Generalized Mean Estimation in Monte-Carlo Tree Search

Tuan Dam[1] , **Pascal Klink**[1] , **Carlo D'Eramo**[1] , **Jan Peters**[1,2] and **Joni Pajarinen**[1,3]
[1]Department of Computer Science, Technische Universität Darmstadt, Germany
[2]Robot Learning Group, Max Planck Institute for Intelligent Systems,Tübingen, Germany
[3]Computing Sciences, Tampere University, Finland
{dam, klink, deramo, peters}@ias.tu-darmstadt.de, joni.pajarinen@tuni.fi

Abstract

We consider Monte-Carlo Tree Search (MCTS) applied to Markov Decision Processes (MDPs) and Partially Observable MDPs (POMDPs), and the well-known Upper Confidence bound for Trees (UCT) algorithm. In UCT, a tree with nodes (states) and edges (actions) is incrementally built by the expansion of nodes, and the values of nodes are updated through a backup strategy based on the average value of child nodes. However, it has been shown that with enough samples the maximum operator yields more accurate node value estimates than averaging. Instead of settling for one of these value estimates, we go a step further proposing a novel backup strategy which uses the power mean operator, which computes a value between the average and maximum value. We call our new approach Power-UCT, and argue how the use of the power mean operator helps to speed up the learning in MCTS. We theoretically analyze our method providing guarantees of convergence to the optimum. Finally, we empirically demonstrate the effectiveness of our method in well-known MDP and POMDP benchmarks, showing significant improvement in performance and convergence speed w.r.t. state of the art algorithms.

1 Introduction

Monte-Carlo Tree Search (MCTS) [Coulom, 2006] is an effective strategy for combining Monte-Carlo search with an incremental tree structure. MCTS is becoming increasingly popular in the community, especially after the outstanding results recently achieved in the game of Go [Silver *et al.*, 2016]. In the last years, the MCTS research has mainly focused on effective ways of expanding the tree, performing rollouts, and backing up the average reward computed from rollouts to the parent nodes. We consider the Upper Confidence bound applied to Trees (UCT) algorithm [Kocsis *et al.*, 2006], which combines tree search with the well-known UCB1 sampling policy [Auer *et al.*, 2002], as an effective way of dealing with the action selection to expand the tree. In UCT, the estimate of the value of each node is computed performing multiple rollouts starting from the node, and updating the node's value as the average of the collected rewards; then, the node's value is backed up to the parent nodes that are updated with the average of the value of the children nodes. Since the action selection policy tends to favor the best actions in the long run, UCT has theoretical convergence guarantees to the optimal value. However, it has been shown that using the average reward for backup leads to an underestimation of the optimal value, slowing down the learning; on the other hand, using the maximum reward leads to an overestimation causing the same learning problems, especially in stochastic settings [Coulom, 2006]. This problem is also evinced in the well-known Q-Learning algorithm [Watkins, 1989], where the maximum operator leads to overestimation of the optimal value [Smith and Winkler, 2006]. Some variants of Q-Learning based on (weighted) mean operators have been successfully proposed to address this issue [Hasselt, 2010; D'Eramo *et al.*, 2016].

In this paper, we introduce a novel backup operator based on a power mean [Bullen, 2013] that, through the tuning of a single coefficient, computes a value between the average reward and the maximum one. This allows to balance between the negatively biased estimate of the average reward, and the positively biased estimate of the maximum reward; in practice, this translates in balancing between a safe but slow update, and a greedy but misleading one. In the following, we propose a variant of UCT based on the power mean operator, which we call Power-UCT. We theoretically prove the convergence of Power-UCT, based on the consideration that the algorithm converges for all values between the range computed by the power mean. We empirically evaluate Power-UCT w.r.t. UCT and the recent MENTS algorithm [Xiao *et al.*, 2019] in classic MDP and POMDP benchmarks. Remarkably, we show how Power-UCT outperforms the baselines both in terms of quality and speed of learning. Thus, our *contribution* is twofold:

1. We propose a new backup operator for UCT based on a power mean, and prove the convergence to the optimal values;
2. We empirically evaluate the effectiveness of our approach comparing it with UCT in well-known MDPs and POMDPs, showing significantly better performance.

The rest of this paper is organized as follows. First we describe related work. Next, we discuss background knowledge

of MCTS, UCB and UCT. Then, we describe the power mean operator and introduce our Power-UCT algorithm. We derive theoretical results and prove convergence to the optimum for Power-UCT. Finally, we present empirical results in both MDP and POMDP problems, showing that Power-UCT outperforms baselines in MCTS.

2 Related Work

Several works focus on adapting how UCB1 [Auer *et al.*, 2002] is applied to MCTS. For this purpose UCB1-tuned [Auer *et al.*, 2002] modifies the upper confidence bound of UCB1 to account for variance in order to improve exploration. Tesauro *et al.* [2012] propose a Bayesian version of UCT, which obtains better estimates of node values and uncertainties given limited experience. However, the Bayesian version of UCT is more computation-intensive. While most work on bandits in MCTS focuses on discrete actions, work on continuous action MCTS also exists [Mansley *et al.*, 2011]. Since our MCTS algorithm is based on the UCT algorithm, which is an extension of UCB1, our method could be applied to all of these MCTS algorithms.

Many heuristic approaches based on specific domain knowledge have been proposed, such as adding a bonus term to value estimates based on domain knowledge [Gelly and Wang, 2006; Teytaud and Teytaud, 2010; Childs *et al.*, 2008; Kozelek, 2009; Chaslot *et al.*, 2008] or prior knowledge collected during policy search [Gelly and Silver, 2007; Helmbold and Parker-Wood, 2009; Lorentz, 2010; Tom, 2010; Hoock *et al.*, 2010]. We point out that we provide a novel node value backup approach that could be applied in combination with all of these methods.

To improve upon UCT algorithm in MCTS, Khandelwal *et al.* [2016] formalizes and analyzes different on-policy and off-policy complex backup approaches for MCTS planning based on techniques in the Reinforcement Learning literature. Khandelwal *et al.* [2016] propose four complex backup strategies: MCTS(λ), MaxMCTS(λ), MCTS$_\gamma$, MaxMCTS$_\gamma$. Khandelwal *et al.* [2016] report that MaxMCTS(λ) and MaxMCTS$_\gamma$ perform better than UCT for certain setup of parameter. Vodopivec *et al.* [2017] proposed an approach called SARSA-UCT, which performs the dynamic programming backups using SARSA [Rummery, 1995]. Both Khandelwal *et al.* [2016] and Vodopivec *et al.* [2017] directly borrow value backup ideas from Reinforcement Learning in order to estimate the value at each tree node. However, they do not provide any proof of convergence.

Instead, our method provides a completely novel way of backing up values in each MCTS node using a power mean operator, for which we prove the convergence to the optimal policy in the limit. The recently introduced MENTS algorithm [Xiao *et al.*, 2019], uses softmax backup operator at each node in combination with E2W policy, and shows better convergence rate w.r.t. UCT. Given its similarity to our approach, we empirically compare to it in the experimental section.

3 Background

In this section, we first discuss an overview of Monte Carlo Tree Search method. Next, we discuss UCB algorithm and subsequently an extension of UCB to UCT algorithm. Finally, we discuss the definition of Power Mean operator and its properties.

3.1 Monte-Carlo Tree Search

MCTS combines tree search with Monte-Carlo sampling in order to build a tree, where states and actions are respectively modeled as nodes and edges, to compute optimal decisions. MCTS requires a generative black box simulator for generating a new state based on the current state and chosen action. The MCTS algorithm consists of a loop of four steps:

- **Selection:** start from the root node, interleave action selection and sampling the next state (tree node) until a leaf node is reached
- **Expansion:** expand the tree by adding a new edge (action) to the leaf node and sample a next state (new leaf node)
- **Simulation:** rollout from the reached state to the end of the episode using random actions or a heuristic
- **Backup:** update the nodes backwards along the trajectory starting from the end of the episode until the root node according to the rewards collected

In the next subsection, we discuss UCB algorithm and its extension to UCT.

3.2 Upper Confidence Bound for Trees

In this work, we consider the MCTS algorithm UCT (Upper Confidence bounds for Trees) [Kocsis *et al.*, 2006], an extension of the well-known UCB1 [Auer *et al.*, 2002] multi-armed bandit algorithm. UCB1 chooses the arm (action a) using

$$a = \underset{i \in \{1 \dots K\}}{\arg\max} \overline{X}_{i,T_i(n-1)} + C\sqrt{\frac{\log n}{T_i(n-1)}}. \tag{1}$$

where $T_i(n) = \sum_{t=1}^{n} \mathbf{1}\{t = i\}$ is the number of times arm i is played up to time n. $\overline{X}_{i,T_i(n-1)}$ denotes the average reward of arm i up to time $n-1$ and $C = \sqrt{2}$ is an exploration constant. In UCT, each node is a separate bandit, where the arms correspond to the actions, and the payoff is the reward of the episodes starting from them. In the backup phase, value is backed up recursively from the leaf node to the root as

$$\overline{X}_n = \sum_{i=1}^{K} \Big(\frac{T_i(n)}{n}\Big) \overline{X}_{i,T_i(n)}. \tag{2}$$

Kocsis *et al.* [2006] proved that UCT converges in the limit to the optimal policy.

3.3 Power Mean

In this paper, we introduce a novel way of estimating the expected value of a bandit arm ($\overline{X}_{i,T_i(n-1)}$ in (1)) in MCTS. For this purpose, we will use the *power mean* [Mitrinovic and Vasic, 1970], an operator belonging to the family of functions

for aggregating sets of numbers, that includes as special cases the Pythagorean means (arithmetic, geometric, and harmonic means):

Definition 1. *For a sequence of positive numbers $X = (X_1, ..., X_n)$ and positive weights $w = (w_1, ..., w_n)$, the power mean of order p (p is an extended real number) is defined as*

$$M_n^{[p]}(X, w) = \left(\frac{\sum_{i=1}^{n} w_i X_i^p}{\sum_{i=1}^{n} w_i} \right)^{\frac{1}{p}}. \quad (3)$$

With $p = 1$ we obtain the weighted arithmetic mean. With $p \to 0$ we have the geometric mean, and with $p = -1$ we have the harmonic mean [Mitrinovic and Vasic, 1970] Furthermore, we get [Mitrinovic and Vasic, 1970]

$$\mathrm{M}_n^{[-\infty]}(X, w) = \lim_{p \to -\infty} \mathrm{M}_n^{[p]}(X, w) = \mathrm{Min}(X_1, ..., X_n), \quad (4)$$

$$\mathrm{M}_n^{[+\infty]}(X, w) = \lim_{p \to +\infty} \mathrm{M}_n^{[p]}(X, w) = \mathrm{Max}(X_1, ..., X_n), \quad (5)$$

The weighted arithmetic mean lies between $\mathrm{Min}(X_1, ..., X_n)$ and $\mathrm{Max}(X_1, ..., X_n)$. Moreover, the following lemma shows that $\mathrm{M}_n^{[p]}(X, w)$ is an increasing function.

Lemma 1. *$M_n^{[p]}(X, w)$ is an increasing function meaning that*

$$M_n^{[1]}(X, w) \leq M_n^{[q]}(X, w) \leq M_n^{[p]}(X, w), \forall p \geq q \geq 1 \quad (6)$$

Proof. For the proof, see [Mitrinovic and Vasic, 1970]. □

The following lemma shows that Power Mean can be upper bound by Average Mean plus with a constant.

Lemma 2. *Let $0 < l \leq X_i \leq U, C = \frac{U}{l}, \forall i \in (1, ..., n)$ and $p > q$. We define:*

$$Q(X, w, p, q) = \frac{M_n^{[p]}(X, w)}{M_n^{[q]}(X, w)} \quad (7)$$

$$D(X, w, p, q) = M_n^{[p]}(X, w) - M_n^{[q]}(X, w). \quad (8)$$

Then we have:

$$Q(X, w, p, q) \leq L_{p,q} D(X, w, p, q) \leq H_{p,q}$$

$$L_{p,q} = \left(\frac{q(C^p - C^q)}{(p - q)(C^q - 1)} \right)^{\frac{1}{p}} \left(\frac{p(C^q - C^p)}{(q - p)(C^p - 1)} \right)^{-\frac{1}{q}}$$

$$H_{p,q} = (\theta U^p + (1 - \theta) l^p)^{\frac{1}{p}} - (\theta U^q + (1 - \theta) l^q)^{1/q},$$

where θ is defined in the following way. Let

$$h(x) = x^{\frac{1}{p}} - (ax + b)^{1/q}$$

where:

$$a = \frac{U^q - l^q}{U^p - l^p}; b = \frac{U^p l^q - U^q l^p}{U^p - l^p} \quad (9)$$

$$x^{'} = \arg\max\{h(x), x \in (l^p, U^p)\} \quad (10)$$

then:

$$\theta = \frac{x' - l^p}{U^p - l^p}.$$

Proof. Refer to Mitrinovic and Vasic [1970]. □

4 Power Mean Backup

As previously described, it is well known that performing backups using the average of the rewards results in an underestimate of the true value of the node, while using the maximum results in an overestimate of it [Coulom, 2006]. Usually, the average backup is used when the number of simulations is low, for a conservative update of the nodes due to the lack of samples; on the other hand, the maximum operator is favoured when the number of simulations is high. We address this problem proposing a novel backup operator for UCT based on the power mean (Equation 3):

$$\overline{X}_n(p) = \left(\sum_{i=1}^{K} \left(\frac{T_i(n)}{n} \right) \overline{X}_{i,T_i(n)}^p \right)^{\frac{1}{p}}. \quad (11)$$

This way, we bridge the gap between the average and maximum estimators with the purpose of getting the advantages of both. We call our approach Power-UCT and describe it in more detail in the following.

4.1 Power-UCT

The introduction of our novel backup operator in UCT does not require major changes to the algorithm. Indeed, the Power-UCT pseudocode shown in Algorithm 1 is almost identical to the UCT one, with the only differences in lines 36 and 49. MCTS has two type of nodes: V_Nodes corresponding to the state-value, and Q_Nodes corresponding to state-action values. An action is taken from the V_Node of the current state leading to the respective Q_Node, then it leads to the V_Node of the reached state. We skip the description of all the procedures since they are well-known components of MCTS, and we focus only on the ones involved in the use of the power mean backup operator. In SIMULATE_V, Power-UCT updates the value of each V_Node using the power mean of its children Q_Nodes, that are computed in SIMULATE_Q. Note that our algorithm could be applied to several bandit based enhancements of UCT, but for simplicity we only focus on UCT.

5 Theoretical Analysis

In this section, we show that Power-UCT can smoothly adapt to all theorems of UCT [Kocsis *et al.*, 2006]. The following results can be seen as a generalization of the results for UCT, as we consider a generalized mean instead of a standard mean as the backup operator. Our main results are Theorem 6 and Theorem 7, which respectively prove the convergence of failure probability at the root node, and derive the bias of power mean estimated payoff. In order to prove them, we start with Theorem 1 to show the concentration of power mean with respect to i.i.d random variables X. Subsequently, Theorem 2 shows the upper bound of the expected number of times when a suboptimal arm is played. Theorem 3 bounds the expected error of the power mean estimation. Theorem 4 shows the lower bound of the number of times any arm is played. Theorem 5 shows the concentration of power mean backup around its mean value at each node in the tree[1].

[1]Refer to [Dam *et al.*, 2019] for proofs of lemmas and theorems.

Algorithm 1 Power-UCT

Input
s: state
a: action
p: Power Mean constant
C: Exploration constant
$N(s)$: number of simulations of V_Node of state s
$n(s,a)$: number of simulations of Q_Node of state s and action a
$V(s)$: Value of V_Node at state s. Default is 0
$Q(s,a)$: Value of Q_Node at state s, action a. Default is 0
$\tau(s,a)$: transition function
γ: discount factor

procedure SELECT_ACTION(s)
    return $\arg\max_a Q(s,a) + C\sqrt{\frac{\log N(s)}{n(s,a)}}$
end procedure

procedure SEARCH(s)
    repeat
        SIMULATE_V(s, 0)
    until TIMEOUT()
    return $\arg\max_a Q(s,a)$
end procedure

procedure ROLLOUT(s, depth)
    if $\gamma^{\text{depth}} < \epsilon$ **then**
        return 0
    end if
    $a \sim \pi_{\text{rollout}}(.)$
    $(s', r) \sim \tau(s,a)$
    return $r + \gamma$ROLLOUT $(s', depth+1)$
end procedure

procedure SIMULATE_V(s, depth)
    $a \leftarrow$SELECT_ACTION(s)
    SIMULATE_Q $(s, a,$ depth$)$
    $N(s) \leftarrow N(s) + 1$
    $V(s) \leftarrow \left(\sum_a \frac{n(s,a)}{N(s)} Q(s,a)^p\right)^{\frac{1}{p}}$
end procedure

procedure SIMULATE_Q(s, a, depth)
    $(s', r) \sim \tau(s,a)$
    if $s' \notin Terminal$ **then**
        if $V(s')$ not expanded **then**
            ROLLOUT$(s',$ depth$)$
        else
            SIMULATE_V$(s',$ depth $+ 1)$
        end if
    end if
    $n(s,a) \leftarrow n(s,a) + 1$
    $Q(s,a) \leftarrow \frac{(\sum r_{s,a}) + \gamma . \sum_{s'} N(s').V(s')}{n(s,a)}$
end procedure

Theorem 1. *If $X_1, X_2, ..., X_n$ are independent with $\mathbf{Pr}(a \leq X_i \leq b) = 1$ and common mean μ, $w_1, w_2, ..., w_n$ are positive and $W = \sum_{i=1}^n w_i$ then for any $\epsilon > 0$, $p \geq 1$*

$$\mathbf{Pr}\left(\left|\left(\frac{\sum_{i=1}^n w_i X_i^p}{\sum_{i=1}^n w_i}\right)^{\frac{1}{p}} - \mu\right| > \epsilon\right)$$

$$\leq 2\exp(H_{p,1})\exp(-2\epsilon^2 W^2 / \sum_{i=1}^n w_i^2 (b-a)^2))$$

Theorem 1 is derived using the upper bound of the power mean operator, which corresponds to the average mean incremented by a constant [Mitrinovic and Vasic, 1970] and Chernoff's inequality. Note that this result can be considered a generalization of the well-known Hoeffding inequality to power mean. Next, given i.i.d. random variables X_{it} (t=1,2,...) as the payoff sequence at any internal leaf node of the tree, we assume the expectation of the payoff exists and let $\mu_{in} = \mathbb{E}[\overline{X_{in}}]$. We assume the power mean reward drifts as a function of time and converges only in the limit, which means that

$$\mu_i = \lim_{n\to\infty} \mu_{in}.$$

Let $\delta_{in} = \mu_i - \mu_{in}$ which also means that

$$\lim_{n\to\infty} \delta_{in} = 0.$$

From now on, let $*$ be the upper index for all quantities related to the optimal arm. By assumption, the rewards lie between 0 and 1. Let's start with an assumption:

Assumption 1. *Fix $1 \leq i \leq K$. Let $\{F_{it}\}_t$ be a filtration such that $\{X_{it}\}_t$ is $\{F_{it}\}$-adapted and $X_{i,t}$ is conditionally independent of $F_{i,t+1}, F_{i,t+2}, ...$ given $F_{i,t-1}$. Then $0 \leq X_{it} \leq 1$ and the limit of $\mu_{in} = \mathbb{E}[\overline{X_{in}}(p)]$ exists. Further, we assume that there exists a constant $C > 0$ and an integer N_c such that for $n > N_c$, for any $\delta > 0$, $\triangle_n(\delta) = C\sqrt{n\log(1/\delta)}$, the following bounds hold:*

$$\mathbf{Pr}(\overline{X}_{in}(p) \geq \mathbb{E}[\overline{X}_{in}(p)] + \triangle_n(\delta)/n) \leq \delta, \quad (12)$$

$$\mathbf{Pr}(\overline{X}_{in}(p) \leq \mathbb{E}[\overline{X}_{in}(p)] - \triangle_n(\delta)/n) \leq \delta. \quad (13)$$

Under Assumption 1, a suitable choice for the bias sequence $c_{t,s}$ is given by

$$c_{t,s} = 2C\sqrt{\frac{\log t}{s}}. \quad (14)$$

where C is an exploration constant.
Next, we derive Theorems 2, 3, and 4 following the derivations in [Kocsis *et al.*, 2006]. First, from Assumption 1, we derive an upper bound on the error for the expected number of times suboptimal arms are played.

Theorem 2. *Consider UCB1 (using power mean estimator) applied to a non-stationary problem where the pay-off sequence satisfies Assumption 1 and where the bias sequence, $c_{t,s}$ defined in (14). Fix $\epsilon \geq 0$. Let $T_k(n)$ denote the number of plays of arm k. Then if k is the index of a suboptimal arm then each sub-optimal arm k is played in expectation at most*

$$\mathbb{E}[T_k(n)] \leq \frac{16C^2 \ln n}{(1-\epsilon)^2 \triangle_k^2} + A(\epsilon) + N_c + \frac{\pi^2}{3} + 1. \quad (15)$$

Next, we derive our version of Theorem 3 in [Kocsis *et al.*, 2006], which computes the upper bound of the difference between the value backup of an arm with μ^* up to time n.

Theorem 3. *Under the assumptions of Theorem 2,*

$$\left| \mathbb{E}\left[\overline{X}_n(p)\right] - \mu^* \right| \leq |\delta_n^*| + \mathcal{O}\left(\frac{K(C^2 \log n + N_0)}{n}\right)^{\frac{1}{p}}.$$

A lower bound for the times choosing any arm follows:

Theorem 4. ***(Lower Bound)*** *Under the assumptions of Theorem 2, there exists some positive constant ρ such that for all arms k and n, $T_k(n) \geq \lceil \rho \log(n) \rceil$.*

For deriving the concentration of estimated payoff around its mean, we modify Lemma 14 in [Kocsis *et al.*, 2006] for power mean: in the proof, we first replace the partial sums term with a partial mean term and modify the following equations accordingly. The partial mean term can then be easily replaced by a partial power mean term and we get

Lemma 7. *Let Z_i, a_i be as in Lemma 13 in [Kocsis* et al.*, 2006]. Let F_i denotes a filtration over some probability space. Y_i be an F_i-adapted real valued martingale-difference sequence. Let X_i be an i.i.d. sequence with mean μ. We assume that both X_i and Y_i lie in the [0,1] interval. Consider the partial sums*

$$S_n = \left(\frac{\sum_{i=1}^{n}(1 - Z_i)X_i^p + Z_i Y_i^p}{n}\right)^{\frac{1}{p}}. \tag{16}$$

Fix an arbitrary $\delta > 0$, and fix $p \geq 1$, and $M = \exp(H_{p,1})$ where $H_{p,1}$ is defined as in Lemma 2. Let $\triangle_n = 9\sqrt{2n \log(2M/\delta)}$, and $\triangle = (9/4)^{p-1}\triangle_n$ let

$$R_n = \mathbb{E}\left[\left(\frac{\sum_{i=1}^{n} X_i^p}{n}\right)^{\frac{1}{p}}\right] - \mathbb{E}[S_n]. \tag{17}$$

Then for n such that $a_n \leq (1/9)\triangle_n$ and $|R_n| \leq (4/9)(\triangle/n)^{\frac{1}{p}}$

$$\mathbf{Pr}(S_n \geq \mathbb{E}[S_n] + (\triangle/n)^{\frac{1}{p}}) \leq \delta \tag{18}$$

$$\mathbf{Pr}(S_n \leq \mathbb{E}[S_n] - (\triangle/n)^{\frac{1}{p}}) \leq \delta \tag{19}$$

Based on the results from Lemma 7, we derive the concentration of estimated payoff around its mean.

Theorem 5. *Fix an arbitrary $\delta > 0$ and fix $p \geq 1$, $M = \exp(H_{p,1})$ where $H_{p,1}$ is defined as in Lemma 2 and let $\triangle_n = (\frac{9}{4})^{p-1}(9\sqrt{2n \log(2M/\delta)})$. Let n_0 be such that*

$$\sqrt{n_0} \leq \mathcal{O}(K(C^2 \log n_0 + N_0(1/2))). \tag{20}$$

Then for any $n \geq n_0$, under the assumptions of Theorem 2, the following bounds hold true:

$$\mathbf{Pr}(\overline{X}_n(p) \geq \mathbb{E}[\overline{X}_n(p)] + (\triangle_n/n)^{\frac{1}{p}}) \leq \delta \tag{21}$$

$$\mathbf{Pr}(\overline{X}_n(p) \leq \mathbb{E}[\overline{X}_n(p)] - (\triangle_n/n)^{\frac{1}{p}}) \leq \delta \tag{22}$$

Using The Hoeffding-Azuma inequality for Stopped Martingales Inequality (Lemma 10 in Kocsis *et al.* [2006]), under Assumption 1 and the result from Theorem 4 we get

Theorem 6. ***(Convergence of Failure Probability)*** *Under the assumptions of Theorem 2, it holds that*

$$\lim_{t \to \infty} \mathbf{Pr}(I_t \neq i^*) = 0. \tag{23}$$

And finally, the following is our main result showing the expected payoff of our Power-UCT.

Theorem 7. *Consider algorithm Power-UCT running on a game tree of depth D, branching factor K with stochastic payoff at the leaves. Assume that the payoffs lie in the interval [0,1]. Then the bias of the estimated expected payoff, $\overline{X_n}$, is $\mathcal{O}(KD(\log(n)/n)^{\frac{1}{p}} + K^D(1/n)^{\frac{1}{p}})$. Further, the failure probability at the root convergences to zero as the number of samples grows to infinity.*

Proof. (Sketch) As for UCT [Kocsis *et al.*, 2006], the proof is done by induction on D. When $D = 1$, Power-UCT corresponds to UCB1 with power mean backup, and our assumptions on the payoffs hold, thanks to Theorem 1, the proof of convergence follows the results as Theorem 3 and Theorem 6.

Now we assume that the result holds up to depth $D - 1$ and consider the tree of depth D. Running Power-UCT on root node is equivalent to UCB1 on non-stationary bandit settings, but with power mean backup. Theorem 3 shows that the expected average payoff converges. The conditions on the exponential concentration of the payoffs (Assumption 1) are satisfied follows from Theorem 5. The error bound of running Power-UCT for the whole tree is the sum of payoff at root node with payoff starting from any node i after the first action chosen from root node until the end. This payoff by induction at depth $D - 1$ in addition to the bound from Theorem 3 when the the drift-conditions are satisfied, and with straightforward algebra, we can compute the payoff at the depth D, in combination with Theorem 6. Since by our induction hypothesis this holds for all nodes at a distance of one node from the root, the proof is finished by observing that Theorem 3 and Theorem 5 do indeed ensure that the drift conditions are satisfied. This completes our proof of the convergence of Power-UCT. Interestingly, the proof guarantees the convergence for any finite value of p. □

6 Experiments

In this section, we aim to answer the following questions empirically: Does the Power Mean offer higher performance in MDP and POMDP MCTS tasks than the regular Mean? How does the value of p influence the overall performance? How does Power-UCT, our MCTS algorithm based on the Power Mean, compare to state-of-the-art methods in tree-search? We choose the recent MENTS algorithm [Xiao *et al.*, 2019], that introduces a maximum entropy policy optimization framework for MCTS, and shows better convergence rate w.r.t. UCT.

For MENTS we find the best combination of the two hyperparameters by grid search. In MDP tasks, we find the UCT

Algorithm	4096	16384	65536	262144
UCT	0.08 ± 0.02	0.23 ± 0.04	0.54 ± 0.05	0.69 ± 0.04
p=2.2	0.12 ± 0.03	0.32 ± 0.04	$\mathbf{0.62 \pm 0.04}$	$\mathbf{0.81 \pm 0.03}$
p=max	0.10 ± 0.03	0.36 ± 0.04	0.55 ± 0.04	0.69 ± 0.04
MENTS	$\mathbf{0.28 \pm 0.04}$	$\mathbf{0.46 \pm 0.04}$	$\mathbf{0.62 \pm 0.04}$	0.74 ± 0.04

Table 1: Mean and two times standard deviation of the success rate, over 500 evaluation runs, of UCT, Power-UCT and MENTS in *FrozenLake* from OpenAI Gym. The top row of each table shows the number of simulations used for tree-search at each time step.

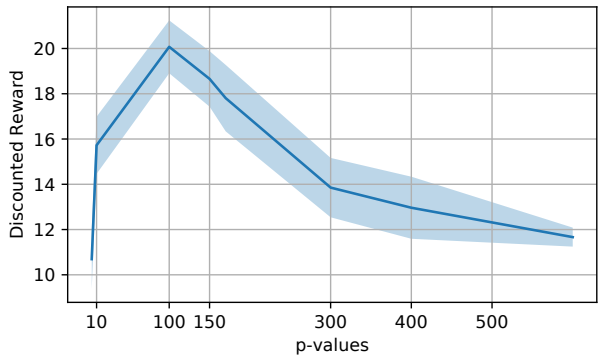

Figure 1: Evaluating Power-UCT w.r.t. different p-values: The mean discounted total reward at 65536 simulations (shaded area denotes standard error) over 100 evaluation runs.

exploration constant using grid search. For Power-UCT, we find the p-value by increasing it until performance starts to decrease.

6.1 *FrozenLake*

For MDPs, we consider the well-known *FrozenLake* problem as implemented in OpenAI Gym [Brockman *et al.*, 2016]. In this problem, the agent needs to reach a goal position in an 8x8 ice grid-world while avoiding falling into the water by stepping onto unstable spots. The challenge of this task arises from the high-level of stochasticity, which makes the agent only move towards the intended direction one-third of the time, and into one of the two tangential directions the rest of it. Reaching the goal position yields a reward of 1, while all other outcomes (reaching the time limit or falling into the water) yield a reward of zero. As can be seen in Table 1, Power-UCT improves the performance compared to UCT. Power-UCT outperforms MENTS when the number of simulations increases.

6.2 Copy Environment

Now, we aim to answer the question of how Power-UCT scales to domains with a large number of actions (high branching factor). We use the OpenAI gym Copy environment where the agent needs to copy the characters on an input band to an output band. The agent can move and read the input band at every time-step and decide to write a character from an alphabet to the output band. Hence, the number of actions scales with the size of the alphabet.

Contrary to the previous experiments, there is only one initial run of tree-search and afterwards, no re-planning between two actions occurs. Hence, all actions are selected according

Algorithm	512	2048	8192	32768
UCT	2.6 ± 0.98	$9. \pm 1.17$	34.66 ± 1.68	$\mathbf{40. \pm 0.}$
$p = 3$	3.24 ± 1.17	$\mathbf{12.35 \pm 1.14}$	$\mathbf{40. \pm 0.}$	$\mathbf{40. \pm 0.}$
$p = \max$	2.56 ± 1.48	9.55 ± 3.06	37.52 ± 5.11	39.77 ± 0.84
MENTS	$\mathbf{3.26 \pm 1.32}$	11.96 ± 2.94	39.37 ± 1.15	39.35 ± 0.95

(a) 144 Actions

Algorithm	512	2048	8192	32768
UCT	1.98 ± 0.63	6.43 ± 1.36	24.5 ± 1.56	$\mathbf{40. \pm 0.}$
$p = 3$	$\mathbf{2.55 \pm 0.99}$	$\mathbf{9.11 \pm 1.41}$	$\mathbf{36.02 \pm 1.72}$	$\mathbf{40. \pm 0.}$
$p = \max$	2.03 ± 1.37	6.99 ± 2.51	27.89 ± 4.12	39.93 ± 0.51
MENTS	2.44 ± 1.34	8.86 ± 2.65	34.63 ± 5.6	39.42 ± 0.99

(b) 200 Actions

Table 2: Mean and two times standard deviation of discounted total reward, over 100 evaluation runs, of UCT, Power-UCT and MENTS in the copy environment with 144 actions (top) and 200 actions (bottom). Top row: number of simulations at each time step.

to the value estimates from the initial search. The results in Tables 2a and 2b show that Power-UCT allows to solve the task much quicker than regular UCT. Furthermore, we observe that MENTS and Power-UCT for $p = \infty$ exhibit larger variance compared to Power-UCT with a finite value of p and are not able to reliably solve the task, as they do not reach the maximum reward of 40 with 0 standard deviation.

6.3 Rocksample and PocMan

In POMDP problems, we compare Power-UCT against the POMCP algorithm [Silver and Veness, 2010] which is a standard UCT algorithm for POMDPs. Since the state is not fully observable in POMDPs, POMCP assigns a unique action-observation history, which is a sufficient statistic for optimal decision making in POMDPs, instead of the state, to each tree node. Similarly to fully observable UCT, POMCP chooses actions using the UCB1 bandit. Therefore, we modify POMCP to use the power mean identically to how we modified fully observable UCT and get a POMDP version of Power-UCT. We also modify POMCP similarly for the MENTS approach. Next, we discuss the evaluation of the POMDP based Power-UCT, MENTS, and POMCP, in the *rocksample* and *pocman* environments [Silver and Veness, 2010]. In both problems, we scale the rewards into [0,1] for MCTS planning and show the plot with real reward scale.

Rocksample. The *rocksample (n,k)* (Smith and Simmons [2004]) simulates a Mars explorer robot in an n x n grid containing k rocks. The task is to determine which rocks are valuable using a long range sensor, take samples of valuable rocks and to finally leave the map to the east. There are $k+5$ actions where the agent can move in four directions (North, South, East, West), sample a rock, or sense one of the k rocks. Rocksample requires strong exploration to find informative actions which do not yield immediate reward but may yield high long term reward. We use three variants with a different number of actions: *rocksample* (11,11), *rocksample* (15,15), *rocksample* (15,35) and set the exploration constant as in [Silver and Veness, 2010] to the difference of the maximum and minimum immediate reward. In Fig. 2, Power-UCT outperforms POMCP for almost all values of p. For sensitivity analysis, Fig. 1 shows the performance of Power-UCT in *rocksample* (11x11) for different p-values at 65536 simulations. Fig. 1

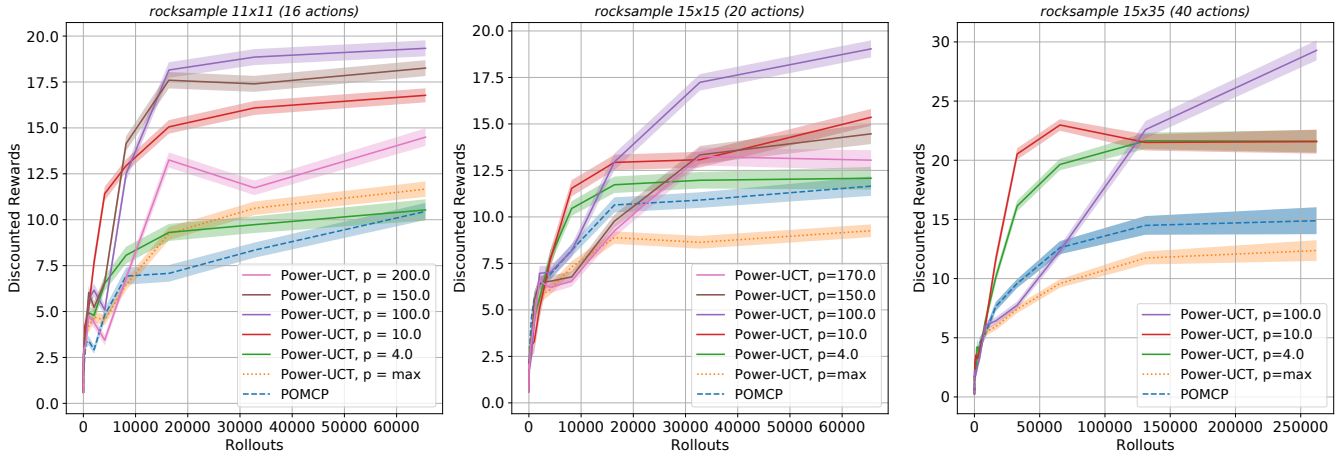

Figure 2: Performance of Power-UCT compared to UCT in *rocksample*. The mean of total discounted reward over 1000 evaluation runs is shown by thick lines while the shaded area shows standard error.

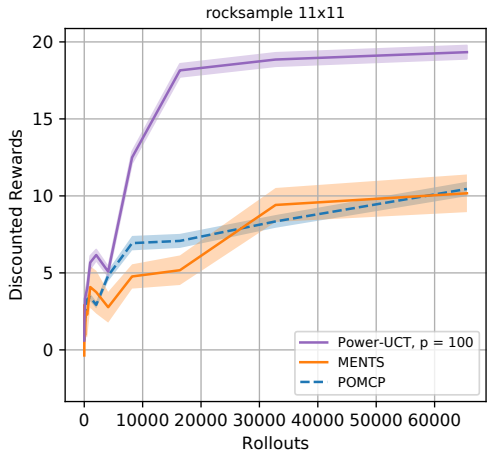

Figure 3: Performance of Power-UCT compared to UCT and MENTS in *rocksample* 11x11. The mean of discounted total reward over 1000 evaluation runs is shown by thick lines while the shaded area shows standard error.

suggests that at least in *rocksample* finding a good p-value is straightforward. Fig. 3 shows that Power-UCT significantly outperforms MENTS in *rocksample* (11,11). A possible explanation for the strong difference in performance between MENTS and Power-UCT is that MENTS may not explore sufficiently in this task. However, this would require more in depth analysis of MENTS.

Pocman. We further evaluate our algorithm in the *pocman* problem [Silver and Veness, 2010]. In *pocman*, an agent called PocMan must travel in a maze of size (17x19) by only observing the local neighborhood in the maze. PocMan tries to eat as many food pellets as possible. Four ghosts try to kill PocMan. After moving initially randomly the ghosts start to follow directions with a high number of food pellets more likely. If PocMan eats a power pill, he is able to eat ghosts for 15 time steps. PocMan receives a reward of -1 at each step he travels, $+10$ for eating each food pellet, $+25$ for eating a ghost and -100 for dying. The *pocman* problem has 4 actions, 1024 observations, and approximately 10^{56} states. Table 3 shows that Power-UCT and MENTS outperform POMCP.

	1024	4096	16384	65536
POMCP	30.89 ± 1.4	33.47 ± 1.4	33.44 ± 1.39	32.36 ± 1.6
$p = max$	14.82 ± 1.52	14.91 ± 1.52	14.34 ± 1.52	14.98 ± 1.76
$p = 10$	29.14 ± 1.61	35.26 ± 1.56	44.14 ± 1.60	$\mathbf{53.30 \pm 1.46}$
$p = 30$	28.78 ± 1.44	33.92 ± 1.56	42.45 ± 1.54	49.66 ± 1.70
MENTS	$\mathbf{54.08 \pm 3.20}$	$\mathbf{55.37 \pm 3.0}$	$\mathbf{53.90 \pm 2.86}$	51.03 ± 3.36

Table 3: Discounted total reward in *pocman* for the comparison methods. Mean $\pm$ standard error are computed from 1000 simulations except in MENTS where we ran 100 simulations.

7 Conclusion

We proposed to use power mean as a novel backup operator in MCTS, and derived a variant of UCT based on this operator, which we call Power-UCT. We theoretically prove the convergence of Power-UCT to the optimal value, given that the value computed by the power mean lies between the average and the maximum. The empirical evaluation on stochastic MDPs and POMDPs, shows the advantages of Power-UCT w.r.t. other baselines.

Possible future work includes the proposal of a theoretically justified or heuristic approach to adapt the greediness of power mean. Moreover, we are interested in analysing the bias and variance of the power mean estimator, or analyse the regret bound of Power-UCT in MCTS. Furthermore, we plan to test our methodology in more challenging Reinforcement Learning problems through the use of parametric function approximators, e.g. neural networks.

Acknowledgements

This project has received funding from SKILLS4ROBOTS, project reference #640554, and, by the German Research Foundation project PA 3179/1-1 (ROBOLEAP), and from the European Union's Horizon 2020 research and innovation programme under grant agreement No. #713010 (GOAL-Robots).

References

[Auer *et al.*, 2002] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[Bullen, 2013] Peter S Bullen. *Handbook of means and their inequalities*. Springer Science & Business Media, 2013.

[Chaslot *et al.*, 2008] Guillaume Chaslot, Mark Winands, Jaap Van Den Herik, Jos Uiterwijk, and Bruno Bouzy. Progressive strategies for monte-carlo tree search. *New Mathematics and Natural Computation*, 4(03):343–357, 2008.

[Childs *et al.*, 2008] Benjamin E Childs, James H Brodeur, and Levente Kocsis. Transpositions and move groups in monte carlo tree search. In *2008 IEEE Symposium On Computational Intelligence and Games*. IEEE, 2008.

[Coulom, 2006] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

[Dam *et al.*, 2019] Tuan Dam, Pascal Klink, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. Generalized mean estimation in monte-carlo tree search, 2019.

[D'Eramo *et al.*, 2016] Carlo D'Eramo, Marcello Restelli, and Alessandro Nuara. Estimating maximum expected value through gaussian approximation. In *International Conference on Machine Learning*, 2016.

[Gelly and Silver, 2007] Sylvain Gelly and David Silver. Combining online and offline knowledge in uct. In *Proceedings of the 24th international conference on Machine learning*, pages 273–280. ACM, 2007.

[Gelly and Wang, 2006] Sylvain Gelly and Yizao Wang. Exploration exploitation in go: Uct for monte-carlo go. In *NIPS: Neural Information Processing Systems Conference On-line trading of Exploration and Exploitation Workshop*, 2006.

[Hasselt, 2010] Hado V Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010.

[Helmbold and Parker-Wood, 2009] David P Helmbold and Aleatha Parker-Wood. All-moves-as-first heuristics in monte-carlo go. In *IC-AI*, pages 605–610, 2009.

[Hoock *et al.*, 2010] Jean-Baptiste Hoock, Chang-Shing Lee, Arpad Rimmel, Fabien Teytaud, Mei-Hui Wang, and Oliver Teytaud. Intelligent agents for the game of go. *IEEE Computational Intelligence Magazine*, 2010.

[Khandelwal *et al.*, 2016] Piyush Khandelwal, Elad Liebman, Scott Niekum, and Peter Stone. On the analysis of complex backup strategies in monte carlo tree search. In *International Conference on Machine Learning*, 2016.

[Kocsis *et al.*, 2006] Levente Kocsis, Csaba Szepesvári, and Jan Willemson. Improved monte-carlo search. 2006.

[Kozelek, 2009] Tomáš Kozelek. Methods of mcts and the game arimaa. 2009.

[Lorentz, 2010] Richard J Lorentz. Improving monte–carlo tree search in havannah. In *International Conference on Computers and Games*, pages 105–115. Springer, 2010.

[Mansley *et al.*, 2011] Chris Mansley, Ari Weinstein, and Michael Littman. Sample-based planning for continuous action markov decision processesgeneralizedfmean. In *Twenty-First International Conference on Automated Planning and Scheduling*, 2011.

[Mitrinovic and Vasic, 1970] Dragoslav S Mitrinovic and Petar M Vasic. *Analytic inequalities*. Springer, 1970.

[Rummery, 1995] Gavin Adrian Rummery. *Problem solving with reinforcement learning*. PhD thesis, University of Cambridge Ph. D. dissertation, 1995.

[Silver and Veness, 2010] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, pages 2164–2172, 2010.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[Smith and Simmons, 2004] Trey Smith and Reid Simmons. Heuristic search value iteration for pomdps. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 520–527. AUAI Press, 2004.

[Smith and Winkler, 2006] James E Smith and Robert L Winkler. The optimizer's curse: Skepticism and postdecision surprise in decision analysis. *Management Science*, 52(3):311–322, 2006.

[Tesauro *et al.*, 2012] Gerald Tesauro, VT Rajan, and Richard Segal. Bayesian inference in monte-carlo tree search. *arXiv preprint arXiv:1203.3519*, 2012.

[Teytaud and Teytaud, 2010] Fabien Teytaud and Olivier Teytaud. On the huge benefit of decisive moves in monte-carlo tree search algorithms. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, pages 359–364. IEEE, 2010.

[Tom, 2010] David Tom. Investigating uct and rave: Steps towards a more robust method. 2010.

[Vodopivec *et al.*, 2017] Tom Vodopivec, Spyridon Samothrakis, and Branko Ster. On monte carlo tree search and reinforcement learning. *Journal of Artificial Intelligence Research*, 60:881–936, 2017.

[Watkins, 1989] C. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, England, 1989.

[Xiao *et al.*, 2019] Chenjun Xiao, Ruitong Huang, Jincheng Mei, Dale Schuurmans, and Martin Müller. Maximum entropy monte-carlo planning. In *Advances in Neural Information Processing Systems*, pages 9516–9524, 2019.