JMLR: Workshop and Conference Proceedings 1–12, 2019

Backpropagation Phase in Upper Confidence Bound for Single Rooted Directed Analytic Graph

Aurélien Pélissier AURELIEN.PELISSI@ENS-PARIS-SACLAY.FR
École Normale Supérieure Paris-Saclay, France
Atsuyoshi Nakamura ATSU@IST.HOKUDAI.AC.JP
Graduate school of Information Science and Technology, Hokkaido University, Japan

Abstract

Monte Carlo tree search (MCTS), and specifically the popular UCT (Upper Confidence Bounds applied to Trees) algorithm, has received considerable interest due to its spectacular success in the difficult problem of computer Go and also proved beneficial in a range of other domains. A major issue that has received little attention in the MCTS literature is the fact that, in most games, different actions can lead to the same state, that may involve a high degree of redundancy in tree representation and unnecessary additional computational cost. We extend UCT to single rooted directed acyclic graph (DAG), and introduce the algorithm α-UCD, (Upper Confidence bound for single rooted Direct acyclic graph with transposition parameter α), that is a parameterized framework to deal with transpositions during the backpropagation phase after each playout. We attempt to show that our algorithm converge to the optimal arm at the root in polynomial time, and the benefit of considering transposition in DAGs is studied on a benchmark Feature Selection MCTS problems.

Keywords: DAG, MCTS, UCT, transpositions

1. Introduction

The combination of Monte Carlo tree search (MCTS) with bandit strategies has proven remarkably efficient and has received considerable interest due to its spectacular success, mainly in the difficult problem of computer Go [Silver et al. (2017)], but also in a wide range of other domains (eg. Optimization, Scheduling, ..) [Browne et al. (2012)]. An important issue that has not seen much attention in the MCTS literature is the fact that, in most games, different actions may lead to the same state (usually referred as transpositions), or states can be revisited [Gusmao and Raiko (2012)]. Thus, the game's state space should be represented as a connected graph, not as a tree.

While the popular UCT algorithm [Kocsis and Szepesvári (2006)] has successfully adapted bandit strategies to growing tree search by finding the optimal arm in polynomial time, it is ignoring the transpositions in connected graph architectures, thus missing the opportunity to share the knowledge between different paths and gain efficiency in the Monte-Carlo search. In this context, there have been attempts to generalize UCT to directed acyclic graph (DAG), for which the consideration for transpositions has already proven to significantly improve the performances of game agents [Saffidine et al. (2012); Childs et al. (2008)].

One challenge that arises from considering DAG architectures over tree architectures is about the backpropagation of the information after each playout. In the case of trees, the traversed nodes are exactly the ancestors of the leaf node from which the evaluation was performed and the backpropagation is straightforward. However, in the case of DAGs, the ancestor nodes are a superset of the traversed nodes and it is not clear how and which ancestor nodes should to be updated. While Monte-Carlo programs usually deal with transpositions in DAGs by updating only the traversed edges [Gaudel and Sebag (2010)], a simple idea would be, as performed by Pélissier et al. (2019), to update every ancestor nodes. However, Saffidine et al. (2012) shows that this backpropagation method may no longer guaranty the optimal convergence of the UCT algorithm.

In this work we extend the UCT algorithm to directed acyclic graphs and propose a parameterized framework that deal with transpositions during the backpropagation phase by updating the ancestor nodes differently depending if they were traversed during the selection phase or not. We introduce the algorithm α-UCD, (Upper Confidence bound for single rooted Direct acyclic graph with transposition parameter α), and we attempt to show that the failure probability at the root converges to zero in polynomial time.

2. Motivation

We consider a fixed connected *DAG* (Directed Acyclic Graph) $\mathcal{D} = (\mathcal{V}, \mathcal{E})$ composed of *nodes* $s \in \mathcal{V}$ related to each other by *directed edges* $(s_1, s_2) \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. For an edge $(s_1, s_2) \in \mathcal{E}$, s_1 is said to be a *parent* of s_2 and s_2 is said to be a *child* of s_1. Node s_1 is said to be an *ancestor* of node s_2 if there is a directed path from s_1 to s_2. For each node $s \in \mathcal{V}$, we denote by $\mathcal{C}(s)$ the set of its children and by $\mathcal{P}(s)$ the set of its parents. The *root* $s_0 \in \{s \in \mathcal{V} \mid \mathcal{P}(s) = \varnothing\}$ is assumed unique, and we finally introduce the leaf node set $\mathcal{L} = \{s \in \mathcal{V} \mid \mathcal{C}(s) = \varnothing\}$. Note that the major difference of a DAG compared to a tree is that the parent set $\mathcal{P}(s)$ may contains more than one node. For each leaf node ℓ, we assume a stochastic oracle $\mathcal{O}_\ell$ that returns a value $X \in [0, 1]$ generated according to an unknown distribution over $[0, 1]$ with mean μ_ℓ for each call.

The value $V(s)$ for any node $s \in \mathcal{V}$ is recursively defined with

$$V(s) = \mu_s \text{ if } s \in \mathcal{L}, \qquad V(s) = \max_{c \in \mathcal{C}(s)} V(c) \text{ otherwise.}$$

The best child of the root s^* is the root's child with highest value,

$$s^* = \operatorname*{argmax}_{s \in \mathcal{C}(s_0)} V(s).$$

2.1. UCT algorithm

The UCT algorithm [Kocsis and Szepesvári (2006)] was introduced to find the best child at the root quickly and accurately by sequentially selecting paths from the root to a leaf in the DAG and calling the corresponding leaf oracle $\mathcal{O}_\ell$ to collect a sample of the leaf to identify s^*. The obtained reward is then backpropagated within the traversed nodes to make better decision in the next iteration. The action selection problem is treated as a separate multi-armed bandit for every node ; at iteration t, for each child $c \in \mathcal{C}(s)$ at a current node s, an upper confidence bound is computed, and the child with the highest upper confidence bound is selected for simulation. When an unvisited node is reached, simulation is performed and the reward is collected.

$$\mathrm{UCB}_{s,c}(t) = \hat{\mu}_c(t) + 2C_p\sqrt{\frac{\ln\left(T_s(t)\right)}{T_c(t)}} \tag{1}$$

with C_p being the exploration parameter. Shah et al. (2019) recently showed that a polynomial expansion parameter is more efficient than the traditional logarithmic one:

$$\mathrm{UCB}_{s,c}(t) = \hat{\mu}_c(t) + 2C_p\sqrt{\frac{T_s(t)^{1/2}}{T_c(t)}} \tag{2}$$

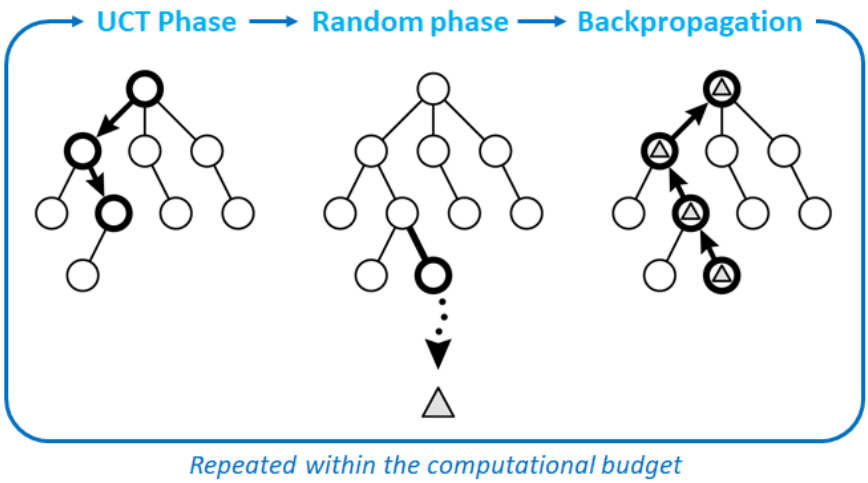

Figure 1: One iteration of the UCT search:(1)Selection, (2)Simulation, (3)Backpropagation.

This UCT frameworks is introduced in the specific case of a tree, which is a DAG where the number of parent for each node is limited to one. UCT is based on the principle that the best moves will be chosen more than the other moves, and consequently the mean of a node should converge towards the mean of its best child [Kocsis and Szepesvári (2006)].

$$\lim_{t\to\infty} \mu(s) = \lim_{t\to\infty} \mu(c^*) \quad \text{with} \quad c^* = \operatorname*{argmax}_{c\in\mathcal{C}(s)} V(c). \tag{3}$$

2.2. Transpositions in DAGs architecture

Introducing transpositions in MCTS is challenging for several reasons. Firstly, since the ancestor nodes are a superset of the traversed nodes during the selection phase, it is not clear how and which ancestor nodes should be updated after each playout. Saffidine et al. (2012) provide two straightforward possible methods: The first one, referred as the *update-descent* procedure, consists in updating only the nodes traversed during the selection phase, while the second one, referred as *update-all*, update every ancestor nodes, with each node being updated no more than once after each playout.

Although the *update-all* method seems to backpropagate the information in the most efficient way, it actually does not guaranty the optimal convergence of the UCT algorithm when the initial guess of the leaves value is inaccurate. As demonstrated by the counter example Figure 2, where equation 3 is not satisfied.

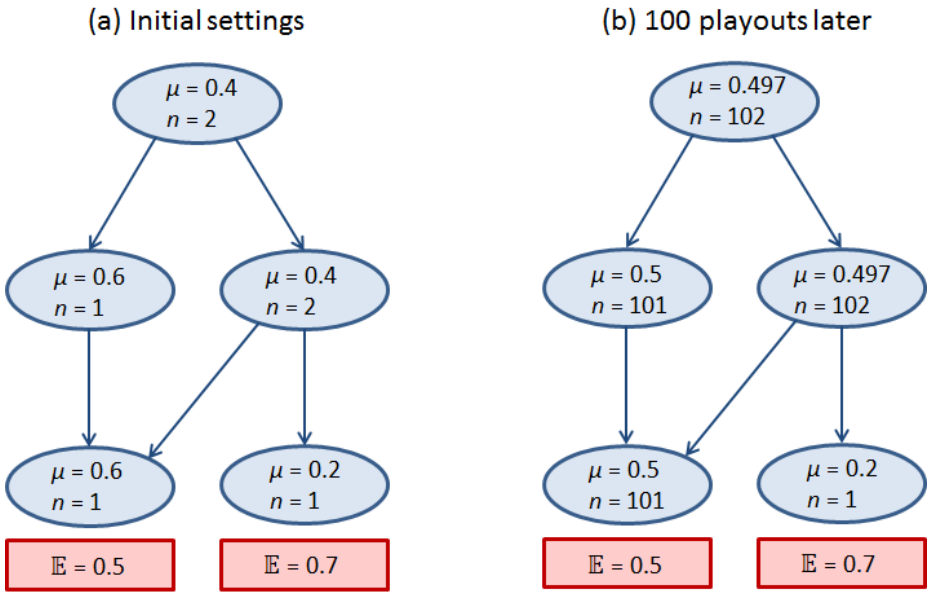

Figure 2: Counter-example for the *update-all* backpropagation procedure. If the initial estimation of the edges is inaccurate, the UCT policy combined with the *update-all* backpropagation procedure will likely lead to errors, *example taken from Saffidine et al. (2012)*.

Secondly, allowing nodes to have more than one parent node leads to the non-intuitive fact that the number of visit of a node s may no longer be equal to the sum of the children visit, even in the case where only the traversed node are updated. As a result, the average relationship between children and parent is also compromised, and this makes the theoretical analysis of UCT [Kocsis et al. (2006); Shah et al. (2019)] no longer valid.

$$\exists s \in \mathcal{V}, \quad T_s \neq \sum_{c \in \mathcal{C}(s)} T_c \quad \text{and} \quad \mu_s \neq \sum_{c \in \mathcal{C}(s)} \frac{T_c}{T_s} \mu_c \tag{4}$$

Using transpositions has a the potential to increase the convergence rate of MCTS, which is particularly important when the node evaluation is significantly more time consuming than the update. We are interested about consequences of transpositions in DAGs for the theoretical analysis of UCT, and in particular, the choice of the backpropagation policy to guaranty convergence to optimal move is the main focus of this article.

3. The α-UCD algorithm

3.1. Proposed algorithm

We denote by $\mathcal{T}(\ell)$ the set of traversed node during the selection phase to reach leaf node ℓ, and we recall that $\mathcal{A}(\ell)$ is the set of ancestor nodes of ℓ. The previous section show that the *update-all* procedure is not desired due to the non guarantied optimal convergence. On the other hand, the *update-descent* procedure miss an opportunity to make better informed decisions in MCTS. To tackle the issue, we introduce a transposition parameter $\alpha \in [0, 1]$, that represent the *importance* of the traversed nodes relatively to the other ancestor nodes during the backpropagation phase, and after each playout, we update the ancestor nodes differently depending if they are from the traversed nodes or not. The intuition behind this is that, while we still gain knowledge from transposition in the DAG, it should not be accounted as strongly as the knowledge chosen during the selection phase, hence the parameter α.

Algorithm Backpropagation α-UCD: Update a node during the backpropagation phase

input : parameter $\alpha \in [0, 1]$, node s, sampled leaf ℓ, reward V

if $s \in \mathcal{T}(\ell)$ **then**

$\mu_s \leftarrow \dfrac{\mu_s T_s + V}{T_s + 1}$

$T_s \leftarrow T_s + 1$

else if $s \in \mathcal{A}(\ell)$ **then**

$\mu_s \leftarrow \dfrac{\mu_s T_s + \alpha V}{T_s + \alpha}$

$T_s \leftarrow T_s + \alpha$

The selection phase of α-UCD algorithm is the same the UCT algorithm, and after each playout, the information is backpropagated as shown on the algorithm above. Note that the number of plays T_s are no longer integers. Here, $\alpha = 0$ corresponds to the *update-descent* policy and $\alpha = 1$ the *update-all* policy.

3.2. Theoretical Analysis

The key aspects for the analysis of α-UCD concern the [C1] Convergence and the [C2] Concentration property of each node values $\bar{X}_{s,n}$:

[C1] Convergence: $\bar{X}_{s,n}$ is assumed to converge to the optimal arm up to a radius ε_0 (drift condition):

$$\lim_{n \to \infty} \mathbb{E}[\bar{X}_{s,n}] = \mu_s \quad \text{and} \quad |\mu_s - \mu^*| < \varepsilon_0$$

.

[C1] Concentration: $\bar{X}_{s,n}$ is assumed to satisfy a concentration property with certain polynomial tail bounds. β, η and ξ being some constant defined in [Shah et al. (2019)], for every $z \geq 1$:

$$\mathbb{P}(n\bar{X}_{s,n} - n\mu_s \geq n^\eta z) \leq \frac{\beta}{z^\xi} \quad \text{and} \quad \mathbb{P}(n\bar{X}_{s,n} - n\mu_s \leq -n^\eta z) \leq \frac{\beta}{z^\xi}$$

Theorem 1 *Given a node s, if all its child nodes $c \in \mathcal{C}(s)$ satisfy the Convergence and Concentration properties, then the node s also satisfies the Convergence and Concentration properties.*

As proved by Shah et al. (2019), with an reasoning by induction starting from the leaf nodes and going up to the tree up to the root, Theorem 1 directly imply that our algorithm α-UCD converge to the best arm at the root in polynomial time. Kocsis et al. (2006) also used a similar procedure in their analysis of UCT (but with exponential tail bounds).

To prove Theorem 1, Kocsis et al. (2006) and Shah et al. (2019) base their analysis on the fact that the value of a given node is by definition the average of its children values:

$$\bar{X}_{s,n} = \frac{1}{n} \sum_{c \in \mathcal{C}(s)} T_c(n) \bar{X}_c. \tag{5}$$

However, this intuitive definition does not hold for DAGs, even for $\alpha = 0$. For our analysis, We are constrained to define the value of a node from all its descendent leaves:

$$\bar{X}_{s,n} = \frac{1}{n} \sum_{\ell \in \mathcal{L}(s)} T_\ell^{tr}(s,n) \bar{X}_\ell^{tr}(s,n) + \alpha T_\ell^{not}(s,n) \bar{X}_\ell^{not}(s,n). \tag{6}$$

With the notation being: $\mathcal{L}(s)$ = set of leaf nodes under s. $T_\ell(s,n)$ is the number of time the leaf ℓ has been pulled under node s after n playouts, where s was tr = traversed and not = not traversed. This definition, mandatory but quite complicated, makes the theoretical analysis difficult, so we take a different approach for our analysis.

3.2.1. An attempt to prove Theorem 1

Rather than studying [C1]Convergence or [C2]Concentration property directly, We define the [E1] Exploration property, that states that α-UCD never stop exploring arms. For all arm i:

$$\forall N \in \mathbb{N}, \exists p \in \mathbb{N}, T_i(p) \geq N$$

We study the bandit problem at a specific node s, and show that even in the worst case where, arms that were not selected during the descent phase are still being updated because they are ancestors of the selected leaf, all the arms verify the Exploration property for $\alpha < \frac{1}{n-1}$, n being the maximum number of arm per nodes.

Let's start with only two child a and b after n_0 playouts, and assume without loss of generality that $\mathrm{UCB}_a(n_0) \geq \mathrm{UCB}_b(n_0)$. We recall from the original description of UCT [Kocsis et al. (2006)] that:

$$\mathrm{UCB}_c(n) = \hat{\mu}_c(n) + 2C_p\sqrt{\frac{\ln(n)}{T_c(n)}} \tag{7}$$

At step n_0, arm a was sampled because its confidence bound was greater than the one of b. Let $p = n - n_0$ be the number of playout needed for arm b to be picked. Let $\delta_{ab} = \mu_a - \mu_b$. To ease the presentation, we denote $T_a(n_0) = T_a$ and $T_b(n_0) = T_b$. Note that $T_a + T_b \neq n_0$ due to the DAG architecture. We assume that the reward when arm a is being picked is always μ_a to simplify the calculation (Although this statement is not true, the value of node a converge to μ_a after a high number of playout). We have:

$$\begin{aligned}
&\mathrm{UCB}_a(n_0+p) - \mathrm{UCB}_b(n_0+p) = \delta_{ab}(n_0+p) + I_{ab}(n_0+p) \\
&\delta_{ab}(n_0+p) \approx \frac{T_b\delta_{ab}(n_0) + \mu_a p\alpha}{\mu_b + p\alpha} \\
&I_{ab}(n_0+p) = 2C_p\sqrt{\ln(n_0+p)}\left[\frac{1}{\sqrt{T_a+p}} - \frac{1}{\sqrt{T_b+p\alpha}}\right]
\end{aligned} \tag{8}$$

Case 1: $T_a > T_b$

This case implies that $I_{ab}(n_0) < 0$, and thus

$$\mathrm{UCB}_a(n_0) - \mathrm{UCB}_b(n_0) = \delta_{ab}(n_0) + I_{ab}(n_0) \tag{9}$$

will inevitably reach to a time p where $\mathrm{UCB}_a(n_0+p) - \mathrm{UCB}_b(n_0+p) < 0$ as μ_a and μ_b converges towards the same value: $\lim_{p\to\infty} \delta_{ab}(n_0+p)) = 0$

Case 2: $T_a \leq T_b$, we assume $\alpha p \gg T_b$

$$\begin{aligned} &\mathrm{UCB}_a(n_0+p) - \mathrm{UCB}_b(n_0+p) < 0 \\ \implies &\frac{1}{\sqrt{T_b + p\alpha}} > \frac{1}{\sqrt{T_a + \alpha}} \\ \implies &T_b + p\alpha < T_a + p \\ \implies &p > \frac{T_b - T_a}{1-\alpha} \end{aligned} \tag{10}$$

Which proves that the exploration property [E1] is true for $\alpha < 1$ and 2 arms

Generalization to n arm:
(Need to provide the details, and this part might be inaccurate)
We have shown that for two arms, α-UCD never stop exploring, as long as $\alpha < 1$. the generalization to multiple arms can be done by induction. Assume that the Exploration property [E1] is true for N arms for some exploration parameter $\alpha < \frac{1}{N-1}$. We add another arm to the tree. By adding this new arm, we increase the chance of one arm being wrongly updated. In the worst case where this new arm always wrongly update the best arm, we obtain:

$$p > \frac{\min_{i \neq a}(T_i - T_a)}{1 - N\alpha} \tag{11}$$

Which is positive only for $\alpha < \frac{1}{N}$.

Here we proved the [E1] Exploration property for any node in the DAG, and only when $\alpha < \frac{1}{n-1}$, n being the maximum number of child nodes per node. It is however not clear to the author how to derive the Convergence and Concentration properties from these results. At least, we can interpret that, even if updating all the ancestors node can lead to sometimes wrongly updating some arm, the parameter α will always guaranty that this arm will be picked so that its value can converge to its real value eventually.

4. Experiment

4.1. Practical implementation

The original description of UCT [Kocsis et al. (2006)] assumes that all the leaf are pulled at least once to avoid infinite confidence bounds. In practice however, trees often have a very high branching factors and it's impractical, so we usually introduce a RAVE score and limit the number of children allowed per nodes [Browne et al. (2012)] ; the number of nodes to be considered is then increased with the number of playouts. In the case where the new node to be added is the parent of some already existing nodes, (it can happen due to the DAG architecture), the value of the new node is immediately set to the average of its existing children, and all of its corresponding edges are also added to the DAG.

4.2. Benchmark on a feature selection problem

Feature selection is a good way to check for the benefit of transposition because it has a high number of transpositions in DAGs representation (Figure 3). To find the best feature subset, Gaudel and Sebag (2010) introduced the algorithm FUSE (Feature Uct SElection), that starts from the empty feature subset and relies on UCT to identify the best leaf, which is taken at the end of the search as the path with the highest average at each step from the root node. In this article we omit the formalism of feature selection problem into a best leaf identification problem, since all the necessary details are already provided in [Pélissier et al. (2019)] and [Gaudel and Sebag (2010)]. The benchmark dataset used in our experiment is the Madelon dataset [Guyon (2003)], which was designed for the NIPS 2003 feature selection challenge [Guyon et al. (2005)].

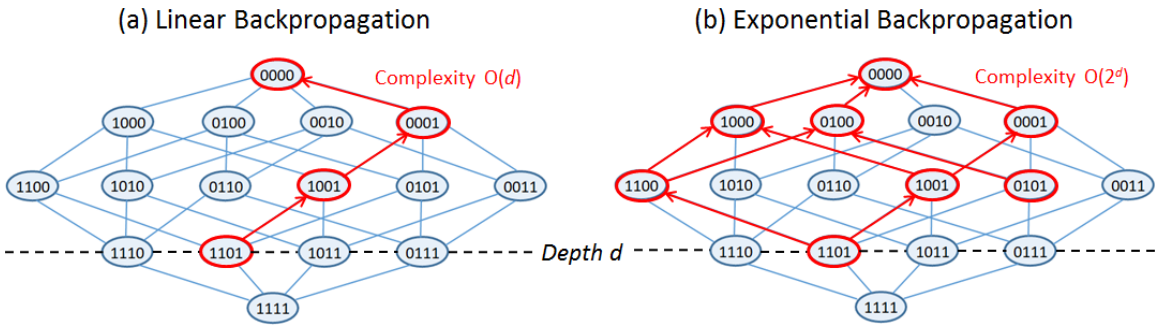

Figure 3: Backpropagation phase for a 4-features selection problem with (a) $\alpha = 0$ and (b) $\alpha = 1$. The red path corresponds to the updated node after sampling the leaf node 1101.

We adapt the algorithm α-UCD into the FUSE algorithm (that we denote α-FUSE) and check how the value of α affects the performances of the search. We run α-FUSE with 50000 iterations[1], independently 100 times for different values of α. Figure 4 plot the average reward obtained from the best feature set at the end of the search for different values of α [0,0.1,0.25,0.5,0.75,1].

1. The `C++` implementation for Feature Selection with α-FUSE is available on Github at https://github.com

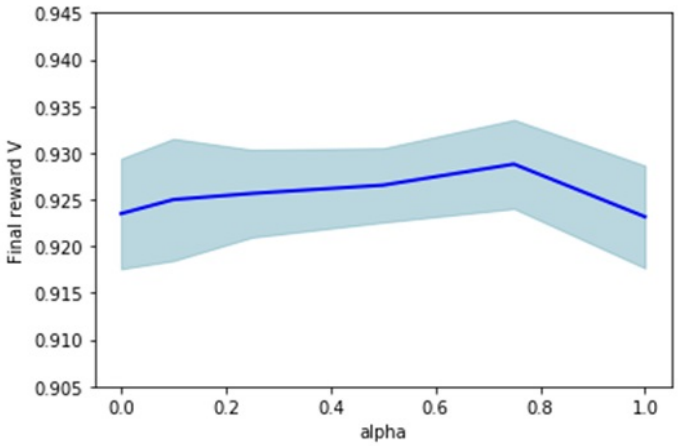

Figure 4: Obtained Reward averaged over 100 trials for different values of α [0,0.1,0.25,0.5,0.75,1]. The shaded area is the 95% confidence interval.

Surprisingly, the results contradict our intuition, and the value of α does not seem to significantly affect the algorithm performance. It seems like the output result is rather dominated by the discrete heuristic procedure that adds new nodes depending on their RAVE score. Another design of experiments, that does not rely on RAVE scores, will be needed for further analysis. In their study, Saffidine et al. (2012) obtained improved performances when using transpositions.

5. Summary and Discussions

We focused on the theoretical analysis of transpositions in UCT, about which very little was known, and we introduced the algorithm α-UCD, that backpropagates the information to all ancestor nodes after each playout in a way that still guaranty asymptotic optimal convergence. Our experiments failed to indicate the dependence of α on the convergence rate

While our article focuses on the confidence bound initially proposed by Kocsis and Szepesvári (2006) ($\sim \sqrt{\log(t)/s}$), Shah et al. (2019) recently showed theoretically that it is better to modify the bias term into $\sqrt{t^{1/2}/s}$. Our framework can be easily generalized to this new formula, and thus has the potential to significantly improve AlphaGo Zero (that also uses a polynomial form for the bias).

Finally, as for further research, one could consider accounting for cycles in the game state space and further generalize α-UCD to connected graph. The question is related to the Graph History Interaction (GHI) problem for which a general solution was proposed by Kishimoto and Müller (2004).

Acknowledgments

Aurélien Pélissier acknowledges the financial support from the École Normale Supérieure (ENS) Paris Saclay and Hokkaido University.

References

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Benjamin E Childs, James H Brodeur, Levente Kocsis, et al. Transpositions and move groups in monte carlo tree search. In *CIG*, pages 389–395, 2008.

Romaric Gaudel and Michele Sebag. Feature selection as a one-player game. In *International Conference on Machine Learning*, pages 359–366, 2010.

António Gusmao and Tapani Raiko. Towards generalizing the success of monte-carlo tree search beyond the game of go. In *ECAI*, pages 384–389, 2012.

Isabelle Guyon. Design of experiments of the nips 2003 variable selection benchmark, 2003.

Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the nips 2003 feature selection challenge. In *Advances in neural information processing systems*, pages 545–552, 2005.

Akihiro Kishimoto and Martin Müller. A general solution to the graph history interaction problem. In *AAAI*, volume 4, pages 644–649, 2004.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

Levente Kocsis, Csaba Szepesvári, and Jan Willemson. Improved monte-carlo search. *Univ. Tartu, Estonia, Tech. Rep*, 1, 2006.

Aurélien Pélissier, Atsuyoshi Nakamura, and Koji Tabata. Feature selection as monte-carlo search in growing single rooted directed acyclic graph by best leaf identification. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 450–458. SIAM, 2019.

Abdallah Saffidine, Tristan Cazenave, and Jean Méhat. Ucd: Upper confidence bound for rooted directed acyclic graphs. *Knowledge-Based Systems*, 34:26–33, 2012.

Devavrat Shah, Qiaomin Xie, and Zhi Xu. On reinforcement learning using monte carlo tree search with supervised learning: Non-asymptotic analysis. *arXiv preprint arXiv:1902.05213*, 2019.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.