Search Algorithms for a Blokus AI

Trieu Ngoc Hoang, Carlos Soto Garcia-Delgado, Martin Michaux, Simon Hilt, Bas Göritzer, and Gyumin Oh

Department of Data Science and Artificial Intelligence
Maastricht University

January 18, 2021

Abstract

This study explores the use of different search strategies when designing an AI that can intelligently play Blokus. Blokus is a well-known multi-player board game with possible exponential moves. Strategies such as Genetic Algorithms, Monte-Carlo Simulations, and MiniMax search are implemented. The performance of these search strategies is tested in different games with perfect information by running self-play experiments and a time limit. It can be argued that Monte Carlo simulations combined with heuristics given by a Genetic Algorithm perform best.

1 Introduction

The first concepts of Artificial Intelligence (AI) were founded around 1950, and Elaine Rich, a well-known computer scientist, summarize the first decades as a study of creating machines that do better than people (Ertel & Black, 2017, pp. 1-3). This article compares different AI solutions playing the board games Blokus. These solutions are based on the three methods: the Genetic Algorithm, Monte Carlo Simulations, and the Minimax. The strength of each algorithm is determined by competition and scoring points for a win and draw. The goal is to find the best-suited algorithm for Blokus, including combinations of the algorithms mentioned earlier.

Blokus is a strategic board game where each player has the full knowledge of the game in-game state. Besides, players can play random moves; there is no randomness in the game, making it solvable with algorithms without any expectation. This means that theoretically, each state's best move can be determined because all the information is present, and no luck is involved. The game can be played from two to four-players in different game variants but the goal, to occupy as much area as possible of the board, stays the same for each variant and player.

The game rules and possible strategies for playing Blokus will be explained in chapter two. Likewise, the basis for the considered algorithms is laid, and known Blokus AIs will be reviewed. The third section is about the applications of different AIs to the game. One AI is implemented on the based on the Genetic Algorithm (GA), two with Monte Carlo (MC) Simulation and two for the MiniMax. Further two combinations of the GA and the MC Simulation are implemented, the optimal heuristic calculated by a GA guides the MC simulations by selecting the best moves to try. These combined algorithms were not yet known and thus, newly implemented. In section four, the foundation and motivation for the different experiments in the form of competition are laid. The results of the experiments are presented and discussed in section five. The sixth and final section will sum up the work combined with an outlook of further research.

2 Blokus Strategies and Existing Agents

The first part of this section is about the rules and strategies of the game Blokus itself and the second part examines already existing algorithms for Blokus.

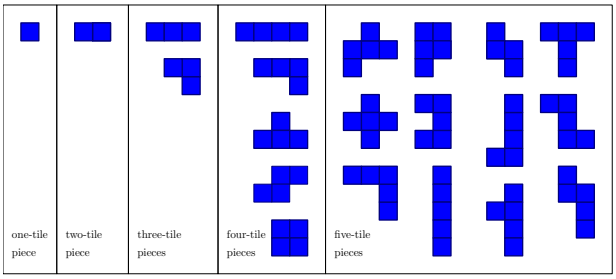

Figure 1: The 21 Blokus Pieces

2.1 About Blokus

Blokus can be played in many different ways; the two variants considered and implemented are the following:

- two-players two-colours (Board size 16x16 tiles)
- four-players four-colours (Board size 20x20 tiles)

For the algorithms' testing and competing algorithms, the 'two-players' variant is important because any algorithms' inter-dependencies can be excluded. Nevertheless, the 'four-player' variant will lead to more complexity due to two additional players and board size (256 tiles to 400 tiles) and, therefore, adapted handling of the AIs. The detailed rules of Blokus can be found in the appendix A.

2.1.1 Strategies to Play

For Blokus, there are a few strategies that can be followed to increase the chance of winning. These strategies will be explained in the following section.

Placing largest piece: In respect of the game's scoring (minus point for every tile left in the base of a player), a strategy that can be played in all

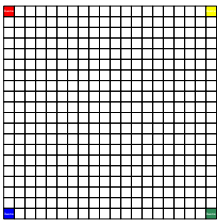

Figure 2: Start positions for 'four-players four-colours' variant

game states is to place the most extensive piece possible. This strategy reduces every round the maximum possible tiles from the base. Also, as the game progresses, more and more tiles of the board will be occupied by pieces, and there is less space to place more significant pieces (Jahanshahi et al., 2013, pp. 150-151).

Closest to the middle: A possible strategy can be to place the pieces, preferably in the middle. The player can place different pieces on the board from the middle and play other strategies because the board edges are as far away as possible (Demertzis & Nikolakaki, 2014, pp. 4-6).

Most compatible corners: Another strategy is by playing pieces so that the player has the most compatible corners after this move. This strategy increases the chance of possible moves for the next round and increases flexibility. A player always has to consider that the game ends if there is no compatible corner in the next round. The compatible corners are visualized in figure 3 (Jahanshahi et al., 2013, pp. 150-151).

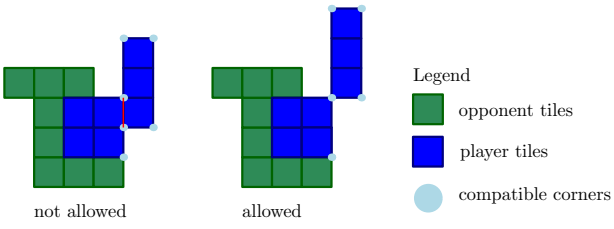

Figure 3: Compatible Corners

Blocks most corner: Based on the previous strategy, another option is to block most of the opponents' compatible corner. However, this strategy is for an advanced state of the game, when pieces of different players are in range. The strategy aims to place a piece such that the opponent(s) have less compatible corners for their next turn. This strategy limits the opponent's players' possibilities and, in the best case, makes the opponent unable to move any further (Jahanshahi et al., 2013, pp. 150-151).

Farthest away from starting point: This chapter's fifth strategy is to place the tiles as far as possible from their starting point. This strategy is most likely a strategy for the beginning of the game. This strategy has the goal of playing large pieces and occupy a large region of the board. Thereby, the player's compatible corners are harder to block, and the player can spread out in other board regions (Demertzis & Nikolakaki, 2014, pp. 4-6).

2.2 Existing Blokus Agents

Existing programs with AI agents for Blokus are Blockem([1,2]), Metablock([3]), Freeblocks3D([4]), Blokish([5]) and Pentobi([6,7]).

Blockem works with a MiniMax algorithm and alpha-beta pruning. The depth of the search is limited to three until the player has less than fourteen pieces. The AI is only available for the 'two-players' 'two colours' variant. **Metablocks** strongest agent is a Minimax algorithm with a five-piece hand for the opening. **Freebloks3D** supports a two-player variant of a 16x16 board and the variant 'two- players two colours' on a 20x20 board, but no information about the algorithms used can be found. There is also no information available about the **Blokish** agents. Nevertheless, the **Pentobi** agent seems to be the most advanced of available Blokus implementations. It also has many different game variants and AIs for these different variants. Pentobi uses opening books, such that the first few pieces are prescribed, but there is no documentation about it (Demertzis & Nikolakaki, 2014, p. 6).

[1] https://github.com/frechilla/blockem
[2] http://blockem.sourceforge.net/
[3] https://code.google.com/archive/p/metablok/
[4] https://github.com/shlusiak/Freebloks-Android
[5] https://github.com/scoutant/blokish
[6] https://pentobi.sourceforge.io/
[7] https://github.com/enz/pentobi

3 Implementation

In this section the different algorithms will be introduced and the implementation will be discussed.

3.1 Genetic Algorithm

The Genetic Algorithm aims to give the optimal weights to different strategies that will lead to optimal decision making. Every time a move is to be chosen, five different strategy-weights are considered (see section 2.1.1 for the strategies).

The population is composed of individuals; each individual is a bot that plays with different (initially random) weights for the strategies. All the individuals in the population play against each other. The winners of every round are matched and reproduce. The new individuals that will form the population will contain the winners' weights with some mutation added (Wirsansky, 2020, pp. 9-13).

As it can be observed, Blokus needs to be played differently at the beginning and the end of the game. Therefore, the game has been divided into three phases (beginning, middle and ending). Both the weights and the splitting point between phases have been calculated using a genetic approach. As a result, the bot has different weights of each strategy for each phase (specific weights and phases are in appendix B).

For optimal performance, the algorithm was trained once to play four-players games on a 20x20 board, and it was also trained for two-players game on a 16x16 board.

However, it can be observed how the bots always try to place the most prominent pieces first, keeping the small ones for the ending. In the beginning, it also tries to aim at the middle of the board and place the most corners. At the middle stage, the bot aims to put pieces far away from where it started to conquer

more space while keeping the right balance between adding and blocking. The ending phase primarily focuses on blocking the most corners from other players while placing the pieces on the previously built space.

This algorithm's strength is that it is greedy; it calculates the best move given at the current situation and not considering future moves. Thus it can calculate a move almost instantly.

Functionality was also added to this algorithm, where it also considers the best move the next player can do (unless, of course, it reached the deepest level of the tree), and the score of that best move for the next player is returned. Then that score is subtracted from the move it began with. This functionality can be turned on and off. If the GA heuristics are to be used as an evaluation function, this functionality is turned off.

3.2 Monte Carlo methods

3.2.1 Monte Carlo Simulation

Blokus is a combinatorial game where each player follows his strategy. Monte Carlo uses randomness for deterministic problems, which are difficult or impossible to solve using other approaches. Monte Carlo simulations' key feature is that it models complete game scenarios and estimates how likely a resulting outcome is. This procedure can be applied to any game whose positions necessarily have a finite number of moves and finite-length (J. A. M. Nijssen, 2013). All feasible moves are determined for each position: several random games are played out to the very end, and the scores are recorded. This process is executed repeatedly, as many times as possible, given that there are usually time or resource limitations in real games. The move leading to the best score is chosen.

Typically, hundreds or thousands of Monte Carlo simulations are performed in a single the game round, each time using different randomly selected values throughout the game. By the end of the procedure, a large number of recorded result-initial move pairs is obtained. Those are used to calculate the probability of reaching various outcomes.

One MC simulation's time complexity is linearly dependent on the number of players and the number of pieces/turns left. This algorithm can be run at the desired time as it aims to compute an approximation by sampling. In the time limit set, the algorithm will make as many simulations as it can. Therefore, the more time it is given, the better the answer will be. The number of explored moves will grow as the game advances. This is because for the same time limit, simulating games until the end will take considerably less than at the beginning. The move to consider is closer to the terminal state, which will reduce the time taken to reach that state, and therefore the number of explored moves will increase exponentially.

3.2.2 Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is a heuristic search algorithm for decision processes (Szepesvári, 2008). The algorithm consists of four steps: selection, expansion, roll out, and backpropagation. First, the algorithm selects the most promising leaf node using the UCB-1 formula. If the node has already been visited, the algorithm adds some children to it and rolls out on the children. Roll-outs follow the same principle of the Monte Carlo search's simulation. If the selected node has never been visited, roll out is performed directly starting from the node. The results then back-propagate to not only the leaf node but also to all of its ancestors (van den Herik et al., 2008, p. 1, p. 4).

For both MC and MCTS, a full expansion approach will not meet the time constraints as it will take too long. If a random expansion is implemented, performance considerably fluctuates. Therefore, the need to implement heuristics to guide the search or expansion arose. These heuristics were defined following the strategies described in section 2.1.1.

3.2.3 Combined Genetic and Monte Carlo (GAMC)

The combination of both the GA and the MC uses each algorithm's strength. The GA, due to its three phases and weights, can preselect certain pieces and locations. This is done by evaluating how good each

possible move is; using the GA heuristics; every move is given a score. The MC algorithm then performs simulations on the best scoring moves to determine the best possible move. Hence, the search focuses on moves, which are, in principle, better than the others. The search is enhanced for the previously described approach.

3.2.4 Combined Monte Carlo Tree Search and Genetic Algorithm (GAMCTS)

The same logic concept implemented in section 3.2.3 is applied to the MCTS. The moves are rated according to the GA Heuristics. Then the nodes corresponding to the best scoring moves are expanded and explored. This leads to a more effective search.

3.3 Minimax Algorithm

Created by Von Neumann and Morgenstern in 1944, the Minimax algorithm is the classic depth-first search technique for sequential two-player games. The search tree is created by recursively expanding all nodes from the root in a depth-first way until either the end of the game or the maximum search depth is reached. In this case, the maximum depth corresponds to the number of a turn multiplied by the number of players (Jones, 2008, pp. 92-93).

However, in Blokus, games are up to four-players; hence variations of the classical MiniMax Algorithm, such as the two following, were implemented to search in multi-player Games (J. Nijssen & Winands, 2013).

In both, each leaf node's heuristic score is calculated with the same phases, strategies, and weights used and trained by the GA.

This algorithm's time complexity is $O(b^d)$, and the space complexity is $O(bd)$, where b is the number of moves to consider at a given position of a board, and d is the depth of the search. These are time complexities without pruning; however, the Max^n search uses shallow pruning and the Paranoid search prunes using α-β pruning.

Moreover, both searches use the killer moves strategy, which is a dynamic move ordering technique. It stores a list of nodes, being limited, used to cut-off other nodes, and their occurrence to keep the best one first. When a new killer move is found, it replaces the move that currently does not cause a cut-off. While expanding a node, the killer moves are being first used and then the remaining ones.

But, since the Blokus game has a huge branching factor and the Minimax algorithm considers everything, the performance of the two searches is very poor.

3.3.1 MiniMax Max^n Search

This pruning technique is a modification of the Minimax pruning search to multi-player games. At each leaf node, each player is awarded a value based on a heuristic evaluation function in the search tree's leaf nodes. These scores are stored in a list of size N, where N is the number of players. The sum of all players' values can be constant and is called the "U" value. It is used in shallow pruning to cut off nodes and increase the algorithm's performance. When backtracking the tree's values, each player always chooses the move that maximizes his score.

3.3.2 MiniMax Paranoid Search

This search is the most similar to the known alpha-beta pruning for two-players games. It assumes that all opponents are in a single team against the root player. This assumption can be reduced to a two-player game where the root player is represented in the tree by MAX nodes and MIN nodes' opponents. This assumption's advantage is $\alpha - \beta$ like deep pruning is possible in the search tree, allowing more in-depth searches in the same amount of time. It can be argued that this search is not a consistent strategy with the goal of Blokus.

4 Experiment Motivation

Based on the enhancements already described in the previous section, some variations need to be evaluated due to experiments within one AI method. In the experiments, different AI's play against each other several times. The winning rate will determine

the strength of the bot. Bots moving first may have more space relative to other players, and therefore the result of the experiments could be biased towards them. This is why bots alternate their order in subsequent games.

There are two parts to the experiments. In the first part, the strength of the variations is determined. In the second part, the best variations are playing against each other to ascertain the best AI.

To determine the best player for two-player mode, the best AIs of each method will play against each other (see section 5.4). To determine the strength of the AIs in four-player mode, each variant of the two best methods will play against each other (see section 5.5).

4.1 Monte Carlo Methods

Concerning the Monte Carlo Method, there are two approaches implemented. The difference between the MCTS and the MC is the tree building. From the experiments, a time limit was set for selecting a move. The same time limit is set for both bots, such that no bot has an advantage over the other one. This way, it can be shown which search technique is better for Blokus.
Further experiments with different time limits have been conducted to see the performance trend of both approaches.

4.2 Genetic Algorithm

The weights for every strategy were differently calculated for two and four-players games. The algorithm was used to calculate phases and weights separately, leading to different bots' behaviour when playing on a 16x16 board a two-players game than playing on a 20x20 board a four-players game. The AI will play significantly less intelligently when not given the weight corresponding to its type of game.

4.3 Combination of Genetic Algorithm and Monte Carlo Methods

Since there are two algorithms based on the Monte Carlo Method, there is a combination of them with the GA. In both of them, the best moves are explored according to GA heuristics. In the experiments, these two algorithms are compared. Again, different time limits are set for both to see a performance trend. The GAMC and GAMCTS start by exploring the best ten moves according to the GA heuristics. The number of moves approaches the range of a few hundred as the game advances.

4.4 MiniMax

The following experiments test the behaviours of the Max^n and Paranoid searches that are different in how they consider the opponents, use the killer moves and prune the nodes when trying to tackle multiplayer problems, however, this is not relevant in two-players game.

In the table 4, it was chosen to make them play against each other in the four-player mode, and in the table 3, the games played between the two searches and the other algorithms have only been considered.

Regarding the parameters, the number of turns taken into account cannot be high due to the high branching factor so that the algorithm will go through all the moves for each player in one round. If a more significant depth were set, the algorithm would take too much time. Both searches use the killer moves strategy, and the limited number of killer moves stored can be modified.

Moreover, in order to be able to perform games in four-players mode, the board is decreased to the size of 10x10 and the number of pieces to nine.

4.5 Game with existing Blokus Agent

To evaluate the best algorithms' strength, each method's best algorithm will play against an already existing Blokus agent. Since both MiniMax variations take too much time, only the GA, MC and GAMC are included. As introduced in section 2.2 the currently strongest Blokus agent is Pentobi. Pentobi itself can be set to different strength levels. Pentobi will be set to the maximum level (seven)

5 Experiment Results

In this section the results of the previous motivated experiments are displayed and discussed.

5.1 Monte Carlo Methods Results

This section is about the competition of the MC and the MCTS for different time limits.

time limit	**Wins P1 MC**	**Wins P2 MCTS**	**Draws**
1	14	9	2
3	16	8	1
10	8	0	2
30	8	2	0
time limit	**Wins P1 MCTS**	**Wins P2 MC**	**Draws**
1	9	16	0
3	11	14	0
10	7	3	0
30	5	5	0

Table 1: Influence of time limit on MCTS and MC (time limit in seconds)

Concerning the results in table 1, the MC algorithm has a win rate of 60 percent on average playing 140 games in the two-player variant against the MCTS. The time limit does not seem to have a significant impact on both players. However, being the first player seems to be a small advantage since the MCTS and MC perform slightly better when they play first.

The MC algorithm's edge can be explained because the simulation number is more remarkable when not building the tree. Each time the MCTS has to expand a node, it needs to evaluate moves to expand, and by doing so, diminishes the time for simulation, even though it should have a more powerful result.

5.2 Combined GA and MC Methods Results

This section is about the competition of the two combined algorithms composed of the GA and the two Monte Carlo methods.

time limit	**Wins P1 GAMC**	**Wins P2 GAMCTS**	**Draws**
1	14	9	2
3	17	8	0
10	7	3	0
30	8	1	1
time limit	**Wins P1 GAMCTS**	**Wins P2 GAMC**	**Draws**
1	7	17	1
3	5	19	1
10	1	8	1
30	0	10	0

Table 2: Influence of time limit on GAMCTS and GAMC (time limit in seconds)

Overall the GAMC (win rate of circa 71 percent) seems to be stronger than the GAMCTS compared in two-player mode. The difference between the GAMCTS and GAMC seems to get bigger the more time both have to simulate games. As expected these results are strongly linked to the results of the Monte Carlo methods.

5.3 MiniMax Searches Results

time limit	**Wins** Max^n	**Wins MC**	**Wins GA**	**Wins GA-MCTS**
1	3	1.666	4.335	1
3	2.1667	9.25	2	6.583
10	2	11.666	1	5.335
30	4.667	11	1	3.335
time limit	**Wins Paranoid**	**Wins MC**	**Wins GA**	**Wins GA-MCTS**
1	2.667	12.5	2.25	2.583
3	1.334	17	0.334	1.334
10	2.665	13	2.667	1.667
30	2.333	10.998	3.333	3.333

Table 3: Experiments on both Minimax searches against other algorithms implemented

depth limit	Wins four-players Max^n	Wins two-players Paranoid	Draws
4	4	2	0

Table 4: Every possible crossing games between the 2 searches (6 in total) have been experienced considering a board of size 10 and the 9 pieces for each player in order to be able test those two algorithms

As expected, the Max^n performed better than the Paranoid search since the latter is not consistent with Blokus.

5.4 Method Comparison two-player Mode

The inner cells of the following table 5 contains the win rate for the first player which is displayed in the first column.

1st \ 2nd	MC	GA	GAMC	MaxN
MC		10/10	8/10	9/10
GA	1.5/10		0/10	1/1
GAMC	4.5/10	10/10		9/10
MaxN	1/10	0/1	0/10	

Table 5: Winrate two-Player

Based on the two-player games performed, the MC is the strongest with a win rate of 90 percent as the first player and around 76 percent as the second player.

5.5 Method Comparison four-player Mode

MC	MCTS	GAMC	GAMCTS
34.5	30.5	32.5	22.5

Table 6: Wins four-Player

Comparing all Monte Carlo variations against each other in all possible combinations in four-player mode with five games each time, the MC, MCTS and GAMC seem to be similarly strong.

5.6 Pentobi Results

In table 7 the number of wins are displayed. The detailed results can be seen in table 23 in the appendix C.

Pentobi	MC	GAMC	GA
3	1	0	0

Table 7: Wins in 4 games

On the one hand, winning one of four games shows that these AIs are going in the right direction, on the other hand, there is still room for improvement.

6 Conclusion

Based on the three chosen methods, a guided MC seems to be the strongest. Since Blokus is a game with a high branching factor, the two MiniMax variations performed poorly. The GA places its pieces based on partly static heuristics and can neither react to other players' moves, nor plan ahead. However, the weights and phases developed by the genetic approach are beneficial for the MC in order to guide the algorithm.

To further enhance the MC based AI, a new heuristic can be implemented, such that the AI does not place pieces in positions where no other player can currently place any piece. Therefore the AI aims to place pieces in contested positions. This strategy can also be considered to enhance the GA.

Due to the limits of the GA decision-making, a Neural Network could be implemented and trained with reinforcement learning to play Blokus. Theoretically the best reply/move for each state can be determined without encoding arbitrary heuristics in a Neural Network. Therefore a Neural Network will find a move in a relatively short amount of time. The Neural Network can also be superior to the Monte Carlo methods.

References

Demertzis, I., & Nikolakaki, S. M. (2014). *A hardware-based minimax implementation of a blokus duo agent reconfigurable digital systems - technical report.* Technical University of Crete, Greece.

Ertel, W., & Black, N. (2017). *Introduction to artificial intelligence.* Springer International Publishing.

Jahanshahi, A., Taram, M. K., & Eskandari, N. (2013). *Blokus duo game on fpga.* doi: 10.1109/CADS.2013.6714256

Jones, M. (2008). *Artificial intelligence: A systems approach: A systems approach.* Jones & Bartlett Learning.

Nijssen, J., & Winands, M. H. (2013). *Search policies in multi-player games.* ICGA Journal. doi: 10.3233/ICG-2013-36102

Nijssen, J. A. M. (2013). *Monte-carlo tree search for multi-player games.* Maastricht University.

van den Herik, H., Xu, X., Ma, Z., & Winands, M. (2008). *Computers and games: 6th international conference, cg 2008 beijing, china, september 29 - october 1, 2008. proceedings.* Springer Berlin Heidelberg.

Wirsansky, E. (2020). *Hands-on genetic algorithms with python : applying genetic algorithms to solve real-world deep learning and artificial intelligence problems.* Birmingham, UK: Packt Publishing.

Appendices

A Blokus rules

Those rules are summarized from the instructions of the Mattel branded Blokus game for two and four-players. There are four colours (red, yellow, green, and blue) to distinguish between the players.

Game setup

1. Each player is assigned 21 different shaped pieces of the same colour shown in figure 1.
2. The board size is automatically chosen in respect of the variant.
3. The first pieces of every colour have to cover the field right in the corner; this field is marked with 'here' in the player's colour.
4. Every piece can be mirrored and rotated in any direction of the player's needs.
5. colours are played in this order: red starts, then yellow, green, blue are following.

During the game

6. Pieces of the same colour have to touch at least one corner, but only on the corners (no flat edges of the same colour pieces can touch). The corners that allow placing a further piece in the next turn are called 'compatible corners' (see figure 3).
7. Pieces placed on them are fixed and can not be moved during the further game.
8. A player must pass his turn if the player cannot place one of his remaining pieces on the board (will happen automatically by the logic).
9. The game ends when all players cannot place any other piece on the board or if all pieces are placed on the board. A pop-up window will show the score for each player.

Scoring

10. Every player gets -1 point for every unit of tiles of their remaining pieces.

11. If every piece is placed on the board, the player gets +15 points and additional +5 points if the last piece placed was the 1x1 piece.

12. The player with the highest score wins the game (round).

B Weights and phases Genetic Algorithm

Strategies:

1. Most compatible corners
2. Blocks most corners
3. Closest to the middle
4. Placing the largest piece
5. Farthest away from starting point

The first column of the table represents the total number of pieces which need to be placed until a phase begins. Every row represents a phase.

4P	**1**	**2**	**3**	**4**	**5**
0	0.53125	0.38739	0.63102	0.80074	0.43989
5	0.03258	0.21644	0.56497	0.89280	0.68175
12	0.06342	0.56151	0.09996	0.94024	0.01381
2P	**1**	**2**	**3**	**4**	**5**
0	0.91887	0.01297	0.09991	0.19596	0.41667
7	0.35297	0.98731	0.05032	0.73727	0.01405
12	0.29614	0.87144	0.40821	0.85131	0.49246

Table 8: Weights and Phases of Genetic Approach

C Detailed result-tables

Game	P1: MC	P2: MCTS	P1: MCTS	P2: MC
1	-19	-25	-30	-20
2	-16	-13	-30	-20
3	-28	-22	-22	-21
4	-26	-19	-30	-19
5	-17	-25	-36	-16
6	-11	-20	-13	-25
7	-18	-24	-15	-14
8	-20	-21	-28	-30
9	-19	-24	-21	-18
10	-28	-20	-34	-20
11	-16	-12	-22	-19
12	-11	-21	-24	-11
13	-17	-17	-20	-30
14	-22	-20	-22	-16
15	-14	-45	-18	-14
16	-16	-17	-13	-37
17	-19	-24	-25	-26
18	-20	-22	-19	-13
19	-25	-34	-44	-15
20	-15	-35	-16	-18
21	-21	-20	-21	-12
22	-22	-25	-13	-21
23	-35	-24	-21	-26
24	-17	-17	-26	-24
25	-26	-23	-16	-21

Table 9: MCTS VS MC (time limit 1 seconds)

Game	P1: MC	P2: MCTS	P1: MCTS	P2: MC
1	-18	-26	-12	-22
2	-21	-24	-32	-15
3	-17	-28	-32	-15
4	-21	-23	-24	-32
5	-27	-21	-30	-12

Table 10: MCTS VS MC (time limit 60 seconds)

Game	P1: MC	P2: MCTS	P1: MCTS	P2: MC
1	-15	-23	-17	-21
2	-21	-25	-22	-26
3	-24	-14	-36	-16
4	-17	-20	-25	-19
5	-15	-26	-15	-13
6	-33	-21	-25	-27
7	-21	-25	-23	-19
8	-21	-18	-27	-24
9	-18	-22	-13	-21
10	-25	-23	-19	-18
11	-13	-20	-36	-28
12	-22	-22	-14	-21
13	-16	-20	-17	-19
14	-20	-21	-17	-21
15	-28	-18	-39	-20
16	-28	-15	-21	-13
17	-26	-30	-16	-19
18	-18	-21	-19	-26
19	-18	-24	-31	-22
20	-37	-19	-16	-25
21	-14	-15	-20	-17
22	-32	-21	-21	-19
23	-15	-20	-30	-25
24	-15	-17	-22	-23
25	-19	-21	-25	-15

Table 11: MCTS VS MC (time limit 3 seconds)

Game	P1: MC	P2: MCTS	P1: MCTS	P2: MC
1	-25	-26	-21	-26
2	-21	-23	-19	-18
3	-19	-25	-17	-21
4	-23	-23	-24	-18
5	-15	-23	-19	-25
6	-21	-25	-23	-41
7	-19	-27	-23	-25
8	-17	-25	-25	-28
9	-21	-25	-12	-14
10	-17	-17	-25	-21

Table 12: MCTS VS MC (time limit 10 seconds)

Game	P1: MC	P2: MCTS	P1: MCTS	P2: MC
1	-18	-22	-27	-25
2	-17	-20	-18	-26
3	-31	-20	-14	-24
4	-14	-21	-19	-12
5	-24	-28	-29	-20
6	-22	-25	-25	-17
7	-26	-17	-22	-23
8	-18	-26	-22	-25
9	-15	-25	-13	-25
10	-15	-27	-24	-23

Table 13: MCTS VS MC (time limit 30 seconds)

Game	P1: GAMC	P2: GA-MCTS	P1: GA-MCTS	P2: GAMC
1	-19	-13	-23	-21
2	-17	-20	-17	-13
3	-16	-20	-20	-14
4	-15	-18	-12	-15
5	-17	-20	-13	-19
6	-15	-20	-14	-10
7	-13	-13	-20	-16
8	-21	-20	-24	-25
9	-44	-24	-7	-20
10	-10	-17	-18	-16
11	-17	-13	-18	-23
12	-17	-16	-20	-16
13	-14	-19	-32	-26
14	-13	-18	-18	-15
15	-14	-13	-15	-12
16	-17	-20	-19	-9
17	-21	-20	-24	-16
18	-20	-19	-26	-21
19	-9	-12	-21	-19
20	-18	-18	-20	-21
21	-27	-16	-34	-22
22	-11	-18	-19	-22
23	-23	-21	-18	-14
24	-9	-19	-21	-12
25	-13	-17	-20	-20

Table 14: GAMCTS VS GAMC (time limit 1 seconds)

Game	P1: GAMC	P2: GA-MCTS	P1: GA-MCTS	P2: GAMC
1	-14	-20	-26	-23
2	-13	-23	-24	-13
3	-23	-24	-15	-12
4	-22	-19	-17	-20
5	-23	-20	-19	-18
6	-23	-20	-13	-17
7	-13	-14	-21	-20
8	-16	-23	-24	-16
9	-17	-13	-16	-16
10	-9	-17	-31	-16
11	-17	-21	-16	-10
12	-16	-13	-20	-23
13	-13	-39	-21	-14
14	-19	-14	-24	-21
15	-14	-26	-17	-27
16	-27	-29	-21	-13
17	-22	-25	-17	-23
18	-17	-14	-26	-23
19	-18	-24	-18	-11
20	-21	-25	-18	-13
21	-12	-21	-17	-14
22	-22	-25	-23	-17
23	-18	-17	-21	-16
24	-15	-16	-14	-9
25	-16	-18	-19	-18

Table 15: GAMCTS VS GAMC (time limit 3 seconds)

Game	P1: GAMC	P2: GA-MCTS	P1: GA-MCTS	P2: GAMC
1	-14	-28	-21	-15
2	-19	-26	-13	-11
3	-8	-18	-18	-17
4	-24	-27	-19	-17
5	-26	-20	-20	-12
6	-14	-9	-16	-10
7	-19	-17	-23	-23
8	-12	-20	-11	-26
9	-16	-20	-32	-18
10	-13	-20	-17	-15

Table 16: MCTS VS GAMC (time limit 10 seconds)

Game	P1: GA-MCTS	P2: GAMC	P3: GA-MCTS	P4: GAMC
1	-15	-9	-13	-23
2	-9	-4	-9	-28
3	-21	-12	-20	-3
4	-14	-5	-4	-13
5	-13	-22	-13	-20
Game	**P1: GAMC**	**P2: GA-MCTS**	**P3: GAMC**	**P4: GA-MCTS**
1	-19	-5	-20	-8
2	-9	-12	-17	-14
3	-9	-15	-5	-35
4	-15	-29	-8	-7
5	-11	-21	-17	-24

Table 17: GAMCTS VS GAMC 4 Players (time limit 3 seconds)

Game	P1: GA-MCTS	P2: GAMC	P3: GA-MCTS	P4: GAMC
1	-11	-15	-13	-30
2	-13	-4	-18	-18
3	-4	-20	-16	-12
4	-8	-16	-13	-27
5	-13	-17	-21	-5
Game	**P1: GAMC**	**P2: GA-MCTS**	**P3: GAMC**	**P4: GA-MCTS**
1	-17	-19	-13	-13
2	-17	-24	-20	-22
3	-10	-27	-21	-13
4	-12	-16	-13	-15
5	-14	-16	-20	-12

Table 18: GAMCTS VS GAMC 4 Players (time limit 10 seconds)

Game	P1: GAMC	P2: GA	P1: GA	P2: GAMC
1	-11	-23	-32	-29
2	-13	-31	-17	-16
3	-21	-22	-21	-13
4	-13	-17	-32	-22
5	-22	-27	-19	-9
6	-15	-19	-36	-19
7	-13	-18	-25	-14
8	-11	-34	-20	-16
9	-14	-18	-28	-11
10	-17	-18	-23	-22

Table 19: GA VS GAMC (time limit 3 seconds)

Game	P1: MC	P2: GAMC	P1: GAMC	P2: MC
1	-17	-19	-15	-18
2	-20	-23	-23	-22
3	-23	-21	-28	-20
4	-21	-23	-26	-21
5	-19	-34	-8	-16
6	-25	-19	-14	-21
7	-19	-30	-22	-22
8	-21	-29	-26	-30
9	-21	-27	-26	-14
10	-9	-21	-23	-22

Table 20: MC VS GAMC (time limit 3 seconds)

Game	P1: MC	P2: GA	P1: GA	P2: MC
1	-19	-21	-22	-13
2	-11	-30	-21	-23
3	-12	-27	-20	-12
4	-16	-24	-35	-30
5	-28	-34	-25	-15
6	-14	-24	-24	-23
7	-12	-14	-20	-18
8	-12	-22	-16	-16
9	-19	-21	-29	-17
10	-15	-22	-38	-18

Table 21: MC VS GA (time limit 3 seconds)

Game	P1: GAMC	P2: GA-MCTS	P1: GAMC	P2: GA-MCTS
1	-13	-23	-25	-14
2	-23	-30	-22	-18
3	-15	-17	-17	-15
4	-16	-30	-26	-14
5	-20	-17	-21	-15
6	-15	-22	-19	-10
7	-17	-17	-20	-19
8	-8	-18	-19	-15
9	-20	-21	-17	-8
10	-17	-22	-27	-22

Table 22: GAMCTS VS GAMC (time limit 10 seconds)

Game	P1: MC	P2: GAMC	P3: GA	P4: Pentobi
1	-17	-34	-21	-3
Game	**P1: Pentobi**	**P2: MC**	**P3: GAMC**	**P4: GA**
1	-9	-4	-20	-18
Game	**P1: GA**	**P2: Pentobi**	**P3: MC**	**P4: GAMC**
1	-13	-4	-32	-28
Game	**P1: GAMC**	**P2: GA**	**P3: Pentobi**	**P4: MC**
1	-26	-33	20	-21

Table 23: Games against Pentobi Results