Decaying Simulation Strategies

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

***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 investigate the application of a decay factor for two domain-independent simulation strategies: the N-gram selection technique (NST) and the move-average sampling technique (MAST). Three decay factor methods, called move decay, batch decay, and simulation decay, are applied. Furthermore, a combination of move decay and simulation decay is also tested. The decay variants are implemented in the GGP program CADIAPLAYER. Four types of games are used: turn taking, simultaneous move, one player, and multiplayer. Except for one-player games, experiments show that decaying can significantly improve the performance of both NST and MAST simulation strategies.**

***Index Terms*—Decay, general game playing (GGP), Monte Carlo tree search (MCTS), N-grams.**

I. Introduction

PAST research in artificial intelligence (AI) has focused on developing programs that can play one game at a high level. These programs generally rely on human-expert knowledge embedded into the programs by the software developers. In general game playing (GGP), the aim is to create programs that can learn to play a wide variety of games at an expert level. As there is no human intervention allowed, one of the main challenges in GGP is to construct programs capable of discovering and applying relevant game knowledge during play. Furthermore, it is no longer possible to determine beforehand which search techniques and enhancements are best suited for the game at hand. To address these challenges, most successful GGP programs incorporate a wide range of AI techniques, such as knowledge representation, knowledge discovery, machine learning, heuristic search, and online optimization.

The first successful GGP programs, such as CLUNEPLAYER [1] and FLUXPLAYER [2], [3], were based on minimax search with an automatically learned evaluation function. CLUNEPLAYER and FLUXPLAYER won the International GGP competition in 2005 and 2006, respectively. However, ever since, GGP programs incorporating Monte Carlo tree search (MCTS)-based approaches have proved more successful in the competition. In 2007, 2008, and 2012, CADIAPLAYER [4], [5] won; in 2009 and 2010, ARY won [6]; and in 2011 and 2013, TURBO TURTLE, developed by Sam Schreiber, was the winner. All three programs are based on MCTS, an approach particularly well suited for GGP because no game-specific knowledge is required besides the basic rules of the game.

The performance of MCTS depends heavily on the simulation strategy employed in the playout phase [7]. As there is no game-dependent knowledge available in GGP, generic simulation strategies need to be developed. Tak *et al.* [8] proposed a simulation strategy based on N-grams, called the N-gram selection technique (NST). The new NST strategy was shown to outperform, on average, the more established move-average sampling technique (MAST) [9], which was employed by CADIAPLAYER when winning the 2008 International GGP competition.

The information gathered by NST and MAST is kept between successive searches. On the one hand, this reuse of information may bolster the simulation strategy as it is immediately known what the strong moves are in the playout. On the other hand, this information can become outdated as moves that are strong in one phase of the game become weak in a later phase. In this paper, we investigate the application of a decay factor for NST and MAST statistics. The idea of decaying statistics was already applied in the discounted upper confidence bounds applied to trees (UCT) algorithm [10]. In that study, decaying proved of limited use, mainly because the UCT statistics were associated with single game positions that do not get outdated (in turn-taking deterministic perfect-information games). However, schemes such as NST and MAST, which generalize statistics across a large set of game positions, may benefit from decaying as the quality of the generalization may change over time with the game situation.

This paper is structured as follows. First, Section II gives the necessary background information about MCTS. Next, the simulation strategies NST and MAST are explained in Section III. The different decay factor methods are discussed in Section IV. Subsequently, Sections V and VI deal with the experimental setup and results. Finally, Section VII gives conclusions and an outlook to future research.

II. Monte Carlo Tree Search

CADIAPLAYER [4], [5] uses MCTS [11], [12] 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 technique that gradually builds up a tree in memory.

Manuscript received July 01, 2013; revised December 17, 2013; accepted February 11, 2014. Date of publication March 11, 2014; date of current version December 11, 2014. This work was supported by The Netherlands Organisation for Scientific Research (NWO) in the framework of the GoGeneral Project 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 6211 LK, The Netherlands (e-mail: mandy.tak@maastrichtuniversity.nl; m.winands@maastrichtuniversity.nl).

Y. Björnsson is with the School of Computer Science, Reykjavík University, Reykjavík 103, Iceland (e-mail: yngvi@ru.is).

Digital Object Identifier 10.1109/TCIAIG.2014.2310782

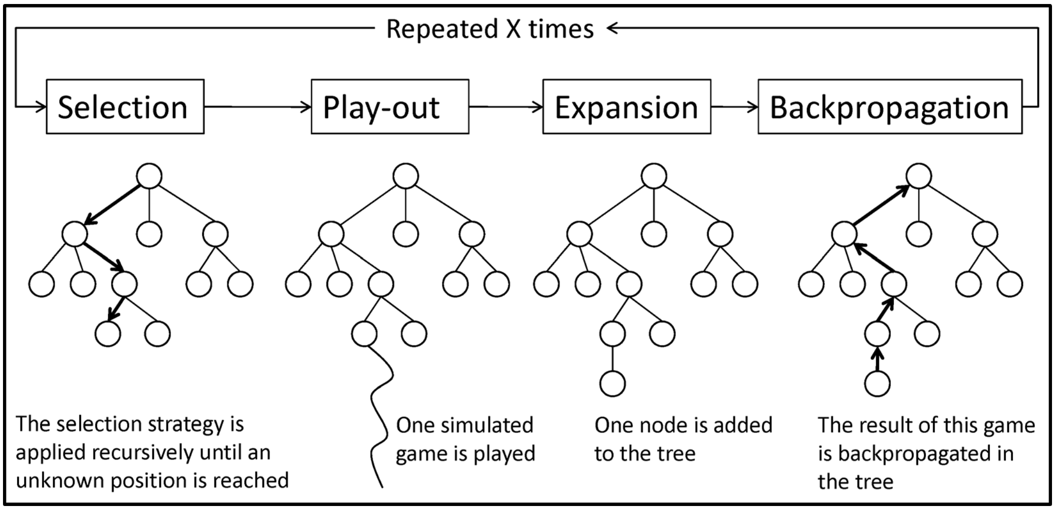

Fig. 1. Four strategic steps in MCTS.

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 [13], 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 playout 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 playout [7].
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 UCT algorithm [11], which employs the UCB1 [14] formula, is applied to determine which moves to select in the tree. At each node s move a^* selected is given by

$$a^* \leftarrow \underset{a \in A(s)}{\operatorname{argmax}} \left\{ Q(s,a) + C\sqrt{\frac{\ln N(s)}{N(s,a)}} \right\} \quad (1)$$

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 playout step are used to determine which move to select. In all other cases, (1) is applied.
2) During the playout step, a complete game is simulated. The most basic approach is to play uniformly random moves. However, the playouts can be improved significantly by playing nonuniform random moves biased by a simulation strategy [7]. The simulation strategies used in this paper are described in Section III.
3) In the expansion step, nodes are added to the tree. In CADIAPLAYER, only one node per simulation is added [12]. 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.
4) In the backpropagation step, the reward obtained in the playout is propagated backward 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 reward that was just obtained. In GGP, the reward lies in the range $[0, 100]$.

More details about the implementation of CADIAPLAYER can be found in [5].

III. SIMULATION STRATEGIES

This section explains the simulation strategies employed in the experiments. Section III-A explains MAST used by CADIAPLAYER when it won the Association for the Advancement of Artificial Intelligence (AAAI) 2008 GGP competition. Section III-B explains NST.

A. Move-Average Sampling Technique

MAST [5], [9] is based on the principle that moves good in one state are likely to be good in other states as well. The history heuristic [15], which is used to order moves in $\alpha\beta$ search [16], is based on the same principle. For each move a, a global average $Q_h(a)$ is kept in memory, which is the average of the returned rewards of the playouts in which move a occurred. These values are used to bias the selection of moves, primarily in the playout phase, but also for tie breaking of unexplored moves in the selection phase. The moves are selected using a softmax-based Gibbs measure [17]

$$P(s,a) = \frac{e^{Q_h(a)/\tau}}{\sum_{b \in A(s)} e^{Q_h(b)/\tau}} \tag{2}$$

where $P(s,a)$ is the probability that move a will be selected in state or node s. 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 encourage exploration of nonvisited moves, the initial $Q_h(a)$ value is set to the maximum possible score of 100.

B. N-gram Selection Technique

NST was introduced by Tak *et al.* [8]. NST keeps track of move sequences as opposed to single moves as in MAST. Tak *et al.* [8] showed that NST often outperforms MAST in GGP.

A method similar to NST has been applied successfully in *Havannah* [18], [19] and *Tron* [20]. Another method similar to NST is called N-gram-average-sampling technique (NAST), which is applied in *Dou Di Zhu*, *Hearts*, and *Lord of the Rings: The Confrontation* [21]. Furthermore, NST also bears some resemblance to the simulation strategy introduced by Rimmel and Teytaud [22], which is based on a tiling of the space of Monte Carlo simulations.

NST is based on N-gram models, which were invented by Shannon [23]. An N-gram model is a statistical model to predict the next word based on the previous $N-1$ words. N-grams are often employed in statistical language processing [24]. N-grams have also been applied in various research on computer games, including predicting the next move of the opponent [25], [26], extracting opening moves [27], ordering moves [28], [29], and detecting forced moves [30].

The N-grams in NST consist of consecutive move sequences z of length 1, 2, and 3. Similar to MAST, the average of the returned rewards of the playouts is accumulated. However, the average reward for a sequence z, here called $R(z)$, is also kept for longer move sequences, as opposed to only single moves.

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. For each player, the extracted move sequences are stored separately.

The move sequences consist of moves from both the current player and 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.

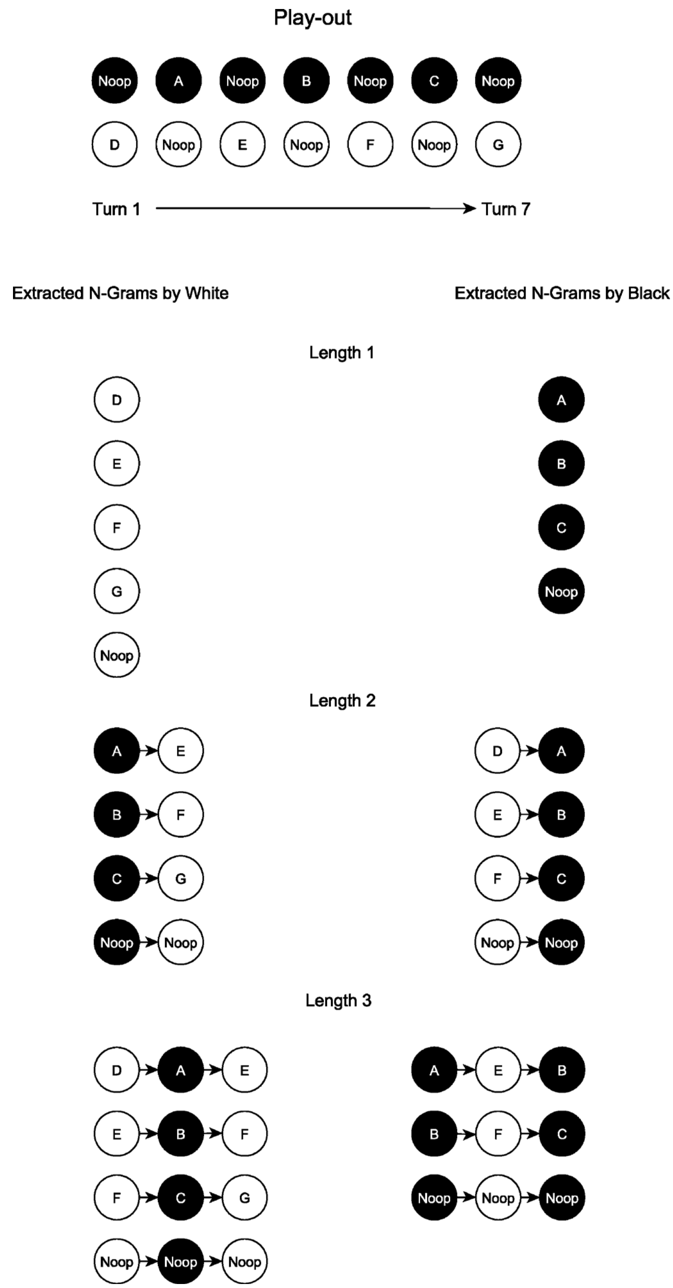

Fig. 2. Extracted N-grams from playout.

Fig. 2 gives an example of a playout. At each step, both players have to choose a move, because all games in GGP are by default simultaneous-move games. The example given here concerns a turn-taking, 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 influence the decision process during the playout.

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-taking 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 at each step. Another possibility would have been to include the moves of all players at each step, but that would result in 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 playout, 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 $T(a)$ for a move by taking the unweighted average of the $R(z)$ values stored for these sequences. In this calculation, the $R(z)$ values for the move sequences of length 2 and length 3 are only taken into account if they are visited at least k times.

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 $R(z)$ value of the move sequence of length 1 (which is the move itself) will be returned. The k parameter thus prevents move sequences with only a few visits from being considered.

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 to bias the selection toward unexplored moves.

In this manner, a score $T(a)$ is assigned to each legal move a in a given state. These scores are then used with ϵ-greedy [31], [32] to determine which move to select. With a probability of $1-\epsilon$, the move with the highest $T(a)$ value is selected, otherwise a legal move is chosen uniformly at random.

IV. Decay Factor

The information gathered by NST and MAST is kept between successive searches. On the one hand, this reuse of information may bolster the simulation strategy as it is immediately known what the strong moves are in the playout. This is especially important in GGP as the number of simulations to gather information is quite low. On the other hand, this information can become outdated as moves that are strong in one phase of the game are weak in another phase. Moreover, statistics can be mostly gathered for a particular part of the search tree that subsequently is not reached as the opponent moves differently from what was anticipated. Therefore, we propose to introduce a decay factor. Applying a decay factor can be done in different ways. For NST in particular, we investigate the following three methods.

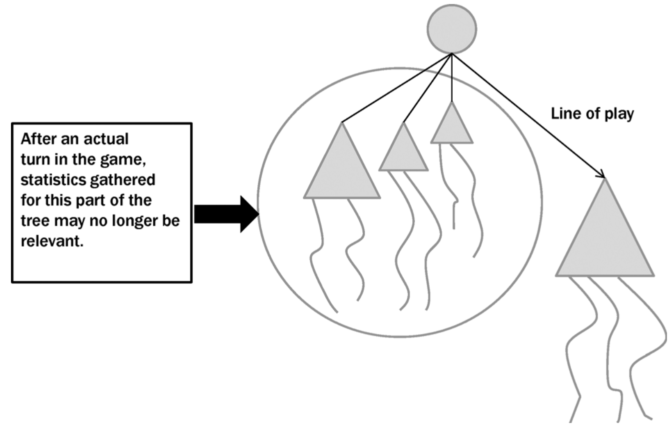

Fig. 3. Why a decay factor can be useful.

For the first two decay methods, all move sequences are multiplied with a decay factor $\gamma \in [0,1]$. In the move decay method, the decay takes place after an actual move is made in the game. The batch decay method takes place after a fixed number of simulations. Both methods can be represented by

$$\forall z \in Z, \qquad V(z) \leftarrow \gamma \cdot V(z). \tag{3}$$

In this equation, Z is a set containing all stored N-grams, z represents an N-gram, and $V(z)$ represents the visit count of N-gram z.

In the third method, called simulation decay, the decay factor ω is applied after each simulation. The decay is only applied to the N-grams that were played in the simulation. If an N-gram occurred multiple times in a simulation, that N-gram will be decayed multiple times. This method is shown below, in which $H \subseteq Z$ represents all the N-grams that occurred in the simulation.

for $i = 1$ to $|H|$ **do**
  $V(i) \leftarrow \omega \cdot V(i)$
end for

We suspect the first method to work best, because after an agent and the opponent(s) actually make their moves, the game state changes and the $R(z)$ values are probably no longer relevant in the new game state. Fig. 3 sketches this phenomenon. Simulations performed in the left part of the tree have updated the $R(z)$ values, while after the actual moves are played, this part of the tree is probably not used anymore. Therefore, the $R(z)$ values were updated based on simulations that do not reflect the actual progress of the game.

We remark that Stankiewicz [18] introduced the first method and showed that, for NST, a decay factor between 0 and 0.25 performs best in *Havannah*. A decay factor of 0 means that the results are reset before each move. NST with a decay factor of 0 resembles in many ways the last-good-reply policy (LGRP) [33], [34]. In LGRP, the most recent successful replies are stored

TABLE I
GAMES USED IN THE EXPERIMENTS

Game	Players	Simul-taneous	Constant-Sum
Sudoku_simple	1	n/a	n/a
StatespaceLarge	1	n/a	n/a
Queens	1	n/a	n/a
Pancakes88	1	n/a	n/a
MaxKnights	1	n/a	n/a
Frogs and Toads	1	n/a	n/a
Zhadu	2	No	Yes
GridGame2	2	No	No
3DTicTacToe	2	No	Yes
TTCC4	2	No	No
Connect5	2	No	Yes
Checkers	2	No	Yes
Breakthrough	2	No	Yes
Knightthrough	2	No	Yes
Othello	2	No	Yes
Skirmish	2	No	No
Merrills	2	No	Yes
Quad	2	No	Yes
Sheep and Wolf	2	No	Yes
Farmers	3	No	No
TTCC4 3P	3	No	No
Chinese Checkers 3P	3	No	No
Battle	2	Yes	No
Chinook	2	Yes	No
Runners	2	Yes	No
Pawn Whopping	2	Yes	Yes

in memory and a reply is removed from memory when it is no longer successful. Note that the three proposed decay methods can be equally well applied to MAST.

A somewhat different approach to decaying UCT values, called discounted UCT, was evaluated by Hashimoto *et al.* [10] in the games *Othello*, *Havannah*, and *Go*. However, the decaying method did not improve performance.

V. EXPERIMENTAL SETUP

The N-gram adjustments are implemented in CADIAPLAYER in order to investigate the effectiveness for GGP. This program is called $\mathrm{CP_{NST}}$. The program using MAST instead of NST is called $\mathrm{CP_{MAST}}$. In Section V-A, an overview is given on the games used in the experiments. In Section V-B, the setup of the experiments is described.

A. Games

The games and their characteristics are shown in Table I. For a brief description of the games, see the Appendix. These games were chosen because they are used in several previous CADIAPLAYER experiments [4], [5], [9], [35]–[38]. *Pawn Whopping* and *Frogs and Toads* were used during the German Open in GGP of 2011 [39]. Furthermore, this selection contains different types of games, namely, one-player games, two-player games, multiplayer games, constant-sum games, and general-sum games. All games can be found on the Dresden GGP Server [40].

B. Setup

In all experiments, two variants of CADIAPLAYER are matched against each other. For NST, ϵ is set to 0.2, because it turned out to work best in [8]. The k parameter is set to 7, because it then makes sure that the N-grams of length 2 and length 3 are not applied when they have been rarely visited. For determining an appropriate value for k, we experimented with different values of k using a smaller test suit where $k = 7$ edged out other settings. However, it seems as if the agent's performance is not that sensitive to the exact value of k (as long as it is not set unreasonably high). For example, our trials with $k \in \{0, 7, 14\}$ resulted in a typical performance difference within $\pm 4\%$ on individual games and a comparable overall average performance. We would thus not expect much different results, even if other (reasonable) values of k were to be chosen.

The τ parameter of the Gibbs measure used in CADIAPLAYER was left unchanged to its preset value of 10.

In GGP, the time setting is defined by a startclock and a playclock. The startclock is the time between the GGP programs receiving the rules and the game starting. The playclock is the time per move. In the experiments, two different time settings are used. Usually, a startclock of 60 s and a playclock of 30 s are employed, but in the experiments where $\mathrm{CP_{NST}}$ plays against $\mathrm{CP_{MAST}}$, the startclock is set to 70 s and the playclock is set to 40 s.

Different time settings are used, because on the one hand, we want to have a high number of simulations per move, but on the other hand, it takes much computation time.

In all experiments, the programs switch roles such that no one has any advantage. For the two-player games, there are two possible configurations. For the three-player games, there are eight possible configurations, but two of these involve the same player three times. Therefore, only six configurations are employed in the experiments [32]. All experiments, except the one-player experiments, are performed on a computer consisting of 64 AMD Opteron 6174 2.2-GHz cores, called GoGeneral. The one-player experiments are performed on a computer consisting of 48 AMD Opteron 6274 2.2-GHz cores, called go4nature01.

VI. EXPERIMENTAL RESULTS

In the experiments, it is examined how the different decay factor methods perform. In the first set of experiments, move decay is tested on $\mathrm{CP_{NST}}$ and $\mathrm{CP_{MAST}}$. After tuning the parameters, the best version of $\mathrm{CP_{MAST}}$ is matched against the best version of $\mathrm{CP_{NST}}$. Furthermore, move decay is also tested on one-player games. In all experiments that follow, only $\mathrm{CP_{NST}}$ is employed, because of computational constraints. In the second and third sets of experiments, batch decay and simulation decay are tested, respectively. In the last set of experiments, simulation decay is mixed with move decay.

The table of the one-player games shows the average score over at least 300 games with a 95% confidence interval. All other tables show the win rate averaged over at least 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, except that draws are counted differently. If all three players obtained the same reward, then the draw is counted as one third of a point.

TABLE II
WIN PERCENTAGE OF CP_{NST} USING MOVE DECAY WITH DIFFERENT VALUES OF γ AGAINST CP_{NST} WITHOUT DECAY, STARTCLOCK = 60 s, PLAYCLOCK = 30 s, ON GOGENERAL

Game	$\gamma=0$	$\gamma=0.2$	$\gamma=0.4$	$\gamma=0.6$	$\gamma=0.8$
Zhadu	26.6 (±3.75)	32.4 (±4.05)	36.5 (±3.80)	47.0 (±3.67)	47.4 (±4.97)
GridGame2	49.4 (±5.38)	49.9 (±3.43)	50.5 (±4.11)	49.8 (±3.43)	49.3 (±4.58)
3DTicTacToe	66.5 (±4.97)	69.0 (±3.53)	66.2 (±4.29)	61.8 (±4.82)	58.2 (±4.85)
TTCC4	27.5 (±4.75)	44.4 (±5.04)	47.9 (±4.51)	52.5 (±5.63)	51.7 (±4.33)
Connect5	61.1 (±4.72)	69.1 (±4.19)	65.7 (±4.02)	66.2 (±3.69)	59.4 (±4.99)
Checkers	45.6 (±4.76)	54.0 (±4.72)	63.3 (±4.41)	60.8 (±5.38)	62.6 (±5.46)
Breakthrough	37.3 (±5.22)	41.9 (±4.36)	45.5 (±4.25)	44.6 (±5.18)	53.6 (±5.62)
Knightthrough	46.4 (±5.62)	38.1 (±4.95)	43.6 (±4.64)	44.1 (±5.58)	54.6 (±5.60)
Othello	36.1 (±5.29)	44.4 (±4.17)	45.2 (±4.02)	49.1 (±5.46)	48.1 (±5.58)
Skirmish	51.0 (±5.28)	49.1 (±5.28)	53.2 (±4.85)	55.1 (±4.40)	52.2 (±4.48)
Merrills	58.3 (±4.32)	58.6 (±5.05)	58.3 (±3.89)	60.7 (±5.02)	55.6 (±5.22)
Quad	61.5 (±3.87)	68.7 (±3.63)	67.2 (±3.37)	65.3 (±3.11)	60.4 (±4.25)
Sheep and Wolf	44.3 (±4.11)	44.0 (±3.41)	47.2 (±4.08)	49.0 (±3.46)	52.2 ±5.47)
Farmers	44.9 (±4.11)	52.7 (±2.62)	53.0 (±3.13)	50.3 (±2.62)	50.3 (±4.10)
TTCC4 3P	53.2 (±5.65)	56.2 (±4.31)	54.6 (±3.96)	54.5 (±3.59)	56.2 (±5.61)
Chinese Checkers 3P	41.4 (±5.09)	50.7 (±4.24)	53.2 (±4.67)	51.3 (±4.29)	51.0 (±5.66)
Battle	56.6 (±5.32)	65.6 (±4.46)	63.8 (±4.15)	64.5 (±5.03)	59.3 (±5.15)
Chinook	45.7 (±5.30)	55.0 (±4.75)	57.3 (±4.36)	56.9 (±5.31)	54.4 (±5.36)
Runners	55.8 (±4.73)	53.4 (±4.66)	49.7 (±4.66)	49.5 (±4.67)	52.1 (±3.98)
Pawn Whopping	47.2 (±2.72)	50.2 (±2.72)	51.5 (±2.71)	50.0 (±2.71)	50.3 (±2.30)

If two players obtained the same, highest reward, the draw is counted as half the point for the corresponding players.

A. Move Decay

1) Move Decay in NST: Table II shows the win rate of CP_{NST} with decay versus CP_{NST} without decay. Note that no decay means that $\gamma = 1$. The results show that decay may improve the program. Furthermore, the results demonstrate that simply resetting the NST statistics at each move (which means $\gamma = 0$) can decrease the performance significantly in some games (i.e., *Zhadu*, *TTCC4*, *Breakthrough*, *Othello*, and *Chinese Checkers 3P*). The best results were obtained for $\gamma = 0.4$ and $\gamma = 0.6$. For picking the best value, there are two criteria of interest: the best overall average performance and robustness. For the latter, we used the metric: the number of games showing statistically significant improvement minus the number of games showing statistically significant deterioration. When overall performance and robustness do not agree on a best setting, some objectivity may be called for. However, in this case, this was unnecessary as the chosen settings were the best according to both metrics (for NST, $\gamma = 0.6$ was a clear winner on both metrics, but for MAST, there was a close call between $\gamma = 0.4$ and $\gamma = 0.6$, both having the same robustness but the former edging out on overall average performance, 57.6% versus 56.8%).

In order to validate the results, the CP_{NST} with $\gamma = 0.6$ is matched against CP_{MAST} with $\gamma = 1$. The reason for choosing $\gamma = 0.6$ for CP_{NST} rather than the seemingly equally performing $\gamma = 0.4$ is because that parameter setting seems to be slightly more robust, that is, it hardly ever performs worse against the nondecaying program. Furthermore, the average over all games is the highest for $\gamma = 0.6$, namely 54.1%. As a reference experiment, CP_{NST} with $\gamma = 1$ played against CP_{MAST} with $\gamma = 1$. The results of the validation are given in Table III. Win rates in bold indicate that they are the highest win rates of their rows. This result shows that, in nine games, the performance of the program with a decay factor of $\gamma = 0.6$ is significantly better

TABLE III
WIN PERCENTAGE OF CP_{NST} USING MOVE DECAY WITH $\gamma \in \{1, 0.6\}$ AGAINST CP_{MAST} WITHOUT DECAY, STARTCLOCK = 70 s, PLAYCLOCK = 40 s, ON GOGENERAL

Game	$\gamma=1$	$\gamma=0.6$
Zhadu	74.9 (±4.51)	**75.5** (±4.39)
GridGame2	52.3 (±3.79)	**52.8** (±4.52)
3DTicTacToe	73.3 (±3.87)	**80.4** (±3.59)
TTCC4	**85.4** (±2.18)	84.4 (±1.69)
Connect5	70.4 (±3.57)	**78.9** (±3.79)
Checkers	68.9 (±5.14)	**80.0** (±4.38)
Breakthrough	63.7 (±3.69)	**72.3** (±2.82)
Knightthrough	47.7 (±5.29)	**50.0** (±5.30)
Othello	**67.4** (±4.54)	67.0 (±4.55)
Skirmish	69.6 (±5.01)	**70.1** (±5.03)
Merrills	44.6 (±2.81)	**50.9** (±2.82)
Quad	79.1 (±2.96)	**92.3** (±2.30)
Sheep and Wolf	61.1 (±3.94)	**61.3** (±4.73)
Farmers	72.2 (±2.64)	**73.1** (±3.11)
TTCC4 3P	53.2 (±3.66)	**58.1** (±2.43)
Chinese Checkers 3P	**57.6** (±4.87)	55.1 (±5.32)
Battle	19.2 (±4.01)	**29.8** (±4.69)
Chinook	73.7 (±2.88)	**79.4** (±1.96)
Runners	35.7 (±4.62)	**36.7** (±4.60)
Pawn Whopping	**52.2** (±2.80)	51.3 (±2.80)

than the program without a decay factor (i.e., *3DTicTacToe*, *Connect5*, *Checkers*, *Breakthrough*, *Merrills*, *Quad*, *TTCC4 3P*, *Battle*, and *Chinook*). In other games, the performance is approximately equal. We suspect that games, in which the quality of a move highly depends on the game state and current phase of the game, can be improved by using a decay factor. Games without this property may profit less from a decay factor. This line of reasoning is supported by the results. Namely, in *Othello*, the decay factor did not improve the results. In this game, there are certain moves that are always good independent of the game state, like placing a stone in the corner.

Also, as reported previously by Tak *et al.* [8], we see that NST is mostly superior to MAST as a general move-selection strategy, with the notable exceptions of the simultaneous-move

TABLE IV
WIN PERCENTAGE OF CP_{MAST} USING MOVE DECAY WITH DIFFERENT VALUES OF γ AGAINST CP_{MAST} WITHOUT DECAY, STARTCLOCK = 60 s, PLAYCLOCK = 30 s, ON GOGENERAL

Game	$\gamma = 0$	$\gamma = 0.2$	$\gamma = 0.4$	$\gamma = 0.6$	$\gamma = 0.8$
Zhadu	52.8 (±3.73)	51.2 (±4.44)	58.9 (±3.71)	54.2 (±3.78)	53.4 (±2.66)
GridGame2	50.0 (±3.56)	50.0 (±4.01)	50.0 (±3.28)	49.9 (±3.26)	50.0 (±2.84)
3DTicTacToe	88.0 (±1.95)	92.4 (±2.00)	91.3 (±1.80)	87.7 (±2.10)	77.3 (±2.35)
TTCC4	48.3 (±3.65)	50.4 (±4.54)	52.7 (±3.81)	52.0 (±3.81)	50.6 (±2.75)
Connect5	77.6 (±3.04)	76.4 (±3.81)	76.3 (±3.16)	68.8 (±3.44)	61.7 (±3.19)
Checkers	59.1 (±4.43)	67.4 (±4.58)	67.4 (±4.28)	65.0 (±4.45)	62.7 (±4.07)
Breakthrough	53.2 (±5.03)	53.0 (±4.11)	56.7 (±4.67)	58.0 (±4.91)	55.4 (±3.53)
Knightthrough	53.2 (±3.92)	54.3 (±3.22)	53.4 (±4.25)	52.2 (±4.21)	52.2 (±2.79)
Othello	43.4 (±5.13)	44.9 (±4.22)	47.9 (±5.56)	46.2 (±5.11)	46.3 (±3.64)
Skirmish	49.6 (±4.08)	48.6 (±5.22)	49.4 (±4.39)	51.6 (±4.40)	48.7 (±2.89)
Merrills	53.5 (±3.71)	54.7 (±3.98)	50.6 (±5.24)	52.1 (±5.23)	51.1 (±3.69)
Quad	72.2 (±2.86)	77.8 (±3.17)	76.6 (±2.72)	73.9 (±2.80)	65.1 (±2.13)
Sheep and Wolf	50.0 (±3.92)	51.2 (±4.40)	51.1 (±3.58)	49.2 (±3.59)	50.4 (±3.15)
Farmers	48.3 (±2.90)	54.3 (±3.25)	53.6 (±2.66)	53.9 (±4.95)	50.5 (±2.33)
TTCC4 3P	51.9 (±3.37)	49.6 (±4.37)	53.8 (±3.64)	54.0 (±3.64)	51.9 (±3.19)
Chinese Checkers 3P	52.9 (±4.06)	53.1 (±5.20)	48.0 (±4.39)	51.8 (±4.36)	49.7 (±2.86)
Battle	50.5 (±3.57)	50.2 (±2.93)	49.4 (±3.84)	52.6 (±3.82)	48.1 (±3.34)
Chinook	53.1 (±3.70)	62.0 (±4.64)	63.5 (±3.86)	60.2 (±3.95)	60.1 (±2.76)
Runners	51.8 (±4.42)	53.8 (±5.03)	51.6 (±4.11)	52.5 (±4.08)	50.3 (±3.55)
Pawn Whopping	49.9 (±2.78)	50.1 (±3.13)	50.6 (±2.56)	49.5 (±4.78)	49.2 (±2.24)

games *Battle* and *Runners*. Both these games could be classified as greedy as opposed to strategic, that is, the same greedy action is often the best independent of the current state and the recent move history (for example, in *Runners*, the furthest advancing action is the best one to take in all game states); such situations are best case scenarios for MAST.

2) Move Decay in MAST: As shown in Section VI-A1, positive results are obtained with decay after an actual move in the game. Therefore, we tested whether move decay also works for other simulation strategies, CP_{MAST} in this case. CP_{MAST} with move decay was matched against CP_{MAST} without decay. The results are shown in Table IV. Again, we see that a decay factor may improve the program. In contrast with NST, simply resetting the statistics each move (which means $\gamma = 0$) has approximately the same or better performance than no decay. The result shows that, in six games, the performance of the program with a decay factor of $\gamma = 0.4$ is significantly better than the program without a decay factor (i.e., *Zhadu*, *3DTicTacToe*, *Connect5*, *Checkers*, *Quad*, and *Chinook*). The performance stays approximately the same in other games. Furthermore, we notice that there is an overlap with NST in the games where decaying is effective (*3DTicTacToe*, *Connect5*, *Checkers*, and *Quad*). This can be explained by the fact that the N-grams of length 1 are in essence the same as MAST, which means that NST will behave similar to MAST when these techniques are changed in the same way (e.g., with a decay factor).

In order to validate the results in a non-self-play experiment, the CP_{MAST} with $\gamma = 0.4$ was matched against CP_{NST} with $\gamma = 1$. CP_{MAST} with $\gamma = 0.4$ is used, because that appears to be the optimal value. It has the highest win rate over all the games, namely, 57.6%. As a reference experiment, CP_{MAST} with $\gamma = 1$ played against CP_{NST}. The results of the validation are given in Table V. It shows again that the same four games profit from a decay factor, namely, *3DTicTacToe*, *Connect5*, *Checkers*, and *Quad*.

3) Move Decay in One-Player Games: The reasoning behind a decay factor is that during a game, the learned information can become outdated when the opponent selects a branch the current player did not investigate thoroughly. In one-player games, this problem does not occur, therefore we do not expect decay to be beneficial in one-player games. Indeed, Table VI shows that there is hardly any improvement by using a decay factor. As it

TABLE V
WIN PERCENTAGE OF CP_{MAST} USING MOVE DECAY AND $\gamma \in \{1, 0.4\}$ AGAINST CP_{NST} WITHOUT DECAY, STARTCLOCK = 70 s, PLAYCLOCK = 40 s, ON GOGENERAL

Game	$\gamma = 1$	$\gamma = 0.4$
Zhadu	20.3 (±3.81)	**24.1** (±4.18)
GridGame2	47.2 (±5.16)	**47.7** (±4.10)
3DTicTacToe	28.2 (±4.00)	**69.8** (±4.00)
TTCC4	16.7 (±3.67)	**17.2** (±3.05)
Connect5	26.7 (±4.21)	**59.6** (±4.23)
Checkers	27.5 (±4.84)	**47.0** (±5.50)
Breakthrough	**31.8** (±4.76)	25.4 (±4.34)
Knightthrough	49.4 (±5.28)	**51.3** (±4.53)
Othello	**34.9** (±4.67)	28.4 (±4.42)
Skirmish	27.8 (±4.94)	**33.8** (±5.25)
Merrills	48.2 (±4.66)	**54.3** (±5.31)
Quad	27.5 (±3.80)	**61.1** (±3.46)
Sheep and Wolf	**36.3** (±4.43)	34.9 (±4.31)
Farmers	33.2 (±3.79)	**33.9** (±3.02)
TTCC4 3P	42.5 (±4.54)	**47.3** (±4.49)
Chinese Checkers 3P	36.7 (±5.20)	**41.8** (±5.32)
Battle	76.9 (±4.02)	**79.0** (±3.25)
Chinook	27.8 (±3.47)	**36.3** (±3.39)
Runners	**67.5** (±4.82)	63.6 (±4.88)
Pawn Whopping	46.5 (±3.77)	**47.7** (±3.01)

TABLE VI
AVERAGE SCORES OF CP_{MAST} USING MOVE DECAY WITH $\gamma \in \{0.4, 1.0\}$ AND CP_{NST} USING MOVE DECAY WITH $\gamma \in \{0.6, 1.0\}$, STARTCLOCK = 70 s, PLAYCLOCK = 40 s, ON GO4NATURE01

Game	CP_{MAST} $\gamma = 0.4$	CP_{MAST} $\gamma = 1.0$	CP_{NST} $\gamma = 0.6$	CP_{NST} $\gamma = 1.0$
Sudoku_simple	38.0 (±0.81)	38.4 (±0.85)	65.5 (±1.20)	62.5 (±1.15)
StatespaceLarge	30.9 (±0.62)	30.3 (±0.53)	29.3 (±0.36)	30.0 (±0.45)
Queens	82.4 (±0.65)	81.5 (±0.67)	86.7 (±0.59)	85.9 (±0.57)
Pancakes88	61.1 (±0.78)	61.0 (±0.80)	55.7 (±0.79)	55.5 (±0.82)
MaxKnights	65.4 (±1.89)	65.9 (±1.98)	67.5 (±1.84)	70.5 (±1.95)
Frogs and Toads	53.8 (±0.66)	54.2 (±0.65)	66.6 (±0.40)	62.9 (±0.38)

TABLE VII
WIN PERCENTAGE OF CP_{NST} USING MOVE DECAY WITH $\gamma = 0.6$ AGAINST CP_{MAST} USING MOVE DECAY WITH $\gamma = 0.4$, STARTCLOCK = 70 s, PLAYCLOCK = 40 s, ON GOGENERAL

Game	
Zhadu	70.3 (±5.17)
GridGame2	52.7 (±2.01)
3DTicTacToe	38.1 (±5.04)
TTCC4	80.9 (±3.67)
Connect5	52.6 (±2.48)
Checkers	65.6 (±3.20)
Breakthrough	73.9 (±4.89)
Knightthrough	44.1 (±2.44)
Othello	65.8 (±5.03)
Skirmish	73.4 (±4.71)
Merrills	51.5 (±2.83)
Quad	52.9 (±2.04)
Sheep and Wolf	63.0 (±5.09)
Farmers	66.0 (±4.19)
TTCC4 3P	56.8 (±3.01)
Chinese Checkers 3P	56.7 (±3.35)
Battle	27.2 (±4.13)
Chinook	70.0 (±4.50)
Runners	34.5 (±4.69)
Pawn Whopping	50.7 (±2.14)

TABLE VIII
WIN PERCENTAGE OF CP_{NST} USING BATCH DECAY WITH BATCH SIZE B AND DECAY FACTOR λ AGAINST CP_{NST} WITHOUT DECAY, STARTCLOCK = 60 s, PLAYCLOCK = 30 s, ON GOGENERAL

B	λ = 0.6	**Connect5** λ = 0.7	λ = 0.8	λ = 0.9
25	54.1 (±5.09)	60.4 (±5.01)	59.4 (±4.86)	64.0 (±4.81)
50	60.5 (±4.04)	60.8 (±4.06)	64.7 (±3.97)	66.6 (±3.92)
100	65.0 (±4.77)	62.0 (±3.64)	60.9 (±5.00)	59.2 (±4.97)
B	λ = 0.6	**Quad** λ = 0.7	λ = 0.8	λ = 0.9
25	65.4 (±4.98)	66.5 (±4.93)	63.4 (±5.03)	68.1 (±4.82)
50	68.7 (±4.80)	67.4 (±4.83)	65.8 (±4.85)	60.2 (±5.04)
100	64.6 (±4.92)	65.0 (±4.93)	60.4 (±4.94)	58.3 (±5.00)
B	λ = 0.6	**Chinook** λ = 0.7	λ = 0.8	λ = 0.9
25	31.7 (±5.04)	33.9 (±5.11)	40.3 (±5.29)	39.3 (±5.29)
50	40.1 (±4.34)	42.8 (±4.38)	43.6 (±4.36)	53.3 (±4.43)
100	45.7 (±5.39)	49.1 (±4.10)	54.3 (±5.44)	54.5 (±5.42)

can be easily detected how many players a game has, there is no problem, because when a player detects that it is in a one-player game, it can switch off the decay method.

4) Move Decay in NST Versus Move Decay in MAST: In [8], it is shown that NST outperforms MAST. The aim of this experiment is to find out whether this relation still holds when move decay is used. In this experiment, NST with move decay is matched against MAST with move decay. Both programs use their optimal parameter settings found in the previous experiments. Table VII shows the results. In 11 games, NST is clearly significantly better than MAST. Only in four games, *3DTicTacToe*, *Knightthrough*, *Battle*, and *Runners*, MAST performs significantly better than NST. These results are in line with the earlier obtained results. For instance, Tables III and V show that for *3DTicTacToe* CP_{MAST} profits much more from the decay factor than CP_{NST} does. Namely, the win rate for CP_{MAST} rises from 28.2% to 69.8% when decay is enabled, while the win rate for CP_{NST} only goes up from 73.3% to 80.4%.

B. Batch Decay

In this experiment, the aim is to find out whether, besides move decay, batch decay also performs well. Batch decay has two parameters, namely a decay factor λ and batch size B. First, the best λ is found by running experiments with $\lambda \in \{0.6, 0.7, 0.8, 0.9\}$ and $B \in \{25, 50, 100\}$ for the three games *Quad*, *Connect5*, and *Chinook*. The results are shown in Table VIII. It shows that for *Chinook*, a λ of 0.9 is clearly the optimal value. Only for this value of λ, the win rate becomes more than 50% for $B = 50$ and $B = 100$. Therefore, we select a λ of 0.9 for the rest of the experiments, because for both *Quad* and *Connect5* the win rates for the three different batch sizes are always above 50%.

In the second experiment, the aim is to find out how batch decay performs in the games for which move decay performed well. The results are shown in Table IX. The best results are obtained for $n = 50$. For this particular value, batch decay always improved the playing strength, except for *Chinook* where there was no positive or negative effect on the playing strength.

In the third experiment, all games (except the one-player games) are employed and CP_{NST} using batch decay with batch size 50 and decay factor 0.9 plays against CP_{NST} without decay. The results of this experiment are shown in Table X. It appears that for *3DTicTacToe* and *Farmers*, batch decay is a little bit better than the results shown for CP_{NST} in Table II at $\gamma = 0.6$. However, for most of the games, move decay is at least as good as batch decay. Furthermore, move decay is much better than batch decay in *Zhadu*, *TTCC4*, and *Breakthrough*. Therefore, batch decay does not seem to be a good alternative to move decay. Furthermore, move decay might be preferred, because it has only one parameter to tune instead of two.

C. Simulation Decay

The goal of this experiment is to find out whether simulation decay can be an alternative to move decay. It has only one parameter ω. This parameter is tuned over the three games *Connect5*, *Quad*, and *Chinook*. The results are shown in Table XI. According to this table, $\omega = 0.94$ performs best. In the next experiment, CP_{NST} with simulation decay $\omega = 0.94$ is matched against CP_{NST} without decay. Comparing the results shown in Table XII with the results of move decay in Table II at $\gamma = 0.6$, it is clear that in most games move decay is at least as good as simulation decay. There are a few exceptions. Move decay seems to be better than simulation decay in *Zhadu*, *TTCC4*, and *Chinese Checkers 3P*, while simulation decay appears to be better in *Breakthrough*, *Knightthrough*, and *Farmers*. However, the overall win rate is comparable with that of move decay with $\gamma = 0.6$, because both overall win rates are around 54%.

D. Simulation Decay Mixed With Move Decay

The aim of the last experiment is to investigate whether combining two decay methods may further improve the program. We choose to mix simulation decay with move decay. The reason to mix these two is that with simulation decay there were improvements in playing strength in some games for which no improvement was observed with move decay and the other way around. For example, in *Breakthrough*,

TABLE IX
WIN PERCENTAGE OF CP_{NST} USING BATCH DECAY WITH BATCH SIZE B AND DECAY FACTOR $\lambda = 0.9$ AGAINST CP_{NST} WITHOUT DECAY, STARTCLOCK = 60 s, PLAYCLOCK = 30 s, ON GOGENERAL

Game	$B = 25$	$B = 50$	$B = 100$	$B = 150$	$B = 200$
3DTicTacToe	68.3 (±3.55)	71.1 (±3.49)	68.5 (±3.57)	67.5 (±3.55)	66.4 (±3.81)
Connect5	64.0 (±4.81)	66.6 (±3.92)	59.2 (±4.97)	60.9 (±3.76)	58.2 (±4.06)
Checkers	60.9 (±5.16)	61.3 (±5.10)	61.1 (±5.17)	58.1 (±5.20)	51.9 (±5.31)
Merrills	57.8 (±5.19)	56.5 (±5.15)	54.0 (±5.22)	53.7 (±5.29)	53.2 (±5.26)
Quad	68.1 (±4.82)	60.2 (±5.04)	58.3 (±5.00)	56.6 (±3.30)	52.9 (±3.55)
Battle	63.6 (±3.88)	62.0 (±3.91)	60.4 (±3.95)	57.9 (±3.98)	57.8 (±4.25)
Chinook	39.3 (±5.29)	53.3 (±4.43)	54.5 (±5.42)	56.4 (±4.06)	57.9 (±4.02)

TABLE X
WIN PERCENTAGE OF CP_{NST} USING BATCH DECAY WITH BATCH SIZE $B = 50$ AND DECAY FACTOR $\lambda = 0.9$ AGAINST CP_{NST} WITHOUT DECAY, STARTCLOCK = 60 s, PLAYCLOCK = 30 s, ON GOGENERAL

Game	
Zhadu	32.6 (±3.32)
GridGame2	48.9 (±3.42)
3DTicTacToe	71.1 (±3.49)
TTCC4	31.2 (±3.68)
Connect5	66.6 (±3.92)
Checkers	61.3 (±5.10)
Breakthrough	31.9 (±5.01)
Knightthrough	42.8 (±4.17)
Othello	47.4 (±5.45)
Skirmish	52.6 (±3.97)
Merrills	56.5 (±5.15)
Quad	60.2 (±5.04)
Sheep and Wolf	42.7 (±3.39)
Farmers	57.7 (±3.90)
TTCC4 3P	54.9 (±3.58)
Chinese Checkers 3P	49.6 (±4.26)
Battle	62.0 (±3.91)
Chinook	53.3 (±4.43)
Runners	55.3 (±4.03)
Pawn Whopping	50.4 (±4.96)

Knightthrough, and *Farmers*, simulation decay seems to perform better than move decay. However, in *Zhadu* and *TTCC4*, move decay appears to be better. We have tested two different settings. In the first setting, CP_{NST} with move decay and $\gamma = 0.6$ and simulation decay with $\omega = 0.94$ plays against CP_{NST} without decay. These settings were the optimal settings when used separately and, therefore, when they are combined, may result in too much decay. Therefore, we also test a second setting where $\gamma = 0.8$ and $\omega = 0.97$.

The results are shown in Table XIII. As expected, the parameters $\gamma = 0.8$ and $\omega = 0.97$ perform better than the settings that were optimal for move decay and simulation decay alone. Nevertheless, mixing the two strategies did not really improve the playing strength, because the overall average is around 54%, which is comparable with that of move decay with $\gamma = 0.6$.

VII. CONCLUSION AND FUTURE WORK

In this paper, we experimented with applying a decay factor to simulation strategies in the domain of GGP. We tested three variants of decaying, namely move decay, batch decay, and simulation decay. Furthermore, we also experimented with combining move decay with simulation decay. Move decay decays after each move, batch decay decays after a fixed number of simulations, and simulation decay decays after each simulation, but only the N-grams/moves that occurred within that simulation. While all decaying variants offer genuine improvements in playing strength, in some games, move decay and simulation decay appear superior.

Move decay was implemented in two well-established methods for simulation biasing in GGP: NST and MAST. CADIAPLAYER, the GGP champion in 2012, was used in the experiments. For both simulation strategies, decaying showed significant performance gains. Moreover, with decaying factors tuned to appropriately balance remembering and forgetting (γ in the range 0.4–0.6), the improvements were robust across a large set of disparate games. One of the recurring challenges in developing new algorithmic techniques and enhancements for GGP is to demonstrate such robustness.

Decaying seems to work especially well in games where the selection of a best action is strongly influenced by local context, e.g., the current game position and recent history. By decaying the search statistics in such games, one still gets the generalization benefits of schemes such as NST and MAST, but with less risk of overgeneralizing.

Our results also confirm previous work suggesting that NST seems overall somewhat superior to MAST as a general move-selection strategy in simulation-based GGP, using both a larger test suite of games and by running more extensive experiments than before. The only notable exceptions were the simultaneous-move games *Battle* and *Runners*, but both these games are somewhat greedy as opposed to strategic, that is, the same greedy action is often the best independent of the current state and the recent move history. Such situations are best case scenarios for MAST.

For future research, it would be interesting to investigate methods for setting in real time the most appropriate decay factor and/or decay method for the game at hand. In this paper, for move decay, we chose to find a single decay factor that works reasonably well across many games, however, our experiments show that no single decay factor or decay method is the best for all the games in our test suite. Therefore, determining the decay method and its parameters online can give substantial improvement. Also of interest is to investigate how a decay factor can be applied to the UCT values. Related work is the discounted UCT, but there was no performance increase measured in *Othello*, *Havannah*, and *Go* [10]. Furthermore, our decaying methods are heuristics, and it would be interesting to investigate whether they can be combined in a way such that they minimize the mean squared error. Also, although we used a constant decaying factor for this work, it might be worthwhile to have it more dynamic, e.g., by being a function of the visit count. Moreover, tuning the parameters when mixing decay strategies could possibly also lead to significantly better results.

TABLE XI
WIN PERCENTAGE OF CP_{NST} USING SIMULATION DECAY WITH DECAY FACTOR ω AGAINST CP_{NST} WITHOUT DECAY, STARTCLOCK = 60 s, PLAYCLOCK = 30 s, ON GOGENERAL

Game	ω = 0.85	ω = 0.90	ω = 0.94	ω = 0.97	ω = 0.99
Connect5	38.6 ($\pm$5.03)	53.2 ($\pm$4.96)	64.5 ($\pm$5.33)	58.4 ($\pm$4.90)	59.2 ($\pm$5.08)
Quad	58.0 ($\pm$5.00)	64.6 ($\pm$4.16)	66.3 ($\pm$4.71)	64.0 ($\pm$4.18)	60.1 ($\pm$4.87)
Chinook	36.5 ($\pm$5.19)	52.4 ($\pm$5.36)	56.6 ($\pm$5.36)	50.9 ($\pm$5.38)	56.6 ($\pm$5.36)

TABLE XII
WIN PERCENTAGE OF CP_{NST} USING SIMULATION DECAY WITH DECAY FACTOR $\omega = 0.94$ AGAINST CP_{NST} WITHOUT DECAY, STARTCLOCK = 60 s, PLAYCLOCK = 30 s, ON GOGENERAL

Game	
Zhadu	39.1 ($\pm$3.64)
GridGame2	49.0 ($\pm$3.45)
3DTicTacToe	65.0 ($\pm$3.60)
TTCC4	39.6 ($\pm$4.13)
Connect5	64.5 ($\pm$5.33)
Checkers	56.0 ($\pm$4.46)
Breakthrough	59.6 ($\pm$4.09)
Knightthrough	52.3 ($\pm$4.28)
Othello	47.8 ($\pm$3.85)
Skirmish	54.3 ($\pm$3.11)
Merrills	59.0 ($\pm$5.04)
Quad	66.3 ($\pm$4.71)
Sheep and Wolf	53.9 ($\pm$2.82)
Farmers	60.4 ($\pm$2.58)
TTCC4 3P	57.0 ($\pm$3.60)
Chinese Checkers 3P	43.6 ($\pm$4.23)
Battle	53.3 ($\pm$3.97)
Chinook	56.6 ($\pm$5.36)
Runners	54.4 ($\pm$3.04)
Pawn Whopping	52.7 ($\pm$2.51)

TABLE XIII
WIN PERCENTAGE OF CP_{NST} USING SIMULATION DECAY AND MOVE DECAY WITH DECAY FACTORS $\gamma \in \{0.6, 0.8\}$ AND $\omega \in \{0.94, 0.97\}$ AGAINST CP_{NST} WITHOUT DECAY, STARTCLOCK = 60 s, PLAYCLOCK = 30 s, ON GOGENERAL

Game	$\gamma = 0.6$, $\omega = 0.94$	$\gamma = 0.8$, $\omega = 0.97$
Zhadu	26.7 ($\pm$2.39)	39.3 ($\pm$3.21)
GridGame2	49.3 ($\pm$2.65)	49.3 ($\pm$3.07)
3DTicTacToe	63.3 ($\pm$2.73)	68.0 ($\pm$3.14)
TTCC4	29.8 ($\pm$2.77)	41.8 ($\pm$3.74)
Connect5	54.9 ($\pm$2.84)	64.2 ($\pm$3.19)
Checkers	49.8 ($\pm$4.80)	55.8 ($\pm$5.58)
Breakthrough	45.7 ($\pm$4.41)	52.0 ($\pm$5.28)
Knightthrough	48.0 ($\pm$3.27)	48.8 ($\pm$3.79)
Othello	40.2 ($\pm$4.13)	45.9 ($\pm$4.87)
Skirmish	51.4 ($\pm$3.40)	54.0 ($\pm$3.95)
Merrills	54.9 ($\pm$3.94)	58.2 ($\pm$4.52)
Quad	63.1 ($\pm$2.44)	64.4 ($\pm$2.77)
Sheep and Wolf	47.7 ($\pm$2.63)	54.1 ($\pm$3.18)
Farmers	59.7 ($\pm$1.98)	61.8 ($\pm$2.28)
TTCC4 3P	55.8 ($\pm$2.76)	55.6 ($\pm$3.21)
Chinese Checkers 3P	41.2 ($\pm$3.22)	52.3 ($\pm$3.79)
Battle	62.1 ($\pm$2.95)	60.3 ($\pm$3.43)
Chinook	48.8 ($\pm$3.09)	55.9 ($\pm$3.56)
Runners	52.5 ($\pm$3.33)	49.5 ($\pm$3.83)
Pawn Whopping	50.7 ($\pm$2.19)	50.9 ($\pm$2.23)

APPENDIX

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

- *Sudoku_Simple* is played on a grid of 9×9 cells. This grid is further divided into nine, 3×3 blocks of nine cells. The aim is to put all numbers from 1 until 9 in the cells of each column, row, and block. The player gets three points for each correctly filled row, column, or block. An additional 19 points is given when the player fills the entire grid correctly.
- *StatespaceLarge* is a game where the player controls a robot by choosing from four different directions. The game ends after 14 steps. The score for the player, which ranges from 7 to 100, depends on the directions chosen per step.
- *Queens* is an instance of the n-queens puzzle, where in this case $n = 10$. The goal is to put ten queens on a 10×10 checkerboard in such a way that these queens do not attack each other. After placing the ten queens, a score is calculated such that there are higher scores when fewer queens are attacking each other. A score of 100 is obtained when there is no queen attacking any other queen.
- *Pancakes88* is a sorting game where eight pancakes have to be put in order. Each move, the player chooses the pancake to flip which will change the order of the pancakes. If the player is able to put the pancakes in order, the score will range from 40 to 100, depending on how many steps it took to rearrange the pancakes. Zero points are scored if, after 40 steps, the pancakes are still not in the correct order.
- *MaxKnights* is similar to *Queens*. It is played on an 8×8 chessboard and each turn the player has to put a chess knight on the board. As soon as one of the knights attacks another knight the game is over. The score for the player depends on the number of knights that are put on the board.
- *Frogs and Toads* is played on two 4×4 boards which are diagonally placed along each other and share the middle cell of this diagonal. In the initial position, the board on the lower right is filled with 15 frogs and the board on the upper left is filled with 15 toads. The cell that is shared by both boards is empty. The goal of the game is to inverse the initial position. To achieve this, the player can move a frog or a toad to an empty adjacent cell (horizontal or vertical) or it may jump (horizontally or vertically) over another frog or toad into an empty cell. After 116 steps, the game ends and points will be given based on how many of the frogs and toads are placed correctly.

The following two-player, turn-taking games are used.

- *Zhadu* is a strategy game consisting of a placement phase and a movement phase. The first piece that is captured determines what other piece needs to be captured in order to win.
- In *GridGame2*, each player has to find a book, a candle, and a bell. A score between 0 and 100 is given, based on how many items were found.

- *3DTicTacToe* is a variant of *TicTacToe*. It is played on a $4 \times 4 \times 4$ cube, and the goal is to align four pieces in a straight line.
- *TTCC4* stands for *TicTacChessCheckersFour*. 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.
- *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 *Five-in-a-Row*.
- *Checkers* is played on an 8×8 board, and the aim is to capture the pieces of the opponent.
- *Breakthrough* is played on an 8×8 board. Each player starts on one side of the board, and the goal is to move one of their pieces to the other side of the board.
- *Knightthrough* is almost the same as *Breakthrough*, but is played with chess knights.
- *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 their own color on the board at the end of the game.
- *Skirmish* is played on an 8×8 board with different kind 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.
- *Merrills* is also known as *Nine Men's Morris*. Both players start with nine pieces each. In order to win, pieces of the opponent need to be captured. The objective is to form a horizontal line or a vertical line of three pieces, called a mill, because pieces in a mill cannot be captured. The game ends when one player has only two pieces left.
- *Quad* is played on a 7×7 board. Each player has "quad" pieces and "white" pieces. The purpose of the "white" pieces is to form blockades. The player that forms a square consisting of four "quad" pieces wins the game.
- *Sheep and Wolf* is an asymmetric 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-taking games are used.

- *Farmers* is a trading game. In the beginning of the game, each player gets the same amount of money. They can use the money to buy cotton, cloth, wheat, and flour. It is also possible to buy a farm or a factory and then the player can produce its own products. The player that has the most money at the end of the game wins.
- *TTCC4 3P* is the same as *TTCC4*, but then with one extra player.
- *Chinese Checkers 3P* is played on a star-shaped board. Each player starts with three pieces positioned in one corner. The aim is to move all these three pieces to the empty corner 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 used.

- *Battle* is played on an 8×8 board. Each player has 20 disks. These disks can move one square or capture an opponent square next to them. Instead of a move, the player can choose to defend a square occupied by their piece. If an attacker attacks such a defended square, the attacker will be captured. The goal is to be the first player to capture ten opponent disks.
- *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. Black and White move their pieces simultaneously like Checkers pawns. As in *Breakthrough*, the first player that reaches the opposite side of the board wins the game.
- In *Runners*, at each turn both players decide how many steps they want to move forward or backward. The aim is to reach the goal location before the opponent does.
- *Pawn Whopping* is similar to *Breakthrough*, but with slightly different movement and is simultaneous.

These games were chosen because they have been used in several previous CADIAPLAYER experiments [4], [5], [9], [35]–[38]. *Pawn Whopping* was used during the German Open in GGP of 2011 [39]. Furthermore, this selection contains different types of games: one-player games, two-player games, multiplayer games, constant-sum games, and general-sum games (e.g., *GridGame2*, *Skirmish*, *Battle*, *Chinook*, *Farmers*, and *Chinese Checkers 3P* belong to the latter).

ACKNOWLEDGMENT

The authors would like to thank M. Lanctot for proofreading the paper, and the anonymous reviewers for their helpful comments.

REFERENCES

[1] J. Clune, "Heuristic evaluation functions for general game playing," in *Proc. 22nd AAAI Conf. Artif. Intell.*, Menlo Park, CA, USA, 2007, pp. 1134–1139.

[2] S. Schiffel and M. Thielscher, "Automatic construction of a heuristic search function for general game playing," in *Proc. 7th Int. Joint Conf. Artif. Intell. Int. Workshop Nonmontonic Reason. Action Change*, 2007.

[3] S. Schiffel and M. Thielscher, "Fluxplayer: A successful general game player," in *Proc. 22nd AAAI Conf. Artif. Intell.*, Menlo Park, CA, USA, 2007, pp. 1191–1196.

[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] H. Finnsson, "Simulation-based general game playing," Ph.D. dissertation, Schl. Comput. Sci., Reykjavík Univ., Reykjavík, Iceland, 2012.

[6] J. Méhat and T. Cazenave, "Ary, a general game playing program," in *Proc. XIIIth Board Games Studies Colloq.*, Paris, France, 2010.

[7] 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.

[8] M. J. W. Tak, M. H. M. Winands, and Y. Björnsson, "N-grams and the last-good-reply policy applied in general game playing," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 2, pp. 73–83, Jun. 2012.

[9] H. Finnsson and Y. Björnsson, "Simulation-based approach to general game playing," in *Proc. 23rd AAAI Conf. Artif. Intell.*, D. Fox and C. Gomes, Eds., Menlo Park, CA, USA, 2008, pp. 259–264.

[10] J. Hashimoto, A. Kishimoto, K. Yoshizoe, and K. Ikeda, "Accelerated UCT and its application to two-player games," in *Advances in Computer Games (ACG 13)*, ser. Lecture Notes in Computer Science, H. J. van den Herik and A. Plaat, Eds. Berlin, Germany: Springer-Verlag, 2012, vol. 7168, pp. 1–12.

[11] 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.
[12] R. Coulom, "Efficient selectivity and backup operators in Monte-Carlo tree search," in *Computers and Games (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.
[13] 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.
[14] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multi-armed bandit problem," *Mach. Learn.*, vol. 47, no. 2-3, pp. 235–256, May 2002.
[15] J. Schaeffer, "The history heuristic," *Int. Comput. Chess Assoc. J.*, vol. 6, no. 3, pp. 16–19, 1983.
[16] D. E. Knuth and R. W. Moore, "An analysis of alpha-beta pruning," *Artif. Intell.*, vol. 6, no. 4, pp. 293–326, 1975.
[17] G. Casella and E. I. George, "Explaining the Gibbs sampler," *Amer. Stat.*, vol. 46, no. 3, pp. 167–174, 1992.
[18] J. A. Stankiewicz, "Knowledge-based Monte-Carlo tree search in Havannah," M.S. thesis, Dept. Knowl. Eng., Maastricht Univ., Maastricht, The Netherlands, 2011.
[19] J. A. Stankiewicz, M. H. M. Winands, and J. W. H. M. Uiterwijk, "Monte-Carlo tree search enhancements for Havannah," in *Advances in Computer Games (ACG 13)*, ser. Lecture Notes in Computer Science, H. J. van den Herik and A. Plaat, Eds. Berlin, Germany: Springer-Verlag, 2012, vol. 7168, pp. 60–71.
[20] C. J. R. H. Laschet, "Selection and play-out enhancements in MCTS for Tron," B.S. thesis, Dept. Knowl. Eng., Maastricht Univ., Maastricht, The Netherlands, 2012.
[21] E. J. Powley, D. Whitehouse, and P. I. Cowling, "Bandits all the way down: UCB1 as a simulation policy in Monte Carlo tree search," in *Proc. IEEE Conf. Comput. Intell. Games*, 2013, pp. 81–88.
[22] 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, Ed. *et al.* Berlin, Germany: , 2010, vol. 6024, pp. 201–210.
[23] C. E. Shannon, "Prediction and entropy of printed English," *Bell Syst. Tech. J.*, vol. 30, no. 1, pp. 50–64, 1951.
[24] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press, 1999.
[25] F. D. Laramée, "Using N-gram statistical models to predict player behavior," in *AI Programming Wisdom*, S. Rabin, Ed. Newton Center, MA, USA: Charles River Media, 2002, vol. 1, ch. 11, pp. 596–601.
[26] I. Millington, *Artificial Intelligence for Games*, 1st ed. San Mateo, CA, USA: Morgan Kaufmann, 2006, ch. 7, pp. 580–591.
[27] T. Nakamura, "Acquisition of move sequence patterns from game record database using N-gram statistics," in *Proc. 4th Game Program. Workshop Jpn.*, 1997, pp. 96–105.
[28] T. Kimura, T. Ugajin, and Y. Kotani, "Bigram realization probability for game tree search," in *Proc. Int. Conf. Technol. Appl. Artif. Intell.*, 2011, pp. 260–265.
[29] J. Hashimoto, "A study on game-independent heuristics in game-tree search," Ph.D. dissertation, Schl. Inf. Sci., Jpn. Ad. Inst. Sci. Technol., Kanazawa, Japan, 2011.
[30] T. Otsuki, "Extraction of 'forced move' from N-gram statistics," in *Proc. 10th Game Program. Workshop Jpn.*, 2005, pp. 89–96.
[31] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, ser. Adaptive Computation and Machine Learning. Cambridge, MA, USA: MIT Press, 1998.
[32] N. R. Sturtevant, "An analysis of UCT in multi-player games," *Int. Comput. Games Assoc. J.*, vol. 31, no. 4, pp. 195–208, 2008.
[33] P. D. Drake, "The last-good-reply policy for Monte-Carlo Go," *Int. Comput. Games Assoc. J.*, vol. 32, no. 4, pp. 221–227, 2009.
[34] H. Baier and P. D. 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.
[35] H. Finnsson, "CADIA-Player: A general game playing agent," M.S. thesis, Schl. Comput. Sci., Reykjavík Univ., Reykjavík, Iceland, 2007.
[36] H. Finnsson and Y. Björnsson, "Simulation control in general game playing agents," in *Proc. Int. Joint Conf. Artif. Intell. Workshop Gen. Game Playing*, Pasadena, CA, USA, 2009, pp. 21–26.
[37] H. Finnsson and Y. Björnsson, "Learning simulation control in general game-playing agents," in *Proc. 24th AAAI Conf. Artif. Intell.*, M. Fox and D. Poole, Eds., Menlo Park, CA, USA, 2010, pp. 954–959.
[38] H. Finnsson and Y. Björnsson, "CadiaPlayer: Search control techniques," *Künstliche Intelligenz*, vol. 25, no. 1, pp. 9–16, 2011.
[39] P. Kissmann and T. Federholzner, "German Open in GGP 2011," [Online]. Available: http://www.tzi.de/~kissmann/ggp/go-ggp/classical/games/
[40] M. Günther and S. Schiffel, "The Dresden GGP Server," 2012 [Online]. Available: http://130.208.241.192/ggpserver/

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 (M'14) 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 (M'09) 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 a Director (and cofounder) 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.