Time Management for Monte Carlo Tree Search

Hendrik Baier and Mark H. M. Winands, *Member, IEEE*

***Abstract*—*Monte Carlo Tree Search* (MCTS) is a popular approach for tree search in a variety of games. While MCTS allows for fine-grained time control, not much has been published on time management for MCTS programs under tournament conditions. This article first investigates the effects of various time-management strategies on playing strength in the challenging game of Go. A number of domain-independent strategies are then tested in the domains Connect-4, Breakthrough, Othello, and Catch the Lion. We consider strategies taken from the literature as well as newly proposed and improved ones. Strategies include both *semi-dynamic* strategies that decide about time allocation for each search before it is started, and *dynamic* strategies that influence the duration of each move search while it is already running. Furthermore, we analyze the effects of time management strategies on the distribution of time over the moves of an average game, allowing us to partly explain their performance. In the experiments, the domain-independent strategy STOP provides a significant improvement over the state of the art in Go, and is the most effective time management strategy tested in all five domains.**

I. Introduction

In tournament gameplay, time is a limited resource. *Sudden death*, the simplest form of time control, allocates to each player a fixed time budget for the whole game. If a player exceeds this time budget, she loses the game immediately. Since longer thinking times typically result in stronger moves, the player's task is to distribute her time budget wisely among all moves in the game. This is a challenging task both for human and computer players. Previous research on this topic [1]–[5] has mainly focused on the framework of $\alpha\beta$ search with iterative deepening. In a number of game domains however, this algorithm is more and more losing its appeal.

Compared to $\alpha\beta$ search, less has been published on *time management* for Monte Carlo Tree Search (MCTS) [6], [7]. MCTS however allows for much more fine-grained time-management strategies due to its *anytime* property. It can be stopped after every rollout and return a move choice that makes use of the complete search time so far, while $\alpha\beta$ searchers can only make use of completely explored root moves of a deepening iteration.

This article investigates and compares a variety of time-management strategies for MCTS. We include newly proposed strategies as well as strategies described in [7] or independently proposed in [6], partly in enhanced form. These strategies are tested in the domains of 13×13 and 19×19 Go, and as far as possible in Connect-4, Breakthrough, Othello, and Catch the Lion.

This article extends on the authors' previous work in [8]. It includes experiments in four additional test domains (Connect-4, Breakthrough, Othello, and Catch the Lion), which also allows for further analysis of the results. The performance of time-management strategies is compared across domains, and a shift of available time towards either the opening or the endgame is identified as an effect influencing the performance of many strategies. Consequently, a methodology is developed to isolate the effect of this shift and judge the effect of a given strategy independently of it. These new contributions are described in Section V.

The structure of this article is as follows. Section II gives an overview of related work on time management for game-playing programs. Section III outlines the approaches to time management studied in this work—both domain-independent techniques and techniques specific to Go. Section IV presents experimental results of all strategies in Go, while Section V gives the results of testing and analyzing the domain-independent strategies in the games of Connect-4, Breakthrough, Othello, and Catch the Lion. Conclusions and future research follow in Section VI.

The authors are with the Games and AI Group, Department of Knowledge Engineering, Maastricht University, Maastricht, The Netherlands.
e-mail: {hendrik.baier,m.winands}@maastrichtuniversity.nl

II. Time Management

The first publication to address the topic of time management in computer games was by Hyatt [3]. He observed that human chess grandmasters do not use an equal amount of time per move, but play standard openings quickly, think longest directly after coming out of the opening, and then play increasingly fast towards the end of the game. He also suggested a technique that lets $\alpha\beta$ search explore a position longer to find a better move if the best move of the last deepening iteration turns out to lose material.

Donninger [2] gave four "golden rules" for the use of time during a chess game, both for human and computer players: "a) Do not waste time in easy positions with only one obvious move. b) Use the opponent's thinking time effectively. c) Spend considerable time before playing a crucial move. d) Try to upset the opponent's timing." Donninger considered rule c) to be the most important one by far, but also the hardest. In this article, we try to approach rules a) and c) simultaneously by attempting to estimate the importance or difficulty of a position and adjusting search time accordingly. Rule b) can be addressed by MCTS engines with *pondering*, thinking during the opponent's turn, which allows to transfer a part of the search tree into the next move search. If the opponent makes an expected move, the relevant part of the search tree can be large enough to make a move without much further thinking on your own time, which takes away the opponent's opportunity to ponder (an example for rule d). Pondering is not considered in this article.

The first systematic evaluation of time-management algorithms for chess was published by Althöfer *et al.* [1]. Amongst others, strategies were proposed to identify trivial moves that

can be made quickly, as well as troublesome positions that require more thinking. The time controls considered, typical for chess, specify a given amount of time for a given number of moves. They are insofar different from sudden death as used in this article as it here does not refer to the number of moves by the player, but only to the total amount of time per game.

The domain of checkers was used by Markovitch and Sella [4] to automatically acquire a simple time-allocation strategy, distributing a fixed number of deep searches among the moves of a game. The authors divided time-management strategies into three categories. (1) *Static* strategies decide about time allocation to all future moves before the start of the game. (2) *Semi-dynamic* strategies determine the computation time for each move before the start of the respective move search. (3) *Dynamic* strategies make "live" timing decisions while the search process is running. This categorization is used in the remainder of this article.

A number of time-management models for modern chess engines were devised and tested by Šolak and Vučković [5]. Their model M2a involved the idea of estimating the remaining number of moves, given the number of moves already played, from a database of master games. We use a similar approach as the basis for our time-management strategies (called EXP). In more sophisticated models, the authors developed definitions for the complexity of a position—based on the number of legal moves—and allocated time accordingly.

Kocsis *et al.* [9] compared temporal difference learning and genetic algorithms for training a neural network to make semi-dynamic timing decisions in the game Lines of Action. The network could set the underlying $\alpha\beta$ program to one of three predefined search depths.

For the framework of MCTS, only two publications exist so far. A number of both dynamic and semi-dynamic time-management heuristics for 19×19 Go were evaluated by Huang *et al.* [7], assuming sudden-death time controls. We implemented and optimized their heuristics as a baseline for our approaches. The ideas of the "unstable evaluation" heuristic (UNST) and the "think longer when behind" heuristic (BEHIND) were first described and tested in [7]. UNST continues searching if after the regular search time, the most-visited move is not the highest-valued move as well. BEHIND searches longer when the player's winrate at the root is low. Enhanced versions are described under the names UNST-L and BEHIND-L in the next Section.

During the preparation of our experiments, Baudiš [6] published brief descriptions of the dynamic and semi-dynamic time management of the state-of-the-art MCTS Go program PACHI. Variants similar to our "close second" heuristic (CLOSE) and a special case of our "early stop" heuristic (STOP_A) were here formulated independently. CLOSE searches longer if the best and second best move are too close to each other. STOP_A stops searching if the currently best move cannot change anymore in the rest of the planned search time. We evaluate these strategies and propose generalized versions with the names CLOSE-L and STOP as well.

Independent from time management considerations, Huang *et al.* [10] proposed two pruning conditions for MCTS: the *absolute pruning condition* and the *relative pruning condition*. These techniques are related to the STOP strategy and are discussed in Section III-B.

III. Time-Management Strategies

In this section, we describe first the semi-dynamic (III-A), and then the dynamic time-management strategies (III-B) for the MCTS framework which were investigated in this article.

A. Semi-Dynamic Strategies

The following five strategies determine the search time for each move *before* the search for this move is started. EXP, OPEN and MID are domain-independent strategies, while KAPPA-EXP and KAPPA-LM are specific to the game of Go.

EXP. The simple EXP strategy for time allocation, used as the basis of all further enhancements in this article, divides the remaining thinking time for the entire game ($t_\text{remaining}$) by the expected number of remaining moves for the player (m_expected) and uses the result as the search time for the next move (t_nextmove). The formula is as follows:

$$t_\text{nextmove} = \frac{t_\text{remaining}}{m_\text{expected}} \tag{1}$$

m_expected can be estimated in various ways. Three heuristics are investigated in this article, two of them game-independent and one game-specific to the game of Go. The first game-independent heuristic (EXP-MOVES) estimates the number of remaining moves given the number of moves already played. An example would be an expectation of 71 remaining moves for the player at the first turn, or an expectation of 27 remaining moves at turn 90. The second game-independent heuristic (EXP-SIM) estimates the number of remaining moves given the length of simulated games in the preceding search. As an example, 63 more moves could be expected if the average simulated game in the search for the preceding move was 200 moves long, or an average simulation length of 60 moves could lead to an expectation of 28 more moves in the actual game. The third heuristic (EXP-STONES) is specific to Go and uses the number of stones on the board as an estimator of remaining game length. 40 stones on the board could be mapped to an expected 50 more moves, while 140 stones on the board could map to an expectation of 12 more moves for the player. Other games may or may not provide other indicators. The parameters for all three heuristics, e.g. the precise mapping from played moves to remaining moves for EXP-MOVES, are determined from a set of 1000 games played in self-play.

OPEN. The OPEN strategy puts emphasis on the opening phase of the game. Formula 2 modifies the search time for every move in the game by multiplying it with a constant "opening factor" $f_\text{opening} > 1$.

$$t_\text{nextmove} = f_\text{opening} \cdot \frac{t_\text{remaining}}{m_\text{expected}} \tag{2}$$

This results in more time per move being used in the beginning of the game than at the end. As opposed to the implicit assumption of Formula 1 that equal time resources should be allocated to every expected move, here it is assumed that

the first moves of a game have greater influence on the final outcome than the last moves and thus deserve longer search times.

MID. Instead of moves in the opening phase, the MID strategy increases search times for moves in the middle game, which can be argued to have the highest decision complexity of all game phases [7]. For this purpose, the time as given by Formula 1 is increased by a percentage determined by a Gaussian function over the set of move numbers, using three parameters a, b, and c for height, position and width of the "bell curve". The formula for the bell curve is

$$f_{\text{Gaussian}}(x) = ae^{-\frac{(x-b)^2}{2c^2}} \tag{3}$$

$$t_{\text{nextmove}} = (1 + f_{\text{Gaussian}}(\text{current move number})) \cdot \frac{t_{\text{remaining}}}{m_{\text{expected}}} \tag{4}$$

KAPPA-EXP. In [11], the concept of *criticality* was suggested for Go—as some intersections on the board are more important for winning the game than others, these should be recognized as "critical" or "hot", and receive special attention or search effort. To identify critical points, statistics are collected during rollouts on which player owns which intersections at the end of each simulation, and on how strongly this ownership is correlated with winning the simulated game [11], [12]. In the KAPPA-EXP strategy, we use a related concept for identifying not only critical intersections from the set of all intersections of a board, but also critical move choices from the set of all move choices in a game. A highly critical move choice is here understood as a choice that involves highly critical intersections. The KAPPA-EXP strategy distributes time proportional to the expected maximum point criticality given the current move number, as estimated from a database of 1000 games played by the program itself. The idea is that the maximum point criticality, taken over the set of all intersections I on the board, indicates how crucial the current move choice is. We chose Formula 5 to represent the criticality of an intersection i in move m of game g—the *kappa statistic*, a chance-corrected measure of agreement typically used to quantify inter-rater reliability [13]. Here, it is employed to quantify agreement between the variables "intersection i is owned by the player at the end of a rollout during m's move search" and "the player wins a rollout during m's move search".

$$\begin{aligned}\kappa_g^m(i) &= \frac{\text{agreement}^m_{\text{observed}} - \text{agreement}^m_{\text{expected}}}{1 - \text{agreement}^m_{\text{expected}}} \\ &= \frac{\frac{o^m_{\text{winner}}(i)}{n} - \left(\frac{o^m_{\text{white}}(i) w^m_{\text{white}} + o^m_{\text{black}}(i) w^m_{\text{black}}}{n^2}\right)}{1 - \left(\frac{o^m_{\text{white}}(i) w^m_{\text{white}} + o^m_{\text{black}}(i) w^m_{\text{black}}}{n^2}\right)}\end{aligned} \tag{5}$$

where n is the total number of rollouts, $o^m_{\text{winner}}(i)$ is the number of rollouts in which point i ends up being owned by the rollout winner, $o^m_{\text{white}}(i)$ and $o^m_{\text{black}}(i)$ are the numbers of rollouts in which point i ends up being owned by White and Black, respectively, and w^m_{white} and w^m_{black} are the numbers of rollouts won by White and Black, respectively. All numbers refer to the search for move m.

For application at move number m during a given game, the average maximum point criticality $\kappa^m_{\text{avg}} = \frac{1}{y}\sum_{g=1}^{y} \max_{i \in I} \kappa^m_g(i)$ is precomputed from a database of y games, linearly transformed using parameters for slope and intercept $s_{\kappa_{\text{avg}}}$ and $i_{\kappa_{\text{avg}}}$, and finally multiplied with the search time resulting in Formula 6.

$$t_{\text{nextmove}} = (\kappa^m_{\text{avg}} \cdot s_{\kappa_{\text{avg}}} + i_{\kappa_{\text{avg}}}) \cdot \frac{t_{\text{remaining}}}{m_{\text{expected}}} \tag{6}$$

KAPPA-LM. Instead of using the expected criticality for the current move number as defined above, the KAPPA-LM strategy uses the observed criticality as computed during the search for the player's previous move in the game. While KAPPA-EXP looks up κ from a table of values computed offline, KAPPA-LM estimates it online using search statistics. This value $\kappa_{\text{lastmove}} = \max_{i \in I} \kappa^{m-2}_{\text{current game}}(i)$ is again linearly transformed using parameters $s_{\kappa_{\text{lastmove}}}$ and $i_{\kappa_{\text{lastmove}}}$, and multiplied with the base search time. The formula is as follows:

$$t_{\text{nextmove}} = (\kappa_{\text{lastmove}} \cdot s_{\kappa_{\text{lastmove}}} + i_{\kappa_{\text{lastmove}}}) \cdot \frac{t_{\text{remaining}}}{m_{\text{expected}}} \tag{7}$$

For both KAPPA-EXP and KAPPA-LM, lower and upper bounds for the κ factor ensure lower and upper bounds for the total search time even in extreme positions. The algorithms are not very sensitive to these parameters, but without them games can be lost (in particular in the online version KAPPA-LM) due to occasional searches receiving too much or almost no time (extreme f values).

B. Dynamic Strategies

The following five strategies make time-allocation decisions for a move search *while* the respective search process is being carried out. BEHIND, UNST, CLOSE and STOP are domain-independent strategies, while KAPPA-CM is specific to the game of Go. Note that our implementations of CLOSE and STOP are only valid if MCTS plays the most-visited move after each move search, which is the case for all MCTS players in this article. Other approaches to CLOSE and STOP are imaginable if MCTS chooses for example the move with the highest estimated value.

BEHIND. As suggested in [7] as the "think longer when behind" heuristic, the BEHIND strategy prolongs the search if the player is falling behind. It triggers if after the regular search time—as computed by the semi-dynamic strategies described above—the win rate of the best move at the root is lower than a threshold v_{behind}. If this is the case, the search is continued for a time interval determined by multiplying the previously used search time with a factor f_{behind}. The rationale is that by using more time resources, the player could still find a way to turn the game around, while saving time for later moves is less important in a losing position. We have also modified this heuristic to check its condition for search continuation repeatedly in a loop. The maximum number of loops until the search is terminated is bound by a parameter l_{behind}. The single-check heuristic is called BEHIND, the multiple-check heuristic BEHIND-L (for "loop") in the following.

UNST. The UNST strategy, called "unstable evaluation" heuristic in [7], prolongs the search if after the regular search time the most-visited move at the root is not the highest-valued move as well. In this case, the search is continued for the previously used search time multiplied with a factor f_{unstable}. The idea is that by searching longer, the highest-valued move could soon become the most-visited and thus change the final move choice. Analogously to the BEHIND-L technique, UNST-L was introduced as an enhancement of UNST that repeatedly checks its trigger condition in a loop. The parameter specifying the maximum number of loops is l_{unstable}.

CLOSE. The proposed CLOSE strategy prolongs the search if after the regular search time the most-visited move and the second-most-visited move at the root are "too close", defined by having a relative visit difference lower than a threshold d_{close}. A similar strategy was developed independently in [6]. The search is then continued for the previously used search time multiplied with a factor f_{close}. Like the UNST strategy, CLOSE aims to identify difficult decisions that can make efficient use of an increase in search time. We propose two variants of this strategy. It can either be triggered only once (CLOSE) or repeatedly (CLOSE-L) after the regular search time is over. For CLOSE-L, a parameter l_{close} defines the maximum number of loops.

KAPPA-CM. Unlike the three dynamic strategies described above, the KAPPA-CM strategy does not wait for the regular search time to end. Instead, it uses the first e.g. 100 milliseconds of the search process to collect criticality data. Then it uses the maximum point criticality of the current move $\kappa_{\text{currentmove}} = \max_{i \in I} \kappa^m_{\text{current game}}(i)$ to modify the remaining search time. The formula is as follows:

$$t_{\text{currentmove}} = (\kappa_{\text{currentmove}} \cdot s_{\kappa_{\text{currentmove}}} + i_{\kappa_{\text{currentmove}}}) \cdot \frac{t_{\text{remaining}}}{m_{\text{expected}}} \quad (8)$$

The remaining search time can be either reduced or increased by this strategy. Upper and lower limits to the total search time apply.

STOP. The proposed "early stop" (STOP) strategy is based on the idea of terminating the search process as early as possible in case the best move cannot change anymore. For STOP, the search speed in simulations per second is measured, and in regular intervals (e.g. 50 rollouts) it is checked how many rollouts are still expected in the remainder of the total planned search time. If the number of simulations required for the second-most-visited move at the root to catch up to the most-visited one exceeds this expected number of remaining simulations, the search can safely be terminated without changing the final outcome.

However, not all of the remaining simulations in a search generally start with the second-most-visited move. Therefore, we introduce a parameter $p_{\text{earlystop}} \leq 1$ representing an estimate of the proportion of remaining rollouts that actually sample the second-most-visited move. The search is terminated if the number of rollouts needed for the second-most-visited move at the root to catch up to the most-visited one exceeds the expected number of remaining rollouts multiplied with $p_{\text{earlystop}}$. When setting this parameter to a value smaller than 1, an unchanged final outcome is no longer guaranteed. Optimal values of $p_{\text{earlystop}}$ have to be determined empirically. The termination criterion of STOP is:

$$\frac{n \cdot \text{timeleft}_n}{\text{timespent}_n} \cdot p_{\text{earlystop}} < \text{visits}_{\text{best}_n} - \text{visits}_{\text{secondbest}_n} \quad (9)$$

where n is the number of rollouts so far, timeleft_n is the rest of the planned search time, timespent_n is the search time already spent, $\text{visits}_{\text{best}_n}$ is the currently highest number of visits of any move at the root, and $\text{visits}_{\text{secondbest}_n}$ is the currently second-highest number of visits of any move at the root. All numbers refer to the state of the search after n rollouts.

If the expected time savings by the STOP strategy are not considered when computing planned search times, savings will accumulate throughout the game and early moves cannot benefit from them. In order to achieve a different distribution of the resulting time savings among all searches in the game, planned search times are multiplied with a parameter $f_{\text{earlystop}} \geq 1$ that is also determined empirically.

In order to test the effects of the two parameters $p_{\text{earlystop}}$ and $f_{\text{earlystop}}$ independently of each other, we introduce the name STOP_B for the special case of STOP with $p_{\text{earlystop}} = 1$ and free parameter $f_{\text{earlystop}}$. This variant can redistribute search time, but never stops a search before the final outcome is definitely known (it uses "safe" stopping). If $p_{\text{earlystop}} = 1$ and $f_{\text{earlystop}} = 1$ (stopping is "safe", and the redistribution of search time is deactivated as well), STOP is identical to a strategy mentioned independently—but not evaluated—in [6]. In the following, we call this special case STOP_A.

The *absolute pruning condition* proposed by Huang *et al.* in [10] can be seen as a weaker form of STOP_A, only stopping if one move has more than half the simulations planned for the entire search. This does not take the visit difference between the most-visited and second-most-visited move into account. The authors also proposed an "unsafe" criterion for pruning: Their *relative pruning condition* excludes individual moves from the search as soon as they are not expected to catch up with another move anymore. This expectation is based on a formula for the upper bound of the remaining simulations for a given move. However, the authors state that their condition is strict and rarely triggers, making a relaxed condition desirable. STOP allows to relax its stopping condition through the $p_{\text{earlystop}}$ parameter.

IV. Experimental Results in Go

All time-management strategies were implemented in OREGO [14] version 7.08. OREGO is a Go program using a number of MCTS enhancements like a transposition table [15], [16], RAVE [17], a simulation policy similar to that proposed in [18], and LGRF-2 [19]. The simulation policy takes domain-specific knowledge into account by trying to save groups under attack (*atari*), attacking opponent groups, and giving preference to moves that match a prespecified set of 3×3 intersection patterns on the board. After each search, the most-sampled move at the root is played. OREGO resigns

if its win rate at the root falls below 10%. The program ran on a CentOS Linux server consisting of four AMD Twelve-Core OpteronT 6174 processors (2.2 GHz). Unless specified otherwise, each experimental run involved 5000 games (2500 as Black and 2500 as White) of OREGO against the classic (non-MCTS-based) program GNU GO 3.8 [20], played on the 13×13 board, using Chinese rules (area scoring), positional superko, and 7.5 komi. The playing strength of a non-MCTS program like GNU GO is difficult to measure since it has weaknesses that can be relatively easily exploited by human players, but it is estimated to play at amateur level (8-12 kyu on the 19×19 board to 5-7 kyu on the 9×9 board). OREGO plays at around 5-6 kyu with 30 minutes per game on the 19×19 board, and probably stronger on the 13×13 board. For testing against GNU GO, OREGO's strength was reduced here by lowering the time to 30 seconds per game unless specified otherwise. GNU GO ran at its default level of 10, with the capture-all-dead option turned on. It had no time limit. OREGO used a single thread and no pondering. The speed of OREGO was about 1850 simulations per second when searching from the initial position. Optimal parameter settings for the time management strategies were found by manually testing a wide range of parameter values, from around 10-20 for strategies with a single parameter to hundreds of settings for strategies with three parameters, with 500 or 1000 games each against GNU GO.

The remainder of this section is structured as follows. In IV-A, the strategies in [7] are tested as a baseline. Next, IV-B presents results of experiments with semi-dynamic strategies. Dynamic strategies are tested in IV-C. Finally, in IV-D the best-performing strategy is compared to the baseline in self-play, as well as to OREGO with fixed time per move. The results of this section have largely been published before in [8]. However, some experiments were re-run. The only significant difference to [8] is the better performance of an improved version of EXP-SIM, and the addition of the BEHIND-L strategy.

A. ERICA-BASELINE

In order to compare the results to a state-of-the-art baseline, the strategies described in [7] were implemented and evaluated. The thinking time per move was computed according to the "basic formula"

$$t_{\text{nextmove}} = \frac{t_{\text{remaining}}}{C} \tag{10}$$

where $C = 30$ was found to be optimal for OREGO, as well as the "enhanced formula"

$$t_{\text{nextmove}} = \frac{t_{\text{remaining}}}{C + \max(\text{MaxPly} - \text{MoveNumber}, 0)} \tag{11}$$

with $C = 20$ and MaxPly $= 40$. The UNST heuristic, using a single loop as proposed in [7], worked best with $f_{\text{unstable}} = 0.5$. The BEHIND heuristic was most successful in OREGO with $v_{\text{behind}} = 0.6$ and $f_{\text{behind}} = 0.75$. Note that BEHIND had not been found to be effective at first in [8]. This is because only win rate thresholds up to 0.5 had been tested originally—a player with a win rate of more than 0.5 cannot be called "behind" after all. See subsection V-F for a discussion of the effect of higher thresholds.

TABLE I
PERFORMANCE OF ERICA'S TIME MANAGEMENT ACCORDING TO [7] IN 13×13 GO.

Player	Win rate against GNU GO	95% conf. int.
Basic formula	28.6%	27.3%–29.9%
Enhanced formula	31.4%	30.1%–32.7%
ERICA-BASELINE	35.3%	34.0%–36.7%

ERICA's time-management strategies were tested against GNU GO using the basic formula, using the enhanced formula, and using the enhanced formula plus UNST and BEHIND heuristic (called ERICA-BASELINE from now on). Table I presents the results—the enhanced formula is significantly stronger than the basic formula ($p<0.01$), and ERICA-BASELINE is significantly stronger than the enhanced formula ($p<0.001$).

B. Semi-Dynamic Strategies

This subsection presents the results for the EXP, OPEN, MID, KAPPA-EXP, and KAPPA-LM strategies in Go.

EXP-MOVES, EXP-SIM and EXP-STONES. As our basic time-management approach, EXP-MOVES, EXP-SIM and EXP-STONES were tested. The first three rows of Table II show the results. As EXP-STONES appeared to perform best, it was used as the basis for all further experiments with the game of Go. Note however that the differences were not statistically significant. The average error in predicting the remaining number of moves was 14.16 for EXP-STONES, 13.95 for EXP-MOVES, and 14.23 for EXP-SIM.

OPEN. According to preliminary experiments with OPEN, the "opening factor" $f_{\text{opening}} = 2.5$ seemed most promising. It was subsequently tested with 5000 additional games against GNU GO. Table II shows the result: EXP-STONES with OPEN is significantly stronger than plain EXP-STONES ($p<0.001$).

MID. Initial experiments with MID showed Formula 3 to perform best with $a = 2$, $b = 40$ and $c = 20$. It was then tested with 5000 additional games. As Table II reveals, EXP-STONES with MID is significantly stronger than plain EXP-STONES ($p<0.001$).

KAPPA-EXP. The best parameter setting for KAPPA-EXP found in preliminary experiments was $s_{\kappa_{\text{avg}}} = 8.33$ and $i_{\kappa_{\text{avg}}} = -0.67$. Lower and upper bounds for the *kappa* factor were set to 0.5 and 10, respectively. Table II presents the result of testing this setting. EXP-STONES with KAPPA-EXP is significantly stronger than plain EXP-STONES ($p<0.001$).

KAPPA-LM. Here, $s_{\kappa_{\text{lastmove}}} = 8.33$ and $i_{\kappa_{\text{lastmove}}} = -0.67$ were chosen for further testing against GNU GO as well. Lower and upper bounds for the *kappa* factor were set to 0.25 and 10. The test result is shown in Table II. EXP-STONES with KAPPA-LM is significantly stronger than plain EXP-STONES ($p<0.001$).

TABLE II
PERFORMANCE OF THE INVESTIGATED SEMI-DYNAMIC STRATEGIES IN 13×13 GO.

Player	Win rate against GNU GO	95% conf. int.
EXP-MOVES	24.0%	22.9%–25.2%
EXP-SIM	23.5%	22.3%–24.7%
EXP-STONES	25.5%	24.3%–26.7%
EXP-STONES with OPEN	32.0%	30.8%–33.4%
EXP-STONES with MID	30.6%	29.3%–31.9%
EXP-STONES with KAPPA-EXP	31.7%	30.5%–33.0%
EXP-STONES with KAPPA-LM	31.1%	29.8%–32.4%
ERICA-BASELINE	35.3%	34.0%–36.7%

C. Dynamic Strategies

In this subsection the results for the BEHIND, UNST, CLOSE, STOP, and KAPPA-CM strategies in the game of Go are given.

BEHIND. Just like the "enhanced formula" of [7], EXP-STONES was found to be significantly improved by BEHIND in OREGO only with a threshold v_{behind} of higher than 0.5. This is also true after the introduction of BEHIND-L, and will be discussed in subsection V-F. The best parameter settings in preliminary experiments were $f_{\text{behind}} = 0.5$ and $v_{\text{behind}} = 0.6$ for BEHIND, and $f_{\text{behind}} = 0.25$, $v_{\text{behind}} = 0.6$, and $l_{\text{behind}} = 2$ for BEHIND-L. Detailed results of an additional 5000 games with these setttings are given in Table III. EXP-STONES with BEHIND is significantly stronger than plain EXP-STONES ($p<0.001$). EXP-STONES with BEHIND-L, however, could not be shown to be significantly stronger than EXP-STONES with BEHIND.

UNST. The best results in initial experiments with UNST were achieved with $f_{\text{unstable}} = 1.5$. For UNST-L, $f_{\text{unstable}} = 0.75$ and $l_{\text{unstable}} = 2$ turned out to be promising values. These settings were tested in 5000 further games. Table III shows the results. EXP-STONES with UNST is significantly stronger than plain EXP-STONES ($p<0.001$). EXP-STONES with UNST-L, in turn, is significantly stronger than EXP-STONES with UNST ($p<0.05$).

CLOSE. The best-performing parameter settings in initial experiments with CLOSE were $f_{\text{close}} = 1.5$ and $d_{\text{close}} = 0.4$. When we introduced CLOSE-L, $f_{\text{close}} = 0.5$, $d_{\text{close}} = 0.5$ and $l_{\text{close}} = 4$ appeared to be most successful. Table III presents the results of testing both variants in 5000 more games. EXP-STONES with CLOSE is significantly stronger than plain EXP-STONES ($p<0.001$). EXP-STONES with CLOSE-L, in turn, is significantly stronger than EXP-STONES with CLOSE ($p<0.001$).

KAPPA-CM. The best parameter setting for KAPPA-CM found in preliminary experiments was $s_{\kappa_{\text{currentmove}}} = 8.33$ and $i_{\kappa_{\text{currentmove}}} = -1.33$. Lower and upper bounds for the *kappa* factor were set to 0.6 and 10. Table III reveals the result of testing this setting. EXP-STONES with KAPPA-CM is significantly stronger than plain EXP-STONES ($p<0.05$). However, it is surprisingly weaker than both EXP-STONES using KAPPA-EXP and EXP-STONES with KAPPA-LM ($p<0.001$). The time of 100 msec used to collect current criticality information might be too short, such that noise is too high. Preliminary tests with longer collection times (keeping the other parameters equal) did not show significantly better results. A retuning of the other parameters with longer collection times remains as future work.

TABLE III
PERFORMANCE OF THE INVESTIGATED DYNAMIC STRATEGIES IN 13×13 GO.

Player	Win rate against GNU GO	95% conf. int.
EXP-STONES with BEHIND	29.9%	28.7%–31.2%
EXP-STONES with BEHIND-L	30.5%	29.2%–31.8%
EXP-STONES with UNST	33.6%	32.3%–34.9%
EXP-STONES with UNST-L	35.8%	34.4%–37.1%
EXP-STONES with CLOSE	32.6%	31.3%–33.9%
EXP-STONES with CLOSE-L	36.5%	35.2%–37.9%
EXP-STONES with KAPPA-CM	27.3%	26.1%–28.6%
EXP-STONES with STOP_A	25.3%	24.1%–26.5%
EXP-STONES with STOP_B	36.7%	35.4%–38.0%
EXP-STONES with STOP	**39.1%**	**38.0%–40.8%**
EXP-STONES	25.5%	24.3%–26.7%
ERICA-BASELINE	35.3%	34.0%–36.7%

TABLE IV
PERFORMANCE OF EXP-STONES WITH STOP VS. ERICA-BASELINE IN 13×13 GO.

Time setting	Win rate against ERICA-BASELINE	95% conf. int.
30 sec sudden death	63.7%	61.5%–65.8%
60 sec sudden death	59.4%	57.2%–61.5%
120 sec sudden death	60.7%	58.5%–62.8%

STOP. The best settings found for STOP were $f_{\text{earlystop}} = 2.5$ and $p_{\text{earlystop}} = 0.4$. This variant significantly outperformed ($p<0.001$) plain EXP-STONES as well as ERICA-BASELINE in 5000 games each against GNU GO. It is also significantly stronger ($p<0.01$) than the best-performing setting of STOP_B with $f_{\text{earlystop}} = 2$. STOP_B in turn is significantly stronger than plain EXP-STONES as well as STOP_A ($p<0.001$). The STOP_A strategy did not show a significant improvement to the EXP-STONES baseline. Table III presents the results.

D. Strength Comparisons

This subsection focuses on the time-management strategy that proved most successful in Go, the STOP strategy. It attempts to answer the question whether STOP's effectiveness generalizes to longer search times (60 seconds, 120 seconds per game) and to the larger 19×19 board size. Furthermore, an indication is given of how strong the effect of time management is when compared to fixed time per move.

Comparison with ERICA-BASELINE on 13×13. Our strongest time-management strategy on the 13×13 board, EXP-STONES with STOP, was tested in self-play against OREGO with ERICA-BASELINE. Time settings of 30, 60, and 120 seconds per game were used with 2000 games per data point. Table IV presents the results. For all time settings, EXP-STONES with STOP was significantly stronger ($p<0.001$).

Comparison with ERICA-BASELINE on 19×19. In this experiment, we pitted EXP-STONES with STOP against

TABLE V
PERFORMANCE OF EXP-STONES WITH STOP VS. ERICA-BASELINE IN 19×19 GO.

Time setting	Win rate against ERICA-BASELINE	95% conf. int.
300 sec sudden death	62.7%	60.6%–64.9%
900 sec sudden death	60.2%	58.0%–62.4%

ERICA-BASELINE on the 19×19 board. The best parameter settings found were $C = 60$, MaxPly $= 110$ and $f_{\text{unstable}} = 1$ for ERICA-BASELINE, and $f_{\text{earlystop}} = 2.2$ and $p_{\text{earlystop}} = 0.45$ for STOP. Time settings of 300 and 900 seconds per game were used with 2000 games per data point. OREGO played about 960 simulations per second when searching from the empty 19×19 board. The results are shown in Table V—for both time settings, EXP-STONES with STOP was significantly stronger ($p<0.001$).

Comparison with fixed time per move. To illustrate the effect of successful time management, two additional experiments were conducted with OREGO using fixed time per move in 13×13 Go. In the first experiment, the time per move (650 msec) was set so that approximately the same win rate against GNU GO was achieved as with EXP-STONES and STOP at 30 seconds per game. The result of 2500 games demonstrated that the average time needed per game was 49.0 seconds—63% more than needed by our time-management strategy. In the second experiment, the time per move (425 msec) was set so that the average time per game was approximately equal to 30 seconds. In 2500 games under these conditions, OREGO could only achieve a 27.6% win rate, 11.5% less than with EXP-STONES and STOP.

V. EXPERIMENTAL RESULTS IN OTHER DOMAINS

The goal of this section is to investigate the generality of the domain-independent strategies described above: the semi-dynamic strategies OPEN and MID, and the dynamic strategies BEHIND, UNST, CLOSE, and STOP. All time-management strategies were therefore tested in Connect-4, Breakthrough, Othello, and Catch the Lion. Unless specified otherwise, each experimental run again involved 5000 games (2500 as Black and 2500 as White) against the baseline player. The results of this section have not been published before.

We used our own engine with EXP-MOVES as the baseline. EXP-MOVES was chosen because EXP-STONES is domain-specific to Go, and EXP-SIM could be problematic in domains such as Catch the Lion which do not naturally progress in every move towards the end of the game (*progression property*, [21]). ERICA-BASELINE was not used as a baseline in other domains than Go since it was proposed specifically for Go and not as a domain-independent technique. The engine uses MCTS with UCB1-TUNED [22] as selection policy and uniformly random rollouts in all conditions. The exploration factor C of UCB1-TUNED was optimized for each domain at time controls of 1 second per move and set to 1.3 in Connect-4, 0.8 in Breakthrough, 0.7 in Othello, and 0.7 in Catch the Lion. After each search, the most-sampled move at the root is played. Draws, which are possible in Connect-4 and Othello, were counted as half a win for both players.

A. Games

This section outlines the rules of the four test domains.

1) Connect-4: Connect-4 is played on a 7×6 board. At the start of the game, the board is empty. The two players alternatingly place white and black discs in one of the seven columns, always filling the lowest available space of the chosen column. Columns with six discs are full and cannot be played anymore. The game is won by the player who succeeds first at connecting four tokens of her own color either vertically, horizontally, or diagonally. If the board is filled completely without any player reaching this goal, the game ends in a draw.

2) Breakthrough: The variant of Breakthrough used in this article is played on a 6×6 board. The game was originally described as being played on a 7×7 board, but other sizes such as 8×8 are popular as well, and the 6×6 board preserves an interesting search space.

At the beginning of the game, White occupies the first two rows of the board, and Black occupies the last two rows of the board. The two players alternatingly move one of their pieces straight or diagonally forward. Two pieces cannot occupy the same square. However, players can capture the opponent's pieces by moving diagonally onto their square. The game is won by the player who succeeds first at advancing one piece to the home row of the opponent, i.e. reaching the first row as Black or reaching the last row as White.

3) Othello: Othello is played on an 8×8 board. It starts with four discs in the middle of the board, two white discs and two black discs. Each disc has a black side and a white side, with the side facing up indicating the player the disc currently belongs to. The two players alternatingly place a disc on the board, in such a way that between the newly placed disc and another disc of the moving player there is an uninterrupted horizontal, vertical or diagonal line of one or more discs of the opponent. All these discs are then turned over, changing their color to the moving player's side, and the turn goes to the other player. If there is no legal move for a player, she has to pass. If both players have to pass or if the board is filled, the game ends. The game is won by the player who owns the most discs at the end.

4) Catch the Lion: Catch the Lion is a simplified form of Shogi. It is included in this work as an example of Chess-like games, which tend to be particularly difficult for MCTS [23].

The game is played on a 3×4 board. At the beginning of the game, each player has four pieces: a Lion, a Giraffe, an Elephant, and a Chick. The Chick can move one square forward, the Giraffe can move one square in the vertical and horizontal directions, the Elephant can move one square in the diagonal directions, and the Lion can move one square in any direction. During the game, the players alternatingly move one of their pieces. Pieces of the opponent can be captured. As in Shogi, they are removed from the board, but not from the game. Instead, they switch sides, and the player who captured them can later on drop them on any square of the board instead of moving one of her pieces. If the Chick reaches the home row of the opponent, it is promoted to a Chicken, now being able to move one square in any direction except for diagonally backwards. A captured Chicken, however, is

TABLE VI
PERFORMANCE OF THE INVESTIGATED TIME-MANAGEMENT STRATEGIES IN CONNECT-4.

Player	Win rate against EXP-MOVES	95% conf. int.
EXP-MOVES with OPEN $f_{\text{opening}} = 2.25$	55.8%	54.4%–57.1%
EXP-MOVES with MID $a = 2.5$, $b = 30$, $c = 20$	57.0%	55.6%–58.4%
EXP-MOVES with BEHIND $f_{\text{behind}} = 1.5$, $v_{\text{behind}} = 0.6$	55.8%	54.4%–57.1%
EXP-MOVES with BEHIND-L $f_{\text{behind}} = 0.75$, $v_{\text{behind}} = 0.6$, $l_{\text{behind}} = 3$	57.0%	55.6%–58.4%
EXP-MOVES with UNST $f_{\text{unstable}} = 0.75$	54.9%	53.6%–56.3%
EXP-MOVES with UNST-L $f_{\text{unstable}} = 1.0$, $l_{\text{unstable}} = 2$	55.8%	54.4%–57.2%
EXP-MOVES with CLOSE $f_{\text{close}} = 1.5$, $d_{\text{close}} = 0.8$	58.7%	57.3%–60.0%
EXP-MOVES with CLOSE-L $f_{\text{close}} = 0.75$, $d_{\text{close}} = 0.8$, $l_{\text{close}} = 3$	59.7%	58.3%–61.1%
EXP-MOVES with STOP_A	50.8%	49.4%–52.2%
EXP-MOVES with STOP_B $f_{\text{earlystop}} = 5$	63.0%	61.6%–64.3%
EXP-MOVES with STOP $f_{\textbf{earlystop}} = 5$, $p_{\textbf{earlystop}} = 0.9$	**65.0%**	**63.7%–66.3%**

TABLE VII
PERFORMANCE OF THE INVESTIGATED TIME-MANAGEMENT STRATEGIES IN BREAKTHROUGH.

Player	Win rate against EXP-MOVES	95% conf. int.
EXP-MOVES with OPEN $f_{\text{opening}} = 1$	50.6%	49.2%–52.0%
EXP-MOVES with MID $a = 1.5$, $b = 50$, $c = 20$	49.7%	48.3%–51.1%
EXP-MOVES with BEHIND $f_{\text{behind}} = 1$, $v_{\text{behind}} = 0.5$	50.2%	48.8%–51.6%
EXP-MOVES with BEHIND-L $f_{\text{behind}} = 0.25$, $v_{\text{behind}} = 0.5$, $l_{\text{behind}} = 3$	50.9%	49.5%–52.3%
EXP-MOVES with UNST $f_{\text{unstable}} = 0.75$	54.2%	52.8%–55.6%
EXP-MOVES with UNST-L $f_{\text{unstable}} = 1.0$, $l_{\text{unstable}} = 3$	55.2%	53.8%–56.6%
EXP-MOVES with CLOSE $f_{\text{close}} = 0.5$, $d_{\text{close}} = 0.3$	53.1%	51.7%–54.5%
EXP-MOVES with CLOSE-L $f_{\text{close}} = 0.5$, $d_{\text{close}} = 0.3$, $l_{\text{close}} = 2$	53.9%	52.5%–55.3%
EXP-MOVES with STOP_A	54.5%	53.1%–55.9%
EXP-MOVES with STOP_B $f_{\text{earlystop}} = 1.25$	56.9%	55.5%–58.2%
EXP-MOVES with STOP $f_{\textbf{earlystop}} = 1.67$, $p_{\textbf{earlystop}} = 0.3$	**58.2%**	**56.8%–59.5%**

demoted to a Chick again when dropped. The game is won by either capturing the opponent's Lion, or moving your own Lion to the home row of the opponent.

B. Connect-4

In Connect-4, a time limit of 20 seconds per player and game was used. A draw was counted as half a win for both players. The baseline player's speed was about 64500 simulations per second when searching from the initial position. Table VI shows the results of all investigated time-management strategies in Connect-4. Each strategy is listed together with the parameter setting that was found to perform best in initial systematic testing. The win rate given in the table was found by using this parameter setting in an additional 5000 games against the baseline.

As Table VI reveals, EXP-MOVES enhanced with the OPEN, MID, BEHIND, UNST, or CLOSE strategies played significantly stronger ($p<0.001$) than plain EXP-MOVES. BEHIND-L, UNST-L, and CLOSE-L could not be shown to significantly improve on BEHIND, UNST, and CLOSE, respectively. STOP improved significantly on EXP-MOVES alone ($p<0.001$) as well as STOP_B ($p<0.05$). EXP-MOVES with STOP_B was still significantly stronger than EXP-MOVES alone ($p<0.001$). As in Go, STOP_A did not have a significant effect.

C. Breakthrough

In Breakthrough, a time limit of 20 seconds per player and game was used. Searching from the initial board position, the baseline player reached about 24800 simulations per second. Table VII displays the results of all investigated time-management strategies in Breakthrough.

As Table VII shows, EXP-MOVES enhanced with the UNST or CLOSE strategies played significantly stronger ($p<0.001$ and $p<0.05$, respectively) than regular EXP-MOVES. Neither OPEN nor MID or BEHIND had a significant effect. BEHIND-L, UNST-L, and CLOSE-L could also not be shown to significantly improve on BEHIND, UNST, and CLOSE. STOP played significantly stronger than the baseline ($p<0.001$). It could not be shown to improve on STOP_B however. STOP ($p<0.001$) as well as STOP_B ($p<0.05$) improved significantly on STOP_A. In contrast to Go and Connect-4, STOP_A already played significantly stronger than the baseline ($p<0.001$).

D. Othello

In Othello, a time limit of 30 seconds per player and game was used. The longer time setting partly compensates for the longer average game length of Othello compared to Connect-4, Breakthrough, and Catch the Lion. The baseline player's speed was about 4400 simulations per second during a search from the initial position. Table VIII shows the results of all investigated time-management strategies in Othello.

Table VIII reveals that EXP-MOVES enhanced with the UNST or MID strategies played significantly stronger ($p<0.05$) than the EXP-MOVES baseline. Neither OPEN nor CLOSE or BEHIND had a significant effect. BEHIND-L, UNST-L, and CLOSE-L could again not be shown to significantly improve on BEHIND, UNST, and CLOSE. STOP played significantly stronger than the baseline ($p<0.05$). Similar to Breakthrough, this was already true for STOP_A. The

TABLE VIII
PERFORMANCE OF THE INVESTIGATED TIME-MANAGEMENT STRATEGIES IN OTHELLO.

Player	Win rate against EXP-MOVES	95% conf. int.
EXP-MOVES with OPEN $f_{\text{opening}} = 1$	50.1%	48.7%–51.5%
EXP-MOVES with MID $a = 2.5$, $b = 50$, $c = 10$	53.2%	51.8%–54.6%
EXP-MOVES with BEHIND $f_{\text{behind}} = 0.5$, $v_{\text{behind}} = 0.5$	49.2%	47.9%–50.6%
EXP-MOVES with BEHIND-L $f_{\text{behind}} = 0.5$, $v_{\text{behind}} = 0.4$, $l_{\text{behind}} = 4$	51.3%	49.9%–52.7%
EXP-MOVES with UNST $f_{\text{unstable}} = 0.5$	53.1%	51.7%–54.5%
EXP-MOVES with UNST-L $f_{\text{unstable}} = 0.5$, $l_{\text{unstable}} = 4$	52.2%	50.8%–53.6%
EXP-MOVES with CLOSE $f_{\text{close}} = 1.0$, $d_{\text{close}} = 0.2$	50.0%	48.6%–51.4%
EXP-MOVES with CLOSE-L $f_{\text{close}} = 0.25$, $d_{\text{close}} = 0.5$, $l_{\text{close}} = 4$	51.4%	50.0%–52.8%
EXP-MOVES with STOP_A	52.9%	51.5%–54.3%
EXP-MOVES with STOP_B $f_{\text{earlystop}} = 1.25$	54.2%	52.8%–55.6%
EXP-MOVES with STOP $f_{\textbf{earlystop}} = 1.25$, $p_{\textbf{earlystop}} = 0.9$	**54.8%**	**53.4%–56.2%**

TABLE IX
PERFORMANCE OF THE INVESTIGATED TIME-MANAGEMENT STRATEGIES IN CATCH THE LION.

Player	Win rate against EXP-MOVES	95% conf. int.
EXP-MOVES with OPEN $f_{\text{opening}} = 1.5$	50.7%	49.3%–52.1%
EXP-MOVES with MID $a = 1.0$, $b = 20$, $c = 10$	51.2%	49.8%–52.6%
EXP-MOVES with BEHIND $f_{\text{behind}} = 0.75$, $v_{\text{behind}} = 0.4$	51.6%	50.2%–53.0%
EXP-MOVES with BEHIND-L $f_{\text{behind}} = 0.25$, $v_{\text{behind}} = 0.3$, $l_{\text{behind}} = 3$	50.5%	49.0%–51.9%
EXP-MOVES with UNST $f_{\text{unstable}} = 0.5$	56.4%	55.0%–57.8%
EXP-MOVES with UNST-L $f_{\text{unstable}} = 0.75$, $l_{\text{unstable}} = 2$	56.0%	54.6%–57.4%
EXP-MOVES with CLOSE $f_{\text{close}} = 1.0$, $d_{\text{close}} = 0.7$	57.4%	56.0%–58.8%
EXP-MOVES with CLOSE-L $f_{\text{close}} = 0.5$, $d_{\text{close}} = 0.7$, $l_{\text{close}} = 3$	55.9%	54.5%–57.2%
EXP-MOVES with STOP_A	50.0%	48.6%–51.4%
EXP-MOVES with STOP_B $f_{\text{earlystop}} = 1.67$	56.9%	55.5%–58.2%
EXP-MOVES with STOP $f_{\textbf{earlystop}} = 5$, $p_{\textbf{earlystop}} = 0.2$	**60.4%**	**59.1%–61.8%**

more general variants of STOP could not be demonstrated to significantly improve on STOP_A in Othello.

E. Catch the Lion

In Catch the Lion, a time limit of 20 seconds per player and game was used. From the initial board, the baseline had a speed of about 18500 simulations per second. Table IX shows the results of all investigated time-management strategies in Catch the Lion.

As Table IX shows, EXP-MOVES enhanced with the UNST or CLOSE strategies played significantly stronger ($p<0.001$) than regular EXP-MOVES. Neither OPEN, MID, nor BEHIND had a significant effect. BEHIND-L, UNST-L, and CLOSE-L could not be shown to significantly improve on their simpler versions BEHIND, UNST, and CLOSE. STOP was significantly stronger ($p<0.001$) than both the EXP-MOVES baseline and STOP_B. STOP_B improved on STOP_A ($p<0.001$). As in Connect-4, STOP_A did not have a significant advantage over the baseline.

F. Discussion of the Results

In the previous subsections, experimental data were ordered by domain. This subsection summarizes and discusses the results ordered by the type of time-management technique instead, in order to allow for comparisons across different domains. Tables X and XI give an overview by recapitulating which strategies were shown to result in a significant increase of playing strength in which game. Table X illustrates the improvements of *simple strategies*—the ones not involving repeated checks in loops, such as UNST—compared to the baseline player. Table XI shows the improvements of the *loop strategies* compared to their simple counterparts, such as UNST-L to UNST. In the following, these results are discussed ordered by time-management strategy.

TABLE X
TIME MANAGEMENT SUMMARY – SIMPLE STRATEGIES. CHECKMARKS IDENTIFY STRATEGIES THAT WERE SHOWN TO SIGNIFICANTLY IMPROVE ON THEIR RESPECTIVE BASELINE (EXP-MOVES IN THE COLUMN GAME). STRATEGIES THAT SHOWED NO SIGNIFICANT IMPROVEMENT ARE LEFT BLANK.

EXP-MOVES with	Connect-4	Breakthrough	Othello	Catch the Lion	13×13 Go
OPEN	✓				✓
MID	✓		✓		✓
BEHIND	✓				✓
UNST	✓	✓	✓	✓	✓
CLOSE	✓	✓		✓	✓
STOP_A		✓	✓		
STOP_B	✓	✓	✓	✓	✓
STOP	✓	✓	✓	✓	✓

OPEN and MID. Some domains, e.g., Connect-4, Othello, and Go, profit from shifting available time from the endgame to the midgame or early game. Intuitively, this is the case in games that allow players to build up a positional advantage, essentially deciding the game result many moves before an actual terminal state is reached. Less effort is therefore required in the endgame for the leading player to keep the lead and execute the win. Search effort is also largely futile for a player who has fallen behind in the endgame, and is more effectively spent in the early or midgame to avoid falling behind in the first place.

TABLE XI
TIME MANAGEMENT SUMMARY – LOOP STRATEGIES. CHECKMARKS IDENTIFY LOOP STRATEGIES THAT WERE SHOWN TO SIGNIFICANTLY IMPROVE ON THEIR RESPECTIVE SIMPLE VERSIONS (E.G. CLOSE-L ON CLOSE). STRATEGIES THAT SHOWED NO SIGNIFICANT IMPROVEMENT ARE LEFT BLANK.

EXP-MOVES with	Connect-4	Breakthrough	Othello	Catch the Lion	13×13 Go
BEHIND-L					
UNST-L					✓
CLOSE-L					✓

Other games, for instance Catch the Lion and Breakthrough, are more tactical in nature and require a higher search effort even in the endgame. One of the reasons is that with sufficient effort, it can be possible throughout the entire game for both players to lure their opponent into *traps* [23]. The player falling behind can therefore still make effective use of additional time in the endgame, while the leading player still needs to spend time to avoid losing her positional gains in this way. See [24] for a discussion of tacticality and traps and a more in-depth comparison of the test domains in this respect.

The following experiment was conducted in order to compare the test domains with respect to the usefulness of spending search time in the endgame. Player A used 1 second thinking time per move. Player B used 1 second until 10 turns before the average game length of the respective domain, and then switched to 100ms per move. 1000 games were played in each domain, with players A and B both moving first in half of the games. In this setting, player B won 49.1% (95% confidence interval: $46.0\% - 52.3\%$) of games in Connect-4 and 49.0% ($45.9\% - 52.2\%$) in Othello—the loss of time in the endgame could not be shown to significantly weaken the player. In Breakthrough however, player B won only 36.6% ($33.6\% - 39.7\%$) of games, and in Catch the Lion only 31.4% ($28.6\% - 34.4\%$). This explains why shifting large amounts of time from the endgame to the opening or midgame is not effective in these domains.

In Go, switching from 1000ms to 100ms in the described way does result in decreased performance as well (38.4% win rate for player B). Even though endgame time does not seem to be wasted time in Go however, moving a part of it towards the opening or midgame is still effective. Note that OREGO resigns whenever its win rate at the root falls below 10%, which cuts off most of the late endgame of Go as human players typically would. This feature is meant to restrict the game to the moves that actually matter. Deactivating resigning makes games much longer on average (231 moves instead of 118 in self-play at 1 second per move), and creates even more opportunity to shift time away from the endgame. Furthermore, it is interesting to note that the optimal parameter settings of the MID strategy in Connect-4 largely turn it into a variant of OPEN by shifting time far to the beginning of the game. In for example Othello and 13×13 Go this is not the case. Figures 1, 2, and 3 show the average time distribution over a game of Connect-4, Othello, and Go, respectively, using the optimized settings of the MID strategy in each game. The figures demonstrate how the peak of the average time spent by MID per move appears at a later stage of the game in Othello and Go than in Connect-4. This is probably explained by the fact that Othello and Go games take much longer than Connect-4 games on average (compare Table XIV). In Othello and Go it is therefore prudent not to spend too much time on the first moves of a game, as potential consequences of actions are not yet visible to MCTS. Connect-4 however requires more search effort in the opening, as games are often decided early. One could say that opening and midgame fall into one phase in a short game such as Connect-4.

In conclusion, OPEN and MID can be useful not only to improve performance in a given game, but also to gain insight into the relative importance of early game, midgame and endgame decisions in the game at hand. Note that this importance is always dependent on the search engine and search timing used—an engine with opening or endgame databases for example might optimally distribute search time differently, and the use of long search times can make the consequences of moves visible somewhat earlier in the game compared to short search times.

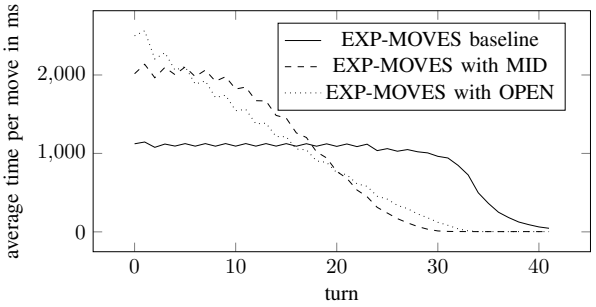


Fig. 1. Average time distribution over a game of Connect-4 with the MID and OPEN strategies.

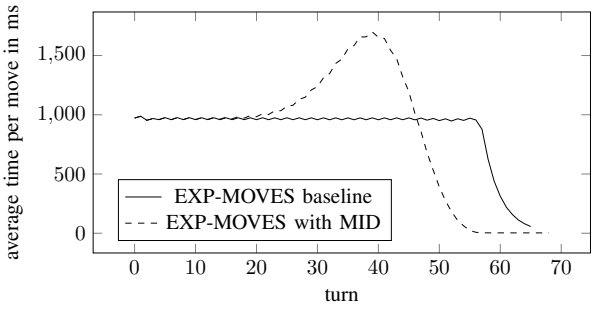


Fig. 2. Average time distribution over a game of Othello with the MID strategy.

BEHIND. Among the tested domains, the BEHIND strategy is improving performance in Connect-4 and 13×13 Go. Looking at the optimal parameter values for these two games however, we can see that search times are prolonged whenever the player's win rate falls below 0.6. This means they are essentially always prolonged in the opening and midgame,

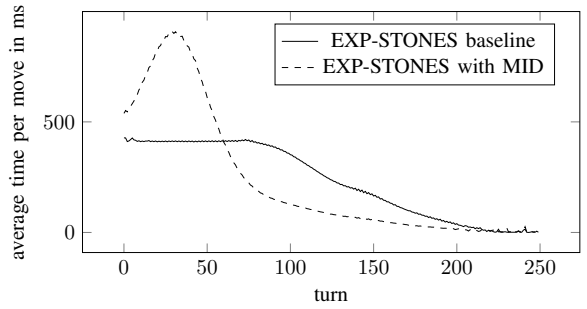


Fig. 3. Average time distribution over a game of 13×13 Go with the MID strategy.

when win rates tend to be close to 0.5. The strategy largely turns from a method to come back from a disadvantageous position into a method to shift more search time to the earlier phase of the game. For Connect-4, Figure 4 shows the similar effects of OPEN and BEHIND at optimal parameter settings when considering the average time distribution over the turns of the game. Since we already know the OPEN strategy to be effective in Connect-4, and OPEN and BEHIND are equally strong according to Table VI, it is an interesting question whether BEHIND provides any additional positive effect beyond this similar time shift.

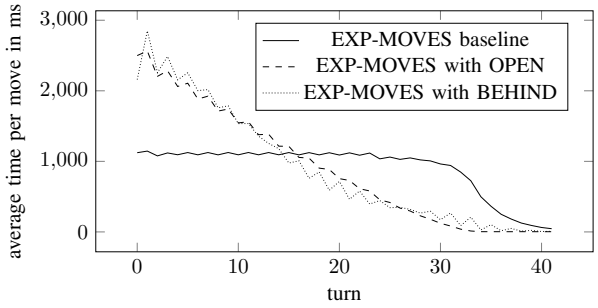


Fig. 4. Average time distribution over a game of Connect-4 with the OPEN and BEHIND strategies.

In order to investigate this question, we determined the average time spent on each turn of the game when using the BEHIND strategy—as presented for Connect-4 in Figure 4—and created a regular MCTS player BEHIND-DISTRIBUTION which directly uses these search times depending on the current turn. Thus, it achieves the same time distribution over the game as BEHIND, while not using the BEHIND method itself. As an example, if BEHIND on average uses 673ms in turn 17 of a game, this is a result of win rates being compared to the threshold parameter v_{behind} and some proportion of searches being prolonged by a factor of f_{behind} in all turns up to 17. BEHIND-DISTRIBUTION just uses 673ms every time it reaches turn 17 in this example, regardless of the win rates observed. By comparing the playing strength of BEHIND-DISTRIBUTION to that of BEHIND we can determine whether the BEHIND method has a significant effect in addition to the time distribution over turns that it causes. BEHIND-DISTRIBUTION was also implemented for 13×13 Go. Since the baseline in Go was EXP-STONES instead of EXP-MOVES, it used the same search times as BEHIND depending on the current number of stones on the board, not depending on the current turn. The results are comparable since the turn and the number of stones on the board correlate strongly.

TABLE XII
PERFORMANCE OF THE DISTRIBUTION PLAYERS IN CONNECT-4, BREAKTHROUGH, OTHELLO, AND CATCH THE LION. 5000 GAMES PER PLAYER WERE PLAYED AGAINST THE EXP-MOVES BASELINE.

Player	Win rate	95% conf. int.	Derived from	Win rate
In Connect-4:				
BEHIND-DISTR.	55.3%	53.9%-56.7%	BEHIND	55.8%
UNST-DISTR.	50.8%	49.4%-52.2%	UNST	54.9%
CLOSE-DISTR.	54.9%	53.5%-56.3%	CLOSE	58.7%
STOP_A-DISTR.	46.1%	44.7%-47.5%	STOP_A	50.8%
STOP_B-DISTR.	52.9%	51.5%-54.3%	STOP_B	63.0%
STOP-DISTR.	55.1%	53.7%-56.5%	STOP	65.0%
In Breakthrough:				
UNST-DISTR.	48.7%	47.3%-50.1%	UNST	54.2%
CLOSE-DISTR.	49.6%	48.2%-51.0%	CLOSE	53.1%
STOP_A-DISTR.	47.6%	46.2%-49.0%	STOP_A	54.5%
STOP_B-DISTR.	47.3%	45.9%-48.7%	STOP_B	56.9%
STOP-DISTR.	47.2%	45.8%-48.6%	STOP	58.2%
In Othello:				
UNST-DISTR.	48.9%	47.5%-50.3%	UNST	53.1%
CLOSE-DISTR.	49.3%	47.9%-50.7%	CLOSE	50.0%
STOP_A-DISTR.	49.1%	47.7%-50.5%	STOP_A	52.9%
STOP_B-DISTR.	49.6%	48.2%-51.0%	STOP_B	54.2%
STOP-DISTR.	50.4%	49.0%-51.8%	STOP	54.8%
In Catch the Lion:				
UNST-DISTR.	51.3%	49.9%-52.7%	UNST	56.4%
CLOSE-DISTR.	50.1%	48.7%-51.5%	CLOSE	57.4%
STOP_A-DISTR.	47.7%	46.3%-49.1%	STOP_A	50.0%
STOP_B-DISTR.	51.8%	50.4%-53.2%	STOP_B	56.9%
STOP-DISTR.	49.3%	47.9%-50.7%	STOP	60.4%

TABLE XIII
PERFORMANCE OF THE DISTRIBUTION PLAYERS IN 13×13 GO. 5000 GAMES PER PLAYER WERE PLAYED AGAINST GNU GO.

Player	Win rate	95% conf. int.	Derived from	Win rate
BEHIND-DISTR.	31.3%	30.0%-32.6%	BEHIND	29.9%
UNST-DISTR.	32.6%	31.3%-33.9%	UNST	33.6%
CLOSE-DISTR.	31.0%	29.7%-32.3%	CLOSE	32.6%
STOP_A-DISTR.	18.7%	17.6%-19.8%	STOP_A	25.3%
STOP_B-DISTR.	29.8%	28.5%-31.1%	STOP_B	36.7%
STOP-DISTR.	32.8%	31.5%-34.1%	STOP	39.1%

As Table XII shows, the win rate of BEHIND-DISTRIBUTION in Connect-4 is not significantly different from that of BEHIND. The same is true in Go, as indicated in Table XIII. BEHIND only appears to improve playing strength in Connect-4 and 13×13 Go due to its time shift to the opening phase. It can be concluded that the BEHIND strategy is not effective in any of the tested domains except for Connect-4 and 13×13 Go, where it can be replaced by a strategy directly manipulating the time distribution over turns (or stones) such as OPEN.

UNST. The UNST strategy is significantly stronger than the baseline in all tested domains. Figure 5 demonstrates that in Go, UNST at optimal parameter settings has the effect of shifting time to the opening, similar to BEHIND. Figure 6 however shows that this is not the case for UNST at optimal

parameter settings in Connect-4. The time distribution of UNST appears nearly identical to that of the EXP-MOVES baseline. In order to test for an effect of the UNST method beyond potential influences on the time distribution over the turns of the game, we constructed a UNST-DISTRIBUTION player for each test domain according to the procedure described for BEHIND-DISTRIBUTION above. The Go player was again based on the time distribution over the number of stones on the board.

Tables XII and XIII show that UNST-DISTRIBUTION is not significantly better than the baseline in any domain but Go, where it is not significantly different from UNST. We can therefore conclude that UNST is useful in Connect-4, Breakthrough, Othello, and Catch the Lion, independently of the time distribution over turns that results from it. In 13×13 Go however, the success of UNST largely depends on a time shift to the opening, similar to BEHIND.

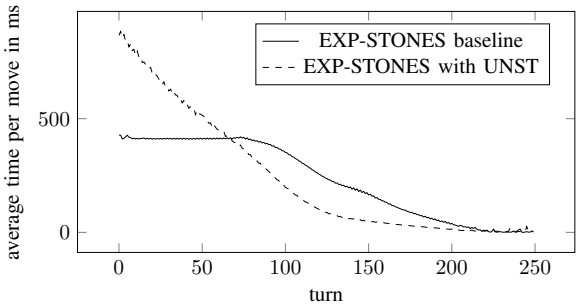


Fig. 5. Average time distribution over a game of 13×13 Go with the UNST strategy.

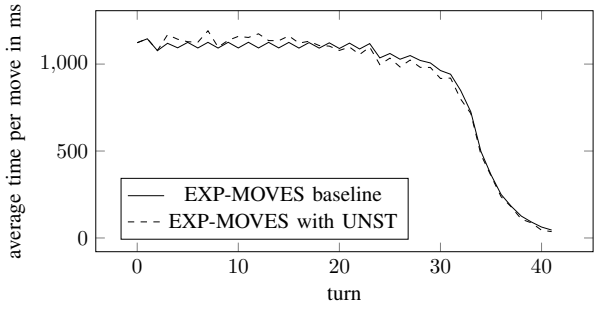


Fig. 6. Average time distribution over a game of Connect-4 with the UNST strategy.

CLOSE. The CLOSE strategy is improving the performance of the baseline in all tested domains except for Othello. As for BEHIND and UNST, a CLOSE-DISTRIBUTION player was created for each domain in order to test the influence of a shifted time distribution over turns of the game that is caused by applying CLOSE. Tables XII and XIII demonstrate that in Breakthrough and Catch the Lion, CLOSE-DISTRIBUTION does not achieve win rates significantly higher than 50%, meaning that CLOSE is playing stronger independently of the time distribution over the game that it causes. In Connect-4, CLOSE-DISTRIBUTION does perform better than EXP-MOVES—but still significantly worse than CLOSE itself, which means that the CLOSE strategy has a positive effect in addition to the time shift. In Go, this additional effect could not be shown to be statistically significant.

It remains to be shown why CLOSE does not perform well in Othello. It could be the case that there are too many situations in which two moves have relatively high and close visit counts, but investing additional search effort does not lead to a better move decision—either because the moves are equally valid, or because the consequences of the two moves cannot be reliably distinguished even with the help of a limited additional amount of time.

STOP. The STOP strategy—as well as the special case STOP_B—significantly improves on the baseline in all tested domains. Furthermore, STOP consistently achieved the highest win rate of all investigated time-management techniques across the five domains and is thus the most successful strategy tested in this article.

We determined the average percentage of the regular search time that was saved per move when STOP, STOP_A or STOP_B were active. For STOP_A/STOP_B, this percentage was 31.1%/32.1% in Connect-4, 24.7%/25.0% in Breakthrough, 23.1%/23.1% in Othello, and 32.4%/33.7% in Catch the Lion. The time shift used by STOP_B thus does not change these values significantly. Note that more than 50% cannot be saved with STOP_B, because the rest of the search time would then still be long enough to turn a completely unsampled move into the most-sampled one. In the most general STOP variant, more than 50% can theoretically be saved, as it gives up the guarantee of never stopping a search whose outcome could still change. The saved percentages of the search time for STOP were 34.5% in Connect-4, 47.7% in Breakthrough, 25.2% in Othello, and 68.5% in Catch the Lion. The increase in saved time is related to the parameter $p_\text{earlystop}$—domains with high optimal $p_\text{earlystop}$ (0.9 in Connect-4 and Othello) only relax the guarantee to a small degree, domains with low optimal $p_\text{earlystop}$ (0.3 in Breakthrough and 0.2 in Catch the Lion) relax it further.

For all STOP variants, the performance of a player imitating their time distribution over the game was tested as well (named STOP_A-DISTRIBUTION etc.). According to Tables XII and XIII, the distribution over turns (or stones) of STOP and STOP_B alone has no significant positive effect on playing strength in all games except for Connect-4 and Go. In Connect-4 and Go, STOP-DISTRIBUTION does have a positive effect, but is still significantly weaker than STOP. The same holds for STOP_B-DISTRIBUTION. In conclusion, STOP and STOP_B are effective in all tested domains, independently of (or in addition to) their time distribution over turns.

The basic strategy STOP_A works in some domains (Breakthrough, Othello) but not in others (Connect-4, Catch the Lion, Go). As the success of STOP_B shows—a strategy that works just like STOP_A but simultaneously shifts time to the opening of the game—this is largely due to STOP_A's adverse effect of shifting time to the endgame. The fact that such time shifts can potentially occur with any given strategy makes studying and comparing these two STOP variants worthwhile. As illustrative examples, see Figures 7 and 8

for a visualization of the effects of STOP_A and STOP_B in Connect-4 and Go, respectively. The time distribution of the most general STOP variant—not shown here for clarity—looks very similar to the STOP_B distribution in both games. An additional argument for the hypothesis that a time shift to the endgame hurts the performance of STOP_A comes from the performance of STOP_A-DISTRIBUTION (see Tables XII and XIII). In all tested domains except for Othello, also representing the only domain where STOP_B does not improve on STOP_A, the time distribution over turns (or stones) of STOP_A hurts performance significantly.

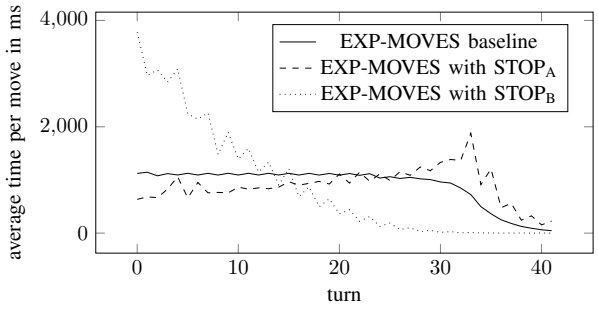


Fig. 7. Average time distribution over a game of Connect-4 with the STOP_A and STOP_B strategies. While STOP_A shifts time to the endgame when compared to the baseline, STOP_B shifts time to the opening phase of the game.

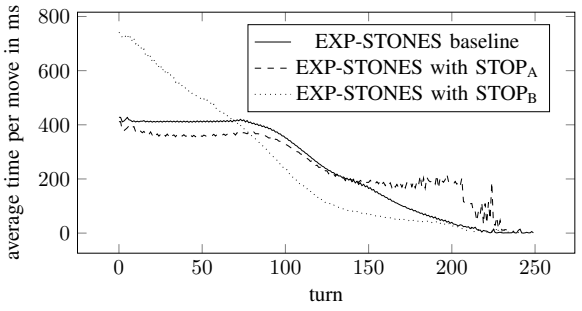


Fig. 8. Average time distribution over a game of 13×13 Go with the STOP_A and STOP_B strategies.

BEHIND-L, UNST-L, and CLOSE-L. As Table XI shows, the repeated check for the termination conditions in UNST-L and CLOSE-L only resulted in significantly stronger play in the domain of Go. None of the other games profited from it. BEHIND-L was not significantly more effective than BEHIND in any tested domain. (Note that UNST, CLOSE, and BEHIND are special cases of UNST-L, CLOSE-L, and BEHIND-L, respectively. Therefore, the looping versions cannot perform worse than the simple versions if tuned optimally.) It is possible that in the set of domains used, only Go is sufficiently complex with regard to branching factor and game length to make such fine-grained timing decisions worthwhile. See Table XIV for a comparison of average game lengths and average branching factors of all five domains investigated. All values are averages from 1000 self-play games of regular MCTS, 1 second per move. In Go, we took the average against GNU GO at 30 seconds per game.

TABLE XIV
GAME LENGTH AND BRANCHING FACTOR IN CONNECT-4, BREAKTHROUGH, OTHELLO, CATCH THE LION, AND 13×13 GO.

	Connect-4	Breakthrough	Othello	Catch the Lion	13×13 Go
Game length	37	29	61	35	150
Branching factor	5.8	15.5	8	10.5	90

All time-management strategies that *prolong* search times when certain criteria are met, such as BEHIND, UNST, and CLOSE, take available time from the later phases and shift it to the earlier phases of the game. Strategies that *shorten* search times based on certain criteria, such as STOP_A, move time from the opening towards the endgame instead. When analyzing the effect of time-management approaches, it is therefore worth testing whether these shifts have a positive or negative effect. Should the effect be negative, STOP_B provides an example of how to possibly counteract it by introducing an explicit shift in the opposite direction.

VI. CONCLUSION AND FUTURE RESEARCH

In this article, we investigated a variety of time-management strategies for Monte Carlo Tree Search, using the games of Go, Connect-4, Breakthrough, Othello, and Catch the Lion as a testbed. This included newly proposed strategies (called OPEN, MID, KAPPA-EXP, KAPPA-LM, and KAPPA-CM) as well as strategies described in [7] (UNST and BEHIND) or independently proposed in [6] (CLOSE and STOP_A), partly in enhanced form (UNST-L, BEHIND-L, CLOSE-L, STOP_B, and STOP). Empirical results show that the proposed strategy EXP-STONES with STOP provides a significant improvement over the state of the art as represented by ERICA-BASELINE in 13×13 Go. For sudden-death time controls of 30 seconds per game, EXP-STONES with STOP increased OREGO's win rate against GNU GO from 25.5% (using a simple baseline) or from 35.3% (using the state-of-the-art ERICA-BASELINE) to 39.1%. In self-play, this strategy won approximately 60% of games against ERICA-BASELINE, both in 13×13 and 19×19 Go under various time controls.

Furthermore, comparison across different games shows that the domain-independent strategy STOP is the strongest of all tested time-management strategies. It won 65.0% of self-play games in Connect-4, 58.2% in Breakthrough, 54.8% in Othello, and 60.4% in Catch the Lion. With the exception of CLOSE in Othello, UNST and CLOSE also prove effective in all domains. Since many time-management strategies result in a shift of available time towards either the opening or the endgame, a methodology was developed to isolate the effect of this shift and judge the effect of a given strategy independently of it.

The following directions appear promising for future research. First, a natural next step is the combined testing and optimization of all above strategies—in order to determine to which degree their positive effects on playing strength can

complement each other, or to which degree they could be redundant (such as OPEN and BEHIND in Connect-4), or possibly interfere. ERICA-BASELINE demonstrates that some combinations can be effective at least in Go. Second, a non-linear classifier like a neural network could be trained to decide about continuing or aborting the search in short intervals, using all relevant information used by above strategies as input. A third direction is the development of improved strategies to measure the complexity and importance of a position and thus to effectively use time where it is most needed. In Go for example, counting the number of independent fights on the board could be one possible, domain-dependent approach. Furthermore, possible interactions of time management strategies with other MCTS enhancements could be studied, such as for instance the sufficiency-based selection strategy by Gudmundsson and Björnsson [25].

ACKNOWLEDGMENT

This work is funded by the Netherlands Organisation for Scientific Research (NWO) in the framework of the project Go4Nature, grant number 612.000.938. The authors would like to thank the reviewers for their valuable comments that helped improve this article.

REFERENCES

[1] I. Althöfer, C. Donninger, U. Lorenz, and V. Rottmann, "On Timing, Permanent Brain and Human Intervention," in *Advances in Computer Chess*, H. J. van den Herik, I. S. Herschberg, and J. W. H. M. Uiterwijk, Eds. Maastricht: University of Limburg, 1994, vol. 7, pp. 285–297.

[2] C. Donninger, "A la recherche du temps perdu: 'That was easy'," *ICCA Journal*, vol. 17, no. 1, pp. 31–35, 1994.

[3] R. M. Hyatt, "Using Time Wisely," *ICCA Journal*, vol. 7, no. 1, pp. 4–9, 1984.

[4] S. Markovitch and Y. Sella, "Learning of Resource Allocation Strategies for Game Playing," *Computational Intelligence*, vol. 12, no. 1, pp. 88–105, 1996.

[5] R. Šolak and V. Vučković, "Time Management during a Chess Game," *ICGA Journal*, vol. 32, no. 4, pp. 206–220, 2009.

[6] P. Baudiš, "MCTS with Information Sharing," Master's thesis, Department of Theoretical Computer Science and Mathematical Logic, Charles University, Prague, Czech Republic, 2011.

[7] S.-C. Huang, R. Coulom, and S.-S. Lin, "Time Management for Monte-Carlo Tree Search Applied to the Game of Go," in *International Conference on Technologies and Applications of Artificial Intelligence*. IEEE Computer Society, 2010, pp. 462–466.

[8] H. Baier and M. H. M. Winands, "Time Management for Monte-Carlo Tree Search in Go," in *13th International Conference on Advances in Computer Games (ACG 2011)*, ser. Lecture Notes in Computer Science, H. J. van den Herik and A. Plaat, Eds., vol. 7168, 2012, pp. 39–51.

[9] L. Kocsis, J. W. H. M. Uiterwijk, and H. J. van den Herik, "Learning Time Allocation Using Neural Networks," in *Second International Conference on Computers and Games (CG 2000)*, ser. Lecture Notes in Computer Science, T. A. Marsland and I. Frank, Eds., vol. 2063. Springer, 2001, pp. 170–185.

[10] J. Huang, Z. Liu, L. Benjie, and X. Feng, "Pruning in UCT Algorithm," in *2010 International Conference on Technologies and Applications of Artificial Intelligence*, 2010, pp. 177–181.

[11] R. Coulom, "Criticality: a Monte-Carlo Heuristic for Go Programs. Invited talk, University of Electro-Communications, Tokyo, Japan." 2009.

[12] S. Pellegrino, A. Hubbard, J. Galbraith, P. Drake, and Y.-P. Chen, "Localizing Search in Monte-Carlo Go Using Statistical Covariance," *ICGA Journal*, vol. 32, no. 3, pp. 154–160, 2009.

[13] J. Cohen, "A Coefficient of Agreement for Nominal Scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[14] P. Drake *et al.*, "Orego Go Program," 2011, Available online: http://legacy.lclark.edu/~drake/Orego.html.

[15] B. E. Childs, J. H. Brodeur, and L. Kocsis, "Transpositions and Move Groups in Monte Carlo Tree Search," in *2008 IEEE Symposium on Computational Intelligence and Games (CIG 2008)*, P. Hingston and L. Barone, Eds., 2008, pp. 389–395.

[16] R. Greenblatt, D. Eastlake III, and S. D. Crocker, "The Greenblatt Chess Program," in *Proceedings of the Fall Joint Computer Conference*, 1967, pp. 801–810.

[17] S. Gelly and D. Silver, "Combining online and offline knowledge in UCT," in *24th International Conference on Machine Learning (ICML 2007)*, ser. ACM International Conference Proceeding Series, Z. Ghahramani, Ed., vol. 227. ACM, 2007, pp. 273–280.

[18] S. Gelly, Y. Wang, R. Munos, and O. Teytaud, "Modification of UCT with Patterns in Monte-Carlo Go," HAL - CCSd - CNRS, Tech. Rep., 2006.

[19] H. Baier and P. Drake, "The Power of Forgetting: Improving the Last-Good-Reply Policy in Monte Carlo Go," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 4, pp. 303–309, 2010.

[20] Free Software Foundation, "GNU Go 3.8," 2009, Available online: http://www.gnu.org/software/gnugo/.

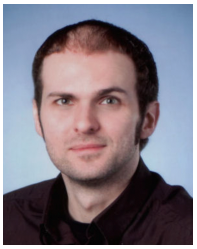
[21] H. Finnsson and Y. Björnsson, "Game-Tree Properties and MCTS Performance," in *IJCAI 2011 Workshop on General Intelligence in Game Playing Agents, GIGA'11*, 2011, pp. 23–30.

[22] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-Time Analysis of the Multiarmed Bandit Problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[23] R. Ramanujan, A. Sabharwal, and B. Selman, "On Adversarial Search Spaces and Sampling-Based Planning," in *20th International Conference on Automated Planning and Scheduling (ICAPS 2010)*, R. I. Brafman, H. Geffner, J. Hoffmann, and H. A. Kautz, Eds., 2010, pp. 242–245.

[24] H. Baier and M. H. M. Winands, "Monte-Carlo Tree Search and Minimax Hybrids," in *2013 IEEE Conference on Computational Intelligence and Games (CIG 2013)*, 2013, pp. 129–136.

[25] S. F. Gudmundsson and Y. Björnsson, "Sufficiency-Based Selection Strategy for MCTS," in *23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, F. Rossi, Ed., 2013, pp. 559–565.



Hendrik Baier holds a B.Sc. in Computer Science (Darmstadt Technical University, Darmstadt, Germany, 2006), an M.Sc. in Cognitive Science (Osnabrück University, Osnabrück, Germany, 2010), and is currently completing a Ph.D. in Artificial Intelligence (Maastricht University, Maastricht, The Netherlands).

Mark Winands received a 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 ICGA Journal and as an associate editor of IEEE Transactions on Computational Intelligence and AI in Games.