Planning and Learning using Adaptive Entropy Tree Search

Piotr Kozakowski  
Faculty of Mathematics, Informatics and Mechanics  
University of Warsaw  
Warsaw, Poland  
p.kozakowski@mimuw.edu.pl

Mikołaj Pacek  
Faculty of Mathematics, Informatics and Mechanics  
University of Warsaw  
Warsaw, Poland  
m.pacek2@student.uw.edu.pl

Piotr Miłoś  
Institute of Mathematics of the Polish Academy of Sciences  
Warsaw, Poland  
pmilos@impan.pl

Abstract—Recent breakthroughs in Artificial Intelligence have shown that the combination of tree-based planning with deep learning can lead to superior performance. We present Adaptive Entropy Tree Search (ANTS) - a novel algorithm combining planning and learning in the maximum entropy paradigm. Through a comprehensive suite of experiments on the Atari benchmark we show that ANTS significantly outperforms PUCT, the planning component of the state-of-the-art AlphaZero system. ANTS builds upon recent work on maximum entropy planning methods - which however, as we show, fail in combination with learning. ANTS resolves this issue to reach state-of-the-art performance. We further find that ANTS exhibits superior robustness to different hyperparameter choices, compared to the previous algorithms. We believe that the high performance and robustness of ANTS can bring tree search planning one step closer to wide practical adoption.

Index Terms—maximum entropy, monte carlo tree search, planning, learning

I. INTRODUCTION

Planning and learning is a fundamental component of intelligent behavior in both biological and artificial agents [1]–[3]. On the one hand, planning can discover high-quality decisions but falls short in large state spaces. On the other hand, learning can harness powerful pattern matching techniques and discover useful state representations, hence significantly reducing the search space [4], [5].

Recently, Monte Carlo tree search (MCTS) planning methods [6], [7], which incorporate learning, played a major role in some of the breakthroughs in deep reinforcement learning (RL) and in the real world domains, ranging from playing complex strategic games [8], through testing security systems [9], physics simulations [10], production management [11], to creative content generation [12]. Consequently, there exists a large body of research aiming to improve the performance of MCTS [4], [13]–[15]. Importantly, MCTS algorithms are modular, allowing to incorporate a broad spectrum of ideas.

One such prominent idea is the Principle of Maximum Entropy (PME) [16], [17], which has improved the performance and increased the robustness of RL methods [18]–[20], and was combined with MCTS in [21], [22]. In particular, the latter papers introduce MENTS and TENTS algorithms, respectively, demonstrating how to utilize pre-trained value networks in maximum entropy planning, and also providing theoretical guarantees. However, the successful scaling of MENTS and TENTS to the online learning setting has been an open problem. In this paper, we show how to close this gap, by introducing Adaptive Entropy Tree Search (ANTS), the first maximum entropy MCTS method showing competitive performance in online learning on the Atari benchmark [23].

ANTS combines three algorithmic enhancements: (a) temperature adaptation, (b) leaf evaluation using raw Q-values, and (c) pseudoreward shaping. The eponymous temperature adaptation introduces a useful reparameterization of the algorithm: instead of controlling the temperature of the action selection distribution, we control the average entropy of that distribution across the nodes in the tree. This parameterization, as we show experimentally, is significantly more robust to different hyperparameter choices. It also has a more intuitive interpretation: the Shannon entropy of a discrete distribution can be regarded as a logarithm of its weighted average branching factor [24]. As such, by setting the level of entropy, we control the breadth of the search tree.

II. BACKGROUND

Reinforcement learning (RL) problems are typically formalized by Markov decision processes (MDPs). An MDP is a tuple \((S, A, \mathcal{P}, R, \gamma)\), where \(S, A\) are the state and action spaces, \(\mathcal{P}(\cdot|s,a)\) is the transition distribution, \(R\) is a scalar reward function and \(\gamma \in [0, 1)\) is the discount factor. The agent’s behavior is described by the policy \(\pi(\cdot|s), s \in S\).

Maximum entropy RL introduces the objective

\[
J(\pi) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t [R(s_t, a_t) + \tau \mathcal{H}(\pi(\cdot|s_t))] \right]
\]

where \(\tau > 0\) is a temperature parameter and \(\mathcal{H}\) is Shannon entropy:

\[
\mathcal{H}(\pi(\cdot|s_t)) = - \sum_{a \in A} \pi(a|s_t) \log \pi(a|s_t). \tag{1}
\]
[19] derives a backup operator, which can be used to solve entropy-regularized RL problems.\[7\]

\[
\begin{align*}
Q(s, a) &\leftarrow R(s, a) + \gamma \mathbb{E}_{s', \sim \pi(s, a)} V(s') \\
V(s) &\leftarrow \tau \log \sum_{a \in A} \exp \left( \frac{1}{\tau} Q(s, a) \right).
\end{align*}
\]

(2)

The fixed point of this backup results in $Q^*, V^*$. The policy $\pi^*(s) \propto \exp \left( \frac{1}{\tau} Q^*(s, a) \right)$ is optimal with respect to $J$.

a) Monte Carlo Tree Search (MCTS): MCTS is a class of planning algorithms in MDPs with discrete action spaces. Below we describe its version related in this paper, referring to [6] for a general discussion. MCTS iteratively builds a planning tree where nodes correspond to states and edges to (state, action) pairs. It consists of four phases: Selection, in which a path from the root to a leaf is selected; Expansion, in which all successors of this leaf are added to the tree as new leaves; Simulation, in which the values of the new leaves are estimated; and Backpropagation, in which the value estimates of the nodes on the path are updated, starting from the bottom.

AlphaZero [4] augments vanilla MCTS with neural network heuristics trained in a self-improvement loop. AlphaZero uses the PUCT selection strategy, in which at state $s$ the next edge/action is chosen to maximize

\[
\text{PUCT}(s, a) = Q(s, a) + c \pi_{\text{prior}}(a|s) \frac{\sqrt{N(s)}}{N(s, a) + 1}.
\]

(3)

where $c > 0$ is a hyperparameter, $\pi_{\text{prior}}(a|s)$ is a prior policy, $N(s)$ is the number of visits in state $s$ and $N(s, a)$ is the number of times action $a$ has been executed in state $s$. In the simulation phase, the value of a leaf is estimated using a value network. This network is trained to estimate the empirical state values, $\pi_{\text{prior}}$ is also a network, trained to predict $\pi_{\text{prior}}(a|s)$.

In this work we consider an expansion phase that adds all successors of a given node to the tree. These new leaves are initialized using a Q-network. The Q-network is trained to estimate the empirical state-action values.

After planning is finished, an action is selected according to a stochastic policy $\pi$ (this approach follows [13]):

\[
\pi(a) \propto N(s_{\text{root}}, a)^{1/\tau_{\text{sel}}},
\]

(4)

where $\tau_{\text{sel}} \in (0, 1)$ is a temperature hyperparameter. The stochastic $\pi$ ensures diverse data for training. If exploration is not needed, i.e., when not training neural networks, we use $\text{argmax}_a N(\text{root}, a)$ (equivalent to $\tau_{\text{sel}} \downarrow 0$).

b) MaxEnt MCTS: The maximum entropy backup [3] can be adapted to MCTS, resulting in the MENTS algorithm [21]. In the selection phase the consequent nodes are chosen according to the E3W sampling strategy:

\[
\begin{align*}
\pi_{\text{E3W}}(a|s) &= (1 - \lambda_s) \pi_\tau(a|s) + \lambda_s \frac{1}{|A|}, \\
\pi_\tau(a|s) &\propto \exp \left( \frac{1}{\tau} Q(s, a) \right),
\end{align*}
\]

(5)

\footnote{In the limit as $\tau \downarrow 0$ we recover the classical Bellman backup.}

where $\lambda_s = \frac{c|A|}{\log(N(s) + 1)}$, $\epsilon > 0$. In the simulation phase, leaves are initialized using:

\[
\begin{align*}
Q(s, a) &\leftarrow \hat{Q}(s, a) - \hat{V}(s) / \tau_{\text{init}}, \\
\hat{V}(s) &\leftarrow \tau \log \sum_{a \in A} \exp \left( \frac{1}{\tau} \hat{Q}(s, a) \right),
\end{align*}
\]

(6)

where $\hat{Q}$ is a Q-network. In the backpropagation phase, node $Q$-values are updated according to [2]. The algorithm described here utilizes Shannon entropy. It can be extended to other types of entropy; in particular [22] derive TENTS, a version using Tsallis entropy:

\[
\mathcal{H}(\pi(s)) = \frac{1}{2} \left( 1 - \sum_{a \in A} \pi(a|s) \right)^2.
\]

(7)

Because optimizing the maximum-entropy RL objective with Tsallis entropy can result in sparse policies, this formulation is particularly beneficial in environments with large action spaces [25].

Algorithm 1 Adaptive Entropy Tree Search.

function $\text{ANTS}(s_{\text{root}}, \hat{t})$

for $i$ in $\{1 \ldots n\}$ do

if $i$ mod $k = 0$ then

$\tau \leftarrow \max(\text{FIND_ROOT}(\mathcal{H}^+), \tau_{\text{min}})$

$\hat{t} \leftarrow \exp(\alpha \log \hat{t} + (1 - \alpha) \log \tau)$

$\text{RECALCULATE}_Q(s_{\text{root}}, \hat{t})$

$s_{\text{leaf}} \leftarrow \text{SELECTION}(s_{\text{root}}, \hat{t})$

children[$s_{\text{leaf}}$] $\leftarrow \text{EXPANSION}(s_{\text{leaf}})$

$\forall a \in A \ Q(s_{\text{leaf}}, a) \leftarrow \hat{Q}(s_{\text{leaf}}, a; \tau)$

$\text{BACKPROPAGATION}(s_{\text{leaf}}, \hat{t})$

sample $a \sim \pi_{\text{E3W}}(\cdot|s_{\text{root}})$

return $a, \hat{t}, Q(s_{\text{root}}, a; \hat{t})$

function $\text{RECALCULATE}_Q(s, \tau)$

for $a \in A$ do

$\text{RECALCULATE}_Q(\text{children}[s][a], \tau)$

$Q(s, a) \leftarrow \text{BACKUP}(s, a, \tau)$

At every timestep we make $n$ MCTS iterations. Every $k$ iterations we adapt the temperature and recalculate the $Q$-values in the tree. Temperature adaptation is described in Section III-A. $Q$-value recalculation involves a custom backup described in Section III-B. The selection and backpropagation phases are shown in Algorithm 2 below. The expansion phase is omitted for clarity. The simulation phase is inlined as a call to the $Q$-network; this is described in detail in Section III-C. Finally, we select an action by sampling from the E3W policy, sharpened by a separate temperature parameter $\tau_{\text{sel}}$, described in Section III-D.
Algorithm 1: Selection and backpropagation phases.

\begin{algorithm}
\begin{algorithmic}
\Function{Selection\(s, \tau\)}\end{algorithmic}
while not IS\_Leaf\(s\) do
\State sample \(a \sim \pi_{\widehat{Q}}^{{\text{E3W}}}(\cdot|s)\)
\State \(s \leftarrow \text{children}[s][a]\)
\Return \(s\)
\EndFunction
\Function{Backpropagation\(s, \tau\)}\end{algorithmic}
while not IS\_Root\(s\) do
\State \(N(s) \leftarrow N(s) + 1\)
\State \(Q(s, a) \leftarrow \text{Backup}(s, a, \tau)\)
\State \(s \leftarrow \text{parent}[s]\)
\EndFunction
\end{algorithm}

In the selection phase, we choose the consecutive nodes until we reach a leaf by sampling from the E3W policy \((5)\). In the backpropagation phase we recalculate the \(Q\)-values of the nodes on the path from the leaf to the root using the backup described in Section III-B.

In addition to the evaluation using pretrained Q-networks, investigated in [21], [22], we extend ANTS to work in the more practical setting of online learning, shown in Algorithm III-D. The learning loop interleaves phases of data collection and training. When collecting an episode, we initialize the temperature to a constant value \(\tau_0\), then pass and update it in the planner calls. We train the Q-network using SGD on samples from the replay buffer, with a loss function \(L\) described in Section III-D.

Algorithm 2: Online learning.

\begin{algorithm}
\begin{algorithmic}
\Function{Learning\_Loop}()
\While {not convergence}
\State episode \(\leftarrow \text{Collect\_Episode}\)
\State replay\_buffer \(\leftarrow\) replay\_buffer \(\oplus\) (episode)
\State\(\bar{Q} \leftarrow \text{SGD}(L, \text{batch}, \bar{Q})\)
\EndWhile
\EndFunction
\Function{Collect\_Episode}
\State \(s \leftarrow s_0; \tau \leftarrow \tau_0\)
\State episode \(\leftarrow (s_0)\)
\While {not IS\_Terminal\(s\)}
\State \(a, \tau, q \leftarrow \text{ANTS}(s, \tau)\)
\State sample \(s \sim \mathcal{P}(\cdot|s, a)\)
\State episode \(\leftarrow\) episode \(\oplus\) (\(a, \tau, q, s\))
\EndWhile
\Return episode
\EndFunction
\end{algorithm}
\end{algorithm}

The algorithms and equations in this paper refer to ANTS in combination with Shannon entropy [1]. It is possible to extend MaxEnt MCTS to other types of entropy, as shown in [22]. We evaluate our method also in a setting with Tsallis entropy [7]; we denote the versions with Shannon and Tsallis entropy as ANTS-S and ANTS-T, respectively.

A. Temperature adaptation

In our method, the temperature \(\tau\) is chosen considering entropies (depending on \(\tau\)) of nodes expanded by the MCTS planner (denoted by \(S_{\text{tree}}\)). Specifically, \(\tau\) is varied so that the mean node entropy matches the prescribed value \(H_{\text{avg}}\). This is achieved by finding a root of the following function:

\[
H^+(\tau) = \frac{1}{|S_{\text{tree}}|} \sum_{s \in S_{\text{tree}}} H(\pi_{\tau}(\cdot|s)) - H_{\text{avg}},
\]

where \(\pi_{\tau}(a|s) \propto \exp\left(\frac{1}{\tau} Q(s, a)\right)\).

Any root-finding algorithm can be applied here. For simplicity, we use the derivative-free Brent algorithm [26]. To avoid numerical issues when \(\tau \approx 0\), we impose a lower bound on the temperature: \(\tau_{\text{min}}\). To avoid instabilities, we smooth the temperature using an exponential moving average. We smooth in log-space to be able to quickly adapt the order of the magnitude. This results in the following two lines in Algorithm [1]

\[
\tau \leftarrow \max(\text{FIND\_ROOT}(H^+, \tau_{\text{min}}))
\]

\[
\hat{\tau} \leftarrow \exp(\alpha \log \hat{\tau} + (1 - \alpha) \log \tau),
\]

where \(\alpha\) is the parameter of the moving average.

B. Pseudoreward shaping

The \text{Backup} used in Algorithms [1] and [2] is a modification of [2]:

\[
Q(s, a; \tau) \leftarrow R(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s, a)} V(s'; \tau)
\]

\[
V(s; \tau) \leftarrow \tau \log \sum_{a \in A} \exp\left(\frac{1}{\tau} Q(s, a; \tau)\right) - \tau H_{\text{max}},
\]

where \(H_{\text{max}}\) is the maximum possible entropy. This modification is designed to prevent the inefficient depth-first behavior, which we observed in our experiments. Since values calculated by [2] include positive entropy pseudorewards, they overestimate the true returns. As a result, the planner can start deepening the same, single path, because it keeps receiving high values along it. The pseudoreward shaping term \(-\tau H_{\text{max}}\) in (8) mitigates this effect. We note that as the subtracted value does not depend on the action, this shaping does not change the optimal policy obtained at infinite computational budget.

C. Q-value estimation

In the simulation phase, we estimate the \(Q\)-values of the newly-expanded leaves in the tree using a given Q-network. We initialize their \(Q\)-values as \(Q(s, a) \leftarrow \hat{Q}(s, a; \tau)\), where \(\hat{Q}\) denotes the Q-network. This initialization method is a great deal simpler than the one used in the original MaxEnt MCTS formulation [6]. In addition to its simplicity, our method yields considerable improvement, as demonstrated in Section IV-D. Another important element is accommodating temperature changes during planning. For this purpose, we inject information about the current temperature \(\tau\) to the network - the \(Q\)-values are influenced by it. We have discovered that giving the network access to this information greatly stabilizes the algorithm.
D. Online learning

The main evaluation setting considered in this paper is that of online learning. Our learning loop is similar to the one used in [4] and shown in Algorithm 3. We emphasize that our algorithmic developments, presented above, and the learning details below were critical for making online learning feasible.

The loop consists of two phases: data collection and training. In the first phase, the planner is run for an episode, using a neural network for initializing the $Q$-values of leaves added to the tree. The episodes, along with state $Q$-values calculated by the planner, are stored in a replay buffer and used to train the neural network in the next phase. We note that the planner $Q$-values are typically more accurate than the ones from the network; this difference provides the learning signal.

During data collection, we use an exploration scheme similar to (4). Actions performed at each step in the environment are sampled from

$$
\pi(a) = \pi_{\tau, \tau_{sel}}(a|s_{root}),
$$

where $\tau_{sel} \in (0,1)$ is an action selection temperature parameter. We use it to interpolate between the E3W (5) and the argmax policy: $a(s) = \text{argmax}_{a \in A} Q(s, a)$. The network is trained with $L_2$ loss on the $Q$-value of the action performed in the collected experience:

$$
L(\hat{Q}, s, a, \tau, q) = (\hat{Q}(s, a; \tau) - q)^2
$$

IV. EXPERIMENTS

In our main series of experiments, we plan using $Q$-networks trained online, as described in Section III-D. Then we present experiments with $Q$-networks pretrained using DQN, as done in [21] and [22]. Next, we provide an empirical comparison between two parameterizations of maximum entropy planning: via temperature and via entropy. Finally, we present ablations of various components of ANTS using pretrained $Q$-networks.

We present the results of ANTS combined with two types of entropy: ANTS-S using Shannon entropy (1), and ANTS-T using Tsallis entropy (7). As baselines we include PUCT, MENTS, and TENTS. PUCT denotes MCTS with the PUCT selection strategy. We run our experiments on a set of 21 Atari games and using the perfect model of the environment, in a setting similar to [22]. To ensure a fair comparison, we ran an extensive hyperparameter tuning of all presented methods, using equal computational budgets. The code and hyperparameter settings are available at https://github.com/adaptive-entropy-tree-search/ants.

A. Main experiment: online learning

In this section we present experiments showing that ANTS is the first maximum entropy MCTS method competitive in the online learning setting. Specifically, ANTS significantly exceeds the results of the prior methods MENTS and TENTS. Moreover, it outperforms the state-of-the-art PUCT (utilized in AlphaZero and MuZero), at the same time being more stable.

As we can see in Figure 1, the performances of both ANTS-S and ANTS-T consistently improve as they collect more data. Both algorithms exhibit similar performance, though ANTS-T appears slightly more sample efficient, while ANTS-S learns more steadily and has lower variance between seeds. PUCT learns slowly and unsteadily, and its median performance eventually starts to deteriorate. MENTS and TENTS quickly plateau at a low score, which confirms the importance of the improvements made in ANTS in the online learning setting.

In these experiments, to collect diverse data for network training, we employ the exploration strategy (9) for ANTS, MENTS, and TENTS, and (4) for PUCT. The hyperparameter $\tau_{sel}$ is tuned for each method separately. We run five experiments for each algorithm and game, each collecting 28000 episodes in total. An important technical step towards stabilizing training was providing the current temperature as an additional input to the $Q$-network, as described in Section III-C.

B. Pretrained $Q$-networks

In this section we present experiments using pretrained $Q$-networks. This evaluation setting allows us to test the properties of the planners in isolation from learning.
shown in Table I. As we can see, ANTS-S performs strongly, achieving the best result or a tie on 14 out of 21 games, on others being close to the best result. Our highly tuned PUCT presents a challenging baseline, outperforming both MENTS and TENTS in almost every case. Both ANTS-S and ANTS-T perform significantly better than MENTS and TENTS, which provides evidence that the improvements made in ANTS are beneficial also when the Q-network is supplied externally.

Interestingly, the performance of both variants of ANTS using pretrained networks is on-par with PUCT, whereas in the online learning experiments ANTS performed better. These results give rise to a hypothesis that the advantage of ANTS over PUCT in online learning is caused by providing better data for training neural networks, rather than better planning using a given neural network.

In Table I we report the mean of 100 episodes. Each episode is run for $10^4$ interactions. For each environment, we bold the methods for which the difference with the highest score is not statistically significant (determined using a $t$-test with $p < 0.05$).

For those experiments, in ANTS, MENTS and TENTS we select an action with the maximum $Q$-value returned by the planner. In PUCT we select an action with the highest visitation count. Similar to [21], [22], we use networks pretrained with DQN for leaf initialization. For this purpose, we utilize the publicly available checkpoints from the Dopamine package [27].

### C. Robustness

We now present a comparison of two parameterizations of maximum entropy planners: by mean entropy, used by ANTS, and by temperature, used by MENTS and TENTS. We evaluate the robustness to different parameter settings of the two parameterizations, as well as their performance when the hyperparameters are tuned.

We measure the robustness by the mean variance score, described below, with lower scores indicating higher robustness. The results are shown in Figure 2.

As we can see, ANTS parameterized by entropy is more robust than the variant parameterized by temperature. In the case of ANTS-S, mean variance of the entropy parameterization is 2 times lower than of the temperature parameterization. In the case of ANTS-T, the difference is much greater: over 8 times. In terms of the absolute performance, ANTS-T parameterized by entropy is noticeably better than its temperature-parameterized counterpart. In the case of ANTS-S, the difference is negligible.

Intuitively, an algorithm can be said to be robust to different hyperparameter choices, if its performance does not vary too much with different hyperparameter settings. To capture this intuition, we define our robustness metric as the variance of human-normalized scores attained at different hyperparameter values, where hyperparameters are chosen from a predefined set. We compute this metric separately for each game, then average it over games:

$$\rho(A, H) = \mathbb{E}_{g \sim G} \text{Var}_{h \sim H} \text{score}(A, h, g),$$

(10)

where $A$ indicates the algorithm using a given parameterization, $H$ is a set of its possible hyperparameter values, $G$
is the set of games, \( \sim \) denotes sampling from the uniform distribution over a finite set, and \( \text{score}(A, h, g) \) is the score of algorithm \( A \) using hyperparameter \( h \) on game \( g \).

To ensure a fair comparison, the hyperparameter sets \( H \) should be of equal size and span the range of sensible values. For the temperature parameterization, the range of possible temperature values is \((0, \infty)\). To handle this infinite interval we choose temperatures equidistant in log-space, with endpoints selected to be the minimum and maximum temperature that worked best on some game during hyperparameter tuning. In the case of entropy parameterization, the interval is \((0, H_{\text{max}})\), where \( H_{\text{max}} \) depends on the entropy type and number of actions. We take values spanning this interval, equidistant in linear space.

D. Ablations

In order to measure the contribution of the different components of ANTS, we perform additional experiments with pretrained networks, removing the individual features of our method. For each variant we report the mean and median human-normalized score. The results are shown in Figure 3. ANTS denotes the full algorithm using Shannon entropy. NO-SHAPING denotes ANTS without pseudoreward shaping and MENTS-EVAL denotes a variant of ANTS where leaf initialization is performed using (6), as done in [21] and [22]. We additionally include NO-E3W, denoting ANTS that samples directly from \( \pi(a|s) \propto \exp(Q(s, a)/\tau) \) during the selection phase, instead of using E3W.

In Figure 3 we can see that \( Q \)-value estimation using raw outputs of the Q-network plays the largest role out of the evaluated components of ANTS. One possible explanation is that leaf initialization using formula (6) retains only information about relations between actions, losing information about the absolute value of a state. This could prevent the planner from a meaningful comparison of nodes higher in the tree. In the case of pseudoreward shaping, the difference is less apparent and only visible in the median human-normalized score plot. Interestingly, NO-E3W performs worse than ANTS in terms of mean score and better in terms of median score. It appears that E3W sampling is necessary on a small number of games, which escalates the mean score, but hurts on most games, which diminishes the median score. It is also worth noticing that the median NO-E3W score has considerably higher variance.

The ablations use the same base hyperparameter as the experiments in Section IV-B. In the MENTS-EVAL experiment, we re-tune \( \tau_{\text{init}} \) jointly on all games, maximizing the mean human-normalized score. In both cases, we use the same set of hyperparameter values as when tuning MENTS.

V. RELATED WORK

Reinforcement Learning
Recent years have brought significant advancements in reinforcement learning due to the use of powerful deep neural network (DNN) function approximators. [28] introduce DQN: Q-learning with DNNs, leading to a human-level performance on the Arcade Learning Environment [23]. The algorithmic progress has been perhaps equally important. [29] combine multiple improvements to DQN, enabling superior performance across the ALE benchmark. [30] advance even further, leading to an algorithm that achieves superhuman performance on all 57 ALE tasks.

Monte Carlo Tree Search
MCTS is a combination of the UCB strategy for multi-armed bandits with tree search [7]. [6] provide a survey of methods in the MCTS family. [4] introduce AlphaZero - MCTS powered with neural networks and self-play, to obtain a system achieving superhuman performance in complex strategic board games: chess, shogi and go. [13] develop MuZero, an extension of AlphaZero with a learned model, and establish a new state-of-the-art on ALE. [14] discover a link between MCTS and policy optimization and derive a novel selection strategy.

Maximum Entropy RL
The Principle of Maximum Entropy (PME) has shown success in modeling human behavior [16]. [31] describe the maximum entropy RL problem as inference in a probabilistic graphical model. [18] analyze the maximum entropy RL problem and show its equivalence to classes of MDPs with uncertain or adversarial reward functions. [32] establish a connection between policy-based and value-based RL under entropy regularization. [19] introduce Soft Q-Learning: a deep RL method derived from soft Q-iteration [2]. [20] present another maximum entropy RL method: Soft Actor-Critic (SAC), achieving state-of-the-art performance in continuous control tasks. [21] use the maximum entropy framework to derive E2W: an optimal sampling strategy for the softmax stochastic bandit problem and introduce MENTS - an MCTS algorithm using this strategy in the selection phase. [22] extend E2W to different types of entropy: Tsallis entropy and relative Shannon entropy, resulting in the E3W sampling strategy. Our work differs from both [21] and [22].
by the introduction of temperature adaptation, simple $Q$-value estimation, and pseudoword shaping.

VI. CONCLUSIONS

In this work, we have presented Adaptive Entropy Tree Search: the first maximum entropy MCTS method competitive in the online learning setting. It delivers superior performance compared to the prior MaxEnt MCTS algorithms MENTS and TENTS, and outperforms the state-of-the-art PUCT. Moreover, ANTS is on par with PUCT when used as a standalone planner, with pretrained neural networks. Importantly, ANTS considerably improves upon the related MENTS and TENTS algorithms in both aspects. We have also demonstrated that the entropy reparameterization introduced by ANTS leads to increased robustness to hyperparameter settings. This makes ANTS particularly viable in real-world applications, where hyperparameter tuning is expensive. Finally, we have verified the design choices in a comprehensive ablation study.

We believe that a promising direction for future research is further analysis of maximum entropy approaches in combination with online learning. The high performance of ANTS in this setting can be attributed to good exploration or exploitation. Both can play a role - on the one hand, maximum entropy planning can be seen as an alternative method of exploration to UCT/PUCT; on the other hand, maximum entropy RL approaches are believed to aid in smoothing the optimization landscape of neural networks [33]. We believe that this line of research would serve as a step towards developing practical planning approaches to challenging real-world problems.

ACKNOWLEDGMENTS

We thank Łukasz Kuc'iński and Konrad Czechowski for their helpful suggestions on the writing. The work of Piotr Kozakowski was supported by the Polish National Science Center grant 2020/37/N/ST6/04113. The work of Mikołaj Pacek was supported by the Digital Poland Operational Programme, financed by the European Structural and Investment Fund, as part of the project AI Tech (POPC.03.02.00-IP01-00-006/20). The work of Piotr Miloś was supported by the Polish National Science Center grant 2017/26/E/ST6/00622. This research was supported by the PL-Grid Infrastructure. Our experiments were managed using https://neptune.ai. We thank the Neptune team for providing us access to the team version and technical support.

REFERENCES

[1] D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick, “Neuroscience-inspired artificial intelligence,” Neuron, vol. 95, no. 2, pp. 245–258, 2017.
[2] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed. USA: Prentice Hall Press, 2009.
[3] Y. Bengio, A. Lodi, and A. Prouvost, “Machine learning for combinatorial optimization: A methodological tour d’horizon,” European Journal of Operational Research, vol. 290, no. 2, pp. 405–421, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221720306895
[4] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, 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. 1144, pp. 1140–1144, 2018.
[5] F. Agostinelli, S. McAleer, A. Shmakov, and P. Baldi, “Solving the rubik’s cube with deep reinforcement learning and search,” Nature Machine Intelligence, vol. 1, no. 8, pp. 356–363, 2019. [Online]. Available: https://doi.org/10.1038/s42256-019-0070-z
[6] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez Liebana, S. Samothrakis, and S. Colton, “A survey of monte carlo tree search methods,” IEEE Transactions on Computational Intelligence and AI in Games, vol. 4, pp. 1–43, 03 2012.
[7] L. Kocsis and C. Szepesvári, “Bandit based monte-carlo planning,” in In: ECML-06. Number 4212 in LNCS. Springer, 2006, pp. 282–293.
[8] C.-S. Lee, M.-H. Wang, G. Chaslot, J.-B. Hoock, A. Rimmel, O. Teytaud, S.-R. Tsai, S.-C. Hsu, and T.-P. Hong, “The computational intelligence of mogo revealed in taiwan’s computer go tournaments,” Computational Intelligence and AI in Games, IEEE Transactions on, vol. 1, pp. 73 – 89, 04 2009.
[9] Y. Tanabe, K. Yoshizoe, and H. Imai, “A study on security evaluation methodology for image-based biometrics authentication systems,” in 2009 IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems, 2009, pp. 1–6.
[10] C. Mansley, A. Weinstein, and M. L. Littman, “Sample-based planning for continuous action markov decision processes,” in Proceedings of the Twenty-First International Conference on International Conference on Automated Planning and Scheduling, ser. ICAPS’11. AAAI Press, 2011, p. 335–338.
[11] G. Chaslot, S. Jong, J.-T. Saito, and J. Uiterwijk, “Monte-carlo tree search in production management problems,” Belgian/Netherlands Artificial Intelligence Conference, 01 2006.
[12] T. Mahlmann, I. Togelius, and G. N. Yannakakis, “Towards procedural strategy game generation: Evolving complementary unit types,” in Applications of Evolutionary Computation, C. Di Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Eknert, A. I. Esparcia-Alcázar, J. J. Merelo, F. Neri, M. Preuss, H. Richter, J. Togelius, and G. N. Yannakakis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 93–102.
[13] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver, “Mastering atari, go, chess and shogi by planning with a learned model,” Nature, vol. 588, pp. 604–609, 12 2020.
[14] J.-B. Grill, F. Altché, Y. Tang, T. Hubert, M. Valko, I. Antonoglou, and R. Munos, “Monte-Carlo tree search as regularized policy optimization,” in Proceedings of the 37th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 3769–3778. [Online]. Available: http://proceedings.mlr.press/v119/grill20a.html
[15] W. Ye, S. Liu, T. Kurutach, P. Abbeel, and Y. Gao, “Mastering atari games with limited data,” in NeurIPS, 2021.
[16] B. D. Ziebart, “Modeling purposeful adaptive behavior with the principle of maximum causal entropy,” Jul 2018. [Online]. Available: https://kilthub.cmu.edu/articles/thesis/Modeling_Purposeful_Adaptive_Behavior_with_the_Principle_of_Maximum_Causal_Entropy/6720692/1
[17] A. Hernandez, R. Hernando, A. Plastino, and A. R. Plastino, “The workings of the maximum entropy principle in collective human behaviour,” Journal of The Royal Society Interface, vol. 10, no. 78, p. 20120758, Jan 2013. [Online]. Available: http://dx.doi.org/10.1098/rsif.2012.0758
[18] B. Eysenbach and S. Levine, “If maxent r is the answer, what is the question?” 2019.
[19] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, “Reinforcement learning with deep energy-based policies,” in Proceedings of the 34th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. International Convention Centre, Sydney, Australia: PMLR, 06–11 Aug 2017, pp. 1352–1361. [Online]. Available: http://proceedings.mlr.press/v70/haarnoja17a.html
[20] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in Proceedings of the 35th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 1861–1870. [Online]. Available: http://proceedings.mlr.press/v80/haarnoja18b.html
[21] C. Xiao, R. Huang, J. Mei, D. Schuurmans, and M. Müller, “Maximum entropy monte-carlo planning,” in Advances in Neural Information
[22] T. Dam, C. D'Eramo, J. Peters, and J. Pajarinen, “Convex regularization in monte-carlo tree search,” 2020.

[23] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, “The arcade learning environment: An evaluation platform for general agents,” Journal of Artificial Intelligence Research, vol. 47, pp. 253–279, 2012, cite arxiv:1207.4708. [Online]. Available: http://arxiv.org/abs/1207.4708

[24] D. Jurafsky and J. H. Martin, https://web.stanford.edu/~jurafsky/slp3/index.html, 2020, accessed: 2021-09-09.

[25] K. Lee, S. Choi, and S. Oh, “Sparse markov decision processes with causal sparse tsallis entropy regularization for reinforcement learning,” IEEE Robotics Autom. Lett., vol. 3, no. 2, pp. 1466–1473, 2018. [Online]. Available: https://doi.org/10.1109/LRA.2018.2800085

[26] R. Brent, Algorithms for minimization without derivatives. Prentice-Hall, 1973.

[27] P. S. Castro, S. Moitra, C. Gelada, S. Kumar, and M. G. Bellemare, “Dopamine: A research framework for deep reinforcement learning,” 2018.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, Feb. 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[29] M. Hessel, J. Modayil, H. van Hasselt, P. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. G. Azar, and D. Silver, “Rainbow: Combining improvements in deep reinforcement learning.” in AAAI, S. A. McIlraith and K. Q. Weinberger, Eds. AAAI Press, 2018, pp. 3215–3222. [Online]. Available: http://dblp.uni-trier.de/db/conf/aaai/aaai2018.html#HesselMHSODPA18

[30] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, Z. D. Guo, and C. Blundell, “Agent57: Outperforming the Atari human benchmark,” in Proceedings of the 37th International Conference on Machine Learning, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 507–517. [Online]. Available: http://proceedings.mlr.press/v119/badia20a.html

[31] S. Levine, “Reinforcement learning and control as probabilistic inference: Tutorial and review,” 2018.

[32] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, “Bridging the gap between value and policy based reinforcement learning,” in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc. 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/97743b083008a894ee7ba16469-Paper.pdf

[33] Z. Ahmed, N. Le Roux, M. Norouzi, and D. Schuurmans, “Understanding the impact of entropy on policy optimization,” in Proceedings of the 36th International Conference on Machine Learning, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 151–160. [Online]. Available: http://proceedings.mlr.press/v97/ahmed19a.html