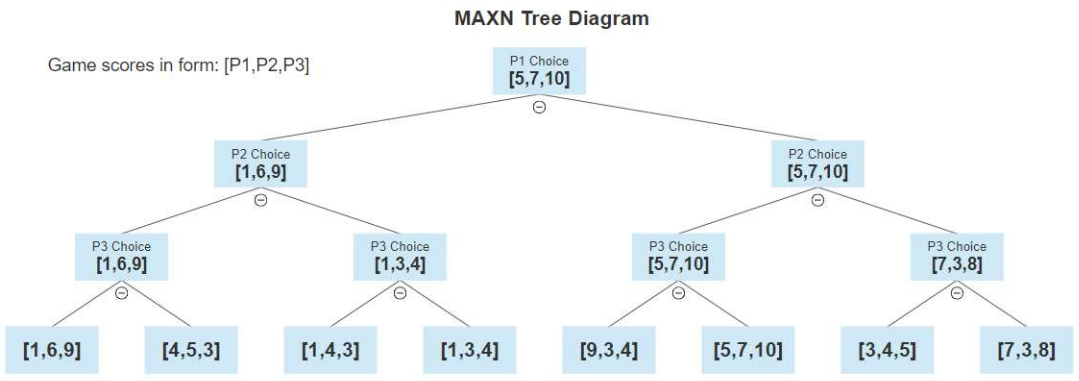


Figure (1)

In the case that the AI gains a substantial advantage over the other players it switches to a defensive strategy in order to preserve its lead. Working under the assumption that the other players will collude in order to increase their individual chances of victory, the defensive algorithm presumes every move made by its opponents is the one that minimizes the AI's score. This algorithm is pure minimax expanded linearly for multiple players. Named the 'paranoid' algorithm, it recursively creates a tree of a set depth with the terminal nodes holding the final score the AI would hold in those states. Paranoid iterates up through the tree, each parent node taking the value of its child that is the minimum. The very initial node, which is move made by the algorithm, selects the child whose value is a maximum. This move does not potentially provide any score gain to the AI but is the move that will most heavily reduce the potential damage.

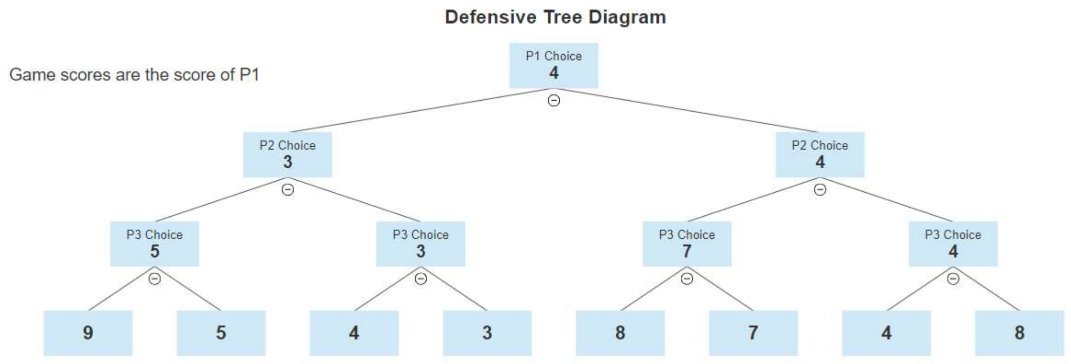


Figure (2)

In the last potential scenario, the AI is in a position in which another player has gained a substantial lead. In this case the algorithm has two potential strategies; play defensively for a draw or play offensively to increase its chance of winning. The strategy in which the AI wants to avoid a loss (-1) it will employ the paranoid algorithm above; reducing its chance of winning (+1) but increasing its chance of a draw (0). If the AI instead employs the higher variance option, an offensive strategy, it will take explicit actions to prevent the leader from winning even if these actions drastically worsen its own situation temporarily. The offensive strategy works very similarly to MAXN as it traverses the MAXN tree assuming the opponents are trying to maximize their own utility. The sole difference being; it selects the move that holds the lowest score for its target opponent instead of maximizing its own utility. Substantial testing will need to be employed to see which option; paranoid or offensive, is a more effective strategy for the given values of winning, losing and drawing.

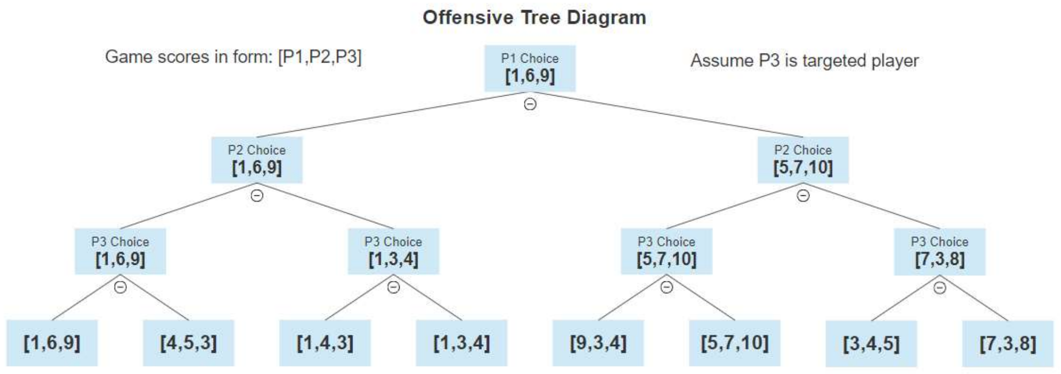


Figure (3)

With these general strategies as the AI’s backbone, many crucial elements need to be decided in order for the algorithm to work effectively; namely the depth to which the searches are to be performed and the overarching method to decide which particular algorithm is chosen for a given game state. The limiting factor on depth performance is the allocated amount of time each move has. As explored in Sturtevant et al the MAXN algorithm can only be shallow pruned, unlike the paranoid algorithm which can be alpha beta pruned (Sturtevant & Korf, 2000). Even with shallow pruning implemented, the max depth search that is viable is 3 layers and that is the depth that all algorithms will be tested too. In order to choose between algorithms, we need to determine the standard to which a player is deemed ‘substantially leading’. As proposed in the paper by Zukerman et all a leading-edge value is found, that is, the utility of the leading player minus the utility of the second player (Zuckerman et al., 2009). If this value is over a threshold value, then that player is deemed ‘sustainably leading’. The utility for each player is given by the number of pieces they have on the board plus the number they have taken, multiplied by the value for each piece in the table below. For the threshold value; the standard value 10 was selected, which could be improved with the help of a genetic algorithm as a future possibility.

Utility Chart

PIECE	VALUE
PAWN	1
KNIGHT	3
BISHOP	3
ROOK	5
QUEEN	9
KING	40

Figure (4)

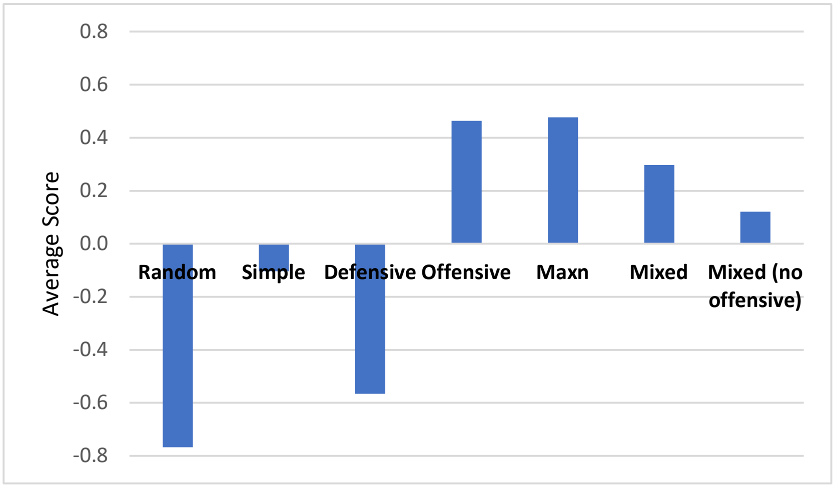
Testing Technique

To properly evaluate the effectiveness of AI’s in question they need to be tested against control agents to see how well each AI performs relative to a baseline standard. The main AI’s to be tested are the combined algorithms; one with offensive play and one without. These will be tested alongside the pure defensive, offensive and MAXN agents to see how well the combined agents perform relative to their individual algorithms. To test the different play styles possible in real gameplay, each AI will be tested against control agents that simulate different styles of ‘real play’. The first control agent is simple a random algorithm that chooses completely random move each turn. The second control agent is a simple algorithm that looks at its possible moves and chooses the one that increases its utility the most.

All 7 algorithms will be tested tournament style in a series of 500 games against randomly selected opponents from the pool of algorithms. After each set of 500 games, the bottom two algorithms will be removed, and the tournament repeated. This testing technique ensures all algorithms get evaluated fairly and properly. The removal of algorithms that perform badly in each tournament will ensure that the complex AI’s are evaluated properly; removing the possibility that an algorithm is seen as overly effective even if it only wins against simple algorithms but fails against complex ones.

7 Agent Results

	Random	Simple	Defensive	Offensive	Maxn	Mixed	Mixed (no offensive)
Games Won	8	64	9	129	119	85	86
Games Lost	173	84	125	21	15	24	58
Games Played	215	193	205	233	218	205	231
Net Score	-165	-20	-116	108	104	61	28
Average Score	-0.767442	-0.103627	-0.565854	0.463519	0.477064	0.297561	0.121212



Rank	Algorithm	Average Score
1	Maxn	0.477064
2	Offensive	0.463519
3	Mixed	0.297561
4	Mixed (no offensive)	0.121212
5	Simple	-0.103627
6	Defensive	-0.565854
7	Random	-0.767442

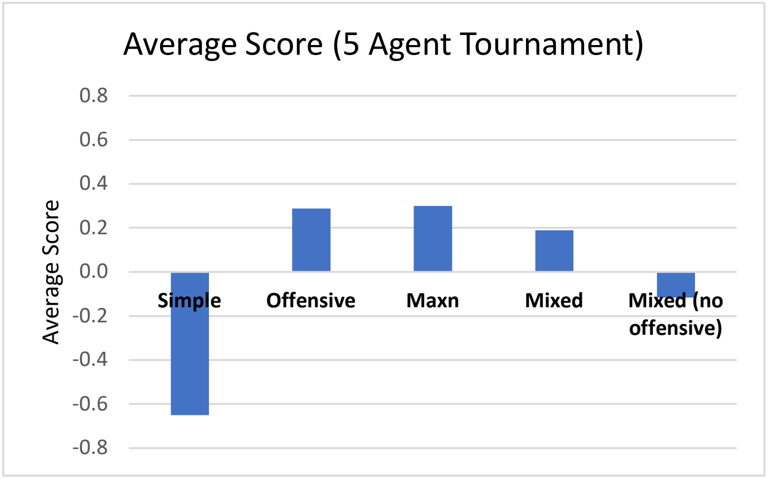
Analysis of 7 Agent Performance

The results comparing all 7 algorithms showed all algorithms performing as intended and producing expected results. The random control agent had the lowest average score, -0.77, losing almost all its games against the other algorithms. The defensive agent came second last, losing over half of its games. This algorithm is purely designed to mitigate risk, and thus, did not win many games. Instead the defensive agent drew 35% of its total games played, a very high percentage relative to the 16% draw rate of the random agent.

Mixed agents expect a degree of target intelligence, that is, opponents that target a player if they hold a substantial lead. If the mixed agent holds such a lead, they act defensively expecting the other players to single them out. This initial tournament had majority of algorithms not employing this strategy, giving these opponents valuable time to catch up while the mixed agent plays defensively. As such, these mixed agents performed worse than the leading agent's; offensive and MAXN, who capitalize against opponents who don't have target intelligence. This type of algorithm is generally more fruitful in the early stages of the tournament as a higher percentage of games will be against less strategic opponents.

5 Agent Results

	Simple	Offensive	Maxn	Mixed	Mixed (no offensive)
Games Won	28	147	128	118	79
Games Lost	232	57	40	60	111
Games Played	314	313	293	305	275
Net Score	-204	90	88	58	-32
Average Score	-0.649682	0.287540	0.300341	0.190164	-0.116364



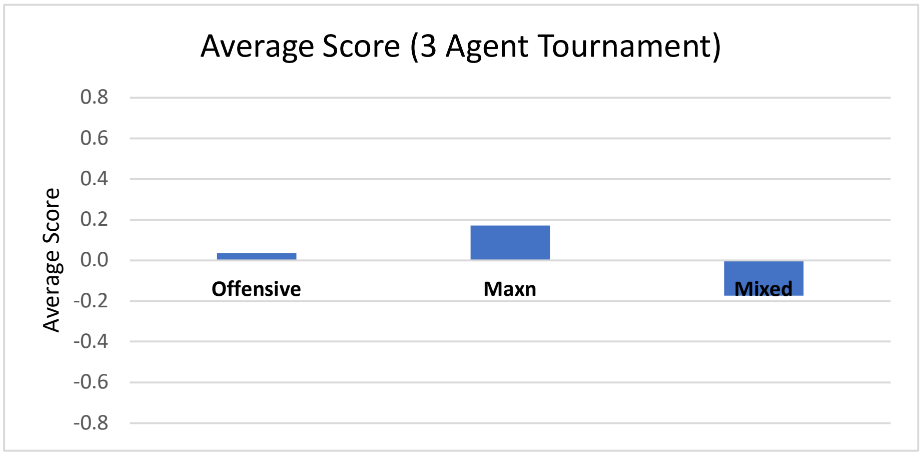
Rank	Algorithm	Average Score
1	Maxn	0.300341
2	Offensive	0.28754
3	Mixed	0.190164
4	Mixed (no offensive)	-0.116364
5	Simple	-0.649682

Analysis of 5 Agent Performance

Similarly, the results from this second tournament conformed to the expected outcomes of each agent. The simple test algorithm lost vast majority of its games against the other algorithms as they all employ a minimum 3 move look ahead relative to its basic 1 move look ahead. The results also highlighted the disadvantage the mixed algorithm had without its offensive attribute. The conclusion for this tournament is that for the current weightings of wins, draws and losses an aggressive strategy against these given opponents is more viable then a reserved one.

3 Agent Results

	Offensive	Maxn	Mixed	Mixed (no offensive)
Games Won	150	230	120	120
Games Lost	135	145	220	220
Games Played	140	120	240	240
Net Score	15	85	-100	-100
Average Score	0.035294	0.171717	-0.172414	-0.172414



Rank	Algorithm	Average Score
1	Maxn	0.171717
2	Offensive	0.035294
3	Mixed	-0.172414

Analysis of 3 Agent Performance and Conclusion

This final set of results displayed the ineffectiveness of the mixed agent and provides insight into possible adaptations for future studies. Results from the tournaments showed the mixed agent was less viable then both the MAXN and offensive agents. Although the margins for this tournament were drastically closer than the previous tests, a range of 0.34 relative to 0.95 and 1.24, the results still highlight that the combination of minimax adaptations was largely unsuccessful. This provides contrary findings to the study in the game hearts by Zuckerman et al which found the combination of algorithms had substantial improvements (Zuckerman et al., 2009). The results from this tournament does not challenge the finding by Zuckerman et al, but instead shows room for possible design and execution improvements of these mixed agents. This area for improvement could start with an adaptation to method that selects which algorithm to employ. A genetic algorithm to choose the most effective threshold value would likely yield more desirable results then the default value used.

Bibliography

Atashpendar, A., Schilling, T., & Voigtmann, T. (2016). Sequencing chess. *Europhysics letters, 116*(1), 10009. doi:10.1209/0295-5075/116/10009

International journal of game theory (Online). (1971). *International journal of game theory (Online)*.

John, S. (2017). *A cultural history of chess-players: Minds, machines, and monsters*. Manchester: Manchester University Press.

Krantz, S. G. (2015). The MiniMax Theorem. In: Chapman and Hall/CRC.

Luckhardt, C. A., & Irani, K. B. (1986). *An algorithmic solution of N-person games*. Paper presented at the Proceedings of the Fifth AAAI National Conference on Artificial Intelligence, Philadelphia, Pennsylvania.

Schadd, M. P. D., & Winands, M. H. M. (2012). Best Reply Search for Multiplayer Games. *IEEE Transactions on Computational Intelligence and AI in Games, 3*(1). doi:10.1109/TCIAIG.2011.2107323

Sturtevant, N., & Korf, R. (2000). *On Pruning Techniques for Multi-Player Games*.

Zuckerman, I., Felner, A., & Kraus, S. (2009). *Mixing search strategies for multi-player games*. Paper presented at the Proceedings of the 21st international jont conference on Artifical intelligence, Pasadena, California, USA.