Prior Estimation in UCT based on the Sequential Halving Algorithm

Yun-Ching Liu[1,a)] Yoshimasa Tsuruoka[1]

Abstract:
Recent developments in the pure exploration formulation of the multi-armed bandits problem, have lead to the development of algorithms that can efficiently identify and estimate the mean reward of the optimal arm. Applications of these "pure exploration bandit algorithms" to game tree search have been of interest in recent years, and most of which are in the modification of the selection stage in the Monte-Carlo Tree Search (MCTS), trying to more effectively identify the principal variation. Instead of identifying the principal variation, another less explored possibility is utilizing the pure exploration bandit algorithms in estimating a prior UCB value for a newly expanded node. In this research, we utilize the sequential halving algorithm, which is a pure exploration bandit algorithm, for obtaining a prior estimation in the UCT algorithm. We have demonstrate the performance and characteristics of our proposed method on the game of 9×9 Go. The experimental results shows that the method is enhanced the performance of the UCT algorithm, and the use of sequential halving algorithm is more effective than uniformly distributing the simulations for prior estimation. The proposed estimation method also displayed an interesting phenomenon of performing better when playing White. These results suggests there are potential in applying simple regret bandit algorithms to prior estimation.

1. Introduction

The application of Monte-Carlo Tree Search (MCTS) in various fields has achieved a number of interesting and important results, especially in the field of Computer Go[1]. The most successful variant of MCTS is the Upper Confidence Bound on Trees (UCT) algorithm, which applies the Upper Confidence Bound (UCB) algorithm to distribute random simulations effectively and efficiently [2].

The UCB algorithm is an algorithm that optimally solves the *multi-armed bandits problem*, or also known as the *"k-armed bandit problem"*[6]. In the multi-armed bandits problem, a player faces a number of slot machines, also known as "one-armed bandits", and each machine produces a random reward according to some unknown probability distribution when it is pulled. The objective of the player is to maximize his or her sum of reward by a sequence of pulls on these machines. The multi-armed bandits problem embodies the essence of the *exploration vs. exploitation dilemma*, in which exploitation is the decision to perform the "current known" optimal action, and exploration is performing an action to gather more information, but the action may not be optimal[7]. The UCB algorithm solves the multi-armed bandits problem by maintaining and upper confidence bound value for each arm, and pulls the arm with the highest value[6].

[1] Department of Electrical Engineering and Information Systems, Graduate School of Engineering, University of Tokyo, Tokyo, Japan.
[a)] cipherman@logos.t.u-tokyo.ac.jp

The UCT algorithm is essentially a Monte-Carlo Method, which performs a series of random simulations or iterations to identify and estimate the value of the optimal move of a position in a game[2]. Each iteration consists of four major stages: *selection*, *expansion*, *simulation*, and *backpropagation*[3]. In the selection stage, we select a leaf node, and then expand the selected leaf node in the expansion stage. From the position of the expanded leaf node, we then play random moves until the end of a game in the simulation stage. Finally, we update relevant information maintained in the nodes along the path that we have just visited, from the expanded leaf node up to the root node. The UCT algorithm continues to perform these iterations until some stopping criteria is met, such as the time limit.

The UCT algorithm mainly uses the UCB algorithm in the selection stage, by viewing each node as an independent instance of the multi-armed bandits problem, where each child node is a single arm. Therefore, an upper confidence bound value is maintained for each expanded node, and the UCT algorithm selects the leaf that has the highest value to expand at each iteration [3]. It has been shown that as the number of iterations $n \to \infty$, the minimax value of the root node can be obtained[2].

Various enhancements for improving the performance of the UCT algorithm have been proposed, and each method has a different focus and approach, for example, some methods have been focusing on modifying the random simulations to approximate "real-games"[3][4], and some have investigated the possibility of performing pruning [5].

One major approach is on obtaining a good prior estimation for a newly expanded node at each iteration, because if the initialized UCB value of the newly expanded nodes are close to its true value, the convergence rate of the tree search can be increased significantly[1][8].

Recent developments in the pure exploration formulation of the multi-armed bandits problem[13], have lead to the development of algorithms that can efficiently identify and estimate the mean reward of the optimal arm[13][20]. Applications of these "pure exploration bandit algorithms" to game tree search have been of interest in recent years [21][23], and most of which are in the modification of the *selection* stage in the Monte-Carlo Tree Search (MCTS); they all try to more effectively identify the principal variation. Instead of identifying the principal variation, another less explored possibility is utilizing the pure exploration bandit algorithms in the estimating a prior UCB value for a newly expanded node.

In this research, we use the sequential halving algorithm[20], which is a pure exploration bandit algorithm, for obtaining a prior estimation in the UCT algorithm. We will first give a brief overview of related research in the next section, and present our proposed method in section 3. We will demonstrate the behaviour of our proposed method in section 4, and finally discuss the results and possible directions for further investigation in section 5.

2. Related Work

In this section, we will begin by reviewing concept of cumulative regret in the conventional setting of the multi-armed bandit problem and the concept of cumulative regret. We will then introduce the pure exploration formulation of the multi-armed bandit problem, the concept of simple regret, and the relation between cumulative and simple regret. Finally, an overview of the developments of bandit algorithms for minimizing simple regret, and MCTS algorithms based on these simple regret bandit algorithms will be given.

2.1 Multi-armed Bandits Problem and Cumulative Regret

The multi-armed bandit problem is a tuple $<\mathcal{A},\mathcal{R}>$, where $\mathcal{A}$ is a known set of m arms (or "slot machines"), and $\mathcal{R}^a(r) = \mathbb{P}[r|a]$ is an unknown probability distribution over rewards. At each round the player selects an action $a_t \in \mathcal{A}$, and the selected machine returns a reward r_t according to R^{a_t}. The objective is to maximize the cumulative reward $\sum_{\tau=1}^{t} r_\tau$.

Suppose the mean reward for machine a is $Q(a) = \mathbb{E}[r|a]$, and the optimal value V^* is defined by $V^* = Q(a^*) = max_{a\in\mathcal{A}}Q(a)$, then the regret is the opportunity loss at a single round $I_t = \mathbb{E}[V^* - Q(a_t)]$. Therefore, the *cumulative regret* is the expected sum of opportunity loss over t rounds

$$R_t = \mathbb{E}[\textstyle\sum_{\tau=1}^{t} V^* - Q(a_\tau)]$$

It is easy to see that as the cumulative reward increases, the cumulative regret will also decrease; therefore, the task of an multi-armed bandits algorithm can also be equivalently formulated as minimizing the cumulative regret.

2.2 Pure Exploration Multi-armed Bandits Problem and Simple Regret

The pure exploration variation of the multi-armed bandits problem was first proposed in 2009 by Bubeck et al.[13]. While the objective in the conventional formulation of the multi-armed bandits problem is to accumulate as much reward as possible, the goal of the pure exploration formulation is to identify the arm that has the highest expected reward, given a pre-determined number of rounds.

In other words, in the pure exploration formulation of the multi-armed bandits problem, after a fixed number of rounds, the algorithm should recommend an arm which it identifies as the optimal arm. Therefore, the algorithm should gather as much information as possible for making the recommendation, and not worry about the cost paid during play.

While the conventional setting was formulated to capture the essence of practical situations, where it is more desirable to minimize the cost during the trials, such as clinical trials[10][11], the pure exploration setting can be found in applications, where it is more desirable that the final product would be the best among all the tested possibilities, such as tests conducted in cosmetics, or channel selection during the initialization of a mobile device[13].

Therefore, in the pure exploration formulation, instead of minimizing the cumulative regret $R_t = \mathbb{E}[\sum_{\tau=1}^{t} V^* - Q(a_\tau)]$, one should minimize the simple regret, which is the difference between the expected reward of the recommended arm and the true optimal arm. The simple regret is defined as

$$r_t = \mathbb{E}[V^* - Q(a_{rec})],$$

where $V^* = Q(a^*) = max_{a\in\mathcal{A}}Q(a)$, is the optimal arm that has the highest mean reward, and $Q(a_{rec})$ is the mean reward of the identified or the recommended arm a_{rec} after t rounds of play.

Since the objective of pure exploration formulation is to "recommend" the arm that has the highest expected reward, apart from the *allocation strategy*, which decides which arm to play in the next round, there is also a need for a *recommendation strategy*, which determines the arm to recommend when the designated number of rounds has finished.

It has been shown that for allocation strategies φ_i on K-arm bandits, in which all the bandits have a (Bernoulli) distribution $\nu_1,\cdots,\nu_K$, there exists a constant C, such that the cumulative regret is bounded by

$$\mathbb{E}[R_t] \leq C\varepsilon(t)$$

where $\varepsilon(t)$ is a function that $\varepsilon : \{1,2,\cdots\} \to \mathbb{R}$. Then, for all the recommendation strategies ψ_i based on these allocation strategies φ_i, these exists a constant D, such that simple regret will have the lower bound of

$$\mathbb{E}[r_t] \geq \frac{\Delta}{2} e^{-D\varepsilon(t)}$$

where Δ is the gap between the mean reward of optimal arm and the second best arm[13].

This result shows that when minimizing the cumulative regret, we are increasing the simple regret at the same time. For example, an optimal algorithm for minimizing the cumulative regret will often have $\varepsilon(t) = \log t$, as it is for the UCB algorithm. But this implies that these optimal algorithms would only decrease the simple regret at a polynomial rate. Therefore, a different approach needs to be taken for tackling the pure exploration formulation of the multi-armed bandit problem.

2.3 Simple Regret Bandit Algorithms

Since the formulation of the pure exploration of multi-armed bandit problem, different efforts have been made to develop algorithms for minimizing the simple regret, and finding the theoretical lower bound.

In the initial development of simple regret bandit algorithms, the approach of considering allocation and recommendation strategy separately, has been taken[13]. Later developed algorithm adopts the approach that systematically ruling out arms that are sub-optimal, and the last remaining arm would be the recommended arm[14][15][16][17].

Recently, the theoretical lower bound has been established and an optimal algorithm, the lil′ UCB algorithm, has been proposed by Jamieson et. al.[19]. Suppose that we are given K armed bandits, and their rewards are decided by the distributions $\nu_1, \nu_2, \cdots, \nu_K \overset{i.i.d}{\sim} \mathcal{N}(\Delta, 1)$, where the gap between any two bandits $\Delta \neq 0$ is unknown. Consider testing whether $\Delta > 0$ or $\Delta < 0$. Let $Y \in \{-1, 1\}$ be the decision of any such test based on T samples and let $\delta \in (0, 1/2)$. If $\sup_{\Delta \neq 0} \mathbb{P}(Y \neq sign(\Delta)) \leq \delta$ then

$$\limsup_{\Delta \to 0} \frac{\mathbb{E}_\Delta[T]}{\Delta^2 \log\log \Delta^2} \geq 2 - 4\delta$$

This theoretical lower bound is derived from law of the iterated logarithm (LIL), which describes the magnitude of the fluctuations of a random walk.

2.3.1 Monte-Carlo Tree Search based on Simple Regret Bandit Algorithms

Recently, there have been increasing interests in applying simple regret bandit algorithms to MCTS.

It has been argued that the pure exploration formulation of the multi-armed bandits algorithm is actually more suited to the task of game tree search, since the objective is not only give an evaluation value of the current game position, but also decide the best move or course of action to execute[21].

The Simple Regret + Cumulative Regret (SR+CR) MCTS sampling scheme takes a hybrid approach in adopting both simple regret and conventional bandit algorithms in MCTS. The overall process of the SR+CR MCTS sampling scheme is mostly the same as the UCT algorithm, except that simple regret bandit algorithms are used in the first level of the game tree, and the UCB algorithm is used for the rest. It has been demonstrated to have better performance than the UCT algorithm in certain situations [21].

On the other hand, the Sequential Halving on Trees (SHOT) algorithm is an MCTS algorithm that is based purely on simple regret bandit algorithms [23]. The SHOT algorithm iteratively expands the game tree, and a fixed budget of playouts are performed in each iteration. The playouts are allocated according to the Sequential Halving algorithm, which is a near optimal simple regret bandit algorithm [20]. The SHOT algorithm has been demonstrated to outperform the UCT algorithm on the game of NoGo.

The H-MCTS algorithm is a hybrid of the SHOT algorithm and the UCT algorithm[22]. Similar to the SR+CR MCTS sampling scheme, the H-MCTS algorithm utilize the SHOT algorithm on top part of the expanded game tree, and the UCT algorithm in deeper depths. The H-MCTS algorithm has shown to outperform the UCT algorithm on various games, such as Amazons, AtariGo, and Breakthough.

3. Proposed Method

As shown in the previous section, most approaches that adopts simple bandit algorithms in MCTS are mainly focused on the selection stage of the MCTS. Rather than the *selection* stage, we investigate the possibility of utilizing simple bandit algorithm for prior estimation in the *expansion* stage of MCTS.

In this section, we will first give some preliminary observation in the general process of game tree search, and then present the Sequential Halving algorithm, which we will adopt in our method. Finally, we will show our method of prior estimation by utilizing the Sequential Halving algorithm.

3.1 Exploration and Evaluation

If we take a step back and observe the general search process on the game tree, we can break the process into two main tasks: *exploration* and *evaluation*. When searching on the game tree, the task of exploration is the decision of which part of the game tree should be expanded, and the task of evaluation is to estimate a value of a node, since we could not search the game tree exhaustively. In the conventional game tree search methods, the expansion and evaluation tasks are mostly achieved by the $\alpha\beta$-search algorithm and evaluation function, respectively. Various methods have been developed to enhance the performance in these two respective tasks. For example, $\alpha\beta$ pruning enhances performance in the exploration task, and auto parameter tuning and machine learning enhances performance in the evaluation task.

The exploration and evaluation tasks can also be observed in MCTS. The exploration task can be mainly observed in the selection stage, since it decides where in the game tree should we expand, and it is mainly dictated by the adopted bandit algorithms. The evaluation task can be observed in the expansion and simulation stage. Similar to that of the conventional game tree search methods, enhancement heuristics can also be roughly categorized in according to these two tasks, for example, progressive pruning [5] in the exploration task, and RAVE in the evaluation task [8].

Since the main objective of the pure exploration multi-armed bandit algorithm is to identify the optimal arm in a given budget of plays, it is natural to expect simple bandit algorithms should estimate the expect reward of each arm efficiently. Therefore, utilizing simple bandit algorithms to enhance the performance of the evaluation task in MCTS seems to be promising possibility.

It has been shown that a good initialization value given to the newly expanded node, can enhance the speed of convergence in MCTS[8], and that simple bandit algorithms seem to be able to estimate the average reward efficiently. Hence, we utilize the simple bandit algorithms in estimating the prior of the newly expanded nodes.

3.2 Sequential Halving Algorithm

The simple bandit algorithm we have adopted is the sequential halving algorithm[20].

The sequential halving algorithm is depicted in Algorithm 1. It divides the given budget of plays into a number of iterations, and in each iteration the number of pulls on each arms are equal. Each iteration will eliminate worst half of the arms, and then retain the top half.

An example is shown in Figure 1. There are four possible actions A_1, A_2, A_3, and A_4. The given budget is 64 plays, and the plays are divided into three iterations. In the first iteration, each action is sampled 8 times, with A_1 winning 7 games, A_2 winning 3 games, A_3 winning 8 games, and A_4 winning 5 games. Hence we eliminate the worst half, namely A_2 and A_4, and retain A_1 and A_3 for further sampling. In the second iteration, we further sample 16 more games on A_1 and A_3. Therefore, together with the first iteration, A_1 and A_3 are sampled a total of 24 times, winning a total of 19 and 22 out of 24 games, respectively. Since we have reached the given number of budget, we recommend the action of A_3 in the third iteration.

The reason for choosing the sequential halving algorithm is that the only parameter we need to give is the total budget of plays, which significantly simplifies the task of performance tuning. Although the lil'UCB algorithm, introduced in Section 2, has been shown to be optimal, and sequential halving is only "near" optimal, the lil'UCB algorithm has more parameters that are needed to tune, which may significantly make the task of identifying the key factors that affect performance more complicated.

3.3 Prior Estimation by Simple Bandit Algorithms

We will use the UCT algorithm as our MCTS algorithm, but instead of just performing a single simulation in the *simluation* stage, as shown in Figure 2(a), we perform a predetermined number of simulations to obtain a more accurate estimation of the UCB value of the newly expanded node, as depicted in Figure 2(b).

Since we are performing search on game trees, the value of the newly expanded node will most likely be close to the value of its optimal child node. Hence, in order to obtain a more accurate estimation, distributing more simulations to a child node that will most likely have the optimal value is more desirable. Hence, we utilize the sequential halving algorithm to distribute the simulations performed for initializing the newly expanded node among its child nodes.

The UCB value of each node is the same as the UCT algorithm, which is defined as follows[2]:

$$UCB_i = v_i + C\sqrt{\frac{\log N}{n_i}}$$

where v_i is the value of node i, n_i is the number of times that node i has been visited, N is the number of times that the parent of node i has been visited, and C is a constant parameter. The results of the simulations are only used for updating the the winning rate v_i in the UCB formula, while the updating for other terms remains the same, that is n_i is initialized to 1. This is due to the fact that the criteria of distributing simulations by sequential halving and UCB are different; therefore, the number of times visited, namely the N and n_i terms, can not be transferable, but the value v_i for each move should be the same.

4. Experimental Results

We performed two experiments to demonstrate the performance and behaviour of our proposed method. The first experiment is a comparison between the pure UCT algorithm and our proposed method. The second experiment is a comparison between uniform and sequential halving distribution of the multiple simulations performed for node initialization.

4.1 Experimental Settings

The experiments are performed on the game of 9×9 Go. A total of 1000 playouts are performed for each tree search, with pure random simulations. A match of 100 games were performed for each comparison experiment, and each method takes turns in playing Black and White.

4.2 Performance Against Pure UCT

The constant c in the UCB formula was set to 0.31 for both pure UCT and UCT with prior estimation. We performed 64 simulations for initializing newly expanded nodes in our proposed method of prior estimation.

Our proposed method won 61 games out of the 100, achieving a winning rate of 61%.

This result shows that our method effectively enhanced the performance of the UCT algorithm, although this is not

Algorithm 1 Sequential Halving Algorithm [20]

```
1: SequentialHalving($budget$)
2: $S \leftarrow [possibleMoves]$
3: while $|S| > 1$ do
4:     for each move $m$ in $S$ do
5:         play($m$)
6:         perform $\lfloor \frac{budget}{|S| \times \lceil \log_2(possibleMoves) \rceil} \rfloor$ playouts
7:         undo($m$)
8:     end for
9:     $S \leftarrow$ the set of $\lceil \frac{|S|}{2} \rceil$ moves with the largest empirical average
10: end while
```

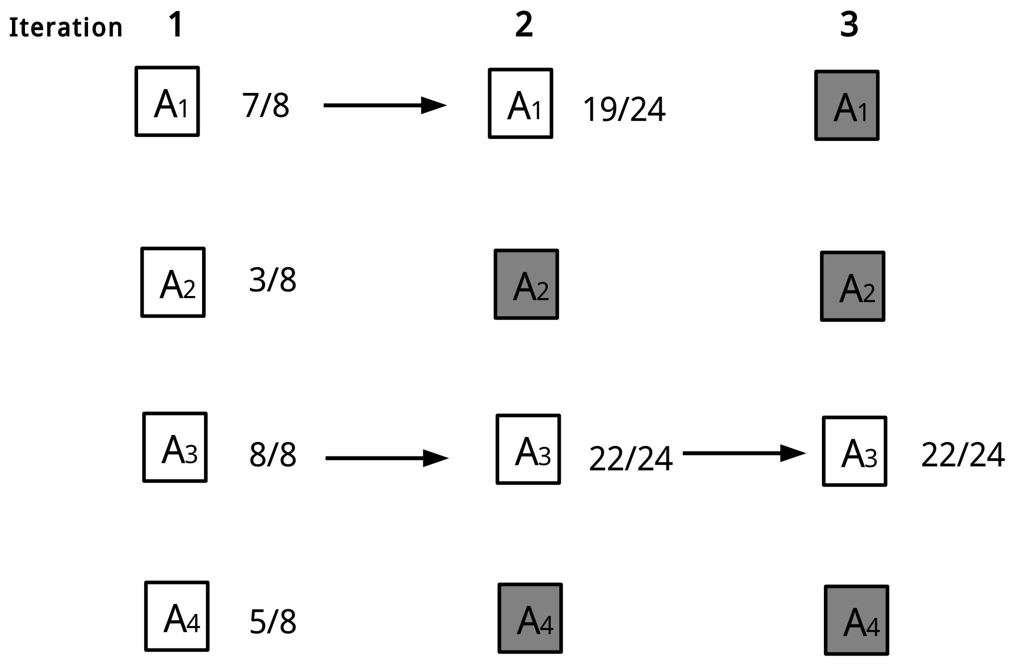

Fig. 1: *An example of the sequential halving algorithm.* There are four possible actions A_1, A_2, A_3, and A_4. The budget is 64 playouts, divided into 3 iterations. The action A_3 is the recommended action of the algorithm.

surprising, since our proposed method performed far more number of simulations despite the same number of playouts are performed for both methods.

An interesting observation is that 43 of the 61 games that our proposed method won are with White, which hints there is an unbalance in the performance of our proposed method.

4.3 Performance Comparison with Uniform Distribution of Simulations

We investigate the difference between utilizing sequential halving and uniform distribution of the multiple simulations performed for prior estimation. There are two possible factors that might affect the performance, namely the number of given budget, and the value of the constant C in the UCB value. We will observe their influence separately.

4.3.1 Comparison between Different Budget

The performance against the uniform distribution of the multiple simulations given different budget are shown in Table 1.

Budget	No. of Win Games	Wins with White
16	55	32
32	52	37
64	50	33
128	34	34

Table 1: Comparison with uniform distribution of initialization simulations. Variation of given simulation budget of simulations for node value initialization.

We can observe that the performance of distribution by sequential halving converges to that of uniform distribution when the number of given simulation budget increases. The winning rate is roughly the same when the budget of 64 is given, but worsens when a budget of 128 is given.

This shows that distribution by sequential halving is more effective in prior estimation with a smaller number of budget, but less effective or even worse when the budget increase.

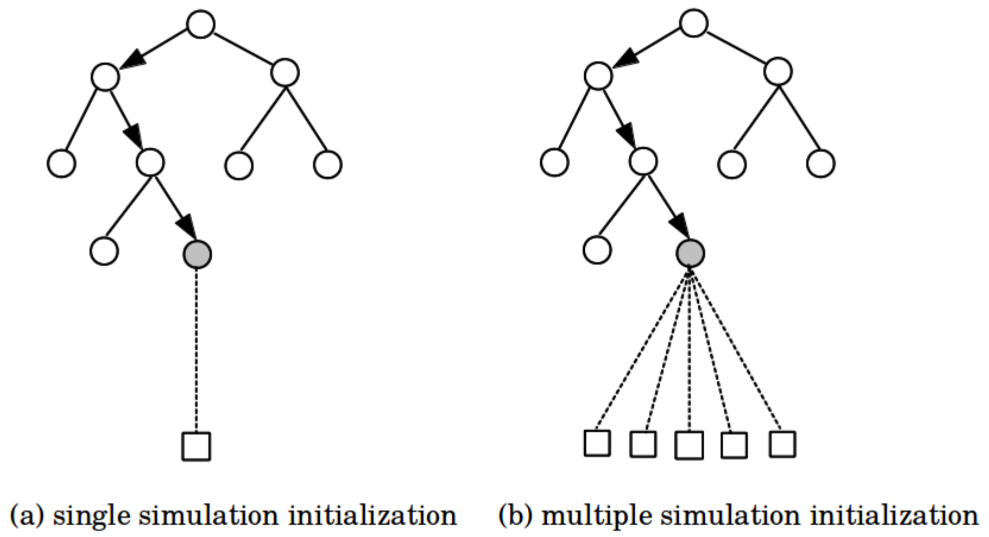
(a) single simulation initialization
(b) multiple simulation initialization

Fig. 2: *Methods of initializing UCB values for newly expanded nodes.* The grey node is the newly expanded node, and the dotted line indicates a single random simulations performed to initialize the UCB of the grey node.

We have also observed the same phenomenon as in the experiment against pure UCT, where our proposed method is more effective when playing with White.

4.3.2 Comparison between Different Setting of Exploration Constant

The performance against the uniform distribution of the multiple simulations given different number of budget are shown in Table 2.

constant C	No. of Win games	Wins with White
0.05	60	37
0.1	58	36
0.3	52	37
0.6	47	36

Table 2: Comparison with uniform distribution of initialization simulations. Variation of different settings of the constant C in the UCB formula.

We can observe that the performance of distribution by sequential halving is more effective with C set to a smaller value. The winning rate is roughly the same when C is set to 0.3, and worsens when set to 0.6.

This shows that distribution by sequential halving has higher accuracy in prior estimation. Therefore, less exploration is needed, and thus the value of C should be smaller.

Again, the same phenomenon has also been observed, where the distribution by sequential halving performs better when playing with White.

5. Conclusion

We have proposed a new method for prior estimation in the UCT algorithm. The method performs a number of predetermined number of simulations distributed by the sequential halving algorithm.Experiments show promising results, and performs particularly well when playing with White, which is a very interesting phenomenon.

This research is a preliminary investigation into the possibility of using simple bandit algorithms for prior estimation in the UCT algorithm. The results has shown some potential in this approach, but it should be noted that the proposed method might still not be practical in real gameplay, because the extra time for performing prior estimation simulations might be better spent on the expansion of the game tree. Modifications, such as logistic regression to model the simple regret bandit based prior estimation, are a possibility for making such an approach practical.

Further investigations are needed to identify the reason to its uneven performance and other characteristics. Explorations in to the possibilities of utilizing different pure exploration bandit algorithms for simulation distribution or integration with other methods, such as RAVE[8], will certainly be of interest.

References

[1] Browne, C., Powley, E.J., Whitehouse, E., Lucas, S.M., Cowling, P.I., Rohlfshagen, Tavener, P.S., Perez, D., Samothrakis, S., Colton, S.: A Survey of Monte Carlo Tree Search Methods. In: IEEE Transactions on Computational Intelligence and AI in Games, 4(1):143 (2012)
[2] Kocsis, L., Szepesvari, C.: Bandit based Monte-Carlo planning. In: Proceedings of the 17th European conference on Machine Learning, pp. 282-293 (2006)
[3] Gelly,S., Wang, Y., Munos,R., Teytaud O.: Modification of UCT with Patterns in Monte-Carlo Go. Technical report, INRIA (2006)
[4] Coulom, R.: Computing Elo Ratings of Move Patterns in the Game of Go. In: ICGA Journal, 30(4), pp.198-207 (2007)
[5] Bouzy, B.: Move-Pruning Techniques for Monte-carlo Go. In: Proceedings of the 11th International Conference on Advances in Computer Games. pp. 104-119 (2006)
[6] Auer, P., Cesa-Bianchi,N., Fischer, P.: Finite-time analysis of the multi-armed bandit problem. In: Machine learning, 47(2-3):235256 (2002)
[7] Sutton, R., Barto, A.: Reinforcement Learning, MIT Press,(1998)
[8] Gelly, S., Silver, D.: Combining Online and Offline Knowledge in UCT. In: Proceedings of the 24th International Conference on Machine Learning. pp. 273-280 (2007)
[9] Chaslot, G., Winands, M., Uiterwijk, J., van den Herik, J., Bouzy,

B.:Progressive Strategies for Monte-Carlo Tree Search. In: New Mathematics and Natural Computation, Vol. 4, No. 3. pp. 343-357 (2008)

[10] Robbins, H: Some aspects of the sequential design of experiments. In: Bulletin of the American Mathematical Society, 58 (5) pp. 527535 (1952)

[11] Lai, T.L.,Robbins, H. (1985): Asymptotically efficient adaptive allocation rules. In: Advances in applied mathematics 6 (1): 4 (1985)

[12] Bubeck, S., Cesa-Bianchi, N.: Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. In: Foundations and Trends in Machine Learning, Vol 5: No 1, pp. 1-122 (2012)

[13] Bubeck, S., Munos, R., Stoltz, G.: Pure exploration in multi-armed bandits problems. In: Proceedings of the 20th International Conference on Algorithmic Learning Theory (2009)

[14] Audibert, J.-Y., Bubeck, S., Munos, R. : Best Arm Identification in Multi-Armed Bandits. In: Proceedings of the 23th annual conference on Computational Learning Theory (2010)

[15] Gabillon, V., Lazaric, A., Ghavamzadeh, M., Bubeck, S.:Multi-bandit Best Arm Identification. In: Proceedings of the Twenty-Fifth Annual Conference on Neural Information Processing Systems (2011)

[16] Kalyanakrishnan, S., Tewari, A., Auer, P., Stone, P.: PAC Subset Selection in Stochastic Multi-armed Bandits. In: Proceedings of the 29th International Conference on Machine Learning (2012)

[17] Bubeck, S., Wang, T., Viswanathan, N.: Multiple Identifications in Multi-Armed Bandits. In: Proceedings of the 30th International Conference on Machine Learning (2013)

[18] Munos, R.:From bandits to Monte-Carlo Tree Search: The optimistic principle applied to optimization and planning. In: Foundations and Trends in Machine Learning, pp. 1-130 (2013)

[19] Jamieson,L., Malloy, M., Bubeck, S., Nowak, R.: lil' UCB: An Optimal Exploration Algorithm for Multi-Armed Bandits. In: Proceedings of the 27th annual conference on Computational Learning Theory (2014)

[20] Karnin, Z., Koren, T., Somekh, O.: Almost Optimal Exploration in Multi-Armed Bandits. In: Proceedings of the 30th International Conference on Machine Learning.pp. 1238-1246 (2013)

[21] Tolpin, D., Shimony, S.E.: MCTS Based on Simple Regret. In: Proceedings of the 26th AAAI Conference on Artificial Intelligence, pp. 570576 (2012)

[22] Pepels, T., Cazenave, T., Winands, M.H.M., Lanctot, M.: Minimizing Simple and Cumulative Regret in Monte-Carlo Tree Search. In: Proceedings of Computer Games Workshop at the 21st European Conference on Artificial Intelligence (2014)

[23] Cazenave, T.: Sequential Halving Applied to Trees. IEEE Transactions on Computational Intelligence and AI in Games (2014)