ME-MCTS: Online Generalization by Combining Multiple Value Estimators

Hendrik Baier , **Michael Kaisers**
Centrum Wiskunde & Informatica, Amsterdam
{hendrik.baier, michael.kaisers}@cwi.nl

Abstract

This paper addresses the challenge of online generalization in tree search. We propose *Multiple Estimator Monte Carlo Tree Search* (ME-MCTS), with a two-fold contribution: first, we introduce a formalization of online generalization that can represent existing techniques such as "history heuristics", "RAVE", or "OMA" – contextual action value estimators or *abstractors* that generalize across specific contexts. Second, we incorporate recent advances in estimator averaging that enable guiding search by combining the online action value estimates of any number of such abstractors or similar types of action value estimators. Unlike previous work, which usually proposed a single abstractor for either the selection or the rollout phase of MCTS simulations, our approach focuses on the combination of multiple estimators and applies them to all move choices in MCTS simulations. As the MCTS tree itself is just another value estimator – unbiased, but without abstraction – this blurs the traditional distinction between action choices inside and outside of the MCTS tree. Experiments with three abstractors in four board games show significant improvements of ME-MCTS over MCTS using only a single abstractor, both for MCTS with random rollouts as well as for MCTS with static evaluation functions. While we used deterministic, fully observable games, ME-MCTS naturally extends to more challenging settings.

1 Introduction

Sequential decision-making problems arise in a variety of domains, and require strong algorithms to find good solutions. Significant progress has been achieved in the field by studying search in games. One particularly successful approach is Monte Carlo Tree Search (MCTS), which excels in large search spaces due to its selective sampling of promising actions. If exploration and exploitation are traded off properly, MCTS has been shown to converge to the optimal policy in the limit [Kocsis and Szepesvári, 2006], while providing approximations at any time. Among other domains, it has been successful in General Game Playing [Finnsson, 2012], General Video Game Playing [Liebana *et al.*, 2016], and was an essential part in recent breakthroughs in deterministic board games [Silver *et al.*, 2017; Silver *et al.*, 2018].

Vanilla MCTS typically stores value estimates in tree nodes that represent states of the underlying domain. There is no sharing of information between different nodes in vanilla MCTS, and the search therefore does not exploit similarity between states. Efforts towards effective generalization have been recently largely focused on *offline* learning, such as the deep neural networks used by AlphaZero for estimating both the value of a visited state as well as the optimal policy in that state [Silver *et al.*, 2018]. Their offline training requires them to generalize across *all* states that are likely to ever be visited in any episode of the domain at hand, and therefore typically needs an extremely large number of training samples and a very powerful function approximator.

In contrast to this, our contribution addresses *online* generalization in search. The promise of online generalization is that it can specialize to the current episode, and learn from the states actually encountered during search. Previous approaches to online generalization have typically either been restricted to the *selection* phase or to the *rollout* phase of MCTS; we aim at applying generalization throughout *entire* MCTS simulations. Previous approaches have also usually been restricted to a *single* technique for value estimation using online generalization; we instead offer as our main contribution a method for combining *multiple* value estimators—called *Multiple Estimator Monte Carlo Tree Search* (ME-MCTS)—where any number of estimators with different generalization strategies can be plugged in.

The estimators we use for testing our approach here are *contextual* action value estimators, also called *history abstractors*. We provide a formalization of this class of value estimators, which includes several online generalization techniques from the literature, and demonstrate empirical performance by combining specific example estimators in deterministic benchmark games from the literature. Note however that ME-MCTS is neither restricted to history abstractors nor to deterministic games.

The remainder of this article is structured as follows: Section 2 describes our concept of contextual value estimates, and the abstractors that provide them. Section 3 explains the use of several value estimators to guide MCTS. Section

4 presents experimental results, both for single abstractors as well as combinations of multiple abstractors, and both for MCTS using rollouts as well as MCTS using evaluation functions. Section 5 relates this paper to the literature, and Section 6 concludes and outlines future work.

2 Contextual Action Value Estimates through History Abstractors

In this section we formalize the concept of history abstractors, which essentially encapsulate an abstraction function together with lookup tables for running average and uncertainty statistics, and we define the specific abstractors used in our experiments presented in Section 4.

We assume fully observable game domains throughout this article. Let the history of the game up to time t be the sequence $h_t = (s_0, a_0, s_1, a_1, \ldots, s_t) \in \mathcal{H}$, with $s_0 \in \mathcal{S}$ the starting state of the game, s_t the current state, and $a_i \in \mathcal{A}$ the actions leading from state to state. A *history abstractor* or simply *abstractor* $A = \{\mathcal{C}, f, Q, N\}$ consists of a set of contexts $\mathcal{C}$; a history abstraction function $f : \mathcal{H} \times \mathcal{A} \to \mathcal{C}$ mapping histories[1] to the contexts that the abstractor is maintaining action value estimates for; action value estimates $Q : \mathcal{C} \times \mathcal{A} \to \mathbb{R}$ for all legal actions in all contexts; and sample counts $N : \mathcal{C} \times \mathcal{A} \to \mathbb{N}$ for all legal actions in all contexts. f defines an equivalence relation on the set of histories $\mathcal{H}$, inducing equivalence classes of histories that are considered similar and generalized over by mapping them to the same context. Q and N are initially zero everywhere.

After completing a simulation (iteration) i of Monte Carlo Tree Search with cumulative return R, an abstractor A updates its statistics (sample count N_A and value estimate Q_A) for every action a_t that was preceded by history h_t in the simulation, using its history abstraction function f_A:

$$\mathbf{c} = f_A(h_t, a_t), \tag{1}$$

$$N_A^i(\mathbf{c}, a_t) = N_A^{i-1}(\mathbf{c}, a_t) + 1, \tag{2}$$

$$Q_A^i(\mathbf{c}, a_t) = Q_A^{i-1}(\mathbf{c}, a_t) + \frac{R - Q_A^{i-1}(\mathbf{c}, a_t)}{N_A^i(\mathbf{c}, a_t)}. \tag{3}$$

During future simulations, these estimates can then be used to choose actions, as outlined in Section 3.

While we are not proposing novel abstractors here, the concept of history abstractor is useful to us in order to subsume a number of online generalization approaches from prior MCTS work. It also defines a common interface that allows us to plug multiple action value estimators of this type into ME-MCTS for our experiments in Section 4.

The "minimal" abstractor uses a separate context for each history: $f_{\text{history}}(h_t, a_t) = h_t$. This is the MCTS tree itself—not generalizing, but providing unbiased value estimates. Note that searching over histories leads to actual *tree* search, while searching over states can lead to a *directed acyclic graph* (DAG), in which a given node can have more than one immediate ancestor. The corresponding abstractor to a DAG would use $f_{\text{state}}(h_t, a_t) = s_t$. For Markovian states, this abstractor A_{state} is unbiased just like A_{history} and converges at least as fast. In this paper however we use pure vanilla MCTS as baseline, without taking transpositions into account. A_{history} replaces the traditional MCTS tree here, which is why we also call it the "tree estimator" even though that tree is represented by Q and N tables just like in any other abstractor.

The "maximal" abstractor only uses a single context for all histories: $f_{\text{global}}(h_t, a_t) = \emptyset$. This corresponds to a global action value table that ignores state, and has for example been used by the *Progressive History* technique (in the selection phase only) [Nijssen and Winands, 2010], or MAST/TO-MAST (in the rollout phase only) [Finnsson and Björnsson, 2010]. While the value estimates of A_{history} only summarize returns for the same action with the same history, A_{global} summarizes returns for each action regardless of history.

Other online generalization techniques use abstractors that fall in between those extremes. The *N-Gram Selection Technique* (NST) for example learns action value estimates, which it only applies to rollouts, in three different contexts: the global context, the context of the immediately preceding action $f_{\text{2-gram}}(h_t, a_t) = a_{t-1}$, and the context of the two preceding actions $f_{\text{3-gram}}(h_t, a_t) = (a_{t-2}, a_{t-1})$ [Tak *et al.*, 2012]. The *PAST* and *FAST* techniques use contexts defined by the predicates that describe states in General Game Playing (GGP), and by higher-level game features that can be inferred from GGP game descriptions, respectively [Finnsson and Björnsson, 2010]. *Opponent Move Abstraction* (OMA), proposed for the selection phase only, uses all previous actions of the same player as context, abstracting away opponent actions [Baier and Kaisers, 2020]. The *Rapid Action Value Estimate* (RAVE) technique [Gelly and Silver, 2007] and its variants, typically only applied in the selection phase, correspond to history abstractors with a slight modification: They map a given history to all of the history's prefixes: $f_{\text{RAVE}}(h_t, a_t) = \{h_0, h_1, \ldots, h_t\}$. Such mappings of histories to *multiple* contexts can result in many updates per sampled action, and therefore lead to extremely fast estimation.

For the experiments in this paper, we consider three abstractors: A_{global} and A_{OMA} as described above, as well as $A_{3\times3}$, whose abstraction function maps h_t, a_t to the 3×3 board squares around the target square of action a_t. This provides a localized view on the part of state that is likely most relevant to a_t, and generalizes to all states which are identical on these squares. We expect $A_{n\times n}$ for some n to work in a wide range of grid-based games with somewhat localized effects of actions. It resembles the local view of the receptive fields in convolutional neural networks, and is similar to an abstraction used by Cazenave [2016].

Beyond the kind of abstractors used in this work, ME-MCTS can also be used with other types of action value estimators. This includes for example abstractors that do not maintain separate value estimates for every legal action, but generalize over actions as well; and estimators that use function approximators instead of lookup tables, which enables them to estimate the values of unseen contexts. Furthermore, there is an approach to online generalization in MCTS

[1] The next potential action a_t is provided to enable abstractors to condition on it, e.g. the abstractor $A_{3\times3}$ described later. We refer to both h_t and (h_t, a_t) as histories in the text, but use notation to clarify where necessary.

that does not use contexts or abstractions as we do here, but is based on a similarity or distance *metric* on states instead [Srinivasan *et al.*, 2015; Xiao *et al.*, 2018]. As long as these methods are able to assess their own uncertainty, they are in principle compatible with ME-MCTS; if the uncertainty can be expressed by pseudocounts, we can in fact apply the exact technique described in the next section.

3 ME-MCTS—Guiding MCTS with Contextual Action Value Estimates

In the previous section, we described how abstractors provide us with contextual action value estimates. In this section, we outline how different action value estimates are used to guide the search in our proposed algorithm ME-MCTS.

Traditionally, MCTS is understood as choosing actions in two distinct phases *in* the tree (selection phase) and *outside* of the tree (rollout phase) [Chaslot *et al.*, 2008a], which together make up all actions of a single *simulation*. In the selection phase, vanilla MCTS chooses actions based on value estimates stored in the tree nodes, using some bandit algorithm such as the classic UCB1 [Auer *et al.*, 2002]. In the rollouts, no value estimates are available in vanilla MCTS, so actions are in the simplest case chosen uniformly at random.

ME-MCTS however chooses all actions in the same way, blurring the traditional distinction between selection and rollout phases. When making an action choice during a given simulation with previous history h_t, it assigns a score to every legal next action a_t. Extending previous work on combining a single abstractor estimate with MC estimates [Gelly and Silver, 2011], this score is a convex combination of all available action value estimates: that of the unbiased estimator A_{history} (the tree), and those of any number of other estimators used. If no other estimators are used, ME-MCTS reduces to vanilla MCTS. After every simulation, all value estimators are updated for all traversed actions in their corresponding context.

For computing the convex combination, ME-MCTS applies insights on optimally combining estimators [Lavancier and Rochet, 2016]. In line with this previous work, we seek to determine the weights $\lambda \in \mathbb{R}^k$, such that the combined action score is defined by $\lambda^\intercal \mathbf{T} = \sum_{i=1}^{k} \lambda_i T_i$, where $\mathbf{T} = (T_1, \dots, T_k)^\intercal$ is a collection of value estimates[2]. We consider only convex combinations, i.e. imposing $0 \leq \lambda_i \leq 1, \forall i$ and $\sum_i \lambda_i = 1$. This hypothesis class strikes a bias-variance tradeoff of being expressive yet retaining identifiable optimal weights, with nuances being discussed in previous work.

We seek to find the optimal weight vector that minimizes mean squared error (MSE), i.e., $\lambda^* = \arg\min_\lambda \mathbb{E}\left(\lambda^\intercal \mathbf{T} - \theta\right)^2 = \arg\min_\lambda \lambda^\intercal \Sigma \lambda$, where θ is the true parameter to be estimated, and $\Sigma = \mathbb{E}\left[(\mathbf{T} - \theta\mathbf{1})(\mathbf{T} - \theta\mathbf{1})^\intercal\right]$ is the MSE matrix. Assuming the MSE matrix is well defined and non-singular, the explicit solution is known to be [Lavancier and Rochet, 2016]

$$\lambda^* = \frac{\Sigma^{-1}\mathbf{1}}{\mathbf{1}^\intercal \Sigma^{-1} \mathbf{1}}. \tag{4}$$

[2]We consider the problem for a specific h_t, a_t here, dropping notational dependency of $\lambda, T, \theta, Q, \Sigma$ etc. on h_t, a_t for simplicity.

In MCTS, the true parameter to be estimated is the expected simulation return. The collection of estimates comprises the unbiased MC estimate along with generalizing estimates, e.g. $\mathbf{T} = (Q_{\text{history}}, Q_{3\times3}, Q_{\text{global}})$. The MSE matrix contains entries $\Sigma_{ij} = \mathbb{E}\left[(T_i - \theta)(T_j - \theta)\right]$. Note that the main diagonal gives $\text{MSE}(T_i) = \sigma_i^2 + b_i^2$, with variance σ_i^2 and bias $b_i = |\mathbb{E}(T_i) - \theta|$. Off-diagonal entries involving the unbiased estimator Q_{history} correspond to covariances, and are thus zero whenever the two respective estimators are independent[3]. Consider the following example MSE matrix, corresponding to the estimator triplet above.

$$\Sigma = \begin{pmatrix} \sigma^2_{\text{history}} & x_{12} & x_{13} \\ x_{12} & \sigma^2_{3\times3} + b^2_{3\times3} & x_{23} \\ x_{13} & x_{23} & \sigma^2_{\text{global}} + b^2_{\text{global}} \end{pmatrix}$$

Since Σ is typically unknown in practice, it must be approximated by some estimate $\hat{\Sigma}$. Since $\hat{\Sigma}$ is used to compute weights for averaging, the most crucial aspect is to capture relative uncertainty in the estimates T_i. Such an estimate of $\hat{\Sigma}$ could be tuned as a meta-parameter and be held constant, or be derived from an approximate model, or from online observations during search. In this work, we compute $\hat{\Sigma}$ based on a model for context-dependent variances (explained below), with tuned meta-parameters for context-independent biases, neglecting co-dependencies (constants $x_{ij} = 0, i \neq j$). We leave it to future work to further refine this estimation.

We estimate the variances assuming a Bernoulli distribution of simulation returns with expectation θ. Let n_i be the number of samples, and $\mathbb{E}(T_i) = \theta_i$ be the underlying expectation for each estimator i, with $\theta_1 = \theta$ (i.e., the first estimator corresponds to the unbiased history estimator). Assuming T_i is a running average estimator of simulation outcomes R_i (here viewed as a random variable over all histories that fall into the same context), $\sigma_i^2 = \text{Var}(T_i) = \frac{\text{Var}(R_i)}{n_i}$. If R_i is a Bernoulli trial with parameter θ_i, $\text{Var}(R_i) = \theta_i(1 - \theta_i)$.

The true expectations θ_i are unknown, and estimator combinations are expected to be most crucial for low sample counts. However, if all initial simulation returns were the same (all one or all zero), then the plugin estimator $\widehat{\text{Var}}(T_i) = \frac{T_i(1-T_i)}{n_i} = 0$, which would under-estimate the uncertainty especially in Q_{history}. Therefore, we instead estimate σ_i^2 by plugging in a maximum entropy estimate of θ_i, or more precisely $\hat{\theta}_i = \arg\max_{\theta_i \in \Theta_i} \theta_i(1 - \theta_i)$ within the simulation returns' Wilson score interval with continuity correction Θ_i.

To trade off exploration and exploitation for our estimator combination, we finally add an upper confidence bound to it. To compute individual upper confidence bounds for all estimates T_i as $U_i = T_i + w_i$, we use the AUER algorithm's confidence width $w_i = c_i \sqrt{\frac{8 \log t}{n_i}}$, where c_i is an exploration parameter and t is time measured in total number of simulations so far. AUER is similar to UCB, and was developed for *sleeping bandits* with changing sets of arms [Kleinberg *et al.*, 2010]. The motivation for this is that the set of legal actions

[3]Since any generalization subsumes the samples of Q_{history}, this independence can in fact be induced by removing local samples from the abstract estimate when computing $\mathbf{T}$. We do not do this here.

is not always constant for all states within a given context. After computing individual upper confidence bounds U_i with AUER, we then also need a way to combine them into an upper confidence bound for our linear combination—built from estimates based on a different number of samples, and with a different bias: The upper confidence bound $U_\lambda = \lambda^\intercal \mathbf{T} + w_\lambda$ for the combined estimator is determined by

$$w_\lambda = \sqrt{\sum_i \lambda_i^2 w_i^2}. \quad (5)$$

This would be an exact upper confidence interval for independent estimators[4] of normally distributed samples, for which the confidence width of averages would yield $w_i = z\sqrt{\text{Var}(T_i)}$, with a scaling factor z. We use it as an approximation of confidence bounds for combining arbitrary widths w_i, which in our experiments come from the AUER formula.

4 Experimental Results

We tested ME-MCTS in four different domains: *Breakthrough*, *Knightthrough*, *Othello*, and *Rolit*. These are fully observable, deterministic, alternating-turn games for two to four players; ME-MCTS does not in principle require these constraints however. All experiments allowed for 250 ms per move, in order to fairly expose the tradeoff between improving search with multiple estimators and the additional computational overhead of computing and combining abstractors. The implemented abstractors each have two parameters: a *bias* parameter that is used to compute their influence on every move choice, and an *exploration factor* for the exploration-exploitation tradeoff of the AUER algorithm. Vanilla MCTS has one parameter: the exploration factor of UCB1. The parameters of all MCTS and ME-MCTS variants were first tuned, followed by a test of the best found parameter settings with at least 1000 games; the results of these tests are presented here.

Our experiments are divided into two groups. In the first group, described in Subsection 4.1, we did state evaluation through Monte Carlo rollouts as in traditional MCTS. In the second group, presented in Subsection 4.2, we assumed that an evaluation function is available for state evaluation, leading to the recently more popular MCTS variants without any Monte Carlo part [Silver *et al.*, 2017]. These typically call the evaluation function as soon as the simulation leaves the tree, omitting the rollout phase entirely. Even though in ME-MCTS, there are other estimators than the tree which could still guide out-of-tree moves, we follow this practice of cutting off as soon as we encounter a history that has not yet been seen and stored in A_{history}, i.e. the tree, and evaluate there.

It is not clear a priori whether abstractors would work better or worse with evaluation functions compared to rollouts: On the one hand, contextual value estimators would profit just like the vanilla MCTS tree from evaluation function returns with higher quality and lower variance than random rollout returns. On the other hand, abstractors can learn from every move in a given simulation—so omitting rollouts results in much less data for them. Our combination of estimators and upper confidence bounds computes weights that respond to relative variance. Even if we devised the variance model for rollouts, the distribution-specific numerator is dominated by the contextual sample counts in the denominator. We may thus expect benefits from online generalization over both MC rollout and evaluation function returns, even if the latter yield a different empirical distribution.

4.1 ME-MCTS using Rollouts

This section summarizes our experiments for MCTS using rollouts for state evaluation. Vanilla MCTS uses uniformly random rollouts, and ME-MCTS simply plays out its simulations to the end of the game. In the first set of experiments, we added each of the three abstractors A_{global}, $A_{3\times3}$, and A_{OMA} individually to ME-MCTS to establish their performance as baseline (only combining them with the tree, A_{history}). In the second set, we added multiple abstractors to ME-MCTS, in order to test our estimator combination method.

Adding Single Abstractors

Figure 1 shows the results of using individual abstractors in ME-MCTS. A_{global} worked well in all four games, $A_{3\times3}$ in all games but Rolit, and A_{OMA} improved MCTS in all domains but Breakthrough. An abstractor can be ineffective in a given domain for various reasons: Its abstraction function may be too costly to compute at the given short time limit for example, or its abstraction may not capture useful similarities between states or histories in the domain at hand. These results primarily serve to give us an impression of which abstractors are at all promising for our estimator combination technique—the quality of an estimator combination of course depends on the quality of the individual estimators available.

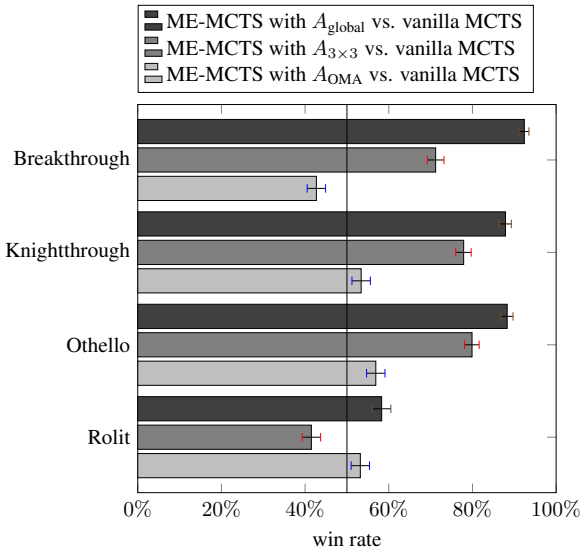

Figure 1: Performance of ME-MCTS using the global, 3×3, and OMA abstractors individually, and using rollouts.

Adding Multiple Abstractors

In our next experiments, we used ME-MCTS to test the combination of A_{global} and $A_{3\times3}$, the combination of A_{global}

[4] $\text{Var}\left(\hat{\theta}_\lambda\right) = \sum_i \lambda_i^2 \text{Var}(T_i) + 2 \sum_{1\le i<j\le k} \lambda_i \lambda_j \text{Cov}(T_i, T_j)$, but independent estimators have zero covariance.

and A_{OMA}, and the combination of $A_{3\times3}$ and A_{OMA}. In each game, we tested these combinations of two abstractors against the best individual abstractor found before, in order to demonstrate the benefit of estimator combination through ME-MCTS. We did not find individually ineffective abstractors to be effective when included in abstractor combinations either. Figure 2 presents the results, showing the best found combination of two abstractors in each domain, and its performance against the best single abstractor according to Figure 1 (which was A_{global} for all games). We can see that in all test domains except Rolit, two abstractors can be more effective at guiding the search than a single one: A_{global} and $A_{3\times3}$ are significantly stronger than only A_{global} in Breakthrough, Knightthrough, and Othello.

In addition, we tested the triple combination of A_{global}, $A_{3\times3}$, and A_{OMA} against the best two abstractors found in each domain. In Rolit, where no combination of two abstractors worked well, we tested the triple combination against the best single abstractor A_{global} instead. Figure 2 shows that three abstractors were significantly better than two in Breakthrough and Othello, while no further improvement over two abstractors could be found in Knightthrough, or over one abstractor in Rolit—the only domain not to show any benefit of estimator combinations here. This could be due to the overhead being too expensive relative to the short time limit and fast random rollouts we used in our experiments. Future work may explore this at other time settings, and with computationally heavier rollouts which would tolerate more overhead[5].

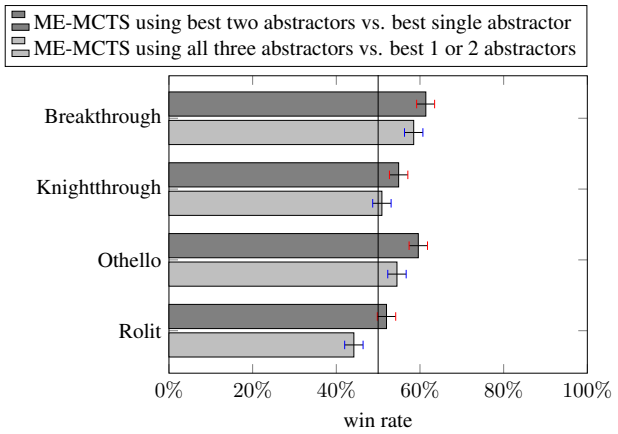

Figure 2: Performance of ME-MCTS using combinations of the global, 3×3, and OMA abstractors, and using rollouts.

4.2 ME-MCTS using Evaluation Functions

This section describes our experiments for MCTS using static evaluation functions for state evaluation. Our evaluation functions had the classic form of linear combinations of game board features. They were tuned in advance, and we ensured that at 250 ms per move, MCTS using them is significantly stronger than MCTS using random rollouts in all domains.

[5]Also note that A_{OMA} works particularly well together with the MCTS enhancement of *progressive widening* [Baier and Kaisers, 2020], which was not considered here for simplicity.

In the first set of experiments, we again tested each of the three abstractors individually in ME-MCTS, and in the second set, we again tested them in combination.

Adding Single Abstractors

Figure 3 shows the results of using single abstractors in ME-MCTS, now using evaluation functions. All abstractors significantly improved vanilla MCTS in all test domains. As a side note, future work could also analyze the connections between offline generalization by an evaluation function, online generalization through abstraction, and possible interactions.

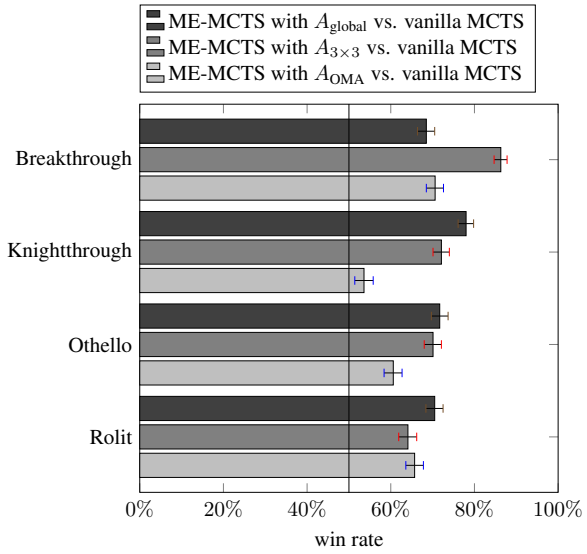

Figure 3: Performance of ME-MCTS using the global, 3×3, and OMA abstractors individually, and using evaluation functions.

Adding Multiple Abstractors

After finding which of the three abstractors individually work best in our test domains, we again tested all combinations of two abstractors against the best single abstractors, and the combination of all three abstractors against the best combinations of two abstractors, as described in the previous subsection. Figure 4 displays the results. In all test domains except Rolit, two abstractors were significantly stronger than a single one again: A_{global} and A_{OMA} are significantly stronger compared to $A_{3\times3}$ in Breakthrough (which is interesting as it shows that the best abstractor "team" does not always have to contain the best "solo" abstractor), A_{global} and A_{OMA} are stronger than only A_{global} in Knightthrough, and A_{global} plus $A_{3\times3}$ outperforms using only A_{global} in Othello. Moreover, in Breakthrough and Knightthrough, the combination of all three tested abstractors also significantly outperformed the best combination of any two abstractors.

Taken together, these results point strongly towards the merit of the idea of combining several estimators in ME-MCTS. Naturally, performance depends on the quality and the computational cost of the available individual estimators. Additional work on optimizations, incremental computations of abstractions, caching strategies etc. could further reduce overhead and thereby make ME-MCTS even more broadly

applicable. In addition, we expect the overhead of individual abstractors as well as their combination in ME-MCTS to be much less relevant when using state-of-the-art deep neural networks for state evaluation instead of the much faster, handcoded functions we used to simplify experimentation. In preliminary experiments in Othello at 1000 simulations per move for example – with the same evaluation function but ignoring overhead – the best tested combination of two abstractors achieved a winrate of 64.8% instead of 58.9% against the best single abstractor, and the combination of all three achieved 55.4% instead of 47.6% against the best two.

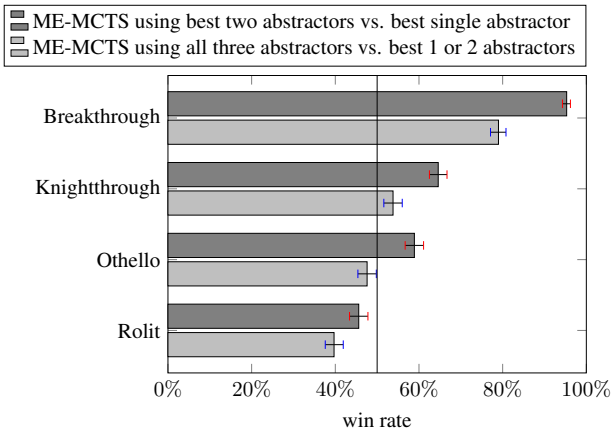

Figure 4: Performance of ME-MCTS using combinations of the global, 3×3, and OMA abstractors, and using evaluation functions.

5 Related Work

Almost all previous online generalization approaches for MCTS were either applied to the selection phase or to the rollouts, but not to *both*. However, applying generalization only during selection means missing out on improving the majority of actions in most simulations, as MCTS trees typically only cover a small part of the full episode; and applying generalization techniques only during rollouts means making them inapplicable to the currently very popular MCTS variants in which rollouts are replaced by static evaluation functions [Silver *et al.*, 2017; Silver *et al.*, 2018]. To our knowledge, the only existing online learning approach applied to both the selection and the rollout phases is based on RAVE [Rimmel *et al.*, 2010]. This partially inspired our approach to using information acquired online throughout entire simulations; however, Rimmel et al. use RAVE estimates in both phases in different ways, while we are aiming at a more unified algorithm for choosing actions that can be applied consistently from the root state of the search to the end of each simulation. They combined no other estimators with RAVE.

Almost all previous online generalization approaches for MCTS used a *single* action value estimator. However, General Game Playing (GGP) led to the development of a family of abstractors [Finnsson and Björnsson, 2010], including MAST, TO-MAST, PAST, and FAST; these were applied to rollout action choices both individually, as well as in combination with RAVE—which is the only case of a combination of two value estimators known to us, and one inspiration for *Multiple Estimator* MCTS.

A further influence for our approach was a note on future work by Powley *et al.* [2013]: The idea of adding multiple different value estimators to MCTS was alluded to here, as well as the idea of combining estimators at different abstraction levels, which allows us to gradually shift our focus from weak abstraction to strong abstraction as we guide the search from the root state to the end of a simulation. However, these ideas were not further developed, implemented or tested.

Combining estimators, such as our abstractors, is an established technique in other fields and the basis of *ensemble learning* for classification and regression, with techniques such as bagging (resampling a dataset) and boosting (re-learning on errors) that improve generalization offline [Mendes-Moreira *et al.*, 2012]. While we also use a linear combination to integrate multiple estimators, we face an online setting and show how several given estimators can be combined to guide search.

6 Conclusions and Further Work

This article has introduced the algorithmic framework ME-MCTS for online generalization in MCTS. Experiments have shown the efficacy of combining multiple types of abstraction in several benchmark games, achieving significant improvements compared to using individual abstraction techniques both in conditions assuming classic rollout-based MCTS, and in conditions using an evaluation function.

For future work, evaluating ME-MCTS in MDPs or stochastic games, partially observable domains, or simultaneous-action domains could offer interesting application perspectives, especially in more "life-like" settings with larger amounts of superfluous information that can be abstracted away without significant loss of performance—board games are already relatively condensed. Furthermore, our work suggests several avenues of improving the performance of state or history abstraction. In practice, it may be most efficient to estimate the MSE matrix state-dependently and online with domain-specific priors that transfer from search to search. One could also initialize individual abstractors with heuristic values, as it has been done for MCTS trees [Chaslot *et al.*, 2008b]. Multi-valued abstractors (mapping a given history to more than one abstraction) might be worth exploring in more depth, in order to learn even more from every single simulation. Currently, RAVE is the only existing algorithm using this approach. It remains an open problem as well how to learn optimal domain abstractions $f_A(h_t, a_t)$ end to end (rather than approximating Q_A directly), e.g. using neural networks. Finally, incorporating linear or non-linear function approximators that can assess their uncertainty—by pseudo-counts or otherwise—will elevate ME-MCTS to an even more general online generalization framework.

Acknowledgments

This work is part of the project *Flexible Assets Bid Across Markets* (FABAM, project number TEUE117015), funded within the Dutch Topsector Energie / TKI Urban Energy by Rijksdienst voor Ondernemend Nederland (RvO).

References

[Auer *et al.*, 2002] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-Time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2-3):235–256, 2002.

[Baier and Kaisers, 2020] Hendrik Baier and Michael Kaisers. Guiding Multiplayer MCTS by Focusing on Yourself. In *IEEE Conference on Games 2020*, pages 550–557. IEEE, 2020.

[Cazenave, 2016] Tristan Cazenave. Playout Policy Adaptation with Move Features. *Theor. Comput. Sci.*, 644:43–52, 2016.

[Chaslot *et al.*, 2008a] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. Monte-Carlo Tree Search: A New Framework for Game AI. In Christian Darken and Michael Mateas, editors, *Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*. The AAAI Press, 2008.

[Chaslot *et al.*, 2008b] Guillaume M. J. B. Chaslot, Mark H. M. Winands, Jaap van den Herik, Jos W. H. M. Uiterwijk, and Bruno Bouzy. Progressive Strategies for Monte-Carlo Tree Search. *New Math. Nat. Comput.*, 4(03):343–357, 2008.

[Finnsson and Björnsson, 2010] Hilmar Finnsson and Yngvi Björnsson. Learning Simulation Control in General Game-Playing Agents. In Maria Fox and David Poole, editors, *Twenty-Fourth AAAI Conference on Artificial Intelligence*. AAAI Press, 2010.

[Finnsson, 2012] Hilmar Finnsson. Generalized Monte-Carlo Tree Search Extensions for General Game Playing. In Jörg Hoffmann and Bart Selman, editors, *Twenty-Sixth AAAI Conference on Artificial Intelligence*. AAAI Press, 2012.

[Gelly and Silver, 2007] Sylvain Gelly and David Silver. Combining Online and Offline Knowledge in UCT. In Zoubin Ghahramani, editor, *Twenty-Fourth International Conference on Machine Learning*, volume 227 of *ACM ICPS*, pages 273–280. ACM, 2007.

[Gelly and Silver, 2011] Sylvain Gelly and David Silver. Monte-Carlo Tree Search and Rapid Action Value Estimation in Computer Go. *Artificial Intelligence*, 175(11):1856–1875, 2011.

[Kleinberg *et al.*, 2010] Robert Kleinberg, Alexandru Niculescu-Mizil, and Yogeshwer Sharma. Regret Bounds for Sleeping Experts and Bandits. *Machine Learning*, 80(2-3):245–272, 2010.

[Kocsis and Szepesvári, 2006] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *17th European Conference on Machine Learning*, pages 282–293, 2006.

[Lavancier and Rochet, 2016] Frédéric Lavancier and Paul Rochet. A general procedure to combine estimators. *Comput. Stat. Data Anal.*, 94:175–192, 2016.

[Liebana *et al.*, 2016] Diego Perez Liebana, Spyridon Samothrakis, Julian Togelius, Tom Schaul, Simon M. Lucas, Adrien Couëtoux, Jerry Lee, Chong-U Lim, and Tommy Thompson. The 2014 General Video Game Playing Competition. *IEEE Trans. Comput. Intell. AI in Games*, 8(3):229–243, 2016.

[Mendes-Moreira *et al.*, 2012] Joao Mendes-Moreira, Carlos Soares, Alípio Mário Jorge, and Jorge Freire De Sousa. Ensemble approaches for regression: A survey. *ACM Comput. Surv.*, 45(1):10, 2012.

[Nijssen and Winands, 2010] J. (Pim) A. M. Nijssen and Mark H. M. Winands. Enhancements for Multi-Player Monte-Carlo Tree Search. In H. Jaap van den Herik, Hiroyuki Iida, and Aske Plaat, editors, *7th International Conference on Computers and Games*, volume 6515 of *LNCS*, pages 238–249. Springer, 2010.

[Powley *et al.*, 2013] Edward Jack Powley, Daniel Whitehouse, and Peter I. Cowling. Bandits All the Way Down: UCB1 as a Simulation Policy in Monte Carlo Tree Search. In *2013 IEEE Conference on Computational Intelligence in Games*, pages 1–8. IEEE, 2013.

[Rimmel *et al.*, 2010] Arpad Rimmel, Fabien Teytaud, and Olivier Teytaud. Biasing Monte-Carlo Simulations through RAVE Values. In H. Jaap van den Herik, Hiroyuki Iida, and Aske Plaat, editors, *7th International Conference on Computers and Games*, volume 6515 of *LNCS*, pages 59–68. Springer, 2010.

[Silver *et al.*, 2017] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the Game of Go without Human Knowledge. *Nature*, 550(7676):354–359, 2017.

[Silver *et al.*, 2018] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-Play. *Science*, 362(6419):1140–1144, 2018.

[Srinivasan *et al.*, 2015] Sriram Srinivasan, Erik Talvitie, and Michael H. Bowling. Improving Exploration in UCT Using Local Manifolds. In Blai Bonet and Sven Koenig, editors, *Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 3386–3392. AAAI Press, 2015.

[Tak *et al.*, 2012] Mandy J. W. Tak, Mark H. M. Winands, and Yngvi Björnsson. N-Grams and the Last-Good-Reply Policy Applied in General Game Playing. *IEEE Trans. Comput. Intell. AI in Games*, 4(2):73–83, 2012.

[Xiao *et al.*, 2018] Chenjun Xiao, Jincheng Mei, and Martin Müller. Memory-Augmented Monte Carlo Tree Search. In *Thirty-Second AAAI Conference on Artificial Intelligence*, pages 1455–1462, 2018.