N-Grams and the Last-Good-Reply Policy Applied in General Game Playing

Mandy J. W. Tak, Mark H. M. Winands, and Yngvi Björnsson

***Abstract*—The aim of general game playing (GGP) is to create programs capable of playing a wide range of different games at an expert level, given only the rules of the game. The most successful GGP programs currently employ simulation-based Monte Carlo tree search (MCTS). The performance of MCTS depends heavily on the simulation strategy used. In this paper, we introduce improved simulation strategies for GGP that we implement and test in the GGP agent CADIAPLAYER, which won the International GGP competition in both 2007 and 2008. There are two aspects to the improvements: first, we show that a simple ϵ-greedy exploration strategy works better in the simulation play-outs than the softmax-based Gibbs measure currently used in CADIAPLAYER and, second, we introduce a general framework based on N-grams for learning promising move sequences. Collectively, these enhancements result in a much improved performance of CADIAPLAYER. For example, in our test suite consisting of five different two-player turn-based games, they led to an impressive average win rate of approximately 70%. The enhancements are also shown to be effective in multiplayer and simultaneous-move games. We additionally perform experiments with the last-good-reply policy (LGRP). The LGRP combined with N-grams is also tested. The LGRP has already been shown to be successful in Go programs and we demonstrate that it also has promise in GGP.**

***Index Terms*—General game playing (GGP), last-good-reply policy (LGRP), Monte Carlo tree search (MCTS), N-grams.**

I. INTRODUCTION

PAST research in artificial intelligence (AI) in games has focused on the development of programs capable of playing one particular game at a world-class level. Such programs have in common that they depend heavily on elaborate game-dependent knowledge, typically provided by their developers. Furthermore, many of the techniques employed in these programs are highly specialized and fine tuned for the particular game at hand. In general game playing (GGP), on the contrary, the aim is to create programs capable of playing a wide range of different games at an expert level, some of which they may never have encountered before. This imposes extra challenges compared to the more traditional approach where the game to play is known beforehand. First, as such programs do not have any game-specific knowledge available beforehand, they have to learn it by themselves during play. Second, it can no longer be decided in advance by the human developer which search method suits best for a particular game. Thus, the program has to decide by itself how to spend the available time searching for the best move. Third, no parameter turning can be done offline, which imposes the application of online machine learning. These three challenges already entail that in GGP a program can become successful only if it incorporates a wide range of different AI techniques, like knowledge representation, knowledge discovery, machine learning, search, and online optimization.

The first successful GGP programs, such as CLUNEPLAYER[1] and FLUXPLAYER[2], [3], were based on minimax with an automatically learned evaluation function. CLUNEPLAYER and FLUXPLAYER won the International GGP competition in 2005 and 2006, respectively. However, GGP programs using a Monte-Carlo-based approach have proved more successful and won the competition in all subsequent years: CADIAPLAYER[4] in 2007 and 2008, Ary[5], [6] in 2009 and 2010, and TURBOTURTLE in 2011 [7].

Monte Carlo tree search (MCTS) [8]–[10] and/or nested Monte Carlo search [6] work well in GGP because no game-specific knowledge, besides the basic rules of the game, is required at the start. Moreover, MCTS can be applied in one-, two-, and multiplayer games, whereas nested Monte Carlo search is particularly suited for one-player games. However, to be truly effective, MCTS requires both a robust strategy for controlling the tradeoff between exploration and exploitation of actions, as well as heuristic knowledge for effectively guiding the MCTS simulations [11]. The contribution of this paper is to propose and evaluate effective methods for performing those tasks in GGP. More specifically, we 1) show that a simple ϵ-greedy exploration strategy works better in the simulation play-outs than the more commonly used softmax-based Gibbs measure; 2) introduce a general framework based on N-grams for learning promising move sequences, which results in much improved play; and 3) show that the last-good-reply policy (LGRP) [12], [13], which is successful in *Go*, also holds promise in GGP.

The paper is structured as follows. In the next two sections, we provide necessary overview background on GGP and MCTS, respectively. The subsequent section describes the details of the simulation guidance approach used in CADIAPLAYER as well as the new approaches introduced in this paper. This is followed by experimental setup and result sections, and finally we conclude and discuss future work.

Manuscript received October 25, 2011; revised February 24, 2012; accepted April 30, 2012. Date of publication May 21, 2012; date of current version June 12, 2012. This work was supported by the Netherlands Organisation for Scientific Research (NWO) in the framework of the project GoGeneral under Grant 612.001.121.

M. J. W. Tak and M. H. M. Winands are with the Games and AI Group, Department of Knowledge Engineering, Faculty of Humanities and Sciences, Maastricht University, Maastricht 6200MD, The Netherlands (e-mail: mandy.tak@maastrichtuniversity.nl; m.winands@maastrichtuniversity.nl).

Y. Björnsson is with the School of Computer Science, Reykjavík University, Reykjavík IS-101, Iceland (e-mail: yngvi@ru.is).

Digital Object Identifier 10.1109/TCIAIG.2012.2200252

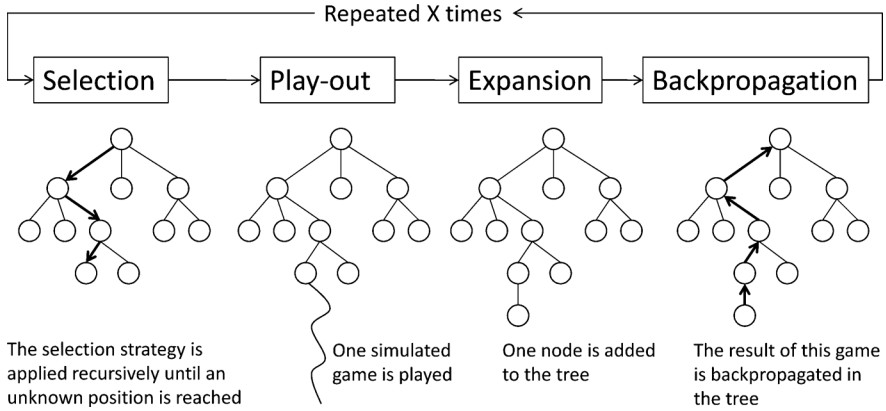

Fig. 1. Four strategic steps in Monte Carlo tree search (adapted from [10]).

II. General Game Playing

The Logic Group at Stanford University (Stanford, CA) initiated the annual International GGP competition to stimulate research in the area of GGP. As a part of the initiative they developed a standard for describing the rules of a game. This section explains briefly how game rules are specified and how matches between GGP agents are conducted.

A. The Game Description Language

The rules of a game are expressed in the Game Description Language (GDL) [14] which is a specialization of the Knowledge Interchange Format (KIF) [15]. It is a first-order logic language for describing knowledge. With GDL *n*-player, deterministic, perfect information, simultaneous-move games can be described. Turn-based games are represented by introducing a so-called *noop* (no operation) move which has no effect and is the only possible move for the player currently not on turn. Recently, GDL-2 [16] was introduced, which allows games with chance and imperfect information to be described.

In GDL, a game state is represented by a set of true facts. The legal moves in that state are described with logical rules. These legal moves define the possible transitions to other states. For a detailed specification of GDL, we refer to [14].

B. Game Master

A game-master server orchestrates games played by GGP agents. The *Dresden GGP Server* is a well-known game-master server hosted online [17]. Furthermore, a standalone Java implementation is freely available under the name *GameController*[18]. The agents register themselves at the server. When the game starts, the game master sends the rules of the game to the players, including their role, the *startclock*, and the *playclock*. The startclock is the time between receiving the rules and the first move. The playclock is the time between each move after play has started.

Each agent sends its move to the game master. If the moves are legal the game master applies them to the current game state. If a player sends an illegal move it is replaced with a random move determined by the game master. The game master informs the agents about all moves played such that each agent can update its internal game state accordingly. The game ends when a terminal state is reached. The agents are then informed about the obtained rewards.

III. Monte Carlo Tree Search

CadiaPlayer [4] uses MCTS [8]–[10] to determine which moves to play. The advantage of MCTS over minimax-based approaches is that no evaluation function is required. This makes it especially suited for GGP in which it is difficult to come up with an accurate evaluation function. MCTS is a best-first search strategy that gradually builds up a tree in memory. Each node in the tree corresponds to a state in the game. The edges of a node represent the legal moves in the corresponding state. Moves are evaluated based on the average return of simulated games.

MCTS consists of four strategic steps [10] which are outlined in Fig. 1. 1) The *selection step* determines how to traverse the tree from the root node to a leaf node L. It should balance the exploitation of successful moves with the exploration of new moves. 2) In the *play-out step*, a random game is simulated from leaf node L until the end of the game. Usually, a *simulation strategy* is employed to improve the play-out [11]. 3) In the *expansion step*, one or more children of L are added. 4) In the *backpropagation step*, the reward R obtained is backpropagated through the tree from L to the root node.

In the following, we describe how these four strategic steps are implemented in CadiaPlayer.

1) In the *selection step*, the upper confidence bounds applied to trees (UCT) algorithm [9] is applied to determine which moves to select in the tree. At each node s move a^* selected is given by

$$a^* = \arg\max_{a \in A(s)} \left\{ Q(s,a) + C\sqrt{\frac{\ln N(s)}{N(s,a)}} \right\} \tag{1}$$

where $N(s)$ is the visit count of s and $N(s,a)$ is the number of times move a is selected in node s. The first term $Q(s,a)$ is the average return when move a is played in state s. The second term increases when state s is visited and siblings of a are selected. If a state s is visited frequently, then even moves with a relatively low $Q(s,a)$ could be selected again at some point, because their second term has risen high enough. Thus, the first

term supports the exploitation of successful moves while the second term establishes the exploration of infrequently visited moves. The C parameter influences the balance between exploration and exploitation. Increasing C leads to more exploration. If $A(s)$, the set of legal moves in state s contains moves that have never been visited before, then another selection mechanism is utilized, because these moves do not have an estimated value yet. If there is exactly one move that has not been visited before, then this one is selected by default. If there are multiple moves that have not been visited before, then the same simulation strategies as used in the play-out step are used to determine which move to select. In all other cases (1) is applied.

2) During the *play-out step* a complete game is simulated. The most basic approach is to play plain random moves. However, the play-outs can be improved significantly by playing quasi-random moves according to a *simulation strategy* [11]. The aim is to improve the performance of the already existing CADIAPLAYER by introducing new simulation strategies. These simulation strategies are described in Section IV.
3) In the *expansion step*, nodes are added to the tree. In CADIAPLAYER, only one node per simulation is added [8]. This node corresponds to the first position encountered outside the tree. Adding only one node after a simulation prevents excessive memory usage, which could occur when the simulations are fast. Other expansion strategies are discussed in [19].
4) In the *backpropagation step*, the reward obtained in the play-out is propagated backwards through all the nodes on the path from the leaf node L to the root node. The $Q(s,a)$ values of all state-move pairs on this path are updated with the just obtained reward. In GGP, the reward lies in the range $[0,100]$.

More details about the implementation of CADIAPLAYER can be found in [4] and [20].

IV. SIMULATION STRATEGIES

A good simulation strategy can improve the play-outs significantly [11]. The aim is to improve CADIAPLAYER by implementing new simulation strategies. The performance of these simulation strategies is compared with the move-average sampling technique (MAST) which was used by CADIAPLAYER when winning the GGP competition in 2008 [21]. The first subsection explains MAST and the adjustments we made to it, the second introduces a new simulation strategy based on N-grams, and the third describes the LGRP [12], [13], which is already applied successfully in *Go* and that we experiment with here in GGP.

A. Move-Average Sampling Technique

The MAST [21] is based on the principle that moves which are good in one state are likely to be good in other states as well. The history heuristic [22], which is used to order moves in $\alpha\beta$ search [23], is based on the same principle. For each move a, a global average $Q_h(a)$ is kept in memory. It is the average of the returned rewards of the play-outs in which move a occurred. These values are utilized for selecting moves in the play-out. Furthermore, if in the MCTS tree a node has more than one unvisited legal move, then the $Q_h(a)$ values of these unvisited moves are employed to determine which move to select. In the original version of MAST this is done using Gibbs measure

$$P(a) = \frac{e^{Q_h(a)/\tau}}{\sum_{b=1}^{n} e^{Q_h(b)/\tau}}.$$

$P(a)$ is the probability that move a will be selected. Moves with a higher $Q_h(a)$ value are more likely to be selected. How greedy the selection is can be tuned with the τ parameter. In order to bias the selection of unexplored moves the initial $Q_h(a)$ value is set to the maximum possible score of 100.

A disadvantage of the Gibbs measure is that the probability of selecting the move with the highest $Q_h(a)$ score is not fixed and unknown. Consequently, it is not assured that moves with the highest $Q_h(a)$ scores will be even chosen at all. Therefore, for a comparison, we implement a different exploration technique also based on the $Q_h(a)$ values, namely ϵ-greedy [24], [25]. With a probability of $1-\epsilon$ the move with the highest $Q_h(a)$ value is selected, and with a probability of ϵ, a legal move is chosen uniformly at random.

B. N-Gram Selection Technique

MAST generalizes the merits of moves from one state to the next without considering the context in which the moves are made. Nonetheless, despite its simplicity, this technique has proved quite successful and, for example, was the main simulation strategy used in CADIAPLAYER when winning the International GGP competition in 2008.

Subsequent work on more context-aware simulation strategies has returned a mixed success. For example, the predicate-average sampling technique (PAST) [26] adds context by taking into account the predicates that are true in the state in which a move is played. However, a disadvantage of PAST is that the benefits coming from having the context added this way were barely sufficient to offset the computational overhead incurred in keeping track of the additional predicates. Experiments revealed that PAST was only slightly better than MAST in *Checkers*, but in three other games PAST did not perform better [27].

The method we propose here, called the *N-gram selection technique (NST)*, keeps track of move sequences instead of single moves. It offers context in a way that is more favorable in terms of added benefits versus computational overhead than, for example, PAST does.

A method similar to NST is applied successfully in *Havannah* [28], [29]. Furthermore, NST also bears some resemblance to the simulation strategy introduced in [30], which is based on a tiling of the space of Monte Carlo simulations.

NST is based on N-gram models, often employed in statistical language processing [31] and in computer games for predicting the next move of the opponent [32], [33]. The N-grams in NST consist of consecutive move sequences S of length 1, 2, and 3. As in MAST the average of the returned rewards of the play-outs is accumulated. However, the average, here called $R(S)$, is kept also for longer move sequences as opposed to single moves only.

The N-grams are formed as follows. After each simulation, starting at the root of the tree, for each player all move sequences of length 1, 2, and 3 that appeared in the simulated game are extracted. The averages of these sequences are updated with the obtained reward from the simulation. It is not checked whether the same move sequence occurred more than once in the simulation. Thus, if there are m occurrences of the same move sequence, then the average of this sequence is updated m times.

The move sequences consist of moves from the current player and moves from the opponent(s). The role numbers $0, 1, 2, \ldots, n-1$, which are assigned to the players at the beginning of a game with n players, are employed in order to determine the move of which opponent to include in the sequences. Suppose that the current player has role number i and there are n players, then the sequences are constructed as follows. A sequence of length 1 consists of just one move of the current player. A sequence of length 2 starts with a move of the player with role $(i+n-1) \bmod n$ and ends with a move of the current player. A sequence of length 3 starts with a move of the player with role $(i+n-2) \bmod n$, followed by a move of the player with role $(i+n-1) \bmod n$ and ends with a move made by the current player. The moves in these sequences are consecutive moves.

Fig. 2 gives an example of a play-out. At each step, both players have to choose a move, because all games in GGP are assumed to be simultaneous-move games. The example given here concerns a turn-based, two-player game, which means that at each step one of the players can only play the *noop* move. The example shows that these *noop* moves are included in the sequences, because NST handles them as regular moves. This does not cause any problem, because these move sequences will only be used during move selection when the player is not really on turn and has the only option of choosing the *noop* move. Therefore, the move sequences containing *noop* moves do not negatively influence the decision process during the play-out.

If the game is truly simultaneous, then at each step all players choose an actual move instead of some players having to choose the *noop* move like in turn-based games. As explained above, NST includes only one move per step in its sequences. This means that for an n-player simultaneous game, moves of $n-1$ players are ignored in each step. Another possibility would have been to include the moves of all players at each step, but that would lead to too specific sequences. The disadvantage of such specific sequences is that fewer statistical samples can be gathered about them, because they occur much more rarely.

In the play-out, and at the nodes of the MCTS tree containing unvisited legal moves, the N-grams are used to determine which move to select. For each legal move, the player determines which sequence of length 1, which sequence of length 2, and which sequence of length 3 would occur when that move is played. The sequence of length 1 is just the move itself. The sequence of length 2 is the move itself appended to the last move played by the player with role $(i+n-1) \bmod n$. The sequence of length 3 is the move itself appended to the previous last move played by the player with role $(i+n-2) \bmod n$ and the last move played by the player with role $(i+n-1) \bmod n$. Thus, in total three sequences could occur. The player then calculates a score for a move by taking the average of the $R(s)$ values stored for these sequences. In this calculation, the $R(s)$ values for the move sequences of length 2 and length 3 are only taken into account if they are visited at least k times. In the performed experiments, $k = 7$. This value was determined by manual tuning.

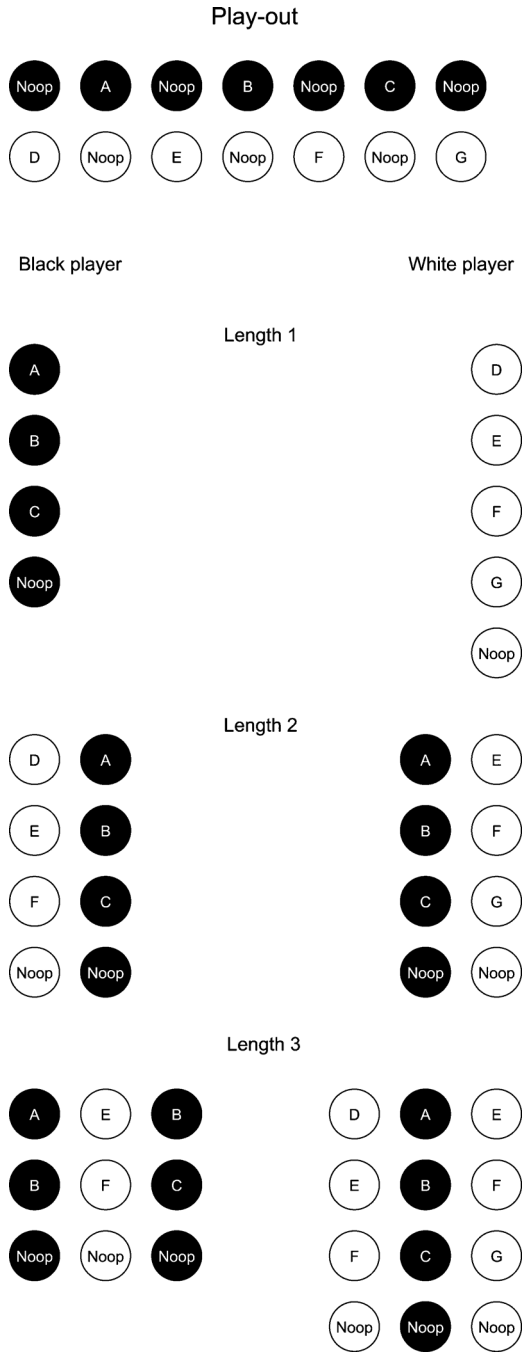

Fig. 2. Extracted move sequences from play-out.

If a move has been played at least once, but the sequences of length 2 and length 3 occurred fewer than k times, then the $R(s)$ value of the move sequence of length 1 (which is the move itself) will be returned.

If a move has never been played before, then no move sequences exist and the calculation outlined above is not possible. In that case the score is set to the maximum possible value of 100 in order to bias the selection toward unexplored moves.

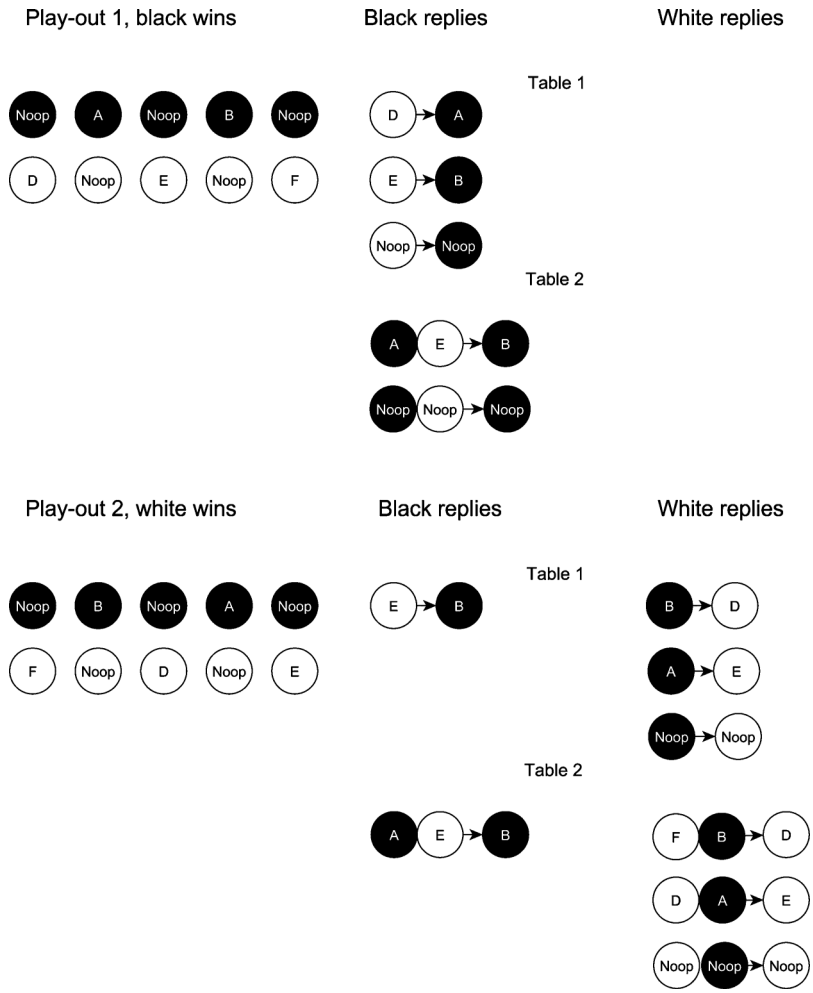

Fig. 3. Updating the reply tables from the last-good-reply policy.

In this manner, a score is assigned to each legal move in a given state. These scores are then used to determine which move to select. This can be done using either a Gibbs measure or an ϵ-greedy strategy.

C. Last-Good-Reply Policy

The LGRP [12], [13] is a simulation strategy in which successful replies that occurred in the play-outs are kept in memory. It bears similarities with the countermove heuristic [34]. The strategy is to play a move that is a successful reply to the immediately preceding moves. If such a legal move is unavailable, then a default fallback strategy decides which move to play. The LGRP has been already applied in *Go* and *Havannah* [12], [28]. In [13], an enhancement is introduced that forgets a stored reply when in a later simulation the reply is unsuccessful. This modification improves the LGRP in *Go*. It is interesting to investigate whether this simulation strategy works in a broader range of games. We thus implement and experiment with the LGRP in CADIAPLAYER.

The implementation of LGRP in CADIAPLAYER is similar to the variant LGRF-2 which is described in [13]. Like in NST, the role numbers $0, 1, 2, \ldots, n-1$ and the number of players n are employed for determining to which moves a best reply is stored. For each player i, two separate tables are created. In one table, the best replies to a previous move of the player with role number $(i+n-1) \bmod n$ are stored. In the second table, the best replies to the previous two moves are stored. This sequence of two moves starts with a move made by the player with role number $(i+n-2) \bmod n$ and ends with a move played by the player with role number $(i+n-1) \bmod n$. The moves have to occur directly after each other.

The tables are updated after each simulation. If the reward obtained by the player is at least as high as the highest reward obtained by any of the players, then the moves made by the player are stored in the table as being a best reply to their immediate predecessor(s). If a best reply is already stored it is overwritten, because only one reply is stored per predecessor move(s). If the reward of the player is lower than that of any of the opponents, then the moves of the player are removed from the table if they were stored as best replies to their immediate predecessor(s).

Fig. 3 gives an example of how the best-reply tables are updated. The first simulation is a win for *black*, so its moves are stored as best replies in the tables. For *white*, nothing is stored because it loses. In the second simulation, *black* loses. Therefore, if an already stored reply occurred, it is removed from the table. In this case, it means that $D \longrightarrow A$, *noop* $\longrightarrow$ *noop*, and *noop*, *noop* $\longrightarrow$ *noop* are deleted. Like in NST, the *noop* moves are included in the tables, but as explained in Section IV-B, this does not cause any problem.

These best-reply tables are used during the play-out and for selecting moves in the MCTS tree when a node has unvisited legal moves. These tables are already employed after the first play-out. The move stored as a best reply to the previous two moves is chosen. These two previous moves form a sequence of consecutive moves, which starts with a move from the player with role number $(i + n - 2) \bmod n$ and ends with a move of the player with role number $(i + n - 1) \bmod n$. If a best reply to this sequence is not legal or unavailable from the tables, then the selected move is the stored best reply to the last move from the player with role number $(i + n - 1) \bmod n$. If also that move is unavailable or illegal, then a default strategy is used. In the experiments, two default strategies are tested. The first one is MAST with ϵ-greedy [24]. The second one is NST with ϵ-greedy.

V. Experimental Setup

We implemented the aforementioned methods in CadiaPlayer to investigate their effectiveness for GGP. Section V-A introduces the games used in the experiments. Section V-B describes the setup of the experiments.

A. Games

In the following, an overview is given of the games employed in the experiments. Note that most of the classic games enlisted below are usually a variant on its original counterpart. The most common adjustments are a smaller board size and a bound on the number of steps. In all experiments, the following five two-player, turn-based games are used.

- *Connect5* is played on an 8×8 board and the player on turn has to place a piece in an empty square. The aim is to place five consecutive pieces of the own color horizontally, vertically or diagonally, like *Go-Moku* or *Five-in-a-Row*.
- *Checkers* is played on an 8×8 board and the aim is to capture pieces of the opponent.
- *Breakthrough* is played on an 8×8 board. Each player starts on one side of the board and the aim is to move one of their pieces to the other side of the board.
- *Othello* is played on an 8×8 board. Each turn a player places a piece of its own color on the board. This will change the color of some of the pieces of the opponent. The aim is to have the most pieces of the own color on the board at the end of the game.
- *Skirmish* is played on an 8×8 board with different kinds of pieces, namely: bishops, pawns, knights, and rooks. The aim is to capture as many pieces from the opponent as possible, without losing too many pieces either.

These games were chosen because they are used in several previous CadiaPlayer experiments [4], [20], [21], [26], [27], [35].

In the final series of experiments additional games are included. This helps with evaluating the methods' robustness across a wider range of games, including simultaneous-move and multiplayer ones. The following two-player, turn-based, games are added.

- *Clobber* is played on a 4×5 board which is completely filled initially. Each turn a player moves one of its pieces to a neighboring cell that has to be occupied by the opponent. The first player that cannot move loses the game. This is a smaller version of *Clobber*, because the tournament board size is 10×10.
- *Sheep and Wolf* is an asymmetrical game played on an 8×8 board. One player controls the Sheep and the other player controls the Wolf. The game ends when none of the players can move or when the Wolf is behind the Sheep. In this case, if the Wolf is not able to move the Sheep wins. Otherwise, the Wolf wins.

The following three-player, turn-based games are added.

- *TTCC4* stands for: *TicTacChessCheckersFour* and is played on a 7×7 board. Each player has a pawn, a checkers piece, and a knight. The aim of each player is to form a line of three with its own pieces.
- *Chinese Checkers* is played on a star-shaped board. Each player starts with three pieces positioned in one arm. The aim is to move all these three pieces to the empty arm at the opposite side of the board. This is a variant of the original *Chinese Checkers*, because according to the standard rules each player has ten pieces instead of three.

The following two-player, simultaneous-move games are added.

- *Chinook* is a variant of *Breakthrough* where two independent games are played simultaneously: one game on the white squares and another one on the black squares.
- In *Runners*, both players decide at each turn how many steps they want to move forward or backward. The aim is to reach a certain location before the opponent does.
- *Fighter* is a fighting game where both players try to hit each other. When a player is hit, its life points go down. The aim is to have more life points than the opponent at the end of the game. The game ends after 25 steps or when one of the players has zero life points.
- *Pawn Whopping* is similar to *Breakthrough*, but with slightly different movement and is simultaneous.

Fighter, *Clobber*, and *Pawn Whopping* were used during the *German Open in GGP* of 2011 [36].

B. Setup

In all experiments, two variants of CadiaPlayer are matched against each other. The startclock and the playclock are set to 60 and 30 s, respectively. In the final experiment, however, we additionally look at the effects of using shorter time controls, where both the startclock and the playclock are set to 10 s. The k parameter of the NST simulation strategy is set to 7. This value was determined by manual tuning. The τ parameter of the Gibbs measure used in CadiaPlayer was left unchanged to the preset value of 10.

In all experiments, the programs switch roles such that none has any advantage. For the two-player games, there are two possible configurations. For the three-player games, there are eight possible configurations, where two of them consist of three times the same player. Therefore, there are only six interesting configurations which are employed in the experiments [25].

In the first experiment, we investigate the effects of using an ϵ-greedy strategy [24] instead of Gibbs measure. The original CadiaPlayer using MAST with Gibbs measure ($\mathrm{CP}_{\mathrm{MAST}}$) plays against the version that uses MAST with an ϵ-greedy

TABLE I
WIN % OF CP_{MASTG} AGAINST CP_{MAST} FOR DIFFERENT VALUES OF ϵ

Game	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.3$	$\epsilon = 0.4$	$\epsilon = 0.5$	$\epsilon = 0.6$	$\epsilon = 0.7$
Connect5	28.1 ($\pm$4.68)	33.6 ($\pm$5.05)	49.7 ($\pm$5.55)	57.9 ($\pm$5.50)	59.6 ($\pm$5.48)	71.7 ($\pm$5.05)	72.7 ($\pm$4.99)
Checkers	74.4 ($\pm$4.53)	73.7 ($\pm$4.68)	69.8 ($\pm$5.09)	71.1 ($\pm$5.06)	65.6 ($\pm$5.33)	68.7 ($\pm$5.19)	62.7 ($\pm$5.42)
Breakthrough	46.8 ($\pm$5.69)	46.2 ($\pm$5.51)	44.6 ($\pm$5.48)	43.0 ($\pm$5.46)	43.5 ($\pm$5.60)	41.3 ($\pm$5.57)	33.1 ($\pm$5.11)
Othello	60.9 ($\pm$5.54)	64.4 ($\pm$5.28)	64.3 ($\pm$5.26)	64.6 ($\pm$5.28)	60.9 ($\pm$5.49)	58.1 ($\pm$5.57)	50.5 ($\pm$5.44)
Skirmish	48.2 ($\pm$5.22)	54.6 ($\pm$5.32)	61.6 ($\pm$5.41)	61.8 ($\pm$5.41)	59.2 ($\pm$5.51)	54.4 ($\pm$5.58)	56.7 ($\pm$5.57)

strategy (CP_{MASTG}). ϵ ranges from 0.1 to 0.7. The aim is to determine whether an ϵ-greedy selection mechanism performs better than Gibbs measure and for which ϵ the best overall performance is obtained. This ϵ will be used in subsequent experiments.

In the second experiment, we compare the performance of the NST and LGRF-2 strategies against that of CP_{MASTG}. In order to make a fair comparison, all the strategies are using an ϵ-greedy strategy (for LGRF-2 this applies to the fallback strategy only). The following three variants of CADIAPLAYER play against CP_{MASTG}. First, CP_{NST} which uses NST as a simulation strategy. Second, $\text{CP}_{\text{LGR}-\text{MASTG}}$ which uses LGRF-2 with CP_{MASTG} as a fallback strategy. Third, $\text{CP}_{\text{LGR}-\text{NST}}$ which uses LGRF-2 with CP_{NST} as a fallback strategy.

The third experiment is similar to the second experiment, but now the three variants as described above play against CP_{MAST} instead of CP_{MASTG}. This means that they play against the original CADIAPLAYER that is using MAST with Gibbs measure. With this experiment, the aim is to examine how much performance is gained (or lost) overall compared with the original CADIAPLAYER when applying the simulation strategies as described in this paper.

The fourth experiment resembles the third experiment, but now all players employ Gibbs measure. The intention of this experiment is to investigate how the new simulation strategies perform when they employ Gibbs measure instead of an ϵ-greedy strategy.

In the fifth experiment, $\text{CP}_{\text{LGR}-\text{NST}}$ plays against CP_{NST}. From the results of the previous experiments, which are shown in Section VI, it became clear that these two simulation strategies performed the best. Therefore, the intention of this experiment is to determine which one is superior.

In the final experiment, CP_{MAST}, CP_{NST}, $\text{CP}_{\text{LGR}-\text{MASTG}}$, and $\text{CP}_{\text{LGR}-\text{NST}}$ are matched against CP_{UCT} which plays uniformly random moves in the play-out phase. This experiment is similar to the one performed in [35] where, in addition, improved versions of CADIAPLAYER were matched against CP_{UCT}. Two different time settings are tested in order to investigate how sensitive the performance of the simulation strategies is to thinking time. In the first time setting, the startclock is set to 60 s and the playclock is set to 30 s. This is the same as in the other experiments. In the second time setting, the startclock and the playclock are both set to 10 s. This is the same time setting as used in the experiment performed in [35]. For CP_{NST}, $\text{CP}_{\text{LGR}-\text{MASTG}}$, and $\text{CP}_{\text{LGR}-\text{NST}}$, ϵ is set to 0.2. Furthermore, in this experiment, eight games are added in order to validate the chosen value of ϵ and in order to examine how the techniques perform in multiplayer games and simultaneous-move games.

VI. EXPERIMENTAL RESULTS

This section discusses the results of the experiments described above. All tables show both a win rate, averaged over at least $\approx$300 games, and a 95% confidence interval. The win rate is calculated as follows. For the two-player games, each game won gives a score of one point and each game that ends in a draw results in a score of half a point. The win rate is the sum of these points divided by the total number of games played. For the three-player games, a similar calculation is performed but the draws are counted differently. If all three players obtained the same reward, then the draw is counted as one third of a point. If two players obtained the same, highest reward, the draw is counted as half a point for the corresponding players.

A. ϵ-greedy Compared With Gibbs Measure

Table I shows the performance of CP_{MASTG} against CP_{MAST} on our benchmark games for different values of ϵ. We see that the ϵ value that gives the best performance differs from one game to the next. For instance, in *Connect5*, CP_{MASTG} performs better when ϵ, and thus exploration, increases. A likely explanation is the lack of moves that are good in general. In *Checkers*, for each of the tested values of ϵ, CP_{MASTG} performs much better than CP_{MAST}. Thus, in *Checkers*, CP_{MASTG} is not highly sensitive to a change in ϵ. In [35], it was demonstrated that in *Checkers* the MAST simulation strategy had only a little performance gain over a fully random simulation strategy. This explains why the introduction of more randomness in CP_{MASTG} has not much influence on its playing strength in *Checkers*. For the other three games, the optimal value of ϵ is around 0.4.

Overall, it was found that $\epsilon = 0.4$ value seems to be a good compromise, resulting in the ϵ-greedy selection player handily outperforming the Gibbs measure on all games except *Breakthrough*. So it seems that, at least for this set of games, an ϵ-greedy selection mechanism is better than Gibbs measure.

A possible explanation is that with Gibbs measure the probability that the move with the highest heuristic score is chosen is not fixed, because it depends on the scores for the other moves as well. Consequently, it is not assured that the move with the highest heuristic score will even be chosen at all. In contrast, with ϵ-greedy, the probability of selecting the move with the highest score is fixed to be 1-ϵ.

Furthermore, with Gibbs measure, the move selection is always biased by the heuristic scores for the moves. A heuristic score gives an estimate of the quality of a move. Consequently, when these heuristic scores do not correspond with the real quality of a move, then it could be that good moves are missed. This is not a problem with the ϵ-greedy strategy because with a probability of ϵ a plain random move is played. Thus, it is less likely that good moves are missed.

TABLE II
WIN % OF CP_{NST} WITH DIFFERENT VALUES OF ϵ AGAINST CP_{MASTG} WITH $\epsilon = 0.4$

Game	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.4$
Connect5	75.7 (±4.86)	76.8 (±4.70)	76.0 (±4.83)
Checkers	49.9 (±4.39)	45.6 (±4.37)	46.5 (±5.64)
Breakthrough	73.7 (±4.98)	77.0 (±4.76)	67.3 (±5.31)
Othello	56.9 (±4.85)	49.8 (±5.66)	46.0 (±5.64)
Skirmish	55.0 (±4.88)	51.0 (±5.57)	51.6 (±5.62)

TABLE III
WIN % OF $\text{CP}_{\text{LGR-MASTG}}$ WITH DIFFERENT VALUES OF ϵ AGAINST CP_{MASTG} WITH $\epsilon = 0.4$

Game	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.4$
Connect5	63.0 (±5.46)	73.5 (±4.99)	76.5 (±4.80)
Checkers	55.3 (±4.87)	47.5 (±5.65)	41.0 (±5.57)
Breakthrough	70.0 (±5.19)	69.1 (±5.20)	69.3 (±5.22)
Othello	38.7 (±5.51)	43.3 (±5.61)	36.5 (±5.45)
Skirmish	44.3 (±4.87)	45.7 (±5.64)	50.8 (±5.62)

TABLE IV
WIN % OF $\text{CP}_{\text{LGR-NST}}$ WITH DIFFERENT VALUES OF ϵ AGAINST CP_{MASTG} WITH $\epsilon = 0.4$

Game	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.4$
Connect5	85.7 (±3.88)	84.2 (±4.13)	81.3 (±4.41)
Checkers	43.5 (±4.35)	41.4 (±4.33)	36.8 (±5.46)
Breakthrough	75.7 (±4.86)	79.7 (±4.55)	80.7 (±4.31)
Othello	42.7 (±5.60)	38.0 (±5.49)	31.3 (±5.25)
Skirmish	52.4 (±4.75)	53.8 (±4.37)	50.7 (±5.66)

TABLE V
WIN % OF CP_{NST} WITH DIFFERENT VALUES OF ϵ AGAINST CP_{MAST}

Game	$\epsilon = 0.1$	$\epsilon = 0.2$
Connect5	73.3 (±4.93)	77.2 (±4.75)
Checkers	66.5 (±5.34)	71.5 (±5.07)
Breakthrough	69.7 (±5.20)	71.3 (±5.12)
Othello	71.0 (±5.13)	73.0 (±5.02)
Skirmish	70.7 (±4.63)	74.7 (±4.44)

B. CP_{MASTG} Compared With CP_{NST} and CP_{LGR}

In this experiment, CP_{MASTG} always uses $\epsilon = 0.4$, because in the previous section it is demonstrated that this leads to the best overall performance. For CP_{NST}, $\text{CP}_{\text{LGR-MASTG}}$, and $\text{CP}_{\text{LGR-NST}}$ three different values for ϵ are used, namely 0.1, 0.2, and 0.4.

Table II shows the results of player CP_{NST} against CP_{MASTG}. For each tested value of ϵ the CP_{NST} player is significantly better in *Connect5* and *Breakthrough*. The best overall performance of CP_{NST} is obtained at $\epsilon = 0.1$. There CP_{NST} demonstrates statistically significant improvement in all games but *Checkers*, where performance is approximately on par. Therefore, based on these results, CP_{NST} is overall clearly a superior player to CP_{MASTG}.

Table III shows the win rate of $\text{CP}_{\text{LGR-MASTG}}$ against CP_{MASTG}. $\text{CP}_{\text{LGR-MASTG}}$ is significantly better in *Connect5* and *Breakthrough*. For the other three games, $\text{CP}_{\text{LGR-MASTG}}$ does not improve upon CP_{MASTG}. Especially in *Othello*, at $\epsilon = 0.4$ and $\epsilon = 0.1$, the performance of $\text{CP}_{\text{LGR-MASTG}}$ is notably worse than that of CP_{MASTG}. A possible explanation is that in *Othello* there are certain moves that are good in general, for example placing stones in the corners because they cannot be captured. This makes *Othello* particularly suited for the MAST simulation strategy. However, LGRF-2 is less appropriate for *Othello*, because when the player has placed a stone in the corner and subsequently lost the game, then this move is removed from the best reply table. Consequently, in the next simulation, it is less likely that the player will place a stone in the corner although it would be a good move.

Finally, Table IV shows the win rate of $\text{CP}_{\text{LGR-NST}}$ against CP_{MASTG}. When this result is contrasted to Table II, we see that the performance of $\text{CP}_{\text{LGR-NST}}$ and CP_{NST} is overall similar, with neither dominating the other. For example, $\text{CP}_{\text{LGR-NST}}$ is better in *Connect5* whereas CP_{NST} is better in *Othello*. In a later experiment (Section VI-E), we match the two players against each other to further investigate their relative strength.

To summarize, two main conclusions can be drawn from these experiments. First, the performance of the simulation strategy depends heavily on the type of game. Second, the NST simulation strategy and the LGRF-2 combined with NST seem to give the best overall performance.

C. CP_{MAST} Compared With CP_{NST}, $\text{CP}_{\text{LGR-MASTG}}$, and $\text{CP}_{\text{LGR-NST}}$

In this experiment, CP_{NST}, $\text{CP}_{\text{LGR-MASTG}}$, and $\text{CP}_{\text{LGR-NST}}$ play against CP_{MAST}, the original CADIAPLAYER equipped with MAST and Gibbs measure. The results, which are shown in Tables V–VII, are in line with the results of the previous experiments. As in the previous experiments, the overall performance of $\text{CP}_{\text{LGR-NST}}$ and CP_{NST} seems to be better than the overall performance of $\text{CP}_{\text{LGR-MASTG}}$. This experiment clearly shows that, at least on this set of five games, CP_{NST} and $\text{CP}_{\text{LGR-NST}}$ improve the playing strength of the original CADIAPLAYER significantly. Especially CP_{NST} seems to be a robust simulation strategy as it is consistently better than the original CADIAPLAYER. Furthermore, $\epsilon = 0.2$ gives the overall best performance, because in this case for each of the five games, all simulation strategies have a win rate above 50% against CP_{MAST}.

D. CP_{MAST} Compared With $\text{CP}_{\text{NST-Gibbs}}$, $\text{CP}_{\text{LGR-MAST-Gibbs}}$, and $\text{CP}_{\text{LGR-NST-Gibbs}}$

The aim of the following experiment is to investigate how the NST and LGR techniques perform in combination with Gibbs measure instead of ϵ-greedy. Therefore, $\text{CP}_{\text{NST-Gibbs}}$, $\text{CP}_{\text{LGR-MAST-Gibbs}}$, and $\text{CP}_{\text{LGR-NST-Gibbs}}$ are matched against CP_{MAST}. The results are shown in Table VIII. By comparing the results with the results in Tables V–VII it is clear that, like for MAST, the ϵ-greedy exploration technique gives an overall better performance than Gibbs measure. An explanation for this result was already given in Section VI-A.

E. CP_{NST} Compared With $\text{CP}_{\text{LGR-NST}}$

From the previous experiments it is clear that CP_{NST} and $\text{CP}_{\text{LGR-NST}}$ have the best overall performance. Therefore, in this experiment, these two players are matched against each other. The win rate of CP_{NST} is shown in Table IX. It clearly

TABLE VI
WIN % OF $\text{CP}_{\text{LGR}-\text{MASTG}}$ WITH DIFFERENT VALUES OF ϵ AGAINST CP_{MAST}

Game	$\epsilon = 0.1$	$\epsilon = 0.2$
Connect5	61.8 (±5.50)	75.3 (±4.88)
Checkers	76.4 (±4.71)	70.9 (±5.07)
Breakthrough	66.0 (±5.36)	62.3 (±5.48)
Othello	55.9 (±4.59)	54.5 (±4.88)
Skirmish	50.3 (±5.66)	58.3 (±5.46)

TABLE VII
WIN % OF $\text{CP}_{\text{LGR}-\text{NST}}$ WITH DIFFERENT VALUES OF ϵ AGAINST CP_{MAST}

Game	$\epsilon = 0.1$	$\epsilon = 0.2$
Connect5	81.2 (±4.42)	82.7 (±4.28)
Checkers	71.2 (±5.13)	62.0 (±5.45)
Breakthrough	69.7 (±5.20)	71.8 (±5.05)
Othello	47.7 (±5.65)	53.8 (±5.15)
Skirmish	66.5 (±5.34)	73.3 (±5.00)

TABLE VIII
WIN % AGAINST CP_{MAST}

Game	$\text{CP}_{\text{NST-Gibbs}}$	$\text{CP}_{\text{LGR-MAST-Gibbs}}$	$\text{CP}_{\text{LGR-NST-Gibbs}}$
Connect5	55.5 (±5.53)	66.3 (±5.26)	62.6 (±5.39)
Checkers	43.0 (±5.59)	38.1 (±5.41)	41.7 (±5.55)
Breakthrough	63.9 (±5.35)	53.5 (±5.55)	61.9 (±5.41)
Othello	50.5 (±5.57)	41.8 (±5.49)	35.0 (±5.31)
Skirmish	50.8 (±5.57)	57.1 (±5.51)	53.7 (±5.55)

TABLE IX
WIN % OF CP_{NST} AGAINST $\text{CP}_{\text{LGR}-\text{NST}}$ FOR DIFFERENT VALUES OF ϵ

Game	$\epsilon = 0.1$	$\epsilon = 0.2$
Connect5	40.2 (±5.55)	43.3 (±4.86)
Checkers	55.5 (±5.62)	58.3 (±5.35)
Breakthrough	40.3 (±5.55)	36.7 (±5.45)
Othello	71.0 (±5.13)	66.8 (±5.33)
Skirmish	51.1 (±4.78)	53.1 (±4.77)

shows that the performance of the simulation strategies depends on the type of game. $\text{CP}_{\text{LGR}-\text{NST}}$ appears to be better in *Connect5* and *Breakthrough* while CP_{NST} is clearly better in *Othello*. The large performance difference in *Othello* again suggests that a strategy like LGRF-2 is not appropriate for *Othello*. Why LGRF-2 seems not appropriate for a game like *Othello* has already been discussed in Section VI-B. Thus, from these experiments, we see that each simulation strategy has its strengths and weaknesses, with neither dominating the other.

F. Comparison With CP_{UCT}

In the final experiment, CP_{MAST}, CP_{NST}, $\text{CP}_{\text{LGR}-\text{MASTG}}$, and $\text{CP}_{\text{LGR}-\text{NST}}$ play against CP_{UCT}. For CP_{NST}, $\text{CP}_{\text{LGR}-\text{MASTG}}$, and $\text{CP}_{\text{LGR}-\text{NST}}$, ϵ is set to 0.2, because this value obtained the best overall performance against CP_{MAST} as shown in Section VI-C. Our test suite is now extended with eight more games. The purpose of this is twofold. First, it allows us to evaluate the performance of the techniques on a wider range of games, including multiplayer and simultaneous-move ones. Second, it validates whether the chosen value for the ϵ parameter performs equally well in games that were not used for tuning the parameter. Table X shows the results when using the same time setting as in the previous experiments.

TABLE X
WIN % AGAINST CP_{UCT}, STARTCLOCK = 60 s, PLAYCLOCK = 30 s

Game	CP_{MAST}	CP_{NST}	$\text{CP}_{\text{LGR-MASTG}}$	$\text{CP}_{\text{LGR-NST}}$
Connect5	66.5 (±5.34)	85.8 (±3.95)	84.0 (±4.15)	**91.3** (±3.18)
Checkers	56.7 (±5.61)	71.0 (±5.11)	**76.9** (±4.69)	69.6 (±5.18)
Breakthrough	86.0 (±3.93)	96.4 (±2.10)	88.0 (±3.68)	**97.3** (±1.82)
Othello	70.1 (±5.15)	**82.8** (±4.27)	70.0 (±5.19)	73.6 (±4.93)
Skirmish	45.0 (±5.63)	**71.5** (±5.11)	56.2 (±5.61)	69.2 (±5.22)
Clobber	52.0 (±5.65)	54.3 (±5.64)	**55.6** (±5.21)	48.7 (±5.64)
Sheep and Wolf	51.3 (±5.66)	65.3 (±5.39)	55.3 (±5.63)	**71.0** (±5.13)
TTCC4	60.6 (±5.05)	**66.0** (±4.89)	44.7 (±5.14)	45.7 (±5.15)
Chinese Checkers	65.1 (±4.85)	**73.0** (±4.22)	59.7 (±4.98)	59.4 (±4.99)
Chinook	79.3 (±4.56)	**90.0** (±3.39)	84.2 (±4.13)	88.7 (±3.59)
Runners	**54.2** (±5.64)	31.3 (±5.25)	37.5 (±5.48)	28.7 (±5.12)
Fighter	49.3 (±5.66)	51.9 (±5.46)	51.5 (±5.66)	**52.8** (±5.48)
Pawn Whopping	**74.3** (±4.91)	74.0 (±4.93)	71.4 (±5.08)	73.2 (±4.93)

Like in the previous experiments, for the first five games, the overall performance of $\text{CP}_{\text{LGR}-\text{NST}}$ and CP_{NST} seems to be better than the overall performance of $\text{CP}_{\text{LGR}-\text{MASTG}}$. Furthermore, $\text{CP}_{\text{LGR}-\text{NST}}$ performs significantly worse than CP_{NST} in *Othello*. This was already observed in Sections VI-B and VI-E. Additionally, the results show that in the first five games CP_{NST}, $\text{CP}_{\text{LGR}-\text{MASTG}}$, and $\text{CP}_{\text{LGR}-\text{NST}}$ have in each game a higher win rate than CP_{MAST}. This is in line with the results shown in Tables V–VII, because for $\epsilon = 0.2$ the win rate of CP_{NST}, $\text{CP}_{\text{LGR}-\text{MASTG}}$, and $\text{CP}_{\text{LGR}-\text{NST}}$ against CP_{MAST} is always above 50%.

The results of the eight additional games indicate that CP_{NST} is a more robust technique than $\text{CP}_{\text{LGR}-\text{MASTG}}$ and $\text{CP}_{\text{LGR}-\text{NST}}$. The overall performance of CP_{NST} in these new games is at least as good as CP_{MAST}. Only in *Runners* CP_{NST} does not perform well. The reason is that in *Runners* there are a small number of legal moves to choose from which do not have intuitive countermoves. The inclusion of N-grams does not add anything, because there is not a direct relation between moves of one player and the other player.

Thus, for CP_{NST}, the chosen value of ϵ works in other games as well. $\text{CP}_{\text{LGR}-\text{MASTG}}$ is not really better than CP_{MAST} and $\text{CP}_{\text{LGR}-\text{NST}}$ leads to mixed results. Furthermore, CP_{NST} performs significantly better in *Chinese Checkers* and *Chinook* compared with CP_{MAST} indicating that the addition of N-grams can also increase the performance in multiplayer games and simultaneous-move games.

The results for *Fighter*, which is a nonboard game, seem to indicate that the N-grams are not beneficial for this game. However, the reason is that in *Fighter* between 70% and 90% of the games ended in a draw. When the drawn games are left out the win rates are 0.43 ($n = 30$), 0.58 ($n = 76$), 0.55 ($n = 85$), and 0.66 ($n = 56$) for CP_{MAST}, CP_{NST}, $\text{CP}_{\text{LGR}-\text{MASTG}}$, and $\text{CP}_{\text{LGR}-\text{NST}}$, respectively. Therefore, it shows that in particular $\text{CP}_{\text{LGR}-\text{NST}}$ seems to work better in *Fighter* than CP_{MAST}.

Table XI shows the results of the same experiment but with a time setting where startclock and playclock both equal 10 s. The performance of CP_{MAST} is not much affected by the shorter time setting. However, the other three simulation strategies show a drastic decrease in performance in *Skirmish* compared with the longer time setting. Furthermore, the win rate of CP_{NST} in *Othello* and the win rates of CP_{NST} and $\text{CP}_{\text{LGR}-\text{NST}}$ in *Sheep and Wolf* and *Chinook* are reduced considerably. Still, there are also improvements. $\text{CP}_{\text{LGR}-\text{MASTG}}$ increases its win rate in *Connect5* by approximately 10% and CP_{NST},

TABLE XI
WIN % AGAINST CP_{UCT}, STARTCLOCK = 10 s, PLAYCLOCK = 10 s

Game	CP_{MAST}	CP_{NST}	$CP_{LGR\text{-}MASTG}$	$CP_{LGR\text{-}NST}$
Connect5	66.2 (±5.35)	86.5 (±3.87)	**94.3** (±2.62)	90.0 (±3.39)
Checkers	56.2 (±5.54)	69.3 (±5.19)	**72.1** (±5.03)	63.0 (±5.46)
Breakthrough	86.3 (±3.89)	**93.3** (±2.82)	91.3 (±3.15)	92.4 (±2.97)
Othello	62.2 (±5.49)	**77.3** (±4.74)	68.0 (±5.28)	58.0 (±5.59)
Skirmish	**45.5** (±5.64)	40.4 (±5.53)	36.8 (±5.46)	36.0 (±5.43)
Clobber	49.0 (±5.66)	54.7 (±5.63)	53.0 (±5.65)	**60.3** (±5.54)
Sheep and Wolf	51.5 (±5.61)	59.3 (±5.56)	53.0 (±5.65)	**61.3** (±5.51)
TTCC4	58.1 (±5.10)	**59.3** (±4.99)	35.6 (±4.95)	37.8 (±4.93)
Chinese Checkers	58.3 (±5.58)	**71.2** (±4.72)	57.2 (±5.11)	51.9 (±5.54)
Chinook	74.2 (±4.87)	**82.7** (±3.91)	77.6 (±4.64)	78.9 (±4.54)
Runners	**50.2** (±5.51)	41.8 (±5.49)	49.8 (±5.57)	44.2 (±5.53)
Fighter	49.8 (±5.66)	**53.7** (±5.64)	52.2 (±5.65)	49.8 (±5.66)
Pawn Whopping	**85.9** (±3.92)	**85.9** (±3.92)	82.0 (±4.30)	85.0 (±4.01)

$CP_{LGR-MASTG}$, and $CP_{LGR-NST}$ increase their win rate in *Runners* substantially. Furthermore, all programs increase their win rate in *Pawn Whopping* by approximately 10%.

Overall, the performance of CP_{NST}, $CP_{LGR-MASTG}$, and $CP_{LGR-NST}$ appears to be better on the longer time setting compared with the shorter time setting. Moreover, the results indicate that CP_{NST}, $CP_{LGR-MASTG}$, and $CP_{LGR-NST}$ benefit much more from longer thinking times than CP_{MAST} does. This result implies that the new simulation strategies will benefit more from faster computer hardware, thus potentially resulting in even more relative gains in playing strength on future computers. The most probable reason is that CP_{MAST} has less overhead than the other three simulation strategies resulting in more simulations per second. For the 13 games used in this experiment CP_{MAST} has on average approximately 83 simulations per second, CP_{NST} has 67 simulations per second, $CP_{LGR-MASTG}$ has 66 simulations per second, and $CP_{LGR-NST}$ has an average of approximately 53 simulations per second.

VII. CONCLUSION AND FUTURE WORK

In this paper, ϵ-greedy strategy was compared with the Gibbs measure as a selection mechanism for the CADIAPLAYER. Furthermore, a new simulation strategy based on N-grams, called NST, was proposed. The NST simulation strategy, compared with MAST, looks to a greater extent at the context in which a move is made. This is done in a natural and computationally economic way by keeping track of scores for move sequences instead of only scores for single moves. Additionally, the LGRP, the variant named LGRF-2, was tested in CADIAPLAYER.

Three important observations drawn from the experimental results are given below.

First, an ϵ-greedy selection mechanism appears to be better than Gibbs measure. For $\epsilon = 0.4$, the program employing ϵ-greedy performs significantly better in four of the five games. A possible explanation is that with Gibbs measure the probability that the move with the highest heuristic score is chosen is not fixed, because it depends on the scores for the other moves as well. Furthermore, with Gibbs measure, the move selection is always biased by the heuristic scores for the moves which may lead to missing a good move in case these heuristics are not entirely correct.

Second, the experiments revealed that LGRF-2 works not only in *Go*, but in other games as well. GGP is an ideal domain to test this because of the wide variety such programs can play. Furthermore, the experiments showed that the fallback strategy is also important, because the LGRF-2 employing NST as a fallback strategy performed much better than LGRF-2 using MAST as a fallback strategy.

Third, the NST simulation strategy based on N-grams is a robust strategy. On a set of five two-player, turn-based games it leads to a win rate of approximately 70% against the original CADIAPLAYER. Moreover, it is shown that the inclusion of N-grams is able to increase the performance in multiplayer games and simultaneous-move games. Furthermore, the new simulation strategies are likely to benefit more than the original strategy from faster future computer hardware.

To conclude, MAST may work well in games where there are consistently good moves, independent of the game state and independent of the moves played by the opponent. The NST simulation strategy may perform well in games where there is a strong relation between the moves of all the players and where certain move sequences are always clearly better than others, independent of the game state. LGRF-2 may function well in games where there is a strong relation between the moves of all the players, depending on the current game state. Due to the forgetting mechanism, it is able to respond quickly to a change in game state. The disadvantage is that LGRF-2 is less robust than NST. A single simulation immediately influences the choices for the next simulation.

The experiments showed that NST and LGRF-2 have promise in GGP. We expect that there is still room for further improvement. Therefore, three directions for future work are given below.

First, from all experiments it became clear that the performance of a specific simulation strategy depends on the type of game. Therefore, interesting future work is to develop techniques that can learn online which simulation strategy to employ.

Second, it became evident that there is no single optimum for the ϵ parameter in an ϵ-greedy selection mechanism. Hence, online parameter tuning is an important subject for future research.

Third, in order to make the NST simulation strategy even more general a method to detect the *noop* moves should be included. This is required for unusual games in which there is a nonstandard turn-taking mechanism. For instance, the game *Pentago* is turn taking, but players get two turns in a row.

REFERENCES

[1] J. Clune, "Heuristic evaluation functions for general game playing," in *Proc. 22nd AAAI Conf. Artif. Intell.*, Menlo Park, CA, 2007, pp. 1134–1139.
[2] S. Schiffel and M. Thielscher, "Fluxplayer: A successful general game player," in *Proc. 22nd AAAI Conf. Artif. Intell.*, Menlo Park, CA, 2007, pp. 1191–1196.
[3] S. Schiffel and M. Thielscher, "Automatic construction of a heuristic search function for general game playing," presented at the 7th IJCAI Int. Workshop Nonmontonic Reason. Action Change, Hyderabad, India, Jan. 7–8, 2007.
[4] Y. Björnsson and H. Finnsson, "Cadiaplayer: A simulation-based general game player," *IEEE Trans. Comput. Intell. AI Games*, vol. 1, no. 1, pp. 4–15, Mar. 2009.
[5] J. Méhat and T. Cazenave, "Ary, a general game playing program," presented at the 13th Board Games Studies Colloq., Paris, France, 2010.
[6] J. Méhat and T. Cazenave, "Combining UCT and nested Monte Carlo search for single-player general game playing," *IEEE Trans. Comput. Intell. AI Games*, vol. 2, no. 4, pp. 271–277, Dec. 2010.
[7] S. Schreiber, "General game playing base package," [Online]. Available: http://code.google.com/p/ggp-base/

[8] R. Coulom, "Efficient selectivity and backup operators in Monte-Carlo tree search," in *CG 2006*, ser. Lecture Notes in Computer Science, H. J. van den Herik, P. Ciancarini, and H. H. L. M. Donkers, Eds. Berlin, Germany: Springer-Verlag, 2007, vol. 4630, pp. 72–83.
[9] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," in *Proceedings of the EMCL 2006*, ser. Lecture Notes in Computer Science, J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds. Berlin, Germany: Springer-Verlag, 2006, vol. 4212, pp. 282–293.
[10] G. M. J.-B. Chaslot, M. H. M. Winands, J. W. H. M. Uiterwijk, H. J. van den Herik, and B. Bouzy, "Progressive strategies for Monte-Carlo tree search," *New Math. Natural Comput.*, vol. 4, no. 3, pp. 343–357, 2008.
[11] S. Gelly and D. Silver, "Combining online and offline knowledge in UCT," in *Proc. 24th Int. Conf. Mach. Learn.*, Z. Ghahramani, Ed., 2007, pp. 273–280.
[12] P. Drake, "The last-good-reply policy for Monte-Carlo Go," *Int. Comput. Games Assoc. J.*, vol. 32, no. 4, pp. 221–227, 2009.
[13] H. Baier and P. Drake, "The power of forgetting: Improving the last-good-reply policy in Monte Carlo Go," *IEEE Trans. Comput. Intell. AI Games*, vol. 2, no. 4, pp. 303–309, Dec. 2010.
[14] N. Love, T. Hinrichs, D. Haley, E. Schkufza, and M. Genesereth, "General game playing: Game description language specification," Stanford Univ., Stanford, CA, Tech. Rep., 2008.
[15] M. R. Genesereth and R. E. Fikes, "Knowledge interchange format, version 3.0 reference manual," Stanford Univ., Stanford, CA, Logic Group Tech. Rep. Logic-92–1, 1992.
[16] M. Thielscher, "A general game description language for incomplete information games," in *Proc. 24th AAAI Conf. Artif. Intell.*, Menlo Park, CA, 2010, pp. 994–999.
[17] The Dresden GGP Server [Online]. Available: http://130.208.241.192/ggpserver/
[18] S. Schiffel, "GameController" [Online]. Available: http://www.general-game-playing.de/downloads.html
[19] T. Yajima, T. Hashimoto, T. Matsui, J. Hashimoto, and K. Spoerer, "Node-expansion operators for the UCT algorithm," in *CG 2010*, ser. Lecture Notes in Computer Science, H. J. van den Herik, H. Iida, and A. Plaat, Eds. Berlin, Germany: Springer-Verlag, 2011, vol. 6515, pp. 116–123.
[20] H. Finnsson, "CADIA-Player: A general game playing agent," M.S. thesis, Schl. Comput. Sci., Reykjavík Univ., Reykjavík, Iceland, 2007.
[21] H. Finnsson and Y. Björnsson, "Simulation-based approach to general game playing," in *Proc. 23rd AAAI Conf. Artif. Intell.*, Menlo Park, CA, 2008, pp. 259–264.
[22] J. Schaeffer, "The history heuristic and alpha-beta search enhancements in practice," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 11, pp. 1203–1212, Nov. 1989.
[23] D. E. Knuth and R. W. Moore, "An analysis of alpha-beta pruning," *Artif. Intell.*, vol. 6, no. 4, pp. 293–326, 1975.
[24] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, ser. Adaptive Computation and Machine Learning. Cambridge, MA: MIT Press, 1998.
[25] N. R. Sturtevant, "An analysis of UCT in multi-player games," *Int. Comput. Games Assoc. J.*, vol. 31, no. 4, pp. 195–208, 2008.
[26] H. Finnsson and Y. Björnsson, "Simulation control in general game playing agents," in *Proc. IJCAI Workshop General Game Playing*, Jul. 2009, pp. 21–26.
[27] H. Finnsson and Y. Björnsson, "Learning simulation control in general game-playing agents," in *Proc. 24th AAAI Conf. Artif. Intell.*, Menlo Park, CA, 2010, pp. 954–959.
[28] J. A. Stankiewicz, "Knowledge-based Monte-Carlo tree search in Havannah," M.S. thesis, Dept. Knowledge Eng., Maastricht Univ., Maastricht, The Netherlands, 2011.
[29] J. A. Stankiewicz, M. H. M. Winands, and J. W. H. M. Uiterwijk, "Monte-Carlo tree search enhancements for Havannah," in *Proceedings of Advances in Computer Games 13*, ser. Lecture Notes in Computer Science, H. J. van den Herik and A. Plaat, Eds. Berlin, Germany: Springer-Verlag, Jul. 2012, vol. 7168, pp. 60–71.
[30] A. Rimmel and F. Teytaud, "Multiple overlapping tiles for contextual Monte Carlo tree search," in *Applications of Evolutionary Computation*, ser. Lecture Notes in Computer Science, C. Di Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekárt, A. Esparcia-Alcazar, C. Goh, J. Merelo, F. Neri, M. Preuß, J. Togelius, and G. Yannakakis, Eds. Berlin, Germany: Springer-Verlag, 2010, vol. 6024, pp. 201–210.
[31] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press, 1999.
[32] I. Millington, *Artificial Intelligence for Games*, 1st ed. San Mateo, CA: Morgan Kaufmann, 2006, ch. 7, pp. 580–591.
[33] F. D. Laramée, "Using N-gram statistical models to predict player behavior," in *AI Programming Wisdom*, S. Rabin, Ed. Newton Center, MA: Charles River Media, 2002, vol. 1, ch. 11, pp. 596–601.
[34] J. W. H. M. Uiterwijk, "The countermove heuristic," *Int. Comput. Chess Assoc. J.*, vol. 15, no. 1, pp. 8–15, 1992.
[35] H. Finnsson and Y. Björnsson, "Cadiaplayer: Search control techniques," *KI J.*, vol. 25, no. 1, pp. 9–16, 2011.
[36] P. Kissmann and T. Federholzner, "German Open in GGP," 2011 [Online]. Available: http://www.tzi.de/~kissmann/ggp/go-ggp/

Mandy J. W. Tak received the M.Sc. degree in operations research from Maastricht University, Maastricht, The Netherlands, in January 2012, where she is currently working toward the Ph.D. degree at the Department of Knowledge Engineering. Her Ph.D. research concerns online learning of search control in general game playing.

Mark H. M. Winands received the Ph.D. degree in artificial intelligence from the Department of Computer Science, Maastricht University, Maastricht, The Netherlands, in 2004.

Currently, he is an Assistant Professor at the Department of Knowledge Engineering, Maastricht University. His research interests include heuristic search, machine learning, and games.

Dr. Winands serves as a Section Editor of the *International Computer Games Association Journal* and as an Associate Editor of the IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES.

Yngvi Björnsson received the Ph.D. degree in computer science from the Department of Computing Science, University of Alberta, Edmonton, AB, Canada, in 2002.

He is an Associate Professor at the School of Computer Science, Reykjavík University, Reykjavík, Iceland, and the Director (and co-founder) of the CADIA research lab. His research interests are in heuristic search methods and search-control learning, and the application of such techniques for solving large-scale problems in a wide range of problem domains, including computer games and industrial process optimization.