# Novelty in Monte Carlo Tree Search

Hendrik Baier, Michael Kaisers

***Abstract*—*Novelty search* has shown benefits in different fields such as evolutionary computing, classical AI planning, and deep reinforcement learning. Searching for novelty instead of, or in addition to, directly maximizing the search objective, aims at avoiding dead ends and local minima, and overall improving exploration. We propose and test the integration of novelty into Monte Carlo Tree Search (MCTS), a popular framework for online RL planning, by linearly combining value estimates with novelty scores during the selection phase of MCTS. We adapt four different novelty measures from the literature (evaluation novelty, state-pseudocounts, feature-pseudocounts, and frequency-thresholding), integrate them into MCTS, and test them in six board games (Connect4, Othello, Breakthrough, Knightthrough, AtariGo and Gomoku). Experiments show improvements for MCTS in a wide range of settings, covering both guidance by handcoded heuristics and neural networks. The results demonstrate potential for these optimistic novelty estimates to achieve online generalisation of uncertainty during search.**

***Index Terms*—novelty, novelty search, Monte Carlo Tree Search, game tree search**

## I. Introduction

Sequential decision-making problems arise in a variety of domains, and significant progress in this area has been achieved by studying search in games. *Monte Carlo Tree Search* (MCTS) in particular handles large search spaces well due to selective sampling of promising actions. It has been shown to converge to the optimal policy in the limit, if exploration and exploitation are traded off properly [1], and it provides approximations at any time. MCTS and its many variants have been successfully applied to countless domains in recent years, for example to General Game Playing [2], to General Video Game Playing [3], and to two-player board games in recent breakthroughs that combined search with deep neural networks [4], [5].

The use of *novelty* in search and optimization has been a significant line of research in several different subfields of AI in the last decade, such as in evolutionary computing, in classical AI planning, and in deep reinforcement learning. Searching for novel states or novel behaviors as an *intrinsic* motivation of an AI agent has shown surprising success compared to exclusively trying to optimize the *extrinsically* given objective function, especially in domains where the gradient of improvement with regard to that objective function is sparse or misleading [6].

In this paper, we propose the integration of novelty search techniques into MCTS, and test them in online RL planning. Specifically, we bias search by linearly combining one of four novelty scores with the value estimates maintained by MCTS before adding the UCB exploration bonus. Unlike in many previous applications of novelty to search, we assume that a somewhat effective heuristic state evaluation function (either handcoded or trained) is already available to guide MCTS; and unlike typical applications of novelty to deep RL, MCTS is able to use count-based uncertainty estimates in its tree. Nevertheless, our results indicate that novelty, as an auxiliary objective that generalizes uncertainty across the state space, can lead to better gameplay through further improvements in MCTS exploration.

This paper extends on a previous workshop paper [7]. To this previous work, we are adding two more test domains (AtariGo and Gomoku); one additional novelty measure inspired by more recent literature on the topic (see III-A4); an analysis of the time overhead of our approach (see IV-D); and we are showing the effectiveness of our approach not only for MCTS guided by handcoded heuristics, but additionally for MCTS guided by learned heuristics, i.e. neural networks in the style of AlphaZero [5]. In particular this last extension demonstrates a much wider applicability of novelty in MCTS, including state-of-the-art MCTS implementations.

The paper is structured as follows: Section II relates this work to the literature. Section III briefly sketches the four different novelty measures used, and how we integrated them into MCTS. Section IV presents experimental results in six board game domains, and Section V discusses conclusions and future work. Code is available online.

## II. Related Work

In evolutionary computation, it has been found that fitness functions, meant to measure progress towards the actual objective of the search, can often be deceptive, and thus lead into dead ends. A sometimes surprisingly effective approach to circumventing this problem is to ignore the objective entirely, and to search only for (behavioral) novelty instead [6], [8]. In this work, we do not aim to completely ignore our objective, i.e. heuristic estimates guiding successful play, but integrate an additional novelty score in order to improve exploration.

In classical AI planning, a simple blind-search procedure called *Iterated Width* (IW) was developed [9]. IW is an iterative breadth-first search that prunes states of insufficient novelty, relaxing the requirement for novelty in every iteration. It was found to be effective in many benchmark planning problems due to their simple enough goal structure, and lead to state-of-the-art performance when integrated with other known planning techniques [9]. As IW does not require knowledge of transitions and goals, it has also been successfully applied

When the authors did the work presented in this paper, the authors were with the Intelligent and Autonomous Systems group, CWI, Amsterdam, the Netherlands. H. Baier (h.j.s.baier@tue.nl) is now with the Information Systems group, TU Eindhoven, Eindhoven, the Netherlands. M. Kaisers (mkaisers@google.com) is now with Google DeepMind.

to simulation-based planning, both in Atari games [10] and in General Game Playing (GGP) [11]. Its pruning criterion, i.e. its novelty measure, was extended to take the reward-so-far into account [12], which improved results in Atari games; and it was modified to utilize heuristic value estimates of states instead when those are available [13]. One of the novelty measures we examine in this work is similar to the ones previously proposed for classical planning [12], [13], where it has been called $h_{BN}$. However, we use it to modify value estimates in the MCTS tree, instead of using it for improving the node ordering in a traditional best first search queue.

In (deep) reinforcement learning, novelty has been studied as a form of intrinsic motivation for the RL agent [14], meant to aid in representation learning as well as in improving exploration. Unlike IW-like algorithms, which tend to use a binary classification of states into novel or non-novel, these RL approaches aim for a finer-grained measure of the agent's uncertainty about its environment, in order to guide the agent towards less familiar regions of the state space and thus encourage learning. In analogy to the tabular case, where we can simply count how often each state has been visited and intrinsically reward the agent for visiting states with lower visit counts, RL novelty techniques are often based on *pseudocounts* – a generalization of state counts which allows to generalize uncertainty across large state spaces. Pseudocounts can be defined based on a *density model* that assigns probabilities to observing a given state [15]. Such pseudocounts have been proposed based on state hashing [16], on neural density models [17], [18], or on successor representations [19]. Two of the novelty measures we study in this work are based on the state-visitation density model [15], as well as on a similar model constructed on a feature representation of the state instead of on the raw state [20]. However, we use it for sample-based planning instead of learning.

The closest work to ours we are aware of is an adaptation of IW-like novelty pruning to Monte Carlo Tree Search, as part of eight enhancements for the application of General Video Game Playing (GVGP) [21]. In contrast to our work in which we bias search, novelty was here used for hard pruning of the tree, which invalidates the convergence guarantees of MCTS. We agree with the authors that "Novelty-Based Pruning (NBP) as proposed in this paper [has] binary effects, in that (...) NBP classifies nodes as either novel or not novel. Perhaps [this method] can be improved by making [it] less binary" [21], which we aim for by proposing a more finely tunable approach that biases the search instead of pruning it. We also differ from Soemers et al. [21] in that we test our approach on adversarial multi-agent environments instead of single-agent ones.

Since the publication of the workshop version of this paper [7], another novelty method has been proposed for Monte Carlo Graph Search (MCGS), a variant of MCTS [22]. This method was specifically intended for sparse reward environments in which no heuristic evaluation function exists, providing some search guidance without relying on "handcrafting of the reward signal for the specific environment" [22]. In contrast to this approach, we demonstrate that novelty can improve exploration not only if it is the only form of search guidance, but *even if* heuristic evaluation functions are already available, either handcrafted (see Subsection IV-A) or indeed learned from scratch with state-of-the-art reinforcement learning methods (see Subsection IV-B). This shows the usefulness of novelty-guided search in a wider range of settings. However, we have included an additional novelty measure inspired by their approach in this paper, see III-A4.

## III. Novelty and MCTS

In this section, we describe the four novelty measures we use in this exploratory study, and how we integrate them into the MCTS framework. All novelty measures assume a state space $S$ with internal structure, with *factored* states $s$ that consist of a vector of distinct atomic components or variables and their assigned values. In board games for example, "square d4" could be such a variable. A variable plus assigned value, e.g. "white piece on d4", is also called a *fact*.[1] We call novelty measures directly based on these atomic state facts "raw-state novelty", and novelty measures based on higher-level features such as those defined by a handcoded heuristic evaluation function "feature-based novelty", although atomic variables could theoretically serve as evaluation features as well.

### A. Defining Novelty

The techniques briefly outlined in this subsection aim at measuring the novelty of a newly discovered state, and are partially modified techniques from the literature.

*1) Evaluation novelty ($N_E^{raw}$):* This technique is adapted from *reward-based novelty* [12], [13]. Given a heuristic state evaluation function $V : S \to \mathbb{R}$ defined on the set of states $S$, and $S_t$ as the set of states observed until time step $t$, the *novelty score* of a feature (either atomic fact or higher-level feature) $f$ at time step $t$ is defined as

$$N_t(f) = \begin{cases} \max\limits_{s \in S_t, f \in s} V(s) & \text{if } f \in s \text{ for some } s \in S_t \\ -\infty & \text{otherwise,} \end{cases} \quad (1)$$

i.e. as the highest evaluation of any state with that particular feature seen so far. Given a state $s$, its *evaluation novelty* $N_E(s)$ is then defined as

$$N_E(s) = \begin{cases} \alpha & \text{if } V(s) > N_t(f) \text{ for some } f \in s \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

where $\alpha$ is a tunable parameter. This means that this novelty measure is binary – it distinguishes only between novel and non-novel – and that a state is considered novel iff for at least one of the features it consists of, the evaluation of the state is higher than that of any state observed before with that particular feature[2].

Evaluation novelty ($N_E$) could in principle be defined on higher-level evaluation features extracted from states as well

[1]*Discrete facts* are often represented as *one-hot encoding* when presented as input to neural networks, and thus their availability is axiomatic across the deep learning literature.

[2]Note that a higher value $N_t(f)$ of a feature $f$ does not mean that $f$ is considered more novel. The values $N_t(f)$ are simply called "novelty scores" because they allow us to compute the novelty $N_E(s)$ of a given state, where higher values indeed mean more novel.

as on raw state facts. Here, we only include experiments based on raw state facts ($N_E^{\text{raw}}$).

*2) Raw-state pseudocount novelty ($N_C^{raw}$):* This technique is adapted from the *$\phi$-Exploration-Bonus algorithm* [20]. It does not make use of a heuristic evaluation function, but of a probability distribution over states. We assume a feature mapping $\phi : S \rightarrow T$ from the state space into an $M$-dimensional feature space $T$, which could in our case either be the space of raw state facts, or a space of higher-level evaluation features. We define a density model $\rho_t(\phi)$ at time $t$ – a probability distribution over this feature space – as the product of independent factor distributions $\rho_t^i(\phi_i)$ over the $M$ individual features:

$$\rho_t(\phi) = \prod_{i=1}^{M} \rho_t^i(\phi_i). \tag{3}$$

For the factor models, we use the empirical estimator

$$\rho_t^i(\phi_i) = \frac{C_t(\phi_i)}{t}, \tag{4}$$

where $C_t(\phi_i)$ is the number of times feature $\phi_i$ has been observed until time step $t$. This allows us to define the *$\phi$-pseudocount* for a given state $s$ at time $t$ as

$$\hat{C}_t^{\phi}(s) = \frac{\rho_t(\phi(s))(1 - \rho_{t+1}(\phi(s)))}{\rho_{t+1}(\phi(s)) - \rho_t(\phi(s))}, \tag{5}$$

where $\rho_t$ is the density model before $\phi(s)$ has been observed, and $\rho_{t+1}$ is the model after the observation [15]. The *pseudo-count novelty* $N_C(s)$ is then defined as

$$N_C(s) = \frac{\alpha}{\sqrt{\hat{C}_t^{\phi}(s)}}, \tag{6}$$

where $\alpha$ is again a tunable parameter [20].

Different feature mappings $\phi$ are imaginable; for the *raw-state* pseudocount novelty tested in this work, we use the atomic facts our states consist of. An atomic fact in chess could for example be "black knight on e5" or "white pawn on a3".

*3) Feature-based pseudocount novelty ($N_C^{eval}$):* The *feature-based* pseudocount novelty variant is computed analogously to the raw-state variant. The difference is that instead of using atomic facts, we compute the density model with the features used by our heuristic evaluation functions, as originally intended for this novelty measure [20]. An evaluation function in chess for example could use more abstract features such as "Black still has both knights", or "White controls the f-file with a rook", which are potentially more meaningful in terms of future rewards.

*4) Raw-state threshold-based novelty ($N_T^{raw}$):* This technique is adapted from the novelty measure used by Tot et al. [22]. Given $S_t$ as the set of states observed until time step $t$, and $S_t[f]$ as the subset of $S_t$ with the feature (either atomic fact or higher-level feature) $f$ ($S_t[f] = \{s \in S_t | f \in s\}$), the *novelty score* of $f$ at time step $t$ is defined as

$$N_t(f) = \begin{cases} \alpha & \text{if } \frac{|S_t[f]|}{|S_t|} < \gamma \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where $\alpha$ and $\gamma$ are tunable parameters – $\gamma$ is the frequency threshold below which a feature counts as novel, and $\alpha$ is the bonus given to novel features. Given a state $s$, its *threshold-based novelty* $N_T(s)$ is then defined as the sum of all its features' novelty scores,

$$N_T(s) = \sum_{f \in s} N_t(f) \tag{8}$$

The novelty measure originally proposed by Tot et al. [22] included an additional mechanism called "novelty inheritance", where nodes (states) reached from a novel node inherit 50% of the parent's novelty score. We do not need this mechanism, as our method for integrating novelty into MCTS, described in the following subsection, already results in the desired result of "encouraging further exploration from the area where a novelty was found" [22].

Just like $N_E$, $N_T$ could in principle be defined on higher-level evaluation features extracted from states as well as on raw state facts. Here, we only include experiments for the version based on atomic facts ($N_T^{\text{raw}}$).

## B. Using Novelty within MCTS

When integrating novelty measures into MCTS, our goal is to improve exploration under short time controls, without losing convergence guarantees in the limit. We achieve this by combining regular value estimates with novelty scores in the selection phase of MCTS, adapting a technique originally proposed for *Rapid Action Value Estimates* (RAVE) [23].

Each MCTS simulation produces an evaluation $V(s)$, returned by the heuristic evaluation function, and additionally a novelty score $N(s)$, produced by the chosen novelty measure, of the state $s$ that was just added to the tree. Just like the value estimate in vanilla MCTS, the additional novelty score is then also backpropagated to all states that were visited in the current simulation; and in each tree node representing one of these states, an average $\bar{N}_a$ is maintained of all novelty scores seen in the subtree below the traversed state-action pair, analogously to how value estimates $\bar{V}_a$ in MCTS tree nodes are formed by averaging over all *evaluations* seen in the subtrees below. For each visited tree node from which an action $a$ was chosen, with $n_a$ the number of times that action has been chosen so far, the backpropagation updates are:

$$\begin{aligned} n_a &= n_a + 1, \\ \bar{V}_a &= \bar{V}_a + \frac{V(s) - \bar{V}_a}{n_a}, \\ \bar{N}_a &= \bar{N}_a + \frac{N(s) - \bar{N}_a}{n_a} \end{aligned}$$

During the selection phase, the selection policy of MCTS can now be changed from the classic UCB1 policy that is based on value alone:

$$\bar{V}_a + k\sqrt{\frac{\ln n}{n_a}}, \tag{9}$$

where $n$ is the number of times the given node has been traversed choosing any action, and $k$ is a factor trading off exploration and exploitation, to a new policy that linearly combines value and novelty averages:

$$b\bar{N}_a + (1-b)\bar{V}_a + k\sqrt{\frac{\ln n}{n_a}}, \quad (10)$$

where $b$ is a weighting coefficient as given by

$$\sqrt{\frac{\beta}{3n_a + \beta}}, \quad (11)$$

with $\beta$ as a tunable parameter regulating how quickly $b$ decays over time, i.e. how quickly novelty is phased out as the node becomes increasingly certain of its value estimates over time. After the search has been completed, MCTS chooses the root action with the highest value estimate (ignoring novelty) for execution; choosing the root action with the highest sample count is another popular option, but did not lead to significantly different results in exploratory experiments.

Note that in the limit, the MCTS search will visit every possible state an infinite number of times, and all novelty averages will approach zero. In short search times however, the parameter $\beta$ helps us to control how quickly MCTS should forget about novelty and focus on value.

## IV. Experimental Results

We tested novelty-enhanced MCTS against vanilla MCTS in six different board game domains: *Connect 4* (on the regular 6x7 board), *Othello* (on 8x8), *Breakthrough* (on 8x8), *Knightthrough* (on 8x8), *Gomoku* (on 9x9), and *AtariGo* (on 7x7). These are fully observable, deterministic, alternating-turn, two-player games, although our approach does not require these constraints. Like the Top Chess Engine Championship (TCEC) with opening books[3], we randomized the first turn of all players in order to achieve more variety in the matches, and fairness was guaranteed by playing two matches from each randomized first turn, with either player moving first.

Vanilla MCTS has one parameter: the exploration factor $k$ of UCB1. All novelty-based approaches have at least two additional parameters: a parameter $\alpha$ that controls the magnitude of the novelty term, and a parameter $\beta$ that controls how quickly the novelty term's influence on the search decreases. $N_T$ additionally uses a parameter $\gamma$ that controls the novelty frequency threshold. The parameters of all agents were first tuned (requiring about 25,000 matches per agent, see Appendix A for details on the tuning method), followed by a test of the best found parameter settings with at least 2000 additional matches. The results of these tests are presented here. In all tables, boldface indicates statistical significance at the 95% confidence level of an improvement over the vanilla MCTS baseline. Our experiments consisted of two sets: The first set was based on MCTS guided by traditional handcoded heuristics, and the second set on MCTS guided by neural networks.

[3]See also https://wiki.chessdom.org/Openings_FAQ.

### A. MCTS Guided by Handcoded Heuristics

In the first set of experiments, all MCTS players used traditional linear heuristic evaluation functions for state evaluation; these evaluation functions also provided us with feature representations for $N_C^{\text{eval}}$ (see Section III-A3). The selection function was UCT. All experiments allowed for either 1000 or 5000 MCTS simulations per move, in order to test whether novelty can improve the sample efficiency of the search. The test domain of AtariGo was left out of this set of experiments, as we did not have an effective handcoded evaluation function for it; we included AtariGo in the experiments with trained evaluation functions described in the next subsection.

The results of our experiments are shown in four tables, depending on the novelty measure used by the enhanced players. The results for pseudocount novelty $N_C^{\text{raw}}$ based on the raw state itself are shown in Table I. The results for the pseudocount novelty measure $N_C^{\text{eval}}$, based on the same features that are used by the heuristic evaluation function in each domain, are shown in Table II. The results for the evaluation-based novelty measure $N_E^{\text{raw}}$ are given in Table III. Finally, the results for the threshold-based novelty measure $N_T^{\text{raw}}$ are shown in Table IV.

| Game | simulations/move | |
|---|---|---|
| | 1000 | 5000 |
| Connect4 | 50.9% | **53.7%** |
| Othello | 50.2% | **56.2%** |
| Breakthrough | 50.3% | **60.9%** |
| Knightthrough | **63.1%** | **79.9%** |
| Gomoku | 52.1% | 49.1% |
| Average | 53.3% | 60.0% |

TABLE I: Winrate of MCTS with raw-state pseudocount novelty ($N_C^{\text{raw}}$) vs. baseline MCTS, using UCT and handcoded heuristics.

| Game | simulations/move | |
|---|---|---|
| | 1000 | 5000 |
| Connect4 | 50.8% | **55.7%** |
| Othello | **52.7%** | **58.0%** |
| Breakthrough | 50.9% | **59.3%** |
| Knightthrough | 52.3% | **72.4%** |
| Gomoku | 50.4% | 48.8% |
| Average | 51.4% | 58.9% |

TABLE II: Winrate of MCTS with feature-based pseudocount novelty ($N_C^{\text{eval}}$) vs. baseline MCTS, using UCT and handcoded heuristics.

We can summarize the results with three observations. First, novelty-enhanced exploration seems to be promising in principle: It led to statistically significant improvements over vanilla UCB1 selection in 22 out of 40 conditions. Second, it seems to be more promising when higher search budgets are available: Only 6 of 20 conditions at 1000 simulations per move are improved, but 16 of 20 conditions at 5000 simulations per move. And interestingly, differences between domains and search budgets were relatively robust to the choice of novelty measure. Whether novelty helps a lot (as in

| Game | simulations/move | |
|---|---|---|
| | 1000 | 5000 |
| Connect4 | **53.2%** | **54.1%** |
| Othello | 50.8% | **56.4%** |
| Breakthrough | **59.7%** | **60.2%** |
| Knightthrough | **62.0%** | **78.0%** |
| Gomoku | 52.3% | 48.1% |
| Average | 55.6% | 59.4% |

TABLE III: Winrate of MCTS with raw-state evaluation novelty ($N_E^{\text{raw}}$) vs. baseline MCTS, using UCT and handcoded heuristics.

| Game | simulations/move | |
|---|---|---|
| | 1000 | 5000 |
| Connect4 | **53.4%** | **54.9%** |
| Othello | 48.8% | **57.4%** |
| Breakthrough | 50.9% | **60.7%** |
| Knightthrough | 48.9% | **78.6%** |
| Gomoku | 51.9% | 47.9% |
| Average | 50.8% | 59.9% |

TABLE IV: Winrate of MCTS with raw-state threshold-based novelty ($N_T^{\text{raw}}$) vs. baseline MCTS, using UCT and handcoded heuristics.

Knightthrough) or not at all (as in Gomoku) seems to depend more on the domain and/or the evaluation function used – possibly on how frequently it is misleading – than on the precise technique chosen for computing novelty.

As expected, the optimal values for $\alpha$ and $\beta$ determined by our experiments tended to be higher for conditions where novelty has a stronger positive effect, and zero or close to zero for conditions where novelty does not help, minimizing its effect. For example, $N_C^{\text{raw}}$ in Knightthrough at 5000 simulations per move worked best with $\alpha = 2$, while for Othello $\alpha = 0.01$ was returned by our optimization, and for Gomoku $\alpha = 0$. The algorithms were not very sensitive to their parameters, and sometimes using a higher $\alpha$ or using a higher $\beta$ even seemed interchangeable to a degree, as they both increase the influence of novelty. Note however that comparisons of the absolute values of these parameters in different domains are difficult, as they depend on the variance of the heuristic evaluation function used: When all heuristic values fall between $0.499$ and $0.501$, a very small weight to a novelty score can already make a large difference when choosing moves.

## *B. MCTS Guided by Neural Networks*

In the second set of experiments, all MCTS players used neural networks both for state evaluation as well as for search guidance via a learned policy. These networks had one shared network body, leading to a value head and a policy head. They were trained with the state-of-the-art AlphaZero approach, and guided MCTS with the same PUCT selection function that AlphaZero uses [5]. Due to computational limitations, the networks consisted of two residual blocks in the body, with 32 and 16 filters respectively; the value head consisted of a single output neuron that approximated the state value, after one fully connected layer of 32 neurons; and the policy head consisted of a number of output neurons sufficient to encode a probability distribution over every legal move in the game at hand, after one fully connected layer whose number of neurons was the smallest power of two that exceeded the number of output neurons. All experiments allowed for either 500 or 2500 MCTS simulations per move. All six test domains were included; but because no handcoded evaluation features were assumed to exist in this set of experiments, pseudocount novelty was only tested based on the raw state ($N_C^{\text{raw}}$) and not on higher-level evaluation features ($N_C^{\text{eval}}$)[4].

The results of our experiments are again shown sorted by the respective novelty measure used to enhance MCTS. The results for raw-state pseudocount novelty $N_C^{\text{raw}}$ are shown in Table V; for raw-state evaluation novelty $N_E^{\text{raw}}$, in Table VI; and for raw-state threshold-based novelty $N_T^{\text{raw}}$, in Table VII.

We can make three interesting observations here again. First, novelty-enhanced exploration also works well for state-of-the-art MCTS implementations guided by *learned* evaluation functions and prior policies – at least with the modestly sized neural network models we were able to test. It led to statistically significant improvements over baseline MCTS in 27 out of 36 conditions. Second, in contrast to MCTS guided by handcoded evaluation functions, the novelty improvement does not seem to increase with longer searches here. We have yet to find a satisfying explanation for this, although we suspect that handcoded search guidance, based on linear combinations of simple features, makes more systematic errors than learned search guidance – which could mean that MCTS guided by such handcoded heuristics could profit less from additional search time unless exploration is somehow enhanced, for example by seeking novelty. However, it is also possible that the stronger effect of novelty-enhanced search simply kicks in at higher budgets than we were able to test with the computationally more demanding network-guided players; this remains to be studied in future work. And third, our novelty-enhanced players guided by learned heuristics show different performance profiles over test domains than the novelty-enhanced players guided by handcoded heuristics tested in the last subsection. For example, the former are much more effective than the latter in Gomoku, while they are much less effective in Knightthrough[5]. One intuitive hypothesis for these differences is that the effect of novelty could primarily depend on the *strength* of a player, i.e. players with higher playing strength might not profit as much from enhancements to their exploration as weaker players. After all, heuristics producing perfect value estimates would of course make all exploration (and all search) unnecessary. This hypothesis is

[4]In future work, it could be possible to extract higher-level features from neural network evaluations as well, for example from the penultimate layer of the value head. Because these features would not be binary, additional modifications of the novelty measures would be necessary.

[5]The precise 50% winrate in some Knightthrough conditions stems from the fact that using no novelty at all worked best here, which led to sets of identical matches being played in self-play. All players played greedily with no stochasticity in their decision-making other than randomizing the order of equally valued child nodes during search. This leads to significant variation for our handcoded evaluation functions, but not for our learned network evaluators, which virtually never value moves equally.

supported for example by the fact that our learned heuristics are much stronger than our handcoded heuristics in Knightthrough, where they profit much less from novelty; but our learned heuristics are somewhat weaker than our handcoded heuristics in Othello, where the networks are not quite deep enough to take interactions of pieces across the whole board into account – and where they profit much more from added novelty guidance. However, it is not clear if there is *always* an inverse correlation of heuristic strength and the usefulness of novelty guidance. We therefore attempt to answer this question in the following subsection.

| Game | simulations/move | |
|---|---|---|
| | 500 | 2500 |
| Connect4 | **53.8%** | **54.3%** |
| Othello | **65.0%** | **75.4%** |
| Breakthrough | **59.1%** | 50.7% |
| Knightthrough | **52.8%** | 50.0% |
| AtariGo | **57.1%** | **53.8%** |
| Gomoku | **55.9%** | **56.8%** |
| Average | 57.3% | 56.8% |

TABLE V: Winrate of MCTS with raw-state pseudocount novelty ($N_C^{\text{raw}}$) vs. baseline MCTS, using PUCT and neural networks.

| Game | simulations/move | |
|---|---|---|
| | 500 | 2500 |
| Connect4 | **57.1%** | **56.4%** |
| Othello | **76.2%** | **60.2%** |
| Breakthrough | **55.3%** | **52.9%** |
| Knightthrough | 50.0% | 50.0% |
| AtariGo | 51.7% | 51.9% |
| Gomoku | **55.7%** | **54.8%** |
| Average | 57.6% | 54.4% |

TABLE VI: Winrate of MCTS using raw-state evaluation novelty ($N_E^{\text{raw}}$) vs. baseline MCTS, using PUCT and neural networks.

| Game | simulations/move | |
|---|---|---|
| | 500 | 2500 |
| Connect4 | **55.1%** | **58.1%** |
| Othello | **71.9%** | **66.3%** |
| Breakthrough | **56.6%** | **54.2%** |
| Knightthrough | 50.0% | 50.0% |
| AtariGo | **55.5%** | 52.5% |
| Gomoku | **57.4%** | **58.0%** |
| Average | 57.8% | 56.5% |

TABLE VII: Winrate of MCTS using raw-state threshold-based novelty ($N_T^{\text{raw}}$) vs. baseline MCTS, using PUCT and neural networks.

## C. Evaluation Strength vs. Usefulness of Novelty Guidance

The question we seek to answer is: Do players with stronger evaluation functions always profit less from adding a novelty bonus to exploration than weaker players? If there turns out to be a consistent inverse correlation between playing strength and the usefulness of novelty guidance, our approach could be considered less promising for the largest and most powerful current models.

Due to computational limitations, we were only able to explore this problem in two domains: Othello and Gomoku. We trained an Othello-playing and a Gomoku-playing network from scratch with AlphaZero [5], extracting the currently strongest network at different stages of training: at six stages in Othello, and at 22 stages in Gomoku for a finer-grained approach. Then we tested in a first experiment whether the networks of later training stages were indeed stronger players than the ones from earlier stages, by letting each of them guide MCTS in 2000 matches against a mix of MCTS players guided by all networks in this set (i.e. the first data point in Othello is from matches played by MCTS using network 1 against MCTS using networks 1 to 6, with an equal number of matches against each). All players used 500 MCTS simulations per move. The results are shown in Figure 1 for Othello and Figure 2 for Gomoku, confirming that the win rate of MCTS increased strongly during training. In Gomoku for example, MCTS with network no. 1 won only 2.5% of matches, while MCTS with network no. 22 could defeat the set of all players in 87.0% of matches. As expected, training stage correlates strongly with playing strength, as training had not yet reached the full potential of the network architecture in either game.

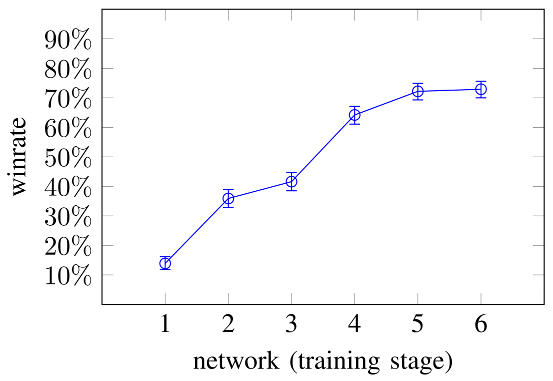

Fig. 1: Winrate of MCTS players using networks of different strength vs. a mix of all such players, in Othello. Error bars show 95% confidence intervals.

In a second experiment, we then tuned novelty-enhanced MCTS (with the novelty measure $N_E^{\text{raw}}$) using each of the extracted networks. The playing strength of these novelty-enhanced players against baseline MCTS using the same networks is shown in Figure 3 for Othello and Figure 4 for Gomoku[6]. As we can see, the novelty guidance has a significant effect for all networks in Othello and for all but 3 networks in Gomoku, from the weakest to the strongest ones. However, there is no clear relationship between the playing strength of MCTS guided by a network and the effect of adding novelty guidance to this MCTS player. The correlation between the two, i.e. the correlation coefficient between the

[6]The exploration factor of baseline MCTS had first been tuned for all networks as well, to ensure the strongest possible vanilla MCTS baseline.

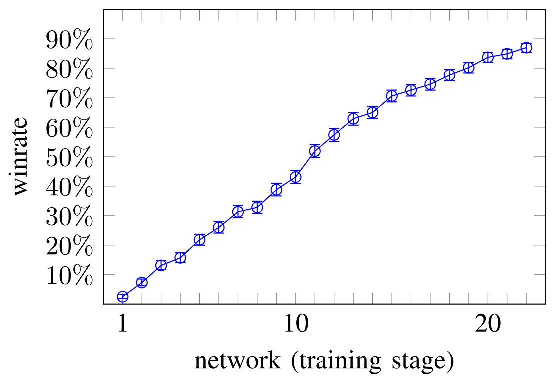

Fig. 2: Winrate of MCTS players using networks of different strength vs. a mix of all such players, in Gomoku. Error bars show 95% confidence intervals.

winrates shown in Figures 2 and 4 for Gomoku, where we computed more data points to increase statistical power, is only $r(20) = .35, p = .105$, thus not significant at the $\alpha = 0.05$ level.

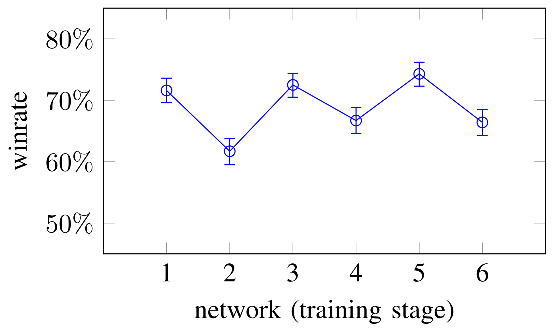

Fig. 3: Winrate of novelty-guided MCTS (with the novelty measure $N_E^{\text{raw}}$) using networks of different strength vs. baseline MCTS using the same networks, in Othello. Error bars show 95% confidence intervals.

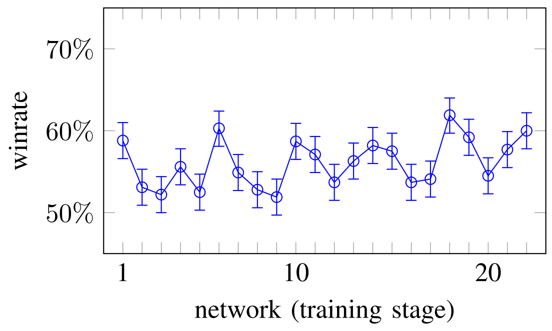

Fig. 4: Winrate of novelty-guided MCTS (with the novelty measure $N_E^{\text{raw}}$) using networks of different strength vs. baseline MCTS using the same networks, in Gomoku. Error bars show 95% confidence intervals.

In conclusion, these exploratory experiments show that the usefulness of novelty guidance cannot always be easily predicted from the strength of a search heuristic. While this needs to be explored in different test domains and with different novelty measures in the future, it could be that the effect of novelty depends in a more subtle way on how often the available heuristics are misleading, how strongly they are misleading, and how systematically they are misleading (e.g. in situations that are more or less relevant for playing strength). There is relatively little theoretical understanding of novelty search in other fields where it is very successful, such as evolutionary computation [24], [25]; exploring this more deeply is interesting future work. Our preliminary results at least indicate that novelty-enhanced MCTS can be effective even when relatively strong search guidance is already available.

## D. Time vs. Usefulness of Novelty Guidance

The experiments described above all compare baseline MCTS to novelty-guided MCTS at equal numbers of MCTS simulations per move. This metric focuses on sample efficiency, but ignores the potential time cost of the novelty enhancements. We therefore conducted additional experiments to measure this time overhead.

In a first set of experiments, we measured the overhead directly. We ran 1000 MCTS searches from the initial state of each game and compared the average time needed by baseline MCTS to the average time needed by novelty-guided MCTS with the novelty measure $N_C^{\text{raw}}$. The other measures showed similar results leading to the same conclusions. Table 5 shows the results for the variants using handcoded evaluation functions to guide the search, which are very fast and efficient, but on average weaker than the learned heuristics. The baselines have a speed of ca. 45k to 75k MCTS simulations per second on our hardware, depending on the domain. Here, novelty-guided exploration leads to a significant slowdown of MCTS. Table 6 shows the results for the variants guided by neural networks, which constitute the state-of-the-art approach, but even with our relatively small networks (as described in Subsection IV-B) have a speed of only 1000-1500 simulations per second. Here, we can observe that the additional overhead for novelty guidance is almost negligible.

In a second set of experiments, we compared baseline MCTS to novelty-guided MCTS in actual gameplay at equal time of 1 second per move, with all algorithms re-tuned for this setting. Our hypothesis was that novelty guidance would still be effective at equal time for MCTS using neural networks, due to the small relative overhead, but ineffective for MCTS using handcoded heuristics, which gets slowed down much more. We here present the results for $N_T^{\text{raw}}$, as the other novelty measures showed similar behavior (see Appendix B). The results for handcoded heuristics and for learned heuristics are shown in Tables VIII and IX, respectively. Interestingly, we can observe that novelty-guided exploration can enhance MCTS play in both settings, with 3 out of 5 domains showing a significant improvement using handcoded heuristics, and 5 out of 6 domains showing a significant improvement using learned neural network heuristics. In the case of the slower network heuristics, this seems to be due to the small relative overhead of novelty computations. In the case of the faster handcoded heuristics however, the reason seems to be that they are still fast enough even with the added novelty guidance – or in other

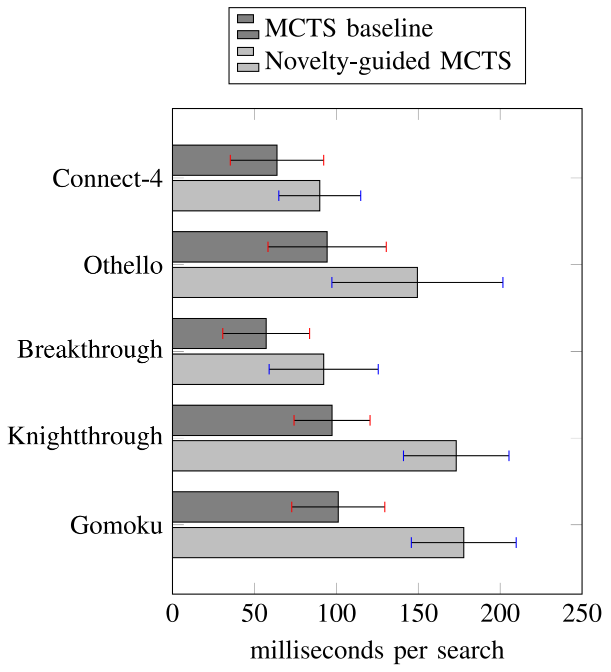

Fig. 5: Speed of novelty-enhanced MCTS (using $N_C^{\text{raw}}$) vs. baseline MCTS, using handcoded search guidance. In milliseconds for an average 5000 simulation move search from the initial state. Error bars show ± 1 standard deviation.

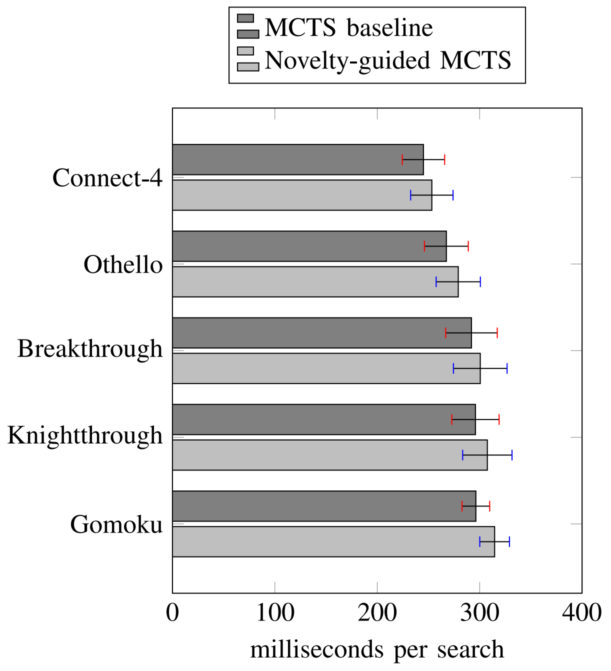

Fig. 6: Speed of novelty-enhanced MCTS (using $N_C^{\text{raw}}$) vs. baseline MCTS, using learned (neural network) search guidance. In milliseconds for an average 500 simulation move search from the initial state. Error bars show ± 1 standard deviation.

words, they are so fast that there are diminishing returns from doing extra MCTS simulations, and it can be worth exchanging them for novelty-enhanced exploration. These results indicate that using novelty as an auxiliary objective of search can be effective for a wide range of both fast and slow simulators.

| Game | win rate against baseline |
|---|---|
| Connect4 | **54.6%** (52.3%-56.7%) |
| Othello | 51.8% (49.6%-54.0%) |
| Breakthrough | **56.4%** (54.2%-58.6%) |
| Knightthrough | **68.7%** (66.6%-70.7%) |
| Gomoku | 51.9% (49.7%-54.1%) |
| Average | 56.7% |

TABLE VIII: Winrate of MCTS using raw-state threshold-based novelty ($N_T^{\text{raw}}$) vs. baseline MCTS, using UCT and handcoded heuristics, at 1 second per move. Intervals indicate 95% confidence.

| Game | win rate against baseline |
|---|---|
| Connect4 | **57.1%** (54.9%-59.3%) |
| Othello | **61.0%** (58.8%-63.1%) |
| Breakthrough | **56.3%** (54.0%-58.4%) |
| Knightthrough | 50.3% (48.1%-52.5%) |
| AtariGo | **52.9%** (50.7%-55.1%) |
| Gomoku | **58.7%** (56.5%-60.8%) |
| Average | 56.1% |

TABLE IX: Winrate of MCTS using raw-state threshold-based novelty ($N_T^{\text{raw}}$) vs. baseline MCTS, using PUCT and neural networks, at 1 second per move. Intervals indicate 95% confidence.

## V. Conclusions and Further Work

In this work, we employed four different state novelty measures with the goal of improving the exploration behavior of MCTS. Results in several board games were promising.

Our method of linearly combining novelty scores with MCTS value estimates exposes a nuance that may be more subtle than using novelty as an added reward bonus in the literature on intrinsic motivation, or as the sole objective in evolutionary search: novelty is here used as a separable heuristic value estimate. This online estimate of *novelty-as-a-value* encodes an optimistic prior: it ascribes higher value to states that are more novel and thus uncertain, and generalises novelty across features. It is combined with the regular action value estimates of the MCTS tree using a weight factor. This weight factor decays to zero as online observations replace the prior that biases the exploration of search. With novelty providing a form of *optimistic online generalisation of uncertainty*, it appears complementary to the exploration/uncertainty term of UCB1 when sufficiently many samples are available.

Future work includes scaling up to more, and more varied, test domains. Multiple types of novelty could be combined during search in order to exploit different ways of generalizing uncertainty online, for example as in *Multiple Estimator MCTS* [26]. Novelty could also be compared to, and combined with, a variety of MCTS enhancements that generalize *value* online, such as for example RAVE and its variants [23], MAST/PAST/FAST [27], or OMA [28].

Future work could also further extend the use of novelty-based approaches specifically in MCTS guided by neural networks, by integrating them in the learning stages of AlphaZero or MuZero frameworks [5], [29]. Thereby, novelty could potentially help improve both learning *and* planning performance.

## Acknowledgments

The authors would like to thank Cesare Carissimo for the interesting discussions on novelty and its motivations.

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). This work used the Dutch national e-infrastructure with the support of the SURF Cooperative using grant no. EINF-7323.

## References

[1] L. Kocsis and C. Szepesvári, "Bandit Based Monte-Carlo Planning," in *17th European Conference on Machine Learning (ECML 2016)*, 2006, pp. 282–293.

[2] H. Finnsson, "Generalized Monte-Carlo Tree Search Extensions for General Game Playing," in *Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI 2012)*, J. Hoffmann and B. Selman, Eds. AAAI Press, 2012.

[3] D. P. Liebana, S. Samothrakis, J. Togelius, T. Schaul, S. M. Lucas, A. Couëtoux, J. Lee, C. Lim, and T. Thompson, "The 2014 General Video Game Playing Competition," *IEEE Trans. Comput. Intell. AI in Games*, vol. 8, no. 3, pp. 229–243, 2016.

[4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the Game of Go without Human Knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[5] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-Play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[6] J. Lehman and K. O. Stanley, "Abandoning Objectives: Evolution Through the Search for Novelty Alone," *Evol. Comput.*, vol. 19, no. 2, pp. 189–223, 2011.

[7] H. Baier and M. Kaisers, "Novelty and MCTS," in *GECCO '21: Genetic and Evolutionary Computation Conference*, K. Krawiec, Ed. ACM, 2021, pp. 1483–1487.

[8] J. Lehman and K. O. Stanley, "Exploiting Open-Endedness to Solve Problems Through the Search for Novelty," in *Artificial Life XI: Eleventh International Conference on the Synthesis and Simulation of Living Systems*, S. Bullock, J. Noble, R. A. Watson, and M. A. Bedau, Eds., 2008, pp. 329–336.

[9] N. Lipovetzky and H. Geffner, "Width and Serialization of Classical Planning Problems," in *20th European Conference on Artificial Intelligence (ECAI 2012)*, ser. Frontiers in Artificial Intelligence and Applications, L. D. Raedt, C. Bessiere, D. Dubois, P. Doherty, P. Frasconi, F. Heintz, and P. J. F. Lucas, Eds., vol. 242, 2012, pp. 540–545.

[10] N. Lipovetzky, M. Ramírez, and H. Geffner, "Classical Planning with Simulators: Results on the Atari Video Games," in *Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, Q. Yang and M. J. Wooldridge, Eds., 2015, pp. 1610–1616.

[11] T. Geffner and H. Geffner, "Width-Based Planning for General Video-Game Playing," in *Eleventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2015)*, A. Jhala and N. Sturtevant, Eds., 2015, pp. 23–29.

[12] A. Shleyfman, A. Tuisov, and C. Domshlak, "Blind Search for Atari-Like Online Planning Revisited," in *Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI 2016)*, S. Kambhampati, Ed., 2016, pp. 3251–3257.

[13] M. Katz, N. Lipovetzky, D. Moshkovich, and A. Tuisov, "Adapting Novelty to Classical Planning as Heuristic Search," in *Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)*, L. Barbulescu, J. Frank, Mausam, and S. F. Smith, Eds., 2017, pp. 172–180.

[14] A. Aubret, L. Matignon, and S. Hassas, "A survey on intrinsic motivation in reinforcement learning," *CoRR*, vol. abs/1908.06976, 2019.

[15] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying Count-Based Exploration and Intrinsic Motivation," in *Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS 2016)*, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 1471–1479.

[16] H. Tang, R. Houthooft, D. Foote, A. Stooke, X. Chen, Y. Duan, J. Schulman, F. D. Turck, and P. Abbeel, "#Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning," in *Thirtieth Annual Conference on Neural Information Processing Systems (NIPS 2017)*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 2753–2762.

[17] G. Ostrovski, M. G. Bellemare, A. van den Oord, and R. Munos, "Count-Based Exploration with Neural Density Models," in *34th International Conference on Machine Learning (ICML 2017)*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70, 2017, pp. 2721–2730.

[18] Y. Burda, H. Edwards, A. J. Storkey, and O. Klimov, "Exploration by Random Network Distillation," *CoRR*, vol. abs/1810.12894, 2018.

[19] M. C. Machado, M. G. Bellemare, and M. Bowling, "Count-Based Exploration with the Successor Representation," in *Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2020)*, 2020, pp. 5125–5133.

[20] J. Martin, S. N. Sasikumar, T. Everitt, and M. Hutter, "Count-Based Exploration in Feature Space for Reinforcement Learning," in *Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI 2017)*, C. Sierra, Ed., 2017, pp. 2471–2478.

[21] D. J. N. J. Soemers, C. F. Sironi, T. Schuster, and M. H. M. Winands, "Enhancements for Real-time Monte-Carlo Tree Search in General Video Game Playing," in *2016 IEEE Conference on Computational Intelligence and Games (CIG 2016)*. IEEE, 2016, pp. 1–8.

[22] M. Tot, M. Conserva, D. P. Liebana, and S. Devlin, "Turning zeroes into non-zeroes: Sample efficient exploration with monte carlo graph search," in *IEEE Conference on Games (CoG 2022)*. IEEE, 2022, pp. 300–306.

[23] S. Gelly and D. Silver, "Combining Online and Offline Knowledge in UCT," in *Twenty-Fourth International Conference on Machine Learning (ICML 2007)*, ser. ACM International Conference Proceeding Series, Z. Ghahramani, Ed., vol. 227. ACM, 2007, pp. 273–280.

[24] S. Kistemaker and S. Whiteson, "Critical factors in the performance of novelty search," in *13th Annual Genetic and Evolutionary Computation Conference, GECCO 2011*, N. Krasnogor and P. L. Lanzi, Eds. ACM, 2011, pp. 965–972.

[25] R. P. Wiegand, "The objective of simple novelty search," in *Proceedings of the Thirty-Third International Florida Artificial Intelligence Research Society Conference*, R. Barták and E. Bell, Eds. AAAI Press, 2020, pp. 166–171.

[26] H. Baier and M. Kaisers, "ME-MCTS: Online Generalization by Combining Multiple Value Estimators," in *30th International Joint Conference on Artificial Intelligence (IJCAI 2021)*, 2021.

[27] H. Finnsson and Y. Björnsson, "Learning Simulation Control in General Game-Playing Agents," in *Twenty-Fourth AAAI Conference on Artificial Intelligence, (AAAI 2010)*, M. Fox and D. Poole, Eds. AAAI Press, 2010.

[28] H. Baier and M. Kaisers, "Guiding Multiplayer MCTS by Focusing on Yourself," in *2020 IEEE Conference on Games (CoG 2020)*, 2020, pp. 550–557.

[29] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. P. Lillicrap, and D. Silver, "Mastering atari, go, chess and shogi by planning with a learned model," *CoRR*, vol. abs/1911.08265, 2019.

[30] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time Analysis of the Multiarmed Bandit Problem," *Mach. Learn.*, vol. 47, no. 2-3, pp. 235–256, 2002.

[31] S. M. Lucas, J. Liu, and D. Pérez-Liébana, "The N-Tuple Bandit Evolutionary Algorithm for Game Agent Optimisation," in *2018 IEEE Congress on Evolutionary Computation (CEC 2018)*. IEEE, 2018, pp. 1–9.

## Appendix A
## On Parameter Tuning

As discussed in Section IV, vanilla MCTS has one parameter: the exploration factor $k$ of UCB1. All novelty-based approaches have at least two additional parameters: a parameter $\alpha$ that controls the magnitude of the novelty term, and a parameter $\beta$ that controls how quickly the influence of the novelty term on the search decreases. $N_T^{\text{raw}}$ additionally uses a parameter $\gamma$ that controls the novelty frequency threshold. This appendix describes in more detail how these parameters were tuned.

**Discretization of parameter spaces.**

All parameters were tuned with discrete optimization methods based on multi-armed bandits. The parameters were therefore discretized. As the optimal order of magnitude of the parameters was not known in advance, an approximately logarithmic discretization of parameter spaces was used, spread over all orders of magnitude that appeared to be promising across all experimental settings and domains. The discretization was coarse enough to make tuning in reasonable time possible within our computational means, while fine enough to not lead to significant changes in experimental results. We tested this by occasionally tuning with finer discretization in the orders of magnitude found optimal by our original tuning approach.

For the experiments reported in the main article and Appendix B, baseline MCTS players were tuned with $k \in \{0.01, 0.02, 0.03, 0.06, 0.1, 0.2, 0.3, 0.6, 1, 2, 3\}$. The exploration factors for novelty-enhanced players were restricted to five values on this scale centered on the optimal baseline value; for example, if the baseline MCTS value was $k = 0.1$, novelty players for the same domain and time setting were tuned with $k \in \{0.03, 0.06, 0.1, 0.2, 0.3\}$. Larger deviations were generally not found to be helpful. The ranges for the novelty-specific parameters were $\alpha, \beta \in \{0, 2 \times 10^{-5}, 3 \times 10^{-5}, 6 \times 10^{-5}, 10^{-4}, 2 \times 10^{-4}, \ldots, 3, 6, 10\}$, and $\gamma \in \{0, 10^{-5}, 2 \times 10^{-5}, 3 \times 10^{-5}, \ldots, 0.03, 0.06, 0.1\}$.

**Bandit-based tuning of vanilla MCTS baselines.**

Since the vanilla MCTS baselines have only one parameter, the exploration factor $k$, they were tuned with a simple approach based on multi-armed bandits. For each test domain, guiding heuristic (handcoded or learned), and time setting, a UCB1-Tuned bandit algorithm [30] was run for 10k matches, with the possible parameter values described above corresponding to the arms. The most-sampled value of $k$ was then chosen as the MCTS baseline for the given domain, guiding heuristic, and time setting.

The opponents for these matches were cycling through a set of MCTS baselines with different exploration factors in $\{0.06, 0.1, 0.2, 0.3, 0.6\}$, guided by the same heuristics as the tuned player, but covering a range of behaviors from more explorative to more exploitative. The single parameter $k$ does not give much flexibility to overfit, and we found the best values for $k$ to generally outperform all other values in the given domain and time setting.

**N-Tuple Bandit tuning of novelty-enhanced players.**

Since the novelty-enhanced players have between two and three parameters with 18 to 25 possible values each, a single multi-armed bandit was not able to tune them within reasonable time - it would have needed up to $25 \times 25 \times 18 = 11,250$ arms. We therefore opted for a tuning approach based on the N-Tuple Bandit Evolutionary Algorithm (NTBEA) [31]. We used an N-Tuple fitness landscape model with a separate multi-armed bandit for each dimension of the search space (each parameter), as well as for each set of two dimensions (pairs of parameters). Please refer to [31] for further details.

The main difference between our approach and the classic NTBEA as described in [31] is that we did not choose the next player to sample with the help of an EA; instead, we chose it by requesting promising parameter values from the same bandits used by the fitness landscape model, in *random order* until all parameters of the next player had a value. In this way, both promising individual parameter settings as well as promising settings for pairs of parameters have a chance to be explored, regardless of the previously sampled player and without any notion of "neighborhood". When using an additional global bandit for all parameters, this approach can be made to converge to the global optimum in the limit, by slowly phasing out all bandits that speed up search by only considering a lower-dimensional projection of the search space. Given the success of this NTBEA variant, it may itself merit rigorous examination in future work.

For each test domain, guiding heuristic (handcoded or learned), and time setting, we ran this algorithm 8 times for 3k matches per run (restarts to avoid local minima). The best players found by each run were then compared based on an additional 1k matches each, determining the final player to be returned.

The opponents for these matches were the best vanilla MCTS baselines found as described in the previous section. Since the optimal amount of exploration is opponent-dependent, we chose to tune Novelty MCTS parameters as a best response to the given baseline - here Vanilla MCTS. We also tried tuning against independent, third opponents; but due to intransitive relations between strategies, having stronger performance against a third opponent than a baseline tuned against the same opponent does not necessarily mean beating that baseline as well. We also tried tuning against the same set of vanilla MCTS opponents our baselines were tuned against; but that sometimes led to players that exploit the weaker opponents in the set, instead of learning to outperform the strongest baselines. Since we were interested in whether novelty-enhanced players can outperform the strongest baselines, we chose these as tuning opponents, leaving further exploration of more broadly generalizing parameter settings for different distributions over opponents to future work.

## Appendix B
## Additional Results at Equal Time per Move

In Subsection IV-D, we show results at equal time per move for novelty measure $N_T^{\text{raw}}$ in Tables VIII and IX, and mention that the other novelty measures showed very similar behavior. In order to demonstrate this in detail, this appendix presents analogous results for all novelty measures. These results include new results for $N_T^{\text{raw}}$ that are slightly different from those in Subsection IV-D, because all results in this appendix had to be re-computed on newer hardware. We did this at 250ms instead of 1s per move to achieve comparable results to the main paper, as the new hardware was considerably faster than the original hardware used for the experiments in the main article.

Tables X to XIII show the results of MCTS guided by hand-coded heuristics. Table X replicates the results of Table VIII on faster hardware and with shorter time limits. Note that the results for $N_E^{\text{raw}}$ and $N_C^{\text{raw}}$ are even more promising than those shown for $N_T^{\text{raw}}$ here and in the main article, with 4 of 5 (Table XIII) and 5 of 5 domains (Table XI) showing a significant improvement, respectively.

| Game | win rate against baseline |
|---|---|
| Connect4 | **52.9%** (50.7%-55.1%) |
| Othello | 50.4% (48.2%-52.6%) |
| Breakthrough | **60.3%** (58.1%-62.4%) |
| Knightthrough | **75.3%** (73.3%-77.2%) |
| Gomoku | 52.1% (49.9%-54.3%) |
| Average | 58.2% |

TABLE X: Winrate of MCTS using raw-state threshold-based novelty ($N_T^{\text{raw}}$) vs. baseline MCTS, using UCT and handcoded heuristics, at 250ms per move. Intervals indicate 95% confidence.

| Game | win rate against baseline |
|---|---|
| Connect4 | **55.4%** (53.2%-57.6%) |
| Othello | **53.4%** (51.2%-55.6%) |
| Breakthrough | **57.3%** (55.1%-59.5%) |
| Knightthrough | **76.6%** (74.7%-78.5%) |
| Gomoku | **58.9%** (56.7%-61.0%) |
| Average | 60.3% |

TABLE XI: Winrate of MCTS using raw-state pseudocount novelty ($N_C^{\text{raw}}$) vs. baseline MCTS, using UCT and handcoded heuristics, at 250ms per move. Intervals indicate 95% confidence.

| Game | win rate against baseline |
|---|---|
| Connect4 | **53.3%** (51.1%-55.6%) |
| Othello | 49.5% (47.3%-51.7%) |
| Breakthrough | **62.4%** (60.2%-64.5%) |
| Knightthrough | **70.6%** (68.5%-72.5%) |
| Gomoku | 52.1% (49.9%-54.3%) |
| Average | 57.6% |

TABLE XII: Winrate of MCTS using feature-based pseudo-count novelty ($N_C^{\text{eval}}$) vs. baseline MCTS, using UCT and handcoded heuristics, at 250ms per move. Intervals indicate 95% confidence.

Tables XIV to XVI show the results for MCTS guided by neural networks. Table XIV is replicating the results of Table IX on faster hardware and with shorter time limits.

| Game | win rate against baseline |
|---|---|
| Connect4 | **54.4%** (52.2%-56.6%) |
| Othello | **52.8%** (50.5%-55.0%) |
| Breakthrough | **59.5%** (57.3%-61.6%) |
| Knightthrough | **74.9%** (72.9%-76.7%) |
| Gomoku | 52.5% (50.3%-54.7%) |
| Average | 58.8% |

TABLE XIII: Winrate of MCTS using raw-state evaluation novelty ($N_E^{\text{raw}}$) vs. baseline MCTS, using UCT and handcoded heuristics, at 250ms per move. Intervals indicate 95% confidence.

| Game | win rate against baseline |
|---|---|
| Connect4 | **55.9%** (53.7%-58.1%) |
| Othello | **59.9%** (57.7%-62.1%) |
| Breakthrough | **54.4%** (52.1%-56.5%) |
| Knightthrough | 50.8% (48.6%-53.1%) |
| AtariGo | **54.8%** (52.6%-57.0%) |
| Gomoku | **57.3%** (55.1%-59.5%) |
| Average | 55.5% |

TABLE XIV: Winrate of MCTS using raw-state threshold-based novelty ($N_T^{\text{raw}}$) vs. baseline MCTS, using PUCT and neural networks, at 250ms per move. Intervals indicate 95% confidence.

| Game | win rate against baseline |
|---|---|
| Connect4 | **53.9%** (51.7%-56.1%) |
| Othello | **62.3%** (60.1%-64.4%) |
| Breakthrough | **53.8%** (51.6%-56.1%) |
| Knightthrough | 49.9% (47.6%-52.1%) |
| AtariGo | **53.1%** (50.9%-55.3%) |
| Gomoku | **56.2%** (54.0%-58.4%) |
| Average | 54.9% |

TABLE XV: Winrate of MCTS using raw-state pseudocount novelty ($N_C^{\text{raw}}$) vs. baseline MCTS, using PUCT and neural networks, at 250ms per move. Intervals indicate 95% confidence.

| Game | win rate against baseline |
|---|---|
| Connect4 | 51.3% (49.1%-53.6%) |
| Othello | **58.2%** (56.0%-60.4%) |
| Breakthrough | **54.8%** (52.6%-57.0%) |
| Knightthrough | 48.4% (46.1%-50.6%) |
| AtariGo | **54.3%** (52.0%-56.4%) |
| Gomoku | **56.0%** (53.7%-58.1%) |
| Average | 53.8% |

TABLE XVI: Winrate of MCTS using raw-state evaluation novelty ($N_E^{\text{raw}}$) vs. baseline MCTS, using PUCT and neural networks, at 250ms per move. Intervals indicate 95% confidence.