An Ensemble-method for uninformed rollout heuristics in MCTS

Bachelorarbeit

Nathan Valenti

TECHNISCHE UNIVERSITÄT DARMSTADT

Technische Universität Darmstadt

Fachbereich Informatik

Fachgebiet Knowledge Engineering

Prof. Dr. Johannes Fürnkranz

Betreuer: Christian Wirth

Bachelorarbeit zu dem Thema:

An Ensemble-method for uninformed rollout heuristics in Monte Carlo Tree Search

Bearbeitet von: Nathan Valenti

Matr.-Nr.: 2459015

Studiengang: Wirtschaftsinformatik (B.Sc.)

Eingereicht am: 1.08.2016

Förmliche Erklärung

Hiermit erkläre ich, Nathan Valenti, geboren am 11.02.1994, an Eides statt, dass ich die vorliegende Bachelorarbeit ohne fremde Hilfe und nur unter Verwendung der zulässigen Mittel sowie der angegebenen Literatur angefertigt habe.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Darmstadt, den 1.08.2016

(Unterschrift)

Contents

1 Abstract

In General game playing artificial intelligences try to play various different games without human guidance. Naturally, there are many challenges in this field, foremost the absence of domain specific knowledge since the agents have to be applicable to conceptually very different types of games.
Monte Carlo Tree Search has proven to be a strong algorithm for this domain. An important aspect of this algorithm are the random rollouts which are used to evaluate a board state. Therefore, numerous enhancements have been proposed to improve the rollout policy. The goal of this thesis is to combine four of these enhancements into an ensemble and to test if this yields significant performance gains.
It will be shown that one of the enhancements, called MAST, is solely accountable for the performance of the entire ensemble but it shows a stronger performance when mixed with random rollouts on some games. Additionally, it will be shown that these performance gains are partly attributed due to shorter, more accurate simulations.

2 Introduction

In recent years there has been a strong development in the field of artificial intelligence. One of the most recent milestones is Google's AlphaGo beating the current human world champion of Go on a standard board [1].

While this agent is an expert in the field of Go, it does not know how to play any other game or perform any other task different from playing Go. More valuable in the search for a general artificial intelligence are thus agents suited to perform multiple different tasks, respectively games, while the information about the specific game at hand is provided at runtime. This is done in the domain of General Game Playing (GGP) [2].

A strong algorithm in this domain has proven to be Monte Carlo Tree Search (MCTS). The algorithm subsequentially builds a game tree, thereby evaluating individual game states with the use of random rollouts. Monte Carlo Tree Search has been a breakthrough for artificial intelligences in the Game of Go and has brought improvements in various other domains [3]. Additionally, some of the strongest GGP-agents use Monte Carlo Tree Search [4, 5].

Much of the success of MCTS is due to using random rollouts to evaluate positions for which no heuristic value exists. Like any other part of the algorithm, there are several ways to refine these rollouts to incorporate knowledge learned about the game. These enhancements differ in the way they gather their knowledge, ranging from the incorporation of prior domain specific knowledge to just using generic features of the game [3].

As these enhancements work differently and thus have strengths and weaknesses in different settings, the goal of this paper is to see whether the combination of several of these enhancements yields for a more stable and overall stronger agent. Therefore, four different enhancements are taken and combined into an ensemble using the simple voting mechanism. The enhancements are chosen to only use generc features of the game as to retain general applicability and not to use domain-specific knowledge. This ensemble is then tested on 8 two-player games to see whether the combination proves beneficial and why.

The remainig paper is organised as follows: Chapter 3 introduces several important concepts and gives an in-depth elaboration of the Monte Carlo Tree Search algorithm and the relevant enhancements. Next, in chapter 4, details are presented about the ensemble method which is used in this thesis. Chapter 5 shows the experimental setting, whose results are presented and discussed in chapter 6. Chapter 7 ends the thesis with the conclusion.

3 Basics

In this chapter the basic concepts are presented and explained. First, an overview over the field of General Game Playing is given. Next, the Monte Carlo Tree Search algorithm is explained in detail. Thereafter, the four enhancements of the random rollouts that have been chosen in this thesis are presented. At last, some general information about ensemble methods is given.

3.1 General Game Playing

3.1.1 Overview

In general game playing, a computer player (also called agent) is expected to be able to play a game based on a formal description of that game. This is especially interesting as these agents cannot rely on domain-specific heuristics and evaluation functions. Therefore the strength of an agent depends solely on its ability to learn during play [2].
To allow for a better development and comparison of agents, the "Game Definition Language" (GDL) has been invented by Genesereth et al. [2] and is now widely used. Being similar to prolog, it is used to describe a game using several key predicates. A specifc state in a game, e.g. a certain position all pieces have on the game board, is now determined by the set of predicates being true for that state. The same goes for moves. An overview over the gdl is provided in [2]. The base version, GDL-1, however, can only describe finite, deterministic games with each player having perfect information. The newer version, GDL-2, introduces the ability to model imperfect information and an element of chance [6].
The way a game works is also widely stadardised. It is splitted into two parts, the starting phase and the playing phase. During the starting phase, an agent can do some preliminary calculations which will help him play the game later on. The length of the starting phase is given by the start clock. Next comes the playing phase. Here the players take simultaneous moves playing the game until a terminal condition is met. Then the game is over and each player is rewarded with a score ranging from 0 to 100. The play phase is limited by the play clock which indicates how much thinking time each player has for one turn [2].
Furthermore, games with alternating moves can easily be modeled. The player who is not currently making a move is given only "noop" as valid possible move. This stands for "no operation" and does not affect the board. In the same way can traditional zero-sum games be modeled by limiting the score to 0 for a loss, 50 for a draw and 100 for a win [2].

3.1.2 GGP-agents

The structures of ggp-agents generally follow a similar principle, consisting of two layers: The basis of each player forms the reasoner or statemachine. It is responsible for processing the rules of the game. It creates an easy representation of the formal ruleset, so the next layer can directly address individual moves and states. On top of that is built the actual strategy, for instance MCTS, which is responsible for picking a move each turn [7]. This is shown in figure 1.

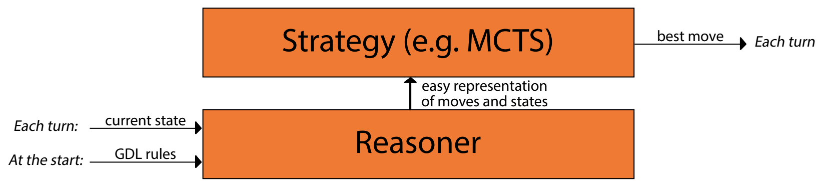

Figure 1: Functionality and structure of a GGP agent

3.2 Monte Carlo Tree Search

3.2.1 Game Trees

A game is often represented as a tree. In this model, the nodes correspond to states, the branches represent the moves (also called actions) which can be taken from the current state.
Monte Carlo Tree Search (MCTS) now iteratively builds the game tree. The aim is to choose the best action possible leading from the current position, taking various future scenarios into account. Therefore the current position corresponds to the root node of the game tree built by MCTS.

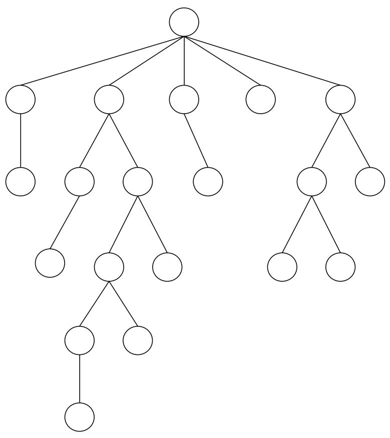

Figure 2: A sample game tree build through MCTS

Although certain game states could be reached via different move combinations in some games, they are kept as different nodes, since this would impose problems onto parts of the MCTS-algorithm [3]. How MCTS can be applied to games represented as directed graphs instead of trees is discussed in [8].

figure 2 shows a small tree built by MCTS. Since it is recursively built by the algorithm it does not represent the whole game, but the states and moves that have been searched. The tree is unbalanced, which is an inherent property of MCTS [3].

3.2.2 The Algorithm

For an algorithm to choose a good move it needs a way to evaluate the states of a game. As seen in section 3.1, some states, in the context of GGP only the terminal states, have values assigned by the game rules. The evaluation is easy in those cases. However, this does not hold true for most inner nodes of the game tree which correspond to non-terminal states [3]. This is problematic for some traditional search methods, often fixed with the use of domain specific evaluation functions. However, such an evaluation function is not always available, foremost not in the context of GGP. Another approach are therefore Monte Carlo simulations. This means that moves are chosen uniformly at random for each player at each turn until a state with a terminal condition is reached. This value is then used as the value of the node being evaluated [3].
A simple Monte-Carlo Method for finding an optimal move is to do a certain number of Monte-Carlo simulations to evaluate the state gained by playing each of the available moves. Since this leaves out potential information from travesing the tree during the simulation, algorithms were proposed, that combine Monte-Carlo simulations with tree-search methods. Thus the algorithm is named Monte Carlo Tree Search [3]. The algorithm can generally be split into four distinct phases:

- Selection phase
- Expansion phase
- Simulation phase
- Backpropagation phase

The selection and the expansion phase are often referred to together as the tree policy [3]. The four phases are illustrated in figure 3 and will now be explained in detail:
During the selection phase the algorithm moves down the already built tree to find a node with branches that have not been taken yet. This node therefore has moves available, which have not been tried out yet during previous iterations. The traversed nodes are marked in bold in figure 3.

There are many options to traverse the search tree, a widely applied one is the UCT-algorithm [3] proposed by Kocsis and Szepesvári [9]. The traversing is done as follows: Starting from

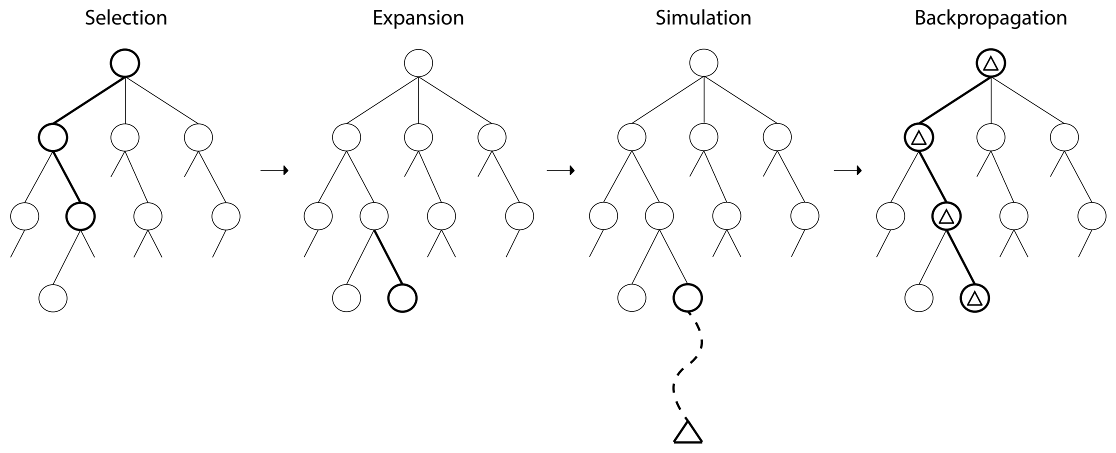

Figure 3: All four phases of Monte Carlo Tree Search

the root node, the algorithm then iteratively descends to the child j which maximises the UCT formula [9]:

$$UCT = \bar{X}_j + 2C_p\sqrt{\frac{2ln(n)}{n_j}}$$

$\bar{X}_j$ is the average score that previous iterations, where this child was involved, have returned. C_p is a constant. n denotes the number of times the parent has been visited while n_j does the same for the child node. As can be seen, the formula is formed as a sum of two parts:
The left term ($\bar{X}_j$) is also called the Exploitation Term. It gets bigger with a higher average score, hence a node is chosen if it has performed well in past iterations. The already good results are exploited as they are chosen more often to confirm whether the good results still hold true. The other term ($2C_p\sqrt{\frac{2ln(n)}{n_j}}$) is called the exploration term. The essential part is $\frac{2ln(n)}{n_j}$ which gets bigger the more often the parent has been visited and the less often the child has been visited. Therefore this term is big for children which have seldom been selected. The basic concept is that there may be good moves which by chance have been assigned a low average on past iterations. These moves are thus tried again and can redeem themselves [3].
This so called exploitation-exploration dilemma is very important for MCTS and its enhancements. On the one hand, the algorithm seeks to confirm its already good moves, hence it exploits the best results. On the other hand there is a chance that the true best move has been poorly evaluated on its first visits hence the algorithm wants to explore lesser visited nodes. The parameter C_p is there to balance the tradeoff between exploitation and exploration [3].
The next phase is the expansion phase. Once a node with unexplored actions has been found, one of them is selected uniformly at random. The tree is then expanded with the new node as shown in Fig. 3, the expanded node is marked again in bold.
From there the simulation phase begins. A Monte Carlo simulation is done to evaluate the current node [3]. The resulting game-specific value is represented as a triangle in figure 3.

The last phase is the backpropagation phase. Now that a game-specific value is available, all nodes that have been selected in the simulation phase and the node having been created in the expansion phase are updated with that value and the new averages are computed.
These four steps are repeated until some computational budget is reached. In the area of General game playing, this budget is usually determined by the play clock (see Sec. 3.1).
Upon having finished, the best move from the root node has to be selected. This can simply be done by selecting the move with the highest average score [3].

3.2.3 Problems and Limitations

MCTS is a very powerful algorithm having had lots of success in different domains. Still, there are some shortcomings worth noting [3].
When confronted with problems having a huge state space, MCTS (and most other traditional search algorithms) will likely fail. Then, domain-specific knowledge is needed to restrict the search to a reasonably sized subtree [3].
Another problem can arise from MCTS using a full simulation to evaluate a game state. If these simulations are particularly costly, MCTS will not be able to perform many simulations, thus its effectiveness is greatly diminished [3, 7].
The algorithm MCTS and several of its enhancements come with parameters which have to be tuned. This is a complex task, especially if one set of parameters has to work in different domains as it can be the case in GGP. As a result, parameters are often tuned by hand, respectively using trial and error testing [3, 10]. Still, some attemps to automatically tune the parameters to the domain at hand exist [3].
Next, the simulation phase uses random rollouts to estimate the value of a game state. Put into context of General Game Playing, this assumes both players taking random moves in any state following the expanded node. This is far from being the optimal strategy both players can use. Therefore chosing a best move based on the assumption that the opponent plays at random may lead to weak choices, as a stronger opponent will play better moves than has been accounted for. This is especially the case when only few simulation can be done and no reasonably sized game tree is built. In this scenario, which is often found in the context of GGP [7], the opponent model used during the tree policy, which is an opponent selecting his best move according to the UCT formula, has only a small effect. Thus improving the simulation policy has proven effective in many circumstances [3].
However, the enhancements bias the simulations which bears the danger of not always being appropriate, thus worsening the performance of the agent in some circumstances [10]. Especially with a high count of simulations, such enhancements generally perform inferior to random rollouts. The main reason being that random rollouts perform a more exhaustive search of the state space while the enhancements drive the rollouts in a certain direction. The latter is good when only few rollouts can be done but is weak if a more exhaustive search is possible [11].

3.3 Enhancements of the Simulation Phase

In this section four enhancements of the simulation phase are presented. They provide alternatives to the standard uniformly distributed rollouts as explained in section 3.2.2. They are all uninformed in that they need no previous knowledge about the game they are playing and they do not learn domain-specific knowledge neither. They only use game primitives such as moves or state predicates [3].

3.3.1 Move-Average Sampling Technique (MAST)

MAST is an enhancement proposed by Finnsson and Björnsson [4]. The basic idea is to manipulate the random rollouts based on the overall results the moves had in the past. It is thus related to the history heuristic [12] which is used in α-β-Search.
MAST works as follows: For each move m available in the game, a record $Q(m)$ is kept. This record saves the averaged score of the move m. If, during an iteration of MCTS, the move m has been chosen, the record $Q(m)$ is updated with the result of the iteration. It is important to say that this happens independently of the state the game was in when chosing move m [4].
During the simulation phase, the random rollouts are now replaced by rollouts following a Gibbs-Distribution. The probability of Move m being chosen in state s is now [4]:

$$P(m) = \frac{\mathrm{e}^{Q(m)/\tau MAST}}{\sum_{b=1}^{n} \mathrm{e}^{Q(b)/\tau MAST}}$$

n represents the number of legal actions in state s, $\tau MAST$ is a parameter greater than 0 to adjust the distribution. $\tau MAST$ approaching 0 stretches the distribution making it more pointed, higher values lengthen it. If a move has not been previously encountered, it receives the maximum score of 100 to enhance exploration [4].

3.3.2 Predicate-Average Sampling Technique (PAST)

PAST is an enhancement proposed by Finnsson and Björnsson [10]. The approach is very similar to MAST (see 3.3.1), but some state information is used here. In the GGP-environment (see 3.1), each state consists of several predicates which hold true in that state. Therefore for each move m played in state s, the average score $Q(m,p)$ is computed for all predicates p of state s. These records are kept throughout the game and continuously updated [10].
During the simulation phase, the random rollouts are now replaced by rollouts following a Gibbs-Distribution. The probability of Move m being chosen in state s is now [10]:

$$P(m) = \frac{max_p(\mathrm{e}^{Q(m,p)/\tau PAST})}{\sum_{b=1}^{n} max_p(\mathrm{e}^{Q(b,p)/\tau PAST})}$$

n represents the number of legal actions in state s, $\tau PAST$ is a parameter greater than 0 to adjust the distribution. $\tau PAST$ approaching 0 stretches the distribution making it more pointed, higher values lengthen it. Additionally, if the resulting estimate for a move is based on a low sample count, it is discarded and the game-average is used instead (which is the MAST-value) to avoid high variance. [10].
Finnsson and Björnsson [10] have also found that PAST has a substantial computational overhead, resulting in fewer overall simulations that can be done.

3.3.3 Contextual Monte Carlo Tree Search

Contextual Monte Carlo Tree Search is an enhancement introduced by Rimmel and Teytaud [13]. The idea is to group moves into coherent Tiles (The enhancement will therefore be called Tiles in short). Here, pairs of moves, which have ocurred during the same playout, are evaluated. More specifically, for each pair of moves (a, b) the average score $Q(a,b)$ is kept. This score is updated whenever these two moves are selected in any order during the same playout.
The random playouts are now replaced by the following procedure: Be P the set of every move already played in this rollout. Furthermore be M the set of all available moves in the current state. Now for all move-combinations in PxM, the Move c with the highest average score $Q_P(b,c)$ is chosen as a preliminary step. The score $Q_P(b,c)$ is the maximum score for the move c over all b $\in P$, while c is the best move in M using this metric. Now the score of c is checked against a lower threshold, which is incorporated by the parameter B. If the score is higher than B, c is kept, otherwise it is discarded. Now the move m for the rollout is chosen according to the following probability distribution [13]:

$$P(m) = \begin{cases} \epsilon + \frac{1-\epsilon}{|M|} & \text{if m equals c} \\ \frac{1-\epsilon}{|M|} & \text{if there is a c, but this is not m} \\ \frac{1}{|M|} & \text{if there is no c} \end{cases}$$

ϵ is a parameter ranging from 0 to 1 denoting the probability the best Tiling is chosen with. This is done in order to allow some exploration according to a ϵ-greedy strategy. If no best Tiling was found or the best Tiling found does not meet the lower bound B, the default policy is used, hence a uniformly random rollout (the third option in the formula). It is important to emphasize that P only refers to the moves played in the rollout, thus during the simulation phase. Moves selected during the selection or expansion phase are not used [13].

3.3.4 Last Good Reply

Last Good Reply (LGR) is an enhancement proposed by Drake [14]. A reply consists of playing a specific move as an answer to a specific move played by the opponent on the turn before. A last good reply is determined as follows: Be j a counter for the turns, if player A won a rollout, each move m_j he made on turn j is considered the last good reply to the move o_{j-1} made by the

opponent a turn before [14].
In the simulation phase, a new move is now chosen according to this policy:
If there is a last good reply m_j available to the move o_{j-1} made by the opponent one turn ago, choose m_j. Otherwise the default policy is used, hence a uniformly random distribution.
This can be written as the following probability distribution:

$$P(m) = \begin{cases} 1 & \text{if m is the last good reply} \\ 0 & \text{if there is a last good reply, but not m} \\ \frac{1}{|M|} & \text{if there is no last good reply} \end{cases}$$

M is the set of all legal moves in this state. If a reply m_j is chosen and the rollout is lost, it is discarded. Similarly if a reply m_j was theoretically available to the move o_{j-1} made by the opponent but m_j could not be chosen because it was not in M and a new reply n_j leads to a win, n_j is now considered the last good reply to o_{j-1} [14].

3.3.5 Related Work

Although the four enhancements presented in section 3.3 provide a good sample out of the realm of uninformed rollout heuristics, there are still other enhancements for the simulation phase (whether they are uninformed or not) worth mentioning.

3.3.5.1 Tree-only MAST

Tree-only MAST is a variation of MAST proposed by Finnsson and Björnsson [10]. It is a variant of MAST in that it only uses the moves played during the simulation and expansion phase to compute the game average. As these two phases operate inside the tree built by MCTS, this enhancement got the prefix Tree-only. It is stronger on Checkers, but otherwise very similar to MAST on the games testet by Finnsson and Björnsson [10].

3.3.5.2 Feature Average Sampling Technique (FAST)

FAST is a method proposed by Finnsson and Björnsson [10]. FAST tries to identify different features of the game, specially piece types and board cells. It then uses template matching to identify common board game features, hence the different piece types and board cells. This enhancement has provided good performance, among others, in the game skirmish, were MAST and PAST have proven weak [10].

3.3.5.3 PoolRave

PoolRave is a method proposed by Rimmel et al. [15]. It is based on the method Rave, which is often succesfully used to improve the selection phase. The method builds general estimates on how good a move is and uses these to influence the random rollouts [15]. It is thus similar to MAST (see section 3.3.1).
Given a state s, be M the set of all legal moves leading from s. For each move m $\in M$, a score $Q(m,s)$ is kept. This score is the average of all rollouts where m has been selected in any state succeeding s. The random rollouts are now replaced with the following policy [15]:
First, a Pool of k moves is build, k being a parameter. For that, the k moves with the highest score $Q(m,s)$ are chosen. Additionally, each move has to be based on a certain number of simulations to be added to the pool. The move being played is now chosen with probability ϵ from the pool, otherwise with a uniformly random rollout [15].

3.3.5.4 Other

Of course, there exist still other enhancements for the simulation phase. Mentionable are the use of patterns or opening books. Patterns describe short sequences of commonly played moves, while opening books provide databases of entire start sequences to the agent. This knowledge is often incorporated directly into the agent and therefore represents a form of prior domain specific knowledge [3].

3.4 Aggregating results

This section will present some common methods for aggregating results. First, two common voting mehtods will be described. Such methods have been used since the mid-50s to combine different outputs of computations. It is used nowadays, among others, to combine multiple statements into a more reliable one. [16].
The term ensemble method itself is often used in machine learning when results are aggregated using a more sophisticated procedure. Out of these one very common method, Bagging, will be presented. The success of such an ensemble method is usually high if the different sources the results come from are weakly correlated and if they tend to make their errors in different situations. Therefore individual errors are mitigated, while shared strengths are kept [17].

Voting schemes

When multiple votes are available, there exist several options (called voting schemes) to pick the resulting vote. The most common ones are [16]:

Unanimous Each voter has to get to the same result.

Majority The cumulated weight has to be greater than half the possible weight.

Weighted voting

Weighted voting is a very general voting technique assigning each vote a weight. Therefore varying importances of the individual votes can be represented. The result is then usually picked as a majority vote [16].

Simple voting

Simple voting is a special case of Weighted voting since all weights are set to one (Or no weights are present, depending on the viewpoint). It is often used when there is only few information available to gather the weights from or when there is some other reason making the computation of weights difficult [16].

Bagging

Bagging is often used in machine learning with great success. To explain the behaviour be E a set of entities that create a result and D a set of data, which all entities use to create their result. Bagging then constructs some subsets of D and gives each entity in E a different subset to create the result from. Then the unweighted average of all results is taken. This is especially useful if the set E consists of a single entity. Through this procedure the results from that entity can substantially be improved [18].

4 Method

This chapter provides details about the implementation and outlines why the four enhancements presented in section 3.3 were chosen. This is done in section 4.1. Thereafter, in section 4.2, the ensemble itself is explained and how the enhancements have been combined.

4.1 Enhancements

As stated in chapter 2 the goal of this paper is to find out whether different heuristics affecting the simulation phase can effectively be combined using an ensemble method. Additionally, the family of enhancements has been reduced to uninformed enhancements, meaning that no domain-specific knowledge, be it injected in advance or learned during play, is used. This is done to obtain a set of comparable enhancements whose interactions can neatly be accessed. Also a general applicability is retained this way.

The basis for a useful combination is that the enhancements are different and have strengths in different situations (see section 3.4). In the best case, the ensemble mitigates downsides of individual enhancements in special cases while preserving the overall playing strength in general. Therefore the enhancements have been chosen to have different underlying assumptions which lead to different behaviour depending on whether or not the assumptions hold true.

MAST assumes that if a move is good in one state, it should be generally good in all other states, too. This is a very general assumption and not necessarily true for all games. PAST thereof chooses the moves depending on the states, more precisely the GDL-predicates forming the states in which they were played. This is more specific than MAST, however, it comes with a bigger computational cost as told in section 3.3.2.

Tiles bases the decision on moves previously played by the same player during the rollout preferring moves which are good in combination. LGR chooses its move based on the last opposing move, which is an entirely new idea.

There are still other enhancements out there, the most important of them have been presented in section 3.3.5. Tree-only MAST and PoolRave are good enhancements but they are conceptually very similar to plain MAST. Since there is not enough theoretical difference, they have been left out of the ensemble. FAST is a very interesting enhancement on its own but it learns domain-specific knowledge during play and is therefore not uninformed. Other enhancements generally try to inject previously gathered domain specific knowledge into the rollouts, prominent examples are the use of patterns or opening books. This is not wanted in this thesis neither is it generally possible in the context of General Game Playing.

4.2 Ensemble

As ensemble method, simple voting is used. While weighted voting is very general and adaptive (see section 3.4), it comes with additional parameters. As already mentioned in section 3.2.3, tuning the parameters is generally a difficult task in MCTS. Additionally, the four enhancements (except for LGR) already have parameters, which increases the overall parameter dimension. To keep the complexity reduced, and to allow for a more accurate optimisation of the parameters already at hand, simple voting is chosen instead. This allows to more precisely access the strengths of the individual enhancements as the probability of refuting a good combination because of bad parameters will be diminished.
More elaborate ensembles, namely Bagging in this case, are not feasible to use under these circumstances, neither. Since the ensemble will be evaluated on a per game basis, there are no subsets to create. Therefore Bagging defaults to simple voting here.
Still, several ways to implement the simple voting method remain. For this thesis, a probabilistic approach has been chosen. The advantages will be discussed towards the end of this section.
All the enhancements are probability-distributions by nature. MAST and PAST use Gibbs-distributions, while Tiles follows an ϵ-greedy distribution. LGR has a deterministic answer if a best reply is present, else it follows the standard uniform distribution. The deterministic reply can be seen as a 0-1 distribution, however.
The distributions of all enhancements are now simply added and the result is divided by 4. This results in a valid probability distribution. Since all distributions have the same weight in the sum, this constitutes a simple voting method. To remind how the enhancements calculate the probability of a move, table 1 provides an overview.

Table 1: Move-Probability calculated by the four enhancements

Enhancement	Probability of chosing move m
MAST	$P(m) = \frac{e^{Q(m)/\tau MAST}}{\sum_{b=1}^{n} e^{Q(b)/\tau MAST}}$
PAST	$P(m) = \frac{max_p(e^{Q(m,p)/\tau PAST})}{\sum_{b=1}^{n} max_p(e^{Q(b,p)/\tau PAST})}$
Tiles	$P(m) = \begin{cases} \epsilon + \frac{1-\epsilon}{\lvert M\rvert} & \text{if m is the best tiling} \\ \frac{1-\epsilon}{\lvert M\rvert} & \text{if there is a best tiling, but not m} \\ \frac{1}{\lvert M\rvert} & \text{if there is no best tiling} \end{cases}$
LGR	$P(m) = \begin{cases} 1 & \text{if m is the best response} \\ 0 & \text{if there is a best response, but not m} \\ \frac{1}{\lvert M\rvert} & \text{if there is no best response} \end{cases}$

For further information about the individual parameters, see section 3.3.

As already mentioned, these probabilities are then summed for each move and divided by 4. The result is the probability of each move being chosen when using the ensemble method.

However, there has been an modification done to PAST at this point: The original method uses a threshold of sample size. If a PAST score is not based on enough samples, the MAST value is chosen instead (as explained in section 3.3.2). This, however, leads to serious problems in the evaluation of the ensemble method. It is no longer possible to separate the two methods MAST and PAST making it very difficult to interpret their contribution to the success of the ensemble later on. Therefore this threshold has been removed. Now PAST only contributes to the ensemble based on the underlying assumption on good state-specific moves. This can now be separated from MAST. That this might worsen the performance of PAST is carefully considered when discussing the results.
There are now several advantages using this probabilistic simple voting method:

- The method does not change the original behaviour of the enhancements with the exception of PAST.
- The method does not introduce any additional parameters. This is very important performance-wise
- In being a probability-distribution, it still allows for exploration. This is the main reason why MAST, PAST and Tiles are probability-distributions instead of simply returning the best value [10, 13]. The method thus preserves this property.

There are still a few things to consider. As all four enhancements can be seen as probability distributions, they assign a high probability to (in their view) good moves and a low one to bad moves.
However, the span between good and bad moves can vary. A potential problem lies with LGR since it does or does not suggest a move following a 0-1 distribution. Therefore the moves suggested by LGR might have a potentially bigger impact than the others using this ensemble method. Likewise, the probability of the best move in Tiles is determined by the parameter ϵ. This parameter is optimised, which will then impact the importance of Tiles in the ensemble. Similar thoughts can be expressed concerning the τ-parameters. Therefore, the optimised parameters will be carefully analysed.
As a short conclusion, the advantages of using a simple probabilistic ensemble method outweigh the disadvantages, so this method has been chosen. Still, several points have been mentioned which deserve careful consideration when the experiments are being discussed.

5 Experimental Setting

In this chapter the experimental setting will be explained in detail. This entails the concrete experimental setup (section 5.1), the parameter optimisation process (section 5.2) and lastly the relevant information about the games used in the experiments (section 5.3).

5.1 Setup

The experimental setup is as follows: Eight games have been chosen to evaluate the ensemble method. All of these will be two-player games, therefore the strength of the ensemble method is measured as the winrate against a standard MCTS Player. This player uses the base MCTS version described in section 3.2.2. The winrate will be determined on a basis of 500 matches. To decouple the winrate of the side an agent played on, each agent will play on each side 250 times.
To fully evaluate the importance of each of the four enhancements to the overall performance, they will by combined into 5 different ensembles. The first one is the base version using all four enhancements. Each of the other four is gained by leaving one of the enhancements out and using just the other three. This is summarized in Table 2. This setting will help to determine how the individual enhancements contribute to the ensemble.

Table 2: All Matches listed

Ensemble	Opponent
All four enhancements	Standard MCTS
PAST, Tiles, LGR (Without MAST)	Standard MCTS
MAST, Tiles, LGR (Without PAST)	Standard MCTS
MAST, PAST, LGR (Without Tiles)	Standard MCTS
MAST, PAST, Tiles (Without LGR)	Standard MCTS

There are some more settings worth mentioning: From the context of General Game Playing (see section 3.1) there are two important parameters, namely start clock and play clock. These are set as follows: The playclock has been set to 5 seconds. Since the agent does not do any calculations during the startclock, this parameter is not relevant for the experiments and can be set to any value.
The optimization and the experiments were run on the FUCHS-Cluster of the Goethe-University in Frankfurt using only nodes of type Istanbul. These nodes have 2 sockets with 6 cores each using a 2.2 GHz Six-Core AMD Opteron(tm) Processor 2427. Each game used only a single core with a memory limit of 2 GB per game. The memory limit was, however, not exhausted, the computations were foremost cpu-bound. The complete experimental setup is summarized in figure 4.

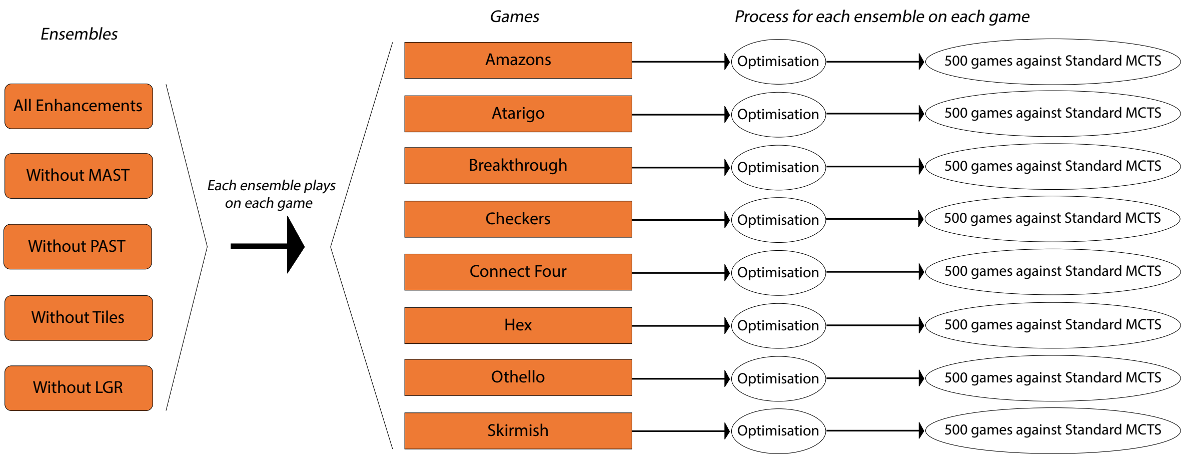

Figure 4: Experimental Setup

5.2 Optimization

Three out of the four enhancements presented in section 3.3 and the MCTS algorithm use parameters for tweaking their performance. This bears the risk of unfitting parameters being chosen and thus influencing the results. Hence the parameters have to be optimised.
For this thesis, the parameters are optimised seperately for each game (as has been shown in figure 4). This has several advantages and downsides: First of all, following a GGP-scheme as explained in [2], a GGP-agent does not know to which game it is going to be applied and therefore cannot use specific parameters for that game unless it stores a set for every game possible. This is hardly achievable and it does not lead to the creation of a general and game-independent agent.
There are several advantages, however. Using game-specific parameters is very good in detecting the principal usefulness of the ensemble method as each game is played with the best possible set of parameters. Since evaluating the general usefulness is the main focus of this thesis, game-specific parameters are chosen for each game. And lastly, it is important to say that if this ensemble method does not prove beneficial using game-specific parameters, it will not do so with general parameters either.
All available parameters are shown in Table 3. It is reminded here that PAST has only one parameter due to the modification explained in section 4.1:

Table 3: Parameters

Parameter	Appearance	Significance
C_p	Base Algorithm	Weighting of exploration vs exploitation
$\tau MAST$	MAST	Streching of the Gibbs distribution
$\tau PAST$	PAST	Streching of the Gibbs distribution
ϵ	Tiles	Greedines of the enhancement
B	Tiles	Threshold for the goodness of a Tiling

As can be seen, Last Good Reply has no parameter.
The parameters have been optimised using random search. This method provides particularly good results while being embarassingly parallel [19], which is a huge advantage in this case. The optimisation function is the winrate against a standard MCTS Player, which is based on 108 matches. This number ensures that the deviation based on a 95% confidence intervall of the true winrate is below 10%. From now on, "one run" means running 108 matches with the same parameter configuration.
The parameters listed in Table 3 can be split into two groups: The first entry, the C_p-paramter belongs to the base MCTS algorithm. It is thus shared with the standard MCTS player which the winrate is tested against. Therefore this parameter is optimised seperately.
The other paramters belong to the enhancements and are only used by the ensemble methods. These parameters are then optimised together. The optimisation process is summarized in figure 5.

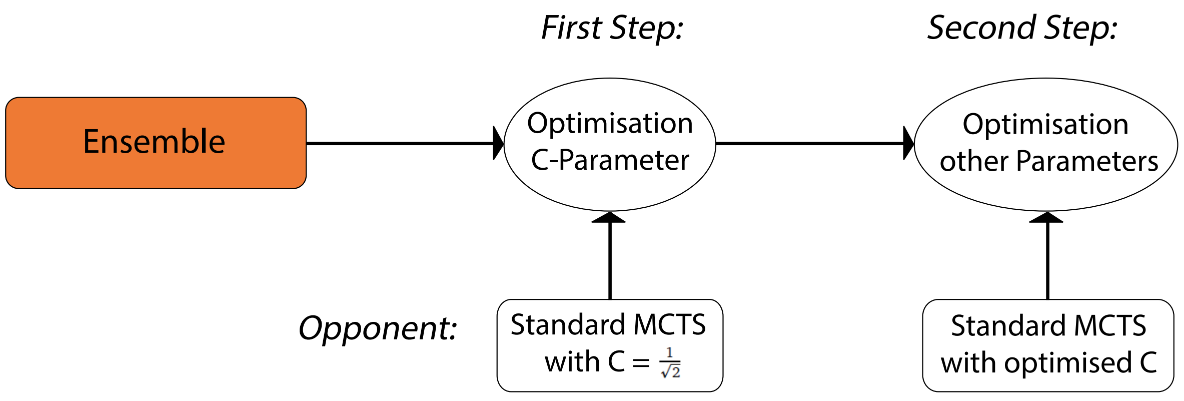
First Step:
Second Step:
Ensemble
Optimisation C-Parameter
Optimisation other Parameters
Opponent:
Standard MCTS with C = $\frac{1}{\sqrt{2}}$
Standard MCTS with optimised C

Figure 5: The optimisation process

The C_p-parameter is optimised based on 90 runs. However, the opposing MCTS player has to use its own C_p-paramter, too, during the matches. For this setting $\frac{1}{\sqrt{2}}$ has been used as shown in figure 5. This value is a theroretical default value since it is proven to make MCTS converge to the optimal policy with infinite iterations [20]. The C_p-paramter is then chosen uniformly random from a 0-1 interval. In theory, the C_p is originally bound to be greater or equal to zero but has no upper bound. The upper bound of 1 has therefore been chosen arbitrarily around the theoretically best value of $\frac{1}{\sqrt{2}}$. Since all optimal C_p parameters turned out to be fairly low, this seems acceptable. Quickly to mention, the scores in GGP are in a range from 0 to 100 as explained in 3.1. However. Kocsis et al. [20] assumed the result to be 0-1 bounded for their convergence proof. Hence, the UCT formula has been modified as follows to satisfy this condition:

$$UCT = \frac{\bar{X}_j}{100} + 2C_p\sqrt{\frac{2ln(n)}{n_j}}$$

The other parameters, $\tau MAST$, $\tau PAST$, ϵ and B are all optimised simultaneously using random search. Table 4 shows the number of runs that have been used for a specific number of parameters:

Table 4: Number of runs for different parameters

Number of Parameters	Number of Runs	Ensembles
2	140	(PAST, MAST, LGR)
3	170	(PAST, Tiles, LGR) and (MAST, Tiles, LGR)
4	200	(All 4) and (PAST, MAST, Tiles)

These four parameters are conceptually different and therefore are sampled differently: ϵ is a probability and is therefore taken uniformly at random from a [0,1]-interval. B is bound by the values a score can take. These range from 0 to 100 (see section 3.1), therefore B is taken uniformly at random from that range.
Both τ-parameters are parameters of a Gibbs-distribution that are greater than zero but unbound in the other direction. Therefore values close to zero convert this into a 0-1 distribution while infinitely big values convert it into a uniform random distribution. The range has therefore be selected to use 0 as lower and 150 as upper bound. The argumentation is similar as with the C_p-parameter: The upper bound has been chosen arbitrarily after having observed that low values are used in practice [10]. Again, the resulting parameters turned out ot be low, so this interval seems acceptable. Additionally, a small change in value has a greater impact when the parameter is close to zero. To account for this effect, the parameter has been chosen according to an exponential distribution. All parameter ranges have been summarized in Table 5.

Table 5: Parameters and their bounds

Parameter	Range	Distribution
$\tau MAST$	(0,150]	exponential
$\tau PAST$	(0,150]	exponential
ϵ	[0,1]	uniform
B	[0,100]	uniform

Generally, there has been put some emphsis on robust parameter optimisation. The goal is not to find the best parameter, but to find a parameter that is fairly reasonable with high probability. So the following approach to choose the resulting parameter has been used:
Using the results, the parameter space of each individual parameter has been divided on the parameter-axis in $\sqrt{n}$ classes with equal density, with n standing for the number of runs, as it is common in descriptive statistics [21]. From each class, the average score has been computed. Then the class with the highest average has been taken and the average over all parameters in this class has been used to select the resulting parameter.
For the parameters of the ensembles this leaves out possible positive interaction effects, but

it guarantees that values are picked which work satisfyingly well with all values of all other parameters.

5.3 Games

As already mentionend in section 5.1, eight games have been chosen to test the ensemble method. First, an overview over all eight games is presented in section 5.3.1, then the game properties will be elaborated in section 5.3.2.
All games are 2-player games. These form a huge family of games, featuring many old board games which are often used for research in the field of artificial intelligence [3]. Therefore, the number of players has been held stable throughout the experiments to allow more finesse and accuracy in the interpretation of the other game properties. Additionally, all games have been chosen to have altenating moves and the terminal scores are fixed as being either 0, 50 or 100, with 0 representing a loss, 50 a draw and 100 a win.

5.3.1 Overview

Amazons

Amazons is a game played on a 10x10 Chess board. Each player controls four queens. Each turn, a player first moves one of his queens and then shoots a diagonal arrow on an empty space. The arrow will block that space for the rest of the game. The first player having no more legal move loses the game.
Amazons has already been used to study Monte Carlo Tree Search, for instance by Kloetzer et al. [22] and Kocsis et al. [20]. The challenge of this game lies in its huge branching factor, yielding often more than 1000 moves to choose from [3].

Atarigo

Go is *the* success-story of Monte Carlo Tree Search [3, 23]. Atarigo is a simplistic variant as the game ends with the first capture and the player having captured the stone wins. This transforms the highly strategic game Go into a tactical one. The board size of this variant is 8x8.

Breakthrough

Breakthrough is a game played on a 8x8 chess board, starting with two lines full of pawns.
Moving one pawn per turn, the first player reaching his opponent's home row (the row farthest from him) wins the game.
Breakthrough has already seen some application in the field of General Game Playing, being used by the developpers of MAST and PAST [4, 10].

Checkers

Checkers, also known as Droughts, is a widely known game played on a standard 8x8 chess board. Players try to capture all opposing pieces while being able to upgrade their own pieces if they reach the opponents home row. However, if no player has won after 102 moves, the game ends in a draw.
Checkers has been used, like Breakthrough, by the developpers of MAST and PAST to evaluate their agents [4, 10].

Connect Four

Connect Four is a widely known game played on a grid, in this case of the dimension 6x8. Each turn a player puts a piece of his color into the lowest line on the grid trying to line four of his pieces in a row, either orthogonally or diagonally.

Hex

Hex is a board game played on an 11x11 grid with hexagonal tiles. The players place one of their pieces each turn aiming at connecting two opposing sides with a line. It has been shown that there exists no draw in hex, meaning one of the players will result to be the winner [3]. Hex is a purely strategic game featuring no short-term option secure a win [11]

Othello

Othello, also known as Reversi, is played on a 8x8 board. Each turn, a player drops a piece on the board. This will flip any opposing pieces between this piece and any other of this players other pieces to his colour. Once the board is full, the player having more pieces of his color on the board wins.
Othello has also been used to by the developpers of MAST and PAST [4, 10]. This game is characteristic for having big swings in the players' score from turn to turn. This poses severe challenge to predict a winner even closely before the end [3].

Skirmish

Skirmish is a game derived from Chess. The version used in this thesis is played on a standard Chess board using the standard Chess setting. The goal of the game is to capture all enemy pieces. However, the game is terminated after 100 moves with the player having captured the most enemy pieces winning the game.
Skirmish has also been used to test MAST and PAST [10].

5.3.2 Game properties

In this section, important properties of the games will be described. This entails:

Depth The number of moves per player that are played until the game is over.

Width The number of different moves a player can chose from during his turn

Standard deviations The standard deviation for the game width and the game depth

Variability of the game depth This is simply the standard deviation divided by the game depth. Since the games differ in game depth, this measurement shows a normalized standard deviation of the game depth.

Tactical or Strategic If the games focuses stronger on short term tactics or long term strategic goals as it is described in [11].

These properties have been averaged over all matches played. An overview is now provided in Table 6:

Table 6: Game properties

Game	Width	Sd Width	Depth	Sd Depth	Var Depth	Tactical/Strategic
Amazons	423.01	530.07	39.22	1.94	0.04	Strategic
Atarigo	56.55	6.34	5.93	2.97	0.5	Tactic
Breakthrough	25.38	2.6	22.54	6.65	0.3	Strategic
Checkers	7.47	2.29	48.59	5.7	0.12	Both
Connect Four	7.13	1.44	16.85	4.82	0.29	Tactic
Hex	48.86	19.46	31.24	5.03	0.16	Strategic
Othello	7.46	2.95	29.02	1.33	0.05	Strategic
Skirmish	27.21	8.21	49.78	0.98	0.02	Both

Sd = Standard deviation
Var = Variability

As can be seen, a wide variety in terms of width and depth has been achieved. Amazons stands out with its huge width of 423.01, this has already been mentioned and is on purpose. With that many moves to chose from, Amazons is at the verge of playability for an MCTS agent. It is thus interesting to see whether the ensemble can prove beneficial under such circumstances.
On the other hand, Connect Four has the smallest width (7.47 and a relatively shallow depth with 16.85). The game tree complexity is thus relatively low for this game, hence it should be easy to play for an MCTS-agent. The other games range from being very short (atarigo with only 5.93 moves on average) to fairly long: Ceckers with 48.59 moves and Skirmish having the longest width with 49.78 moves on average.
Also an interesting property is the variability of the game depth. Atarigo, Breakthrough and Connect Four allow for most flexible game endings, with Hex and Checkers being also somewhat variable. On the other hand, games like Skirmish, Amazons and Othello are relatively inflexible

and remain mostly and at a fixed game depth.
As mentioned in the individual game descriptions, Atarigo is the most tactical game, focussing on very short-term goals while on the other hand Hex is the most strategic one with only long-term goals to consider. Since being tactic or strategic is a simple cassification, the games surely differ on how much these traits apply to them. This will be mentioned in detail whenever necessary during the analysis of the experiments.

6 Results and Discussion

In this chapter the results of the experiments will be presented and discussed. Section 6.1 presents all results and remarks special points of interest. These are then revisited in section 6.2 for discussion and further analysis. Based on these findings a new method is proposed and analysed in section 6.2.8.

6.1 Results

6.1.1 Advanced Game properties

This section provides an overview about some algorithm-specific game properties, namely state advances (table 7), the simulation count (table 8) and the simulation length (table 9). A state advance is a procedure done by the reasoner (see section 3.1.2), moving from one state to the next. The number of advances is a good indication in how fast the reasoner works. However, since the resoner used stays the same for all ensembles, the number of state advances is determined by the computational overhead induced by the enhancements.
The simulation count is the number of simulations (see section 3.2.2) the agent does per 5 second interval. It also shows how many states the agent evaluates per turn. The results have been measured in each game and are averaged over all turns. The data concerning the standard MCTS player has been gathered in a match against another standard MCTS player.

Table 7: State Advances

Game	Standard MCTS	All	No MAST	No PAST	No Tiles	No LGR
Amazons	733.23	554.72	564.46	652.65	558.17	559.39
Atarigo	2446.77	1739.61	1735.76	2223.34	1756.05	1738.97
Breakthrough	3962.1	2578.71	2731	3131.78	2672.62	2616.09
Checkers	2121.66	1795.24	1816.21	1992.29	1845.21	1801.55
Connect Four	4843.2	4129.1	4185.47	4673.12	4352.89	4279.97
Hex	699.99	457.03	466.78	683.38	467.22	463.83
Othello	76.79	75.46	75.61	76.36	76.36	75.35
Skirmish	3007.77	1749.38	1768.17	2441.1	1941.54	1752.53

The state advances follow a clear trend in all games: The standard MCTS player achieves more advances than any ensemble. Leaving MAST or LGR out usually does not change the number of advances substantially.
Leaving Tiles out allows for substantially more advances on some games, for instances in the games Checkers, Connect Four, Skirmish and Hex.

The biggest impact, however, has PAST. Leaving it out allows always (with the exception of Othello) for a big increase in state advances. In some games, like Atarigo, the number of state advances after leaving PAST out is closer to the standard MCTS player than to the ensemble with all enhancements.
It is remarkable how low the number of state advances of Othello are (around 76 on all games). This is especially interesting since all other games have over 400 state advances on all agents.

Table 8: Simulation count

Game	**Standard MCTS**	All	No MAST	**No PAST**	No Tiles	No LGR
Amazons	34.24	25.12	24.49	29.66	23.94	24.51
Atarigo	**84.82**	120.77	77.05	**179.74**	114.90	114.96
Breakthrough	**96.17**	212.29	120.32	**299.32**	160.16	175.18
Checkers	61.7	55.05	53.32	59.59	55.41	53.58
Connect Four	**409.55**	443.41	369	**486.55**	461.75	420.11
Hex	**18.82**	16.12	13.6	**22.91**	16.24	15.15
Othello	2.31	2.23	2.29	2.25	2.3	2.22
Skirmish	104.14	60.09	58.09	84.06	64.15	58.79

Simulation count of the standard MCTS player compared to "No Past":
bold - The simulation count of "No Past" is significantly higher than the one of the standard MCTS player (5% significance level)

Table 9: Simulation length

Game	**Standard MCTS**	All	No MAST	**No PAST**	No Tiles	No LGR
Amazons	21.41	22.08	23.05	22	23.32	22.82
Atarigo	**28.85**	14.40	22.53	**12.37**	14.66	15.13
Breakthrough	**41.2**	12.15	22.69	**10.46**	16.69	14.93
Checkers	32.39	32.61	34.06	33.43	33.30	33.62
Connect Four	**11.83**	9.31	11.34	**9.6**	9.43	10.19
Hex	**37.19**	28.35	34.32	**29.83**	28.77	30.62
Othello	33.24	33.83	33.02	33.94	33.2	33.94
Skirmish	28.88	29.11	30.44	29.04	30.27	29.81

Simulation length of the standard MCTS player compared to "No Past":
bold - The simulation length of "No Past" is significantly lower than the one of the standard MCTS player (5% significance level)

The simulation count (in table 8), shows some interesting properties: First of all, othello has a very low simulation count (Only about 2.3 on average), thus following the trend it set with the state advances. Aside from that, two phenomena can be observed: There are games, especially Atarigo and Breakthrough, where the basic ensemble method achieves more simulations than the standard MCTS player. This trend is spiked for the version without PAST, on Atarigo, Breakthrough, Connect Four and Checkers it achieves significantly more simulations that the standard MCTS player. On this games the ensembles make shorter simulations than the standard MCTS player (this is confiremd in table 9). On the other hand, in the games Amazons, Checkers and

Skirmish the simulation count is lower. In othello, it remains basically unchangend for all agents. In both tables concerning the simulations, the colums of the standard MCTS player and the agent without PAST are marked specially for better readability.

6.1.2 Experiments

Table 10 presents the performance of the different ensembles tested. The values correspond to the winrate against the standard MCTS agent in %. The asterix denotes a significant deviation from the version using all enhancements (significance level: 5%). The version using all enhancements will therefore be now referred to as the base version.

Table 10: Winrates of the different ensembles

Game	All	Without MAST	Without PAST	Without Tiles	Without LGR
Amazons	44.25	40.67	53.37*	46.23	46.83
Atarigo	67.86	52.38*	77.78*	68.45	68.45
Breakthrough	88.89	62.3*	93.45*	90.48	89.29
Checkers	47.22	48.91	55.56*	50.4	49.8
Connect Four	53.57	43.95*	54.37	55.36	55.75
Hex	57.74	48.21*	63.89*	64.48*	59.92
Othello	54.56	50.2	52.08	55.06	50.3
Skirmish	35.12	36.11	43.85*	37.7	34.33

Four things can be noticed immediately:

- The method without Mast is strictly worse than the base version being significantly worse on half of the games (Atarigo, Breakthrough, Connect Four and Hex) while being significantly better in none
- The method without PAST is strictly better than the base version being significantly better in most games and being significantly worse in none (Connect Four and Othello being the only games without a significant difference in performance)
- the method without Tiles is relatively equal to the base version being only significantly better in hex
- the method without LGR is equal to the base version, showing no significant deviation
- the variation of the simulation length (see section 6.1.1 seems to correlate with the winrate. Exemplary the games atarigo and breakthrough have a shorter simulation length for ensembles with a particularly high winrate (e.g. without PAST). For hex, this effect ist less pronounced.

As MAST has been the only enhancement with a negative impact when being left out, a version only using MAST has been run. It has been optimised using 90 runs. Table 11 shows its performance compared to the ensemble without PAST, which has formed the strongest agent until now. The asterix indicates a significant difference to the agent withouth PAST at a 5% significance level.

Table 11: Winrates of the Mast agent

Game	Without PAST	MAST
Amazons	53.37	54.17
Atarigo	77.78	64.29*
Breakthrough	93.45	92.66
Checkers	55.56	64.19*
Connect Four	54.37	56.25
Hex	63.89	59.13
Othello	52.08	52.88
Skirmish	43.85	45.83

As can be seen from the results, the agent just using MAST only differs significantly on the games Atarigo and Checkers. However, while that agent performs significantly better on Checkers, it is worse on Atarigo.

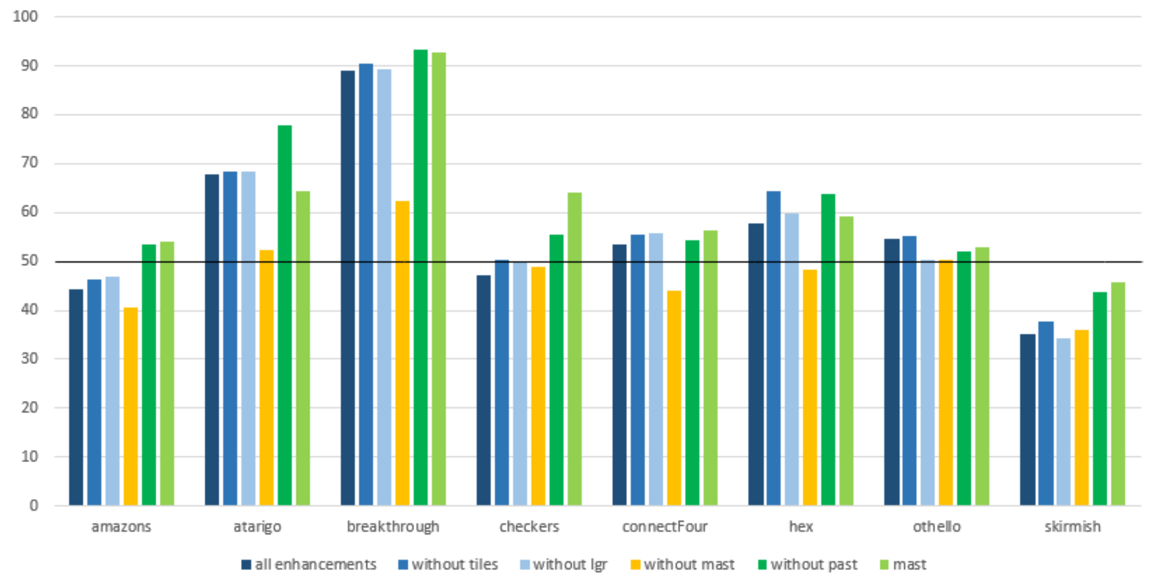

Figure 6: Winrates of all agents

All winrates are summarized in Fig. 6. The first three bars correspont to the base verion, without Tiles and without LGR. As already mentioned they are usually equally high (in Hex, without Tiles stands out). Next comes without MAST, which is generally lower than all other bars. Lastly there are without PAST and MAST, which are generally of equal height, too, with the exceptions of Atarigo and Checkers. For additional distinction, these groups are coloured. The threshold of a 50% winrate is specially marked to allow for a quick overview whether an improvement compared to the standard MCTS player has been achieved.

6.1.3 Optimized Parameters

This section presents the optimised parameters starting with the C_p-parameter in section 6.1.3.1. Next the the parameters specific to the ensemble methods are shown in section 6.1.3.2.

6.1.3.1 C-Parameter

The parameters have been optimized using a robust parameter optimisation as described in section 5.2. Table 12 shows the values of the optimised C_p-parameters.

Table 12: Values of the C-Parameter

Game	Parameter
Amazons	0.143
Atarigo	0.076
Breakthrough	0.078
Checkers	0.154
Connect Four	0.167
Hex	0.107
Othello	0.327
Skirmish	0.062

As can bee seen the parameter has fairly low values compared with the default value of $\frac{1}{\sqrt{2}}$ [20]. The exception is Othello, which has a comparable high parameter value (0.327). This is almost twice as high as the second highest parameter (Connect Four with 0.167).

6.1.3.2 Other Parameters

All tables containing the optimised parameters for each ensemble can be found at the end of this section. They will now be analysed.
First of all, the $\tau MAST$-values have an interesting aspect worth pointing out: For the games Atarigo, Breakthrough, Checkers, Connect Four and Hex, they have relatively low values, especially for the ensemble without PAST. In fact, they are never bigger than 6. When Breakthrough is taken out, the parameter does not exceed the value 3 on these games, while often being below 1. Low values spike the distribution (see section 3.3.1), thus giving moves with a high average an especially high probability.
For the games Amazons, Othello and Skirmish, however, the $\tau MAST$-parameters also take high values in some ensembles. Especially Skirmish stands out, which has an average $\tau MAST$-value of 37 over all ensembles.
For the $\tau PAST$-values, however, no similar effects can be observed. They differ very much between the games and the ensembles.

Comparing the $\tau MAST$-values for the agent just using MAST (table 17) to the one without PAST (table 15) there are some clear differences: The $\tau MAST$-values of the agent just using MAST are substantially higher on all games. Checkers is a notable exception with the $\tau MAST$-value staying very low at 0.474.

Table 13: Parameter values for the basic ensemble

Game	$\tau MAST$	$\tau PAST$	B	ϵ
Amazons	52.806	7.052	34	0.97
Atarigo	0.515	10.427	44.5	0.668
Breakthrough	3.361	55.413	61.7	0.93
Checkers	1.018	10.551	38.3	0.479
Connect Four	1.28	27.33	96.1	0.333
Hex	0.725	22.34	87.2	0.765
Othello	62.097	9.027	18.4	0.082
Skirmish	97.785	61.058	83.8	0.209

Table 14: Parameter values for the ensemble without Tiles

Game	$\tau MAST$	$\tau PAST$
Amazons	13.645	2.212
Atarigo	1.745	72.351
Breakthrough	5.016	0.441
Checkers	0.48	58.379
Connect Four	1.718	44.528
Hex	1.506	2.512
Othello	3.162	8.794
Skirmish	5.972	56.328

Table 15: Parameter values for the ensemble without PAST

Game	$\tau MAST$	B	ϵ
Amazons	0.287	94.071	0.937
Atarigo	0.85	17.4	0.594
Breakthrough	0.955	66.4	0.588
Checkers	0.374	47.9	0.848
Connect Four	1.856	86.5	0.162
Hex	0.647	81.5	0.743
Othello	0.785	44	0.404
Skirmish	12.891	11.7	0.395

Table 16: Parameter values for the ensemble without MAST

Game	$\tau PAST$	B	ϵ
Amazons	0.238	91.4	0.185
Atarigo	11.112	30	0.711
Breakthrough	58.839	67.786	0.830
Checkers	80.573	40.071	0.938
Connect Four	20.445	81.643	0.45
Hex	6.138	89.214	0.422
Othello	7.384	79.929	0.482
Skirmish	12.446	24.6	0.968

Table 17: Parameter values for the Mast agent

Game	$\tau MAST$
Amazons	2.768
Atarigo	5.931
Breakthrough	14.794
Checkers	0.474
Connect Four	12.41
Hex	9.523
Othello	4.742
Skirmish	20.587

Table 18: Parameter values for the ensemble without LGR

Game	$\tau MAST$	$\tau PAST$	B	ϵ
Amazons	1.709	0.58	23.9	0.904
Atarigo	0.481	31.154	29.133	0.378
Breakthrough	5.77	0.261	84.4	0.043
Checkers	0.625	6.925	31.3	0.751
Connect Four	2.698	88.928	69.7	0.262
Hex	0.587	1.49	67	0.202
Othello	4.735	0.316	53.87	0.952
Skirmish	70.728	57.661	78.733	0.367

6.2 Discussion

In this section the results will be discussed. This starts with the varying simulation length in section 6.2.1. Next, general findings concerning individual games will be addressed in section 6.2.2. Thereafter follow detailed discussions about the individual enhancements. Lastly, a new method, which is based on these findings, is presented and analysed in section 6.2.8.

6.2.1 Efficient Simulations

In section 6.1.1 the effect was observed, that on several games, foremost atarigo and breakthrough, a substantial decrease in simulation length can be seen when an improved agent is compared to the standard MCTS player. There are several factors contributing to this behaviour: First of all, the game has to allow for some flexibility in game depth. As has been explained in section 5.3.2, this is not the case for all games, but Breakthrough and Atarigo are the two most flexible games when the variability of depth is taken as measurement. In Amazons and Skirmish, for instance, the game depth is rather fixed hence the simulation length did not change. The correlation between change in simulation length and standard deviation of depth is -0.55 (-0.66 if the variability is used instead of the standard deviation, see section 5.3.2). For this value and all subsequent values the data of the agent using all four enhancements has been used. So a higher flexibility allows for lower simulation lengths. However, the correlation is not that high so there are other more relevant factors.
In section 6.1.2 the observation was made, that an especially high decrease in simulation length could be seen with agents having a high winrate on that game. This connection is shown in figure 7. On the horizontal axis is plotted the deviation in the simulation length, meaning how much shorter the simulations of the ensemble have been. The vertical axis shows the winrate. It can be seen that most points line up on the black line plotted, which has been calculated using

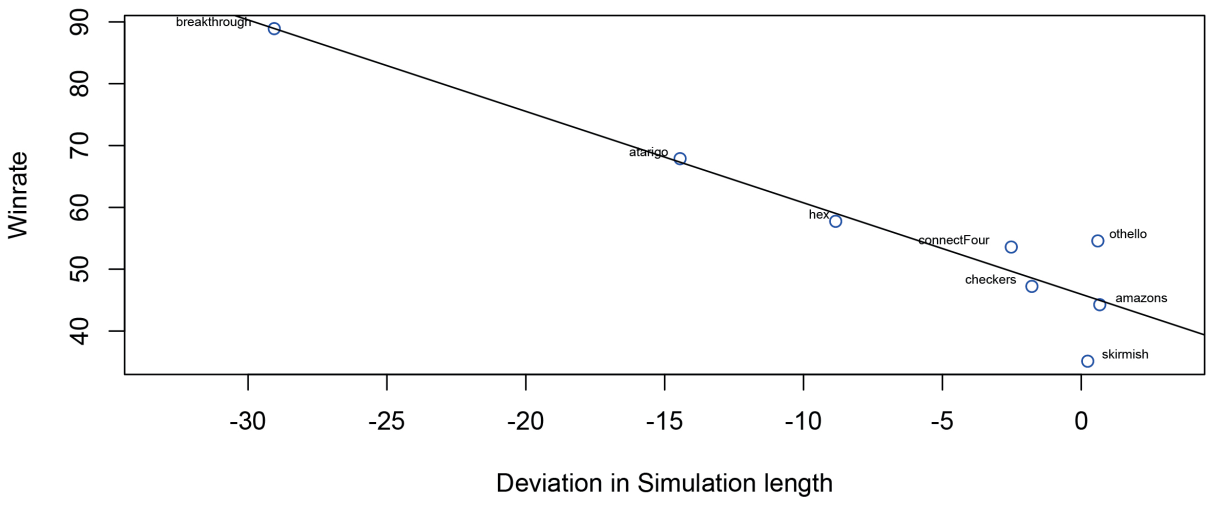

Figure 7: Winrate and simulation length compared

a linear regression. In fact, the correlation between change in simulation length and winrate is -0.94, which is very strong. The minus indicates a shorter simulation length acounting for a higher winrate. It makes sense that shorter simulations are more precise and result in better estimates of a certain state. There is less chance involved, as the variance for each move adds up during a rollout. For games like atarigo and breakthrough, the ensemble thus makes more efficient moves which lead to a faster terminal state than random rollouts, which in turn leads to a more accurate estimate and thus to a higher winrate.

6.2.2 Games

Before evaluating individual games some attention is given to the general game properties. In the previous section a correlation between the variability of the game depth and the winrate of the agent using all enhancements has been found. Since all agents follow a similar trend on each game, the agent using all enhancements will be used for subsequent calculations. Looking at the depth alone, is has a correlation of -0.67 to the winrate. This means a shorter depth supports a higher winrate. Interestingly, the correlation between winrate and standard deviation of depth is 0.62 (0.69 using the variability of depth). This indicates that a game offering more flexibility is more susceptible to be improved as the possibility of replacing lengthy rollouts with great variance by shorter, more precise ones, is important.
For the width, the connection is not as strong. The correlation between width and winrate is -0.25, the one between the standard deviation of the width and the winrate is -0.3. This is a weak correlation though generally present.
As conclusion smaller games, especially when depth is concerned, lead to a better improvement. This makes sense as all enhancements rely on moves being sufficiently evaluated before, be this either in combination with other moves or in general. This is harder to achieve in bigger games, as there will be more moves and more combinations hence limiting the accuracy of the evaluation of individual moves. Interesting is, that the standard deviation of width (the correlation is -0.3) is seen negatively although the correlation is low, while the standard deviation of depth (the correlation is 0.62) greatly helps the performance. However, a strong change in width corresponds to a strong change in available moves, meaning that there can be completely different moves from state to state. This might be hard to capture for the enhancements in general. The correlation-coefficients between the winrate and the game properties are summarized in table 19.

Table 19: Correlation between Winrate and Game properties

	Depth	Sd Depth	Var Depth	Width	Sd Width
Winrate	-0.67	0.62	0.69	-0.25	-0.3

Secondly, the C_p-parameters deserve some attention. As stated in section 5.3 all games only allow draws, wins and losses and nothing in between. It therefore takes multiple evaluations to get a reliable estimate of the true worth of a move. Hence exploitation of known moves is preferred over exploration of new ones which results in the overall very low C_p-parameters.

Amazons

As mentioned in section 5.3.2, the game Amazons is difficult to play for a MCTS agent. With an average branching factor of 423 and a simulation count of 34.24 for the standard mcts agent, the tree building is very limited until the very end of the game. In this environment, the enhancements show little effect, with no enhancement achieving a winrate of above 55%. Hence this game seems to complex to effectively be enhanced in such a setting as the current.

Atarigo

Atarigo seems very susceptible to improvement according to the results with the ensembles. They all achieve a winrate above 65% with the exception of the agent without MAST. This can be explained: The game ends relatively fast (5.93 moves per player) and is very variable in depth (0.5 as variability, this is the highest value for all games). It thus greatly benefits from shorter and more efficient simulations (as described in section 6.2.1) because they have a more accurate result.

Breakthrough

Breakthrough shows the biggest overall improvement with the winrates of MAST and without PAST being above 90%. Also, this game shows some variability in depth, having the second highest variability with 0.03. Since the goal of the game is to get to the enemy home line faster than the opponent, efficient and more acccurate simulations seem very important for this game. Therefore injecting a random move in an otherwise optimal sequence of play can turn it into a losing one if the difference to the opposing best sequence was small. This can thus completely change the evaluation of a position.

Checkers

Checkers is very unimpressed by the ensemble in general, with winrates being in the region of 55%. Just using MAST, however, increases the performance to 64.19%. This will be looked into in section 6.2.7 covering the MAST agent.

Connect Four

Connect Four shows small improvements on all ensembles (except for the version without MAST). Since Connect Four achieves 409.55 simulations on an standard MCTS player while having only a width of 7.13, a reasonably sized tree is built. Therefore the simulation-specific enhancements have less effect since the tree policy takes most of the responsibility for the success of the agent. Hence the differences in the winrates remain small.

Hex

Hex shows some substantial performance gains through the ensembles. It is the most strategic of all games used and has very stable board position. This means that the importance of certain moves change only slowly throughout the game. This can be captured very good by the ensembles, thus resulting in an improvement of around 10% compared to the standard MCTS agent.

Othello

Othello shows winrates around 55% for all ensembles except the one without MAST. This game, however, achieves only 2.23 simulations per average. This is neither induced by a huge width nor depth (Othello has the second smallest width of all games and a modest depth). The states are simply hard to prove for the reasoner, as Othello has also by far the lowest amount of state advances (see section 6.1.1). This also explains why the number of state advances in Othello stay similar for all agents, since any calculations done by the enhancements are of little importance when compared to the costly state advances. Since evaluating such a low amount of moves per turn results in very high variance, the enhancements do not have a great impact, either. The same can be said about the C_p-parameter, which is comparably high for this game (see section 6.1.3.1). Since different values of the parameter showed no substantial difference in performance during the optimisation process (also the correlation between the C_p-parameter and the winrate is -0.04), the actual value is determined by luck and has thus no meaningful interpretation.
Still, the standard MCTS player achieves a winrate of 84.42% against a random player using the optimised C_p-parameter. Therefore MCTS is a meaningful player in othello, however, fine tuning the algorithm with parameters and enhancements has little impact.

Skirmish

Skirmish is the only game where all ensembles worsen the performance. Being very similar to chess, the moves of Skirmish change their importance drastically dependent on the actual board position. No enhancement manages to capture the entirety of the actual board position, hence they all introduce negative bias to the random rollouts. Using the ensemble to mix the different biases together does not help either, as the agent using just MAST turned out to be the strongest on this game.

Generally can be said that all ensembles follow a similar trend as can be seen in figure 6. Therefore the uninformed rollout heuristics do not offer the desired diversity on the different games.

6.2.3 Without MAST

Omitting MAST seems detrimental to the performance of the ensemble method. Only for the game Breakthrough is this version considerably above 50% winrate, while it is below it on most other games. Therefore MAST seems to play a crucial role in the success of the ensemble method. The biggest losses are on Atarigo and Breakthrough, where the winrate dropped about 20% compared to the base version and Hex, with a drop close to 10%.
With Hex and Atarigo this can easily be explained since placing the stones on key positions on the board is certainly beneficial throughout the game. In Breakthrough, moving certain paws achieves a similar effect, hence the assumptions for the enhancement MAST (see section 4.1) are strongly true for these games. This is coherent with the findings in [10], where MAST achieved a great improvement on Breakthrough.
This is also backed up by the results shown in 6.1.3 in that the MAST parameters are generally small and thus try to gain impact on the ensemble using a sharp distribution. For the games Othello, Amazons and Skirmish, where the winrate of the ensemble without MAST did not drop significantly, the parameters are not that small either.
Also MAST does not seem to induce computational overhead as the number of state advances (see section 6.1.1) does not substantially change from the base version when MAST is left out. This makes sense since MAST only needs to compute one average for every move in the game.

6.2.4 Without PAST

Based on section 6.1.1, PAST seems to be computational wearisome in most games. This has already been stated in section 3.3.2 and is thus confirmed here. Having to calculate a score for each move-predicate pair is generally very expensive.
Omitting PAST seems very promising for the performance of the ensemble method, hence PAST has a very poor effect on it. As consequence PAST seems to perform very poorly on most games, at least it cannot make up for the loss of state advances.
This seems to contradict the findings in [10] where PAST was shown to have a very good performance on the game Breakthrough. However, this is likely due to the modification done to PAST as explained in section 4.2. Omitting the threshold for low variance samples seems to be a bad idea. However, with this threshold, PAST is increasingly mixed with MAST. Actually, using the parameters of [10] on the game of breakthrough, MAST is used 88% of the time while PAST only in 12% (sampled from 100 games). This means that, in fact, it is MAST doing all the work which explains the similarity of the winrates of MAST and PAST in [10].
Therefore, when omitting the threshold it is PAST doing all the work, which does not perform well. Hence using state-specific knowledge based on the state-predicates does not work in this form.

However, this result has to be taken with care. While both works are in the context of general game playing, Finnsson and Björnsson [10] used a play clock of 10 seconds instead of 5 as this thesis uses. They also have a different state machine and they could have slightly different versions of the game, too. These factors can lead to a change in the performance of PAST. What can be said with clarity, however, is that PAST cannot be generalised to different experimental settings as it does not work with the one used in this thesis.
This is also coherent with the findings in 6.1.3 in that the parameter of PAST seems to take random values for the different games using different ensembles. This would be hardly possible if there existed true values of PAST for which the enhancement had a considerable positive effect.

6.2.5 Without Tiles

Tiles requires to compute a score for every pair of moves already played, this can be quite costly when the depth increases. As shown in section 5.3.2, Checkers, Skirmish and Hex have a high depth. On these games the agent without Tiles has a substantially higher state advance count than the base version. Likewise, for the game Atarigo, having a very short depth, Tiles imposes no substatial overhead.
Omitting Tiles does not seem to make any difference to the ensemble at all, except for the game of hex. This can have several reasons: Tiles is a good enhancements in some circumstances as shown in [13]. However, the improvement shown there was only about 8% which is not comparable with the improvements achieved by MAST in [10]. So, while Tiles can still be a good improvement, it seems to lack a big impact to reasonably influence the ensemble.
A second explanation is that the settings used by Rimmel and Teytaud [13] with 1000 simulations per turn are not applicable in this context with most simulation counts being below 100. Therefore the possibility exists that Tiles simply does not work under these circumstances and consequently shows no improvement in the ensemble.
Tiles being bad in Hex seems to contradict the findings [13] at the first glance. There, it was tested on the game Havannah, which can be seen as being equal to Hex with the addition of offering short term win-conditions and thus tactical elements. However, as was shown in [11], MCTS agents win by these short-term win-conditions when using low simulations per game (1000 is seen as low in that paper). Therefore, the results found in the game of Havannah are not applicable to Hex under these circumstances. Furthermore, in Hex being such a strategic game, it is resonable that playing two certain pieces together is not beneficial. This might bias the agent to converge towards more short-term solutions while losing the focus on the general strategic position on the board. Therefore Tiles introduces negative bias and worsens the performance.

6.2.6 Without LGR

Omitting LGR seems to make absolutely no difference when looking at the winrate. Based on the data in 6.1.1, the computational overhead is also nonexistend.

The reason here, similar to Tiles, can be seen in the weak improvement Tiles provides. LGR has been tested against baseline MCTS players by Teytaud and Dehos [11], never achieving an improvement above 8%. Also, the results were tested in circumstances very different from this setting with the simulation count ranging from 1000 to 10000 per turn.
On the other hand, LGR is not bad for the ensemble, either, meaning that it does not bias the moves in a bad direction.

6.2.7 Only MAST

As can be seen in table 11 the agent using just MAST hardly differs from the ensemble without PAST. This can be explained using the τMAST parameter values:
Higher values make the distribution more uniform, hence more exploration is done during the rollouts. This effect is comparable to mixing MAST with other enhancements. That explains why the τMAST parameters of the agent using just MAST (table 17) are substantially higher (around 7 on average) than the same parameter from the ensemble without PAST (see table 15) as they are almost all below 1. This is not true for Checkers, however. The $\tau MAST$-value stays small at 0.474. Unlike the other values this sharp MAST-value cannot be achieved by mixing MAST with other enhancements. This explains why the agent just using MAST is significantly better than the agent without PAST on Checkers. That Checkers likes such a sharp distribution can be explained since checkers is the game with the second highest depth (48.59) of all games. Long depths and therefore long simulations create a high variance in the outcome of the simulation as the variance of every random move adds up. Therefore Checkers prefers its rollout policy strongly biased towards the best move in each step to reduce this possible variance. As explained in section 6.2.2, the game with the highest depth of 49.78, Skirmish, does not benefit from MAST therefore this effect cannot be observed. Both games have quite a strong lead on the game with the third highest depth, Amazons, which has a depth of 39.22 moves per player. Amazons has a MAST-parameter of 2.768 on the agent just using MAST, which is comparably small, too.
For Atarigo things look differently. The version without PAST achieves a win rate almost 10% higher than the agent just using MAST. It therefore seems like the exploration offered by the enhancements LGR and Tiles is better than increasing the τMAST-value. Table 20 lists the performance of an agent using MAST and LGR and an agent using MAST and Tiles on both Atarigo and Checkers in order to determine which of these two enhancements is responsible for the performance difference. Additionally, the percentage of random rollouts done by each enhancement is also listed.

MAST/Tiles performs about 3% better in both cases, both results being in between the performances of just MAST and without PAST. This offers no new hindsight to the performance of Checkers. However, no single enhancement can be accounted for the performance difference in Atarigo, since they perform both very similar.
Tiles does a very high amount of random rollouts at about 98% in both cases. This is partly induced by very low ϵ-parameters. On the other hand, LGR performs less random rollouts, but

Table 20: Winrate and random rollouts of MAST/LGR and MAST/Tiles

Game	W MAST/LGR	W MAST/Tiles	R MAST/LGR	R MAST/Tiles
Atarigo	71.43	74.41	67.22	98.07
Checkers	58.43	61.51	90.42	98.64

W = Winrate
R = Percentage Random Rollouts done

still does so more than two thirds of the time on both games. Therefore the small performance advantage from MAST/Tiles over MAST/LGR is possibly due to the increased amount of random rollouts and not due to the overall benefit of the enhancement. This explains why both enhancements perform very similarly.
As first conclusion, mixing MAST with random rollouts seems promising on the game Atarigo. Though this has only been observed on Atarigo, this game is special in having by far the shortest depth of all games (5.93 moves per player) and in being the most tactical game. In this context some unbiased exploration seems beneficial as it allows for more variety in the search.

6.2.8 Strongest Agent

Based on the findings in the previous sections, a new method has been constructed. It follows a simple basic idea: As seen from the performance of without PAST, very sharp values of τMAST work relatively good when mixed with random rollouts. As seen from the performance of MAST, the ratio of mixture is not always the same for every game. A sharp τMAST-value, however, leads to the MAST-distribution focussing on the strongest move. The new method will combine both properties, taking the best move from MAST into account and mixing it with random rollouts. Therefore the following rollout distribution is proposed, with the probability of move m being chosen as:

$$P(m) = \begin{cases} \epsilon + \frac{1-\epsilon}{|M|} & \text{if m is the best move} \\ \frac{1-\epsilon}{|M|} & \text{otherwise} \end{cases}$$

M is the set of all available moves at the current state. ϵ is a parameter to be optimised. The method follows an ϵ-greedy distribution. The best move is taken with probability ϵ, otherwise a move is chosen uniformly at random.
The enhancement uses therefore 1 parameter and has been optimised using 90 runs. The resulting parameters are shown in table 21.

The higher the parameter ϵ the stronger is the focus on the best move. As can be seen, the game Checkers, which had previoulsy focused on a sharp τMAST-value, now focussed strongly on the best move with an ϵ-value of 0.94. The contrary is Skirmish. Since MAST does not add benefits to this game, it has made the ϵsmall to try to effectively play as the standard MCTS player.
The winrate of the new agent is presented in table 22 and compared to MAST and without PAST. The asterix indicates a significant deviation of the new method to whichever of the other two

Table 21: Parameters

Game	ϵ
Amazons	0.5418
Atarigo	0.5959
Breakthrough	0.754
Checkers	0.9434
Connect Four	0.1642
Hex	0.464
Othello	0.9214
Skirmish	0.0398

Table 22: Winrate of the new agent

Game	Without PAST	MAST	New Agent
Amazons	53.37	54.17	52.98
Atarigo	77.78	64.29	75.20
Breakthrough	93.45	92.66	94.05
Checkers	55.56	64.19	68.75
Connect Four	54.37	56.25	53.67
Hex	63.89	59.13	65.87
Othello	52.08	52.88	52.78
Skirmish	43.85	45.83	47.52

agents had the higher winrate for this game (5% significance level). As can be seen, there is none. The new agent achieves a 4% improvement on Checkers, but with a p-value of 0.1029 this does not quite make it to the 5% threshold. It is therefore not statistically significant.
The winrate on Skirmish, despite not being significantly different from the winrate of the MAST agent, opproaches the 50% mark and is not significantly different from that mark either. It is still possible, that at an ϵ-value of 0.0398, the MAST move still has a negative influence on Skirmish. Because of the robust parameter optimisation, this was the smallest value the parameter could take.

7 Conclusion

This paper has investigated whether an ensemble method is applicable to uninformed rollout heuristics for MCTS. This was done in the field of General Game Playing, testing the method on eight two-player games. There have been several conclusions drawn from these experiments, they are presented in the following:

- On games with variable game length, such as Atarigo, Breakthrough and Hex, agents with a higher winrate showed a substantial reduction in simulation length. This can be used in the future to extract more information out of the Monte Carlo simulations as the simulation length could be used as an additional indicator for playing strength.
- The ensemble has not shown to significally benefit from the different types of enhancements. Only one particular enhancement (MAST) proved essential to the ensemble and dominated the others in terms of performance. The idea that different enhancements prove significant dependent on the game did not hold true.
- PAST generally worsenend the performance of the agent indicating that the enhancement itself performs poorly in this particular test setting
- MAST has been shown to increase its performance when mixed with a different type of exploration. Therefore a new method has been constructed, effectively mixing MAST with random rollouts. This method has proven to be at least as good as the previously known strongest agent on each game with only statistically insignificant deviations.

There are important limitations of this paper, hence there remain various fields for further research. This paper has only tested 4 simulation phase enhancements using simple voting on eight two player games in the context of General Game Playing. First of all, these finding, especially the new method, have only been testet on two-player games. To generalise this in the context of GGP, the new method has to be tested on single-player and other multiplayer-games as well. Additionally, since this method still uses one parameter, it can be researched if there exists a reliable value that performs well on a wide variety of games.
Secondly, the ensembe only tested uniformed rollout heuristics. While this is a good first step, part of the missing success could be explained in those methods being to similar in effect. Therefore an ensemble with different enhancements could prove successfull. A start can be FAST (see section 3.3.5), which has a very strong performance on Skirmish, where none of the agents tested achieved an improvement. Also Tree-only MAST can be worth looking at as MAST has proven very strong, so adding a tweaked version of it can be beneficial in an ensemble.
Furthermore, this thesis has shown that strong simulation phase enhancements like MAST can be improved by adding random rollouts to it. This can be tested on other enhancements, too, which can possibly lead to some meaningful performance gains.

Lastly, all of the above points can be tested if they hold true outside of the GGP-context. These are often environments with very different settings e.g. more simulation being done per turn, which may greatly influence the results.

List of Figures

List of Tables

Bibliography

[1] Moyer, "Alphago beats human world champion in go," http://www.theatlantic.com/technology/archive/2016/03/the-invisible-opponent/475611/, 2010, accessed: 2016-05-23.

[2] M. Genesereth, N. Love, and B. Pell, "General game playing: Overview of the aaai competition," *AI magazine*, vol. 26, no. 2, p. 62, 2005.

[3] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 4, no. 1, pp. 1–43, 2012.

[4] H. Finnsson and Y. Björnsson, "Simulation-based approach to general game playing." in *AAAI*, vol. 8, 2008, pp. 259–264.

[5] J. Méhat and T. Cazenave, "Combining uct and nested monte carlo search for single-player general game playing," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 2, no. 4, pp. 271–277, 2010.

[6] M. Thielscher, "A general game description language for incomplete information games." in *AAAI*, vol. 10. Citeseer, 2010, pp. 994–999.

[7] S. Schiffel and Y. Bjornsson, "Efficiency of gdl reasoners," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 6, no. 4, pp. 343–354, 2014.

[8] B. E. Childs, J. H. Brodeur, and L. Kocsis, "Transpositions and move groups in monte carlo tree search," in *Computational Intelligence and Games, 2008. CIG'08. IEEE Symposium On*. IEEE, 2008, pp. 389–395.

[9] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *Machine Learning: ECML 2006*. Springer, 2006, pp. 282–293.

[10] H. Finnsson and Y. Björnsson, "Learning simulation control in general game-playing agents." in *AAAI*, vol. 10, 2010, pp. 954–959.

[11] F. Teytaud and J. Dehos, "On the tactical and strategic behaviour of mtcs when biasing random simulations," *ICGA Journal*, vol. 38, no. 2, pp. 67–80, 2015.

[12] J. Schaeffer, "The history heuristic and alpha-beta search enhancements in practice," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 11, no. 11, pp. 1203–1212, 1989.

[13] A. Rimmel and F. Teytaud, "Multiple overlapping tiles for contextual monte carlo tree search," in *Applications of Evolutionary Computation*. Springer, 2010, pp. 201–210.

[14] P. Drake, “The last-good-reply policy for monte-carlo go,” *International Computer Games Association Journal*, vol. 32, no. 4, pp. 221–227, 2009.

[15] A. Rimmel, F. Teytaud, and O. Teytaud, “Biasing monte-carlo simulations through rave values,” in *International Conference on Computers and Games*. Springer, 2010, pp. 59–68.

[16] B. Parhami, “Voting algorithms,” *Reliability, IEEE Transactions on*, vol. 43, no. 4, pp. 617–629, 1994.

[17] T. G. Dietterich, “Ensemble methods in machine learning,” in *Multiple classifier systems*. Springer, 2000, pp. 1–15.

[18] L. Breiman, “Bagging predictors,” *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[19] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.

[20] L. Kocsis, C. Szepesvári, and J. Willemson, “Improved monte-carlo search,” *Univ. Tartu, Estonia, Tech. Rep*, vol. 1, 2006.

[21] L. Fahrmeir, R. Künstler, I. Pigeot, and G. Tutz, *Statistik - Der Weg zur Datenanalyse*. Heidelberg: Springer-Verlag, 2011.

[22] J. Kloetzer, H. Iida, and B. Bouzy, “The monte-carlo approach in amazons,” in *Proceedings of the Computer Games Workshop*, 2007, pp. 185–192.

[23] A. Rimmel, O. Teytaud, C.-S. Lee, S.-J. Yen, M.-H. Wang, and S.-R. Tsai, “Current frontiers in computer go,” *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 2, no. 4, pp. 229–238, 2010.