Improved Monte-Carlo Search

Levente Kocsis, Csaba Szepesvári, Jan Willemson

MTA SZTAKI, Kende u. 13-17, Budapest, Hungary-1111, {kocsis,szcsaba}@sztaki.hu
University of Tartu, Institute of Computer Science, Liivi str. 2, Tartu, Estonia, jan@ut.ee

Abstract. Monte-Carlo search has been successful in many non-deterministic games, and recently in deterministic games with high branching factor. One of the drawbacks of the current approaches is that even if the iterative process would last for a very long time, the selected move does not necessarily converge to a game-theoretic optimal one. In this paper we introduce a new algorithm, UCT, which extends a bandit algorithm for Monte-Carlo search. It is proven that the probability that the algorithm selects the correct move converges to 1. Moreover it is shown empirically that the algorithm converges rather fast even in comparison with alpha-beta search. Experiments in Amazons and Clobber indicate that the UCT algorithm outperforms considerably a plain Monte-Carlo version, and it is competitive against alpha-beta based game programs.

1 Introduction

Over the years, Monte-Carlo simulation based search algorithms proved to be successful in many non-deterministic and imperfect information games, including backgammon [24], poker [4] and Scrabble [21]. Recently, Monte-Carlo search proved to be competitive in deterministic games with large branching factor, viz. in Go [5]. Monte-Carlo search seems to be one of the few feasible approaches for attacking many search problems underlying RTS games as these problems are often non-deterministic with enormous branching factors [8].

Monte-Carlo search works by iteratively generating sample game episodes (i.e. move sequences going until the end of the game). Subsequently, the starting move leading most frequently to a win is played. The efficiency of Monte-Carlo search heavily depends on the way moves are sampled during the episodes. In current implementations, the moves are sampled using a probability distribution derived from some player model, or a uniform distribution, or a distribution biased by the success of the considered moves elsewhere in the search. One of the drawbacks of the current approaches is that even if the iterative process would run for a long time, the selected move does not necessarily correspond to the game theoretic optimum unlike for most alpha-beta based search algorithms. In this paper we are interested in Monte-Carlo search algorithms with two important properties: (1) small error probability if the algorithm is stopped prematurely, and (2) convergence to the best (in minimax sense) move if enough time is given.

In order to find the best move in the root, one has to determine the best moves in the internal nodes as well (at least along the candidate principal variations).

Since the estimates of the values of the alternative moves rely on the estimates of the values of the (best) successor nodes, we must have small estimation errors for the latter ones. Hence the problem reduces to getting the estimation error decay quickly. In order to achieve this, the algorithm must balance between testing an alternative that looks currently the best (to obtain a precise estimate) and the exploration of other alternatives (to ensure that some good alternative is not missed). This observation serves as the main motivation for the algorithm developed in this paper.

As *multi-armed bandits* represent the archetypical example for exploration-exploitation tradeoffs, we base our algorithm on a particular bandit algorithm. The algorithm chosen, UCB1, due to Auer et al. [2] is known to solve the exploration-exploitation tradeoff in an optimal manner up to a constant factor. The new algorithm, described in Section 2 is called UCT.[1] The convergence of UCT is proven for the infinite memory case, while heuristics based on transposition tables are suggested to overcome memory limitations. The convergence rate is measured empirically for random trees in Section 3. Tournament performance is measured in the game of Amazons and Clobber. Our conclusions are given in Section 4.

2 The UCT algorithm

2.1 The algorithm

UCT is a Monte-Carlo search algorithm with a specific randomized move selection mechanism. The pseudocode of a generic Monte-Carlo search routine is given in Figure 1. The search algorithm iteratively generates game episodes (line 3), and returns the move leading most frequently to a win (line 5).[2] The game episodes are generated by the *search* function that selects and effectuates a move recursively while a terminal[3] node is reached. The value propagation is done in negamax style, which is a natural choice for minimax trees. Adapting it for choice nodes, or cases where MIN/MAX nodes are not strictly alternating is trivial. The result propagated downwards is stored by adjusting the average value for the given node-move pair and by incrementing a counter. This is implemented as part of the transposition table storage mechanism (not shown). For Monte-Carlo versions that do not base their move selection in internal nodes on the result of previous episodes, line 12 is required only in the root node. The effectiveness of the search algorithm depends on the sampling of the moves (line 10). In the plain Monte-Carlo search (referred to in what follows by MC)

[1] UCB stands for Upper Confidence Bound, while UCT stands for 'UCB extended for trees'.

[2] The function *bestMove* is trivial and its code is therefore omitted.

[3] It is possible to stop earlier as well, and to return an evaluation value instead of the game result. For the sake of clarity, we restrict our analysis to stopping at terminal nodes, but we expect that by using evaluation functions the strength of an UCT based game program should improve.

```
1: procedure MonteCarloSearch(position)
2: repeat
3:     search(position, rootply)
4: until Timeout
5: return bestMove(position);

6: function search(position, depth)
7: if GameOver then
8:     return GameResult
9: end if
10: m := selectMove(position, depth);
11: v := −search(position after move m, depth + 1);
12: Add entry (position, m, depth, v, ...) to the TT
13: return v;
```

Fig. 1. Pseudocode of generic Monte-Carlo search

the moves are sampled uniformly. The sampling of the UCT algorithm is based on UCB1, which we describe now.

Consider a bandit problem with K arms, defined by the sequence of random payoffs X_{it}, $i = 1, \ldots, K$, $t \geq 1$, where each i is the index of a gambling machine (the "arm" of a bandit). Successive plays of machine i yield the payoffs X_{i1}, X_{i2}, ... For simplicity, we shall assume that X_{it} lies in the interval $[0, 1]$. An allocation policy is a mapping that selects the next arm to be played based on the sequence of past selections and the payoffs obtained. The expected regret of an allocation policy A after n plays is defined by

$$R_n = \max_i \mathbb{E}\left[\sum_{t=1}^{n} X_{it}\right] - \mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{T_i(n)} X_{i,t}\right],$$

where $T_i(n) = \sum_{s=1}^{n} \mathbb{I}(I_s = i)$ is the number of times arm i was played up to time n, $I_t \in \{1, \ldots, K\}$ is the index of the arm selected at time t. Thus, the regret is the loss due to the fact that the policy does not always play the best machine. It is known that in there is no policy whose regret would grow slower than $O(\ln n)$ for a large class of payoff distributions [15]. A policy is said to resolve the exploration-exploitation tradeoff if its regret growth rate is within a constant factor of the best possible regret rate.

Algorithm UCB1, whose finite-time regret is studied in details in [2] is a simple algorithm that succeeds in resolving the exploration-exploitation tradeoff in this sense. It chooses the arm with the best upper confidence bound:

$$I_t = \operatorname*{argmax}_{i \in \{1,\ldots,K\}} \left\{\overline{X}_{i,T_i(t-1)} + c_{t-1,T_i(t-1)}\right\}, \tag{1}$$

where $c_{t,s}$ is a bias sequence chosen to be

$$c_{t,s} = \sqrt{\frac{2\ln t}{s}}. \tag{2}$$

The bias sequence is such that if X_{it} were i.i.d. (or form a martingale difference process shifted by a constant) then the inequalities

$$\mathbb{P}\left(\overline{X}_{is} \geq \mu_i + c_{t,s}\right) \leq t^{-4}, \tag{3}$$

$$\mathbb{P}\left(\overline{X}_{is} \leq \mu_i - c_{t,s}\right) \leq t^{-4} \tag{4}$$

were satisfied. This follows from Hoeffding's (or more generally, the Hoeffding-Azuma) inequality (see Lemma 8).

Unlike in [2], we allow the mean-value of the payoffs $X_{i\cdot}$ to drift as a function of time. Our main assumption is that the expected values of the averages

$$\overline{X}_{in} = \frac{1}{n}\sum_{t=1}^{n} X_{it} \tag{5}$$

converge. We let $\mu_{in} = \mathbb{E}\left[\overline{X}_{in}\right]$ and

$$\mu_i = \lim_{n\to\infty} \mu_{in}. \tag{6}$$

Further, we define δ_{in} by

$$\mu_{in} = \mu_i + \delta_{in}. \tag{7}$$

Since we allow for a more general payoff process, at this point we just make the assumption that appropriate bias sequences exist. As part of the proof of the main result, we will show that this indeed holds (giving an explicit formula for the bias term).

In the UCT algorithm we model the move selection problem as a separate multi-armed bandit for every (explored) internal node. The arms correspond to the moves and the payoff to the result of the game episode that traverses the node. The sampling function of the UCT algorithm is given in Figure 2. The code given is a canonical version. In specific problems, several enhancements can be used. Since the value converges faster closer to the terminal nodes it is natural to decay the bias sequence with distance from the root (depth). In the experiments $(\ln t/s)^{(D+d)/(2D+d)}$ is used, where D is the estimated game length starting from the node, and d is the depth of the node in the tree.

The provided code suggests that ties are broken in the order of generation. Unless this is intended because of some move-ordering algorithm, random tie-breaking is preferable.

2.2 The UCT algorithm with limited memory

In most search algorithms, transposition tables (TT) are used for storing (information gathered for) the nodes of the search tree. Since the size of the TT is typically smaller than the number of nodes investigated, more than one node is mapped to an entry, and often nodes cannot be stored in the TT or must be deleted from it. In alpha-beta variants, a deleted node may be re-searched when the information is necessary. In UCT, the situation is not that simple. Let us

```
 1: function selectMove(position, depth)
 2: nMoves := # available moves in position
 3: nsum := 0 {nsum will contain # times the descendants of position are considered}
 4: for i := 1 to nMoves do
 5:     Let tte[i] be the TT entry matching the ith descendant of position
 6:     if the entry tte[i] is invalid then
 7:         return random move in position
 8:     end if
 9:     nsum := nsum + tte[i].n {tte[i].n = # times the ith descendant is considered}
10: end for
11: maxv := −∞;
12: for i := 1 to nMoves do
13:     if tte[i].n = 0 then
14:         v := +∞ {Give high preference to an unvisited descendant}
15:     else
16:         v := tte[i].value + √(2 ∗ ln(nsum)/tte[i].n)
17:     end if
18:     if v > maxv then
19:         maxv := v
20:         Let m be the ith move
21:     end if
22: end for
23: return m
```

Fig. 2. Pseudocode of the UCT sampling function. Line 7 is required only for the limited memory version.

consider the tree from Figure 3, left, and assume that we can store only nodes A and B, but not C. If we first search B and store its value then in subsequent searches node C will always be preferred not because it is good, but because it is not stored. If A is a MAX node and C is worse than B, the process will converge to a suboptimal value. There are two solutions for such situations. The first is to switch to random sampling in node A, if one of its child nodes cannot be stored. This modification of the UCT algorithm is indicated in line 7 of Figure 2. Note that the TT access in line 5 results in an invalid entry if the a node cannot be stored and an empty one (with $n = 0$) if the node is not present in the TT, but it can be stored. Alternatively, if not all the moves can be stored and moves with values above some threshold are available, one of those moves can be selected. When there is no such move we fall back to the first solution.

The convergence to the optimal move is ensured by the use of the UCT sampling rule. Thus, it is desirable to be able to use it in nodes where the most information is gathered, and in those which affect most the values close to the root. We found empirically that these requirements are satisfied sufficiently well with a two-level replacement scheme [6], that uses the depth of the node as replacement criterion for the first entry, and the number of times the parent of the node was reached for the second entry. These replacement criteria ensure that the nodes close to the root and the nodes explored often (i.e. having larger impact) are stored. The transposition table that uses the above described replacement scheme will be referred as TwoBigP.

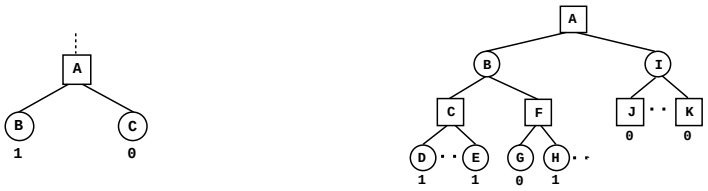

Fig. 3. Sample minimax trees. Squares indicate MAX nodes, and circles indicate MIN nodes. The values below to the nodes represent their (minimax) value.

2.3 Theoretical analysis

We start by analysing UCB1 for non-stationary bandit problems. Remember that by assumption $0 \le X_{it} \le 1$. Quantities related to the optimal arm shall be upper indexed by a star, e.g., μ^*, $T^*(t)$, $\overline{X}_t^*$, etc. For the sake of easy referencing, we summarize the assumptions on the rewards here:

Assumption 1 *Fix $1 \le i \le K$. Let $\{\mathcal{F}_{it}\}_t$ be a filtration such that $\{X_{it}\}_t$ is $\{\mathcal{F}_{it}\}$-adapted and $X_{i,t}$ is conditionally independent of $\mathcal{F}_{i,t+1}, \mathcal{F}_{i,t+2}, \dots$ given $\mathcal{F}_{i,t-1}$. Then $0 \le X_{it} \le 1$ and the limit of $\mu_{in} = \mathbb{E}\left[\overline{X}_{in}\right]$ exists, Further, we assume that there exist a constant $C_p > 0$ and an integer N_p such that for $n \ge N_p$, for any $\delta > 0$, $\Delta_n(\delta) = C_p\sqrt{n \ln(1/\delta)}$, the following bounds hold:*

$$\mathbb{P}\left(n\overline{X}_{in} \ge n\mathbb{E}\left[\overline{X}_{in}\right] + \Delta_n(\delta)\right) \le \delta,$$
$$\mathbb{P}\left(n\overline{X}_{in} \le n\mathbb{E}\left[\overline{X}_{in}\right] - \Delta_n(\delta)\right) \le \delta.$$

Note that under Assumption 1 a suitable choice for $c_{t,s}$ such that (3)–(4) are satisfied (for $t \ge N_p$) is given by

$$c_{t,s} = 2C_p\sqrt{\frac{\ln t}{s}}. \tag{8}$$

In what follows, first Theorem 1 of [2] that bounds the expected number of times when some suboptimal arm is played is generalized (Theorem 2). The next theorem bounds the difference of μ^* and the total payoff received up to some time n. Note that compared to the stationary case we get an additional term due to the drifts of the payoffs. We also give a lower bound on the number of trials of each of the arms that is used to derive an exponential tail inequality for the estimated payoff (Theorem 5). The next theorem, building on the previous results shows that the probability of failure vanishes with time. Based on these results we prove our main result, showing the consistency of the UCT algorithm.

We let $\Delta_i = \mu^* - \mu_i$. Since δ_{it} converges by assumption to zero, for all $\epsilon > 0$ there exists an index $N_0(\epsilon)$ such that if $t \ge N_0(\epsilon)$ then $|\delta_{it}| \le \epsilon\Delta_i/2$ and $|\delta_{j^*,t}| \le \epsilon\Delta_i/2$, whenever i is the index of a suboptimal arm and j^* is the index of an optimal arm. In particular, it follows that for any optimal arm j^*, $t \ge N_0(\epsilon)$, $|\delta_{j^*,t}| \le \epsilon/2 \min_{\{i \,|\, \Delta_i > 0\}} \Delta_i$.

Theorem 2 *Consider UCB1 applied to a non-stationary problem where the payoff sequence satisfies Assumption 1 and where the bias sequence, $c_{t,s}$, used by UCB1 is given by (8). Fix $\epsilon > 0$. Let $T_i(n)$ denote the number of plays of arm i. Then if i is the index of a suboptimal arm then*

$$\mathbb{E}\left[T_i(n)\right] \leq \frac{16C_p^2 \ln n}{(1-\epsilon)^2 \Delta_i^2} + N_0(\epsilon) + N_p + 1 + \frac{\pi^2}{3}.$$

Proof. Fix the index i of a suboptimal arm. We follow the proof of Theorem 1 in [2]. Let

$$A_0(n,\epsilon) = \min\{s \,|\, c_{t,s} \leq (1-\epsilon)\Delta_i/2\}$$

By the definition of $c_{t,s}$, $A_0(n,\epsilon) = \left\lceil \frac{16C_p^2 \ln n}{(1-\epsilon)^2\Delta_i^2} \right\rceil$. We let

$$A(n,\epsilon) = \max(A_0(n,\epsilon), N_0(\epsilon), N_p).$$

By definition,

$$\begin{aligned}
T_i(n) &= 1 + \sum_{t=K+1}^{n} \mathbb{I}(I_t = i) \\
&\leq A(n,\epsilon) + \sum_{t=K+1}^{n} \mathbb{I}(I_t = i, T_i(t-1) \geq A(n,\epsilon)) \\
&\leq A(n,\epsilon) + \sum_{t=1}^{n} \sum_{s=1}^{t-1} \sum_{s'=A(n,\epsilon)}^{t-1} \mathbb{I}(\overline{X}_s^* + c_{ts} \leq \overline{X}_{i,s'} + c_{t,s'}).
\end{aligned}$$

We claim that for $n \geq t \geq s' \geq A(n,\epsilon)$ we have $\mu_t^* \geq \mu_{it} + 2c_{t,s'}$.[4] Indeed, since $n \geq t$ and $c_{t,s}$ increases in t, $c_{t,s'} \leq c_{n,s'}$. Since $c_{t,s}$ decreases in s, and $s' \geq A(n,\epsilon) \geq A_0(n,\epsilon)$, $c_{n,s'} \leq c_{n,A_0(n,\epsilon)}$ and by the definition of A_0, $c_{n,A_0(n,\epsilon)} \leq (1-\epsilon)\Delta_i/2$. Hence, $2c_{t,s'} \leq (1-\epsilon)\Delta_i$. Further, since $t \geq A(n,\epsilon) \geq N_0(\epsilon)$, we have that $\delta_{it} \leq \epsilon\Delta_i$. Hence, $\mu_t^* - \mu_{it} - 2c_{t,s'} = \Delta_i - |\delta_t^*| - \delta_{it} - 2c_{t,s'} \geq \Delta_i - \epsilon\Delta_i - (1-\epsilon)\Delta_i = 0$.

Now, if both $\overline{X}_s^* > \mu_t^* - c_{ts}$ and $\overline{X}_{i,s'} < \mu_{it} + c_{t,s'}$ then using $\mu_t^* \geq \mu_{it} + 2c_{t,s'}$ we get $\overline{X}_s^* + c_{ts} > \overline{X}_{i,s'} + c_{t,s'}$. Hence, $\mathbb{I}(\overline{X}_s^* + c_{ts} \leq \overline{X}_{i,s'} + c_{t,s'}) \leq \mathbb{I}(\overline{X}_s^* + c_{ts} \leq \mu_t^*) + \mathbb{I}(\overline{X}_{i,s'} \geq \mu_{it} + c_{t,s'})$. Plugging this inequality into the bound on $T_i(n)$, we may finish the proof as in [2], taking expectations of both sides, and exploiting (3), (4):

$$\begin{aligned}
\mathbb{E}\left[T_i(n)\right] &\leq A(n,\epsilon) + 1 + \frac{\pi^2}{3} \\
&\leq \left\lceil \frac{16C_p^2 \ln n}{(1-\epsilon)^2\Delta_i^2} \right\rceil + N_0(\epsilon) + N_p + 1 + \frac{\pi^2}{3}.
\end{aligned}$$

[4] For $n < A(n,\epsilon)$, we have $T_i(n) \leq n < A(n,\epsilon)$, so w.l.o.g. we may assume that $n \geq A(n,\epsilon)$.

Theorem 3 *Let*

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

Under the assumptions of Theorem 2,

$$\left|\mathbb{E}\left[\overline{X}_n\right] - \mu^*\right| \leq |\delta_n^*| + O\left(\frac{K(C_p^2 \ln n + N_0)}{n}\right), \tag{9}$$

where $N_0 = N_0(1/2)$.[5]

Proof. Without the loss of generality we assume that there is a unique "best arm". We denote the index of this arm by i^*. By the triangle inequality, $|\mu^* - \mathbb{E}\left[\overline{X}_n\right]| \leq |\overline{\mu}^* - \overline{\mu}_n^*| + |\overline{\mu}_n^* - \mathbb{E}\left[\overline{X}_n\right]| = |\delta_n^*| + |\overline{\mu}_n^* - \mathbb{E}\left[\overline{X}_n\right]|$. We bound the last term as follows:

$$\begin{aligned} n|\overline{\mu}_n^* - \mathbb{E}\left[\overline{X}_n\right]| &= \left|\sum_{t=1}^{n} \mathbb{E}\left[X_t^*\right] - \mathbb{E}\left[\sum_{i=1}^{K} T_i(n)\overline{X}_{i,T_i(n)}\right]\right| \\ &= \left|\sum_{t=1}^{n} \mathbb{E}\left[X_t^*\right] - \mathbb{E}\left[T^*(n)\overline{X}^*_{T^*(n)}\right]\right| + \mathbb{E}\left[\sum_{i=1, i\neq i^*}^{K} T_i(n)\overline{X}_{i,T_i(n)}\right], \end{aligned}$$

where we have exploited that by our assumptions on the payoffs $0 \leq \overline{X}_{i,T_i(n)}$. Since also $\overline{X}_{i,T_i(n)} \leq 1$ holds, the last term can be bounded by the expected total number of times a suboptimal arm was played up to time n. Hence, this term is bounded by $O(K(C_p^2 \log n + N_0))$ by Theorem 2.

In order to bound the first term let us note that $T^*(n)\overline{X}^*_{T^*(n)} = \sum_{t=1}^{T^*(n)} X_t^*$. and that

$$D_n \stackrel{\text{def}}{=} \sum_{t=1}^{n} \mathbb{E}\left[X_t^*\right] - \mathbb{E}\left[\sum_{t=1}^{T^*(n)} X_t^*\right] = \mathbb{E}\left[\sum_{t=1}^{n} X_t^* - \sum_{t=1}^{T^*(n)} X_t^*\right] = \mathbb{E}\left[\sum_{t=T^*(n)+1}^{n} X_t^*\right].$$

Hence, $D_n \geq 0$. Further, using $X_t^* \leq 1$ we may bound the last term from above by $\mathbb{E}\left[n - T^*(n)\right]$, which is just $\sum_{i\neq i^*} \mathbb{E}\left[T_i(n)\right]$ and hence by Theorem 2, $D_n = O(K(C_p^2 \log n + N_0))$. Collecting the terms yields the bound in (9).

The following theorem provides a lower-bound on the number of times an arm is pulled.

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

Proof. The proof is elementary and is hence omitted.

[5] The choice of $\epsilon = 1/2$ is admittedly arbitrary. We attempt no optimization of the bounds.

Theorem 3 bounded the expected estimation error. As shown by the next result, the estimated optimal payoff concentrates quickly around its mean:

Theorem 5 *Fix an arbitrary $\delta > 0$ and let $\Delta_n = 9\sqrt{2n\ln(2/\delta)}$. Let n_0 be such that*

$$\sqrt{n_0} \geq O(K(C_p^2 \ln n_0 + N_0(1/2))).$$

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

$$\mathbb{P}\left(n\overline{X}_n \geq n\mathbb{E}\left[\overline{X}_n\right] + \Delta_n\right) \leq \delta,$$
$$\mathbb{P}\left(n\overline{X}_n \leq n\mathbb{E}\left[\overline{X}_n\right] - \Delta_n\right) \leq \delta.$$

Proof. Our main tool to develop the bounds will be Lemma 14. For the sake of simplicity, we assume that the payoffs of the optimal arm are i.i.d. The general case can be treated similarly, as indicated in the proof of Lemma 14. Let Z_t be the indicator of the event that a suboptimal arm is chosen at time step t. Then by Theorem 2, $\mathbb{E}\left[\sum_{t=1}^{n} Z_t\right] \leq O(K\ln(n))$. Hence, a_t can be chosen to be $O(K(C_p^2\ln(t)+N_0(1/2)))$. Further, X_t of Lemma 14 is identified with the payoff sequence of the best arm. We let Y_t denote the payoff received at time step t. By assumption, X_t, Y_t lie in the $[0,1]$ interval and $n\overline{X}_n = \sum_{t=1}^{n}(1-Z_t)X_t + Z_tY_t$. Note that R_n, as defined in the lemma corresponds to the expected total regret at time n. By Theorem 2, $R_n = O(K(C_p^2\ln n + N_0(1/2)))$. Let n_0 be an index such that if $n \geq n_0$ then $a_n \leq \Delta_n/9$ and $R_n \leq 2\Delta_n/9$. Such an index exists since $\Delta_n = O(\sqrt{n})$ and $a_n, R_n = O(\ln n)$. Hence, for $n \geq n_0$, the conditions of Lemma 14 are satisfied and the desired tail-inequalities hold for $\overline{X}_n$. Since for $\delta \leq 1$, $\Delta_n = 9\sqrt{2n\ln(2/\delta)} \geq 9\sqrt{2n\ln(2)}$, it follows that n_0 can be selected independently of δ. In fact, for a suitable choice of a constant c, n_0 is the first integer such that $\sqrt{n} \geq c(K(C_p^2\ln n + N_0(1/2)))$ is suitable for n_0. This finishes the proof of the theorem.

Finally, we are in the position to prove an upper bound on the failure of the algorithm after time t:

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

$$\lim_{t\to\infty} P(I_t \neq i^*) = 0.$$

Note that the previous results imply only that $I_t \neq i^*$ happens with decreasing frequency, but they do not imply that the probability of suboptimal choices would converge to zero. (Indeed, one may imagine an algorithm where the probability of suboptimal choice is 1 for episodes of index $\{2^k\}_k$, whilst the algorithm select suboptimal choices with decreasing frequency.)

Proof. Fix $\epsilon > 0$. We would like to show that if t is sufficiently large than $P(I_t \neq i^*) \leq \epsilon$.

Let i be the index of a suboptimal arm and let $p_{it} = \mathbb{P}\left(\overline{X}_{i,T_i(t)} \geq \overline{X}^*_{T^*(t)}\right)$ from above. Clearly, $P(I_t \neq i^*) \leq \sum_{i \neq i^*} p_{it}$. Hence, it suffices to show that $p_{it} \leq \epsilon/K$ holds for all suboptimal arms for t sufficiently large.

Clearly, if $\overline{X}_{i,T_i(t)} < \mu_i + \Delta_i/2$ and $\overline{X}^*_{T^*(t)} > \mu^* - \Delta_i/2$ then $\overline{X}_{i,T_i(t)} < \overline{X}^*_{T^*(t)}$. Hence,

$$p_t \leq \mathbb{P}\left(\overline{X}_{i,T_i(t)} \geq \mu_i + \Delta_i/2\right) + \mathbb{P}\left(\overline{X}^*_{T^*(t)} \leq \mu^* - \Delta_i/2\right).$$

The first probability can be expected to be converging much slower since $T_i(t)$ converges slowly. Hence, we bound it first.

In fact,

$$\mathbb{P}\left(\overline{X}_{i,T_i(t)} \geq \mu_i + \Delta_i/2\right) \leq \mathbb{P}\left(\overline{X}_{i,T_i(t)} \geq \overline{\mu}_{i,T_i(t)} - |\delta_{i,T_i(t)}| + \Delta_i/2\right).$$

Without the loss of generality, we may assume that $|\delta_{i,t}|$ is monotone decreasing. Hence, $|\delta_{i,T_i(t)}| \leq |\delta_{i,\lfloor \rho \log t \rfloor}|$ by Theorem 4.

Whenever $\lfloor \rho \log t \rfloor > N_0(\Delta_i/4)$ then $|\delta_{i,T_i(t)}| \leq \Delta_i/4$. Therefore

$$\mathbb{P}\left(\overline{X}_{i,T_i(t)} \geq \mu_i + \Delta_i/2\right) \leq \mathbb{P}\left(\overline{X}_{i,T_i(t)} \geq \overline{\mu}_{i,T_i(t)} + \Delta_i/4\right).$$

Now, let a be an index such that if $t \geq a$ then $(t+1)\mathbb{P}\left(\overline{X}_{i,t} \geq \overline{\mu}_{i,t} + \Delta_i/4\right) < \epsilon/(2K)$. Such an index exist by our assumptions on the concentration properties of the average payoffs. Then, for $t \geq a$

$$\begin{aligned}\mathbb{P}\left(\overline{X}_{i,T_i(t)} \geq \overline{\mu}_{i,T_i(t)} + \Delta_i/4\right) &\leq \mathbb{P}\left(\overline{X}_{i,T_i(t)} \geq \overline{\mu}_{i,T_i(t)} + \Delta_i/4, T_i(t) \geq a\right) \\ &\quad + \mathbb{P}\left(T_i(t) \leq a\right).\end{aligned}$$

Since the lower-bound on $T_i(t)$ grows to infinity as $t \to \infty$, the second term becomes zero when t is sufficiently large. The first term is bounded using the method of Lemma 10. By choosing $b = 2a$ we get

$$\begin{aligned}&\mathbb{P}\left(\overline{X}_{i,T_i(t)} \geq \overline{\mu}_{i,T_i(t)} + \Delta_i/4, T_i(t) \geq a\right) \\ &\qquad \leq (a+1)\mathbb{P}\left(\overline{X}_{i,a} \geq \overline{\mu}_{i,a} + \Delta_i/4\right) + \mathbb{P}\left(T_i(t) \geq 2b\right) \\ &\qquad \leq \epsilon/(2K),\end{aligned}$$

where we have assumed that t is large enough so that $\mathbb{P}\left(T_i(t) \geq 2b\right) = 0$.

Bounding $\mathbb{P}\left(\overline{X}^*_{T^*(t)} \leq \mu^* - \Delta_i/2\right)$ by $\epsilon/(2K)$ can be done in an analogous manner. Collecting the bounds yields that $p_{it} \leq \epsilon/K$ for t sufficiently large.

Unfortunately, our methods are too crude to derive a meaningful convergence rate result on the failure probability. This is because we don't have sufficiently strong bounds for the concentration of $T_i(t)$ for suboptimal arms. However, we conjecture that the convergence rate is $(\log(t)/t)^\kappa$ for $0 < \kappa \leq 1$.

Now follows our main result:

Theorem 7 *Consider algorithm UCT running on a game tree of depth D, branching factor K with stochastic payoffs 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 $O((KD\log(n)+K^D)/n)$. Further, the failure probability at the root converges to zero as the number of samples grows to infinity.*

Proof. The proof is done by induction on D. Consider first the case $D = 1$ (in this case, actually, UCT just corresponds to UCB1). Our assumptions on the payoffs hold, thanks to Hoeffding's inequality. Now the result on the bias follows directly from Theorem 3 and consistency follows from Theorem 6.

Now, assume that the result holds for all trees of up to depth $D-1$ and consider a tree of depth D. Let us only concentrate on the root node. We claim that from the point of the root node, running UCT is equivalent to running UCB1 with non-stationary, correlated payoffs for the various moves (arms). Fix a move i. In fact, the payoff for move i of the root at time t will depend on all previous "entries" into the subtree originating at the successor node of move i. For simplicity we shall denote this node by i, as well. We claim that the payoff process experienced at node i will satisfy the conditions required by Theorems 2–6. First, the payoffs lie in the interval $[0,1]$. Now, since the tree starting at node i has depth $D-1$, by the induction hypothesis we may apply Theorem 3 to show that the expected average payoff converges. That the conditions on the exponential concentration of the payoffs are satisfied follows from Theorem 5. Since this holds for any i, it follows by Theorem 3 that the bias *at the root* converges at the rate of

$$|\delta_n^*| + O(K(\ln n + N_0)/n),$$

where δ_n^* is the rate of convergence of the bias for the best move and

$$N_0 = \min\{\, n \,|\, |\delta_{in}| \le \frac{1}{2}\Delta_i,\ i \neq i^* \,\}.$$

Now, by the induction hypothesis,

$$|\delta_{in}| = O((K(D-1)\log(n) + K^{D-1})/n), \quad i = 1, \ldots, K.$$

Hence, $N_0 = O(K^{D-1})$, yielding the desired result for the bias at the root. The proof is finished by noting that the failure probability converges to zero thanks to Theorem 6.

Note that it follows from the proof that when the payoffs are deterministic then the bias terms prescribe too much exploration at the nodes immediately preceding the leaves. Here, no exploration would be needed at all. This can be achieved gradually by making the bias more uniform. From this, one might conjecture that more uniform bias terms are desirable in the vicinity of the leafs. Indeed, it is reasonable to use stronger exploration bonus close to the root: at the beginning of runs the large unexplored parts of the tree can be expected to behave "randomly".

In fact, for deterministic problems, convergence can be shown for a larger class of bias terms: the role of the bias term can be viewed as taking care of the shifts in the payoff in the subtrees as time goes by. However, we do not pursue this direction further in this paper.

2.4 Related research

Besides the research already mentioned on Monte-Carlo search in games, we believe that the closest to our work is the work of Peret and Garcia [18], who considered single-agent stochastic search in the context of Markovian Decision Problems. They extended the sparse sampling procedure due to Kearn's et al. [13]. The algorithm of Kearns et al. builds a fixed-depth lookahead tree by randomly sampling a fixed number of successor nodes at each stage. Peret and Garcia proposed to guide this tree building process by sampling actions selectively. They compared three strategies: uniform sampling (uncontrolled search), Boltzmann-exploration based search (the values of actions are transformed into a probability distribution, i.e., samples better looking actions are sampled more often) and a heuristic, interval-estimation based approach. They observed that in their domain ('sailing') lookahead pathologies are present when the search is uncontrolled. Experimentally, both the interval-estimation and the Boltzmann-exploration based strategies were shown to avoid the lookahead pathology and to improve upon the basic procedure by a large margin. We note that Boltzmann-exploration is another widely used bandit strategy, known under the name of "exponentially weighted average forecaster" in the on-line prediction literature (e.g. [3]). Boltzmann exploration as a bandit strategy is inferior to UCB in stochastic environments (its regret grows with the square root of the number of samples), but is preferable in adversary environments where UCB does not have regret guarantees. We have also experimented with the Boltzmann-exploration based strategy and found that in the case of our domains it performs significantly worse than the upper-confidence value based algorithm described here. Another related recent work is due to Wang et al. who also considered single-agent problems and looked at Bayesian procedures [25]. Both Peret and Garcia and Wang et al. consider only the case when the full tree fits into the memory.

Chang et al. [7], on the other hand, considered the other limit: they sample the tree in a depth-first, recursive manner: At each node they simulate (recursively) a sufficient number of samples to compute a good approximation of the value of the node. The subroutine returns with an approximate evaluation of the value of the node, but the returned values are not stored (so when a node is revisited, no information is present about which actions can be expected to perform better). Similar to our proposal, they suggest to use the average values and sampling is controlled by upper-confidence bounds. They prove similar results to our case, though, due to the independence of samples the analysis of their algorithm is much easier. They also experimented with propagating the maximum of the values of the children and a number of other combinations. Some of these combinations outperformed propagating the maximum value, which itself was found to be superior to propagating the average values.

We have also considered if it would be more beneficial to approximate the values of intermediate nodes by the maximum or minimum value of the child nodes (depending whether the node is MAX or MIN node). We have found that this choice has two main drawbacks. First, the value of the child might not be stored (due to the lack of memory). In this case, the samples generated through the child are lost (which is not the case for averaging, when it samples of a child do influence the values of its parent). Second, the minimax update may produce undesired sudden changes in the value of a node. Let us consider the example in Figure 3, right. Since B is a more promising alternative, most of the samples were generated in the subtree rooted in B. Now, a path through G is sampled, with 0 being its terminal value. Assuming that H was not sampled before, in the case of minimax update the value of B suddenly changes from 1 to 0. Following this, due to the bias term of the UCT sampling, B will not be sampled, until the same number of samples are generated for the alternatives in A as were sampled beforehand for B.

Note: That this is a problem follows only under a hypothesis of correlated payoffs. Discuss this!

If the average values were propagated, this would not happen, since the value of B would decrease by a smaller amount, and the next sample would (probably) lead to H (F having a lower value would be preferred by MIN, and H is unexplored, thus it is preferred due to the bias term). We note that the second drawback is present only when sampling performed using UCT, and not with uniform sampling. Compared to MC, the minimax update with uniform sampling (denoted in the following by MMMC) has even the advantage of converging to the minimax optimal if enough time end memory is available. However, we shall see in Section 3.1 that MMMC may be way slower converging even in the lack of memory limitations.

Monte-Carlo search of game trees was considered by theoretical computer scientist. An early result of this work is that for binary AND/OR trees a simple randomisation is capable of beating deterministic algorithms improving their expected running time (for *any* given AND/OR tree) to be sublinear in the number of leaves of the tree [23]. Lately, sharp tail probability bounds were derived for the distribution of the number of leaves read by the algorithm [14].

3 Experiments

3.1 Experiments with random trees

P-game tree [22] is a minimax tree where a randomly chosen value is assigned to each move. The value of a leaf node is given by the sum of the move values along the path. If the sum is positive, the result is a win for MAX, if negative it is a win for MIN, and it is draw if the sum is 0. In the experiments, for the moves of MAX the value was chosen uniformly from the interval $[0, 127]$ and for MIN from the interval $[-127, 0]$.[6] P-game trees are considered a good approximation for games that associate certain (hypothetical) values to moves such as Go or trick-scoring card games. We have performed two sets of experiments: (1) experiments

[6] This is slightly different from [22], where 1 and -1 was used only.

for measuring the convergence rate of UCT with unlimited memory, and (2) experiments for measuring the influence of memory limitation.

Convergence with unlimited memory. First, we compared the performance of four search algorithms: alpha-beta (AB), plain Monte-Carlo search (MC), Monte-Carlo search with minimax value update (MMMC), and the UCT algorithm. The failure rate of the four algorithms is plotted as function of iterations in Figure 4. Figure 4, left corresponds to trees with branching factor (B) two and depth (D) twenty, and Figure 4, right to trees with branching factor eight and depth eight. The failure rate represents the frequency of choosing the incorrect move if stopped after a number of iterations. For alpha-beta it is assumed that it would pick a move randomly, if the search has not been completed within a number of leaf nodes.[7] Each data point is obtained by averaging the results obtained over 200 random trees and 200 runs for each of the trees. We observe that for both tree types UCT is converging to the correct move (i.e. zero failure rate) within a similar number of leaf nodes as alpha-beta does. Moreover, if we accept a small failure rate, UCT may even be faster. As expected, MC is converging to failure rate levels that are significant, and it is outperformed by UCT uniformly for all the iteration numbers. We remark that the failure rate for MMCS is higher than that of MC, although MMMC would eventually converge to the correct move if it were given sufficient time to run.

Second, we measured the convergence rate of UCT as a function of search depth and branching factor. The required number of iterations to obtain failure rate smaller than some fixed value is plotted in Figure 5. We observe that for P-game trees, UCT is converging to the correct move in $O(B^{D/2})$ (the curve is roughly parallel to $B^{D/2}$ on log-log scale), similarly to alpha-beta. For higher failure rates, UCT seems to converge faster than $O(B^{D/2})$.

Effects of limited memory. In these experiments UCT was tested with TwoBigP transposition tables of various sizes. The hash-key of a node was defined as the index of the node in the preorder traversing of the entire tree. Four different sizes were used for the transposition table: 100,000, 10,000, 5,000 and 1,000 entries. The failure rate of UCT with limited memory is plotted as a function of the number of iterations in Figure 6. We observe that the convergence rate is barely influenced in the case of reasonably large transposition tables. Note that even for the biggest transposition tables, the number of entries significantly smaller than the number of nodes searched. It only happens for very small transposition table sizes when UCT's failure rate fails to converge to zero, though even in this case the failure rate is quite small (less than 5 percent).

[7] The algorithms are compared in terms of the number of leaf nodes evaluated. The motivation of this is that in most 'knowledgeable' programs the total computational cost is dominated by the cost of processing leaves. Further, the total number of nodes expanded for the Monte-Carlo variants is D times the number of leaf nodes. Hence, the impact of not counting internal nodes is thought to be insignificant, at least what matters the qualitative convergence behaviour of the rate of convergence.

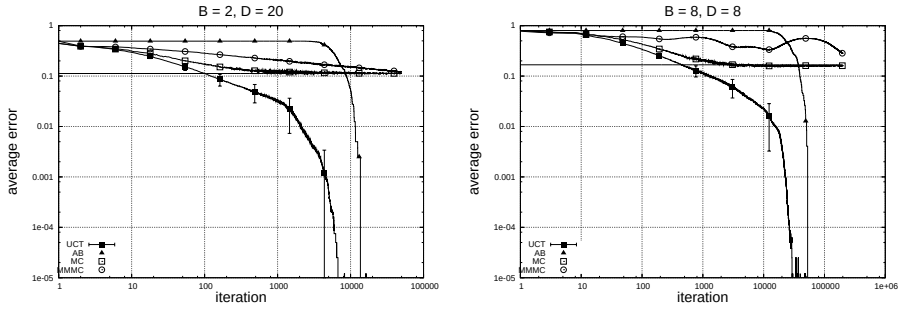

Fig. 4. Failure rate in P-games. The 95% confidence intervals are also shown for UCT.

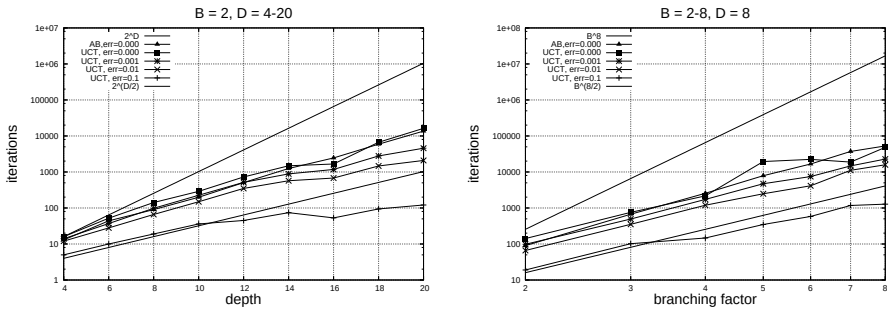

Fig. 5. Convergence in P-games.

Therefore, we conclude that the convergence properties of UCT practically carry over to the limited memory case when UCT is used in conjunction with transposition tables used in today's game programs. We consider the investigation of the behaviour of UCT with limited memory as an important open theoretical issue.

3.2 Amazons

The game of Amazons is played by two sides on a 10×10 board. Each side has four amazons. A move has two parts: first an amazon is moved in the same way as the queens in chess, and second the amazon shoots an arrow. The arrow travels the same way as the amazons. An obstacle is placed on the square where the arrow landed. Neither amazons nor arrows can move over or on obstacles or other amazons. The last player who makes a move wins.

Since the game is convergent and the terminal condition is easy to test, Monte-Carlo search methods can be applied easily. Due to the high branching factor, Amazons is an attractive domain for selective search algorithms in general, and Monte-Carlo search in particular. Indeed, several top programs are employing selective search (see e.g. [9, 17]). High branching and long games are making it difficult for UCT to converge to the game theoretic value. Therefore,

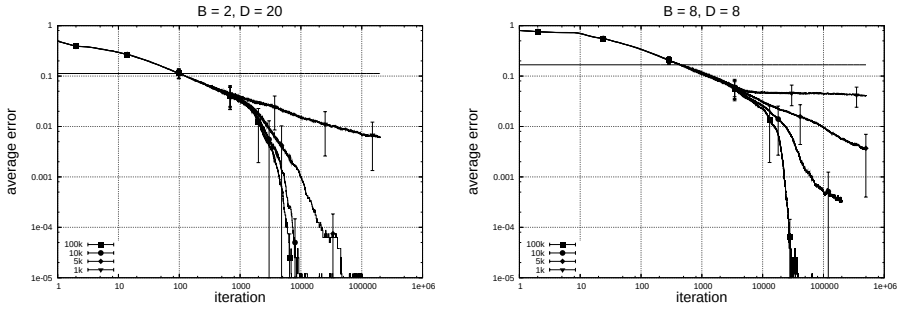

Fig. 6. Failure rate in P-games for UCT with limited memory

	0.01s	0.1s	1s	5s
Result for UCT	$392 = 39,2\%$	$489 = 48,9\%$	$810 = 81\%$	$960 = 96\%$

Table 1. UCT vs. MC in Amazons. Game results with various time limits per move.

we test whether the sampling of UCT is paying off for the extra computation needed.

We have implemented the UCT and the MC search algorithms for Amazons. For UCT a transposition table with 2^{20} entries is used, similar to the one described in Section 3.1. In the start position, UCT is parsing $2 \cdot 10^6$ nodes per second (or 15,000 iterations per second), whilst MC processes $3.6 \cdot 10^6$ nodes per second (or 27,000 iterations per second). We have tested the two algorithms in 1000 game matches with various time limits per move. The results are summarised in Table 1. We observe that for very short time limits MC can exploit its speed advantage. For regular time limits (that are closer to tournament conditions) the loss in speed is quickly offset, and the performance of UCT against MC converges to values close to 100 percent.

3.3 Clobber

The game of Clobber [1] is played by two sides on a chess board. A move consists of moving a piece to a neighbouring square, provided there was a piece of opposing colour, which then gets removed from the board. The players move alternately and the last player who makes a move wins. In the initial position all white squares are occupied by white pieces and all black squares by black pieces. Clobber is a convergent game and the terminal condition is easy to test.

We implemented UCT and MC search algorithms for Clobber and as a benchmark we used MILA, the winner of the 2005 Computer Olympiad [26].[8] Being more optimised, MILA is searching 150,000 nodes per second compared to UCT's

[8] At the Olympiad, MILA, based on the award-wining MIA engine [27], won by 8-0 against ClobberA, a program with a pure MC engine and an endgame solver.

	UCT 5s vs MILA 5s	UCT 30s vs MILA 30s	UCT 30s vs MILA 5s
Result for UCT	$43 = 21,5\%$	$53 = 26,5\%$	$89 = 44,5\%$

Table 2. MILA vs. UCT in Clobber. Game results with various time limits per move.

80,000 on AMD Athlon 64 3GHz computer. The programs were tested in 200-game matches. With 30 seconds per move limitation for both sides, MC (without endgame knowledge) was able to win 17,5% of the games against MILA and 15% of the games against UCT. Results for the games UCT vs. MILA can be found in Table 2. We can see that when MC algorithms are given 30 seconds, UCT performs significantly better than MC (both in direct match and when compared against MILA). We also note that increasing search time increases relative performance against MILA. However, even with significantly more time given, UCT still stays behind MILA. The position estimations given by UCT did converge to the correct minimax values towards the endgame, but mostly MILA was able to see the win earlier. Thus we conclude that while being definitely superior to MC, the UCT approach needs further development.

4 Conclusions

In this article we introduced a new Monte-Carlo search algorithm, called UCT, which extends the bandit algorithm UCB1 for game-tree search. For the unlimited memory case, we proved that the UCT algorithm is consistent: The probability of selecting the optimal move converges to 1. A new transposition-table replacement scheme, TwoBigP is suggested for the limited memory case. The performance of UCT was tested experimentally in random (P-game) trees, and in two games, namely Amazons and Clobber. The P-game experiments have shown that UCT does converge rather fast to the minimax optimal move. The convergence rates are of order $B^{D/2}$, same as for alpha-beta for the trees investigated. Moreover, we observed that the convergence is not impaired significantly when transposition-table with realistic sizes are used. Both the Amazons and the Clobber experiments indicate that UCT outperforms considerably the plain Monte-Carlo search. In Clobber, UCT scored 26,5% against the current top program, which is a rather promising result. We expect that by adding knowledge to UCT (e.g. in the form of evaluation functions), eventually UCT algorithm will become competitive with current tournament programs.

A Technical Details

Let $\mathcal{F}_t$ denote a filtration over some probability space, Y_t be an $\mathcal{F}_t$-adapted real valued martingale-difference sequence. Define the partial sum martingale $S_n = \sum_{t=1}^{n} Y_t$, $n \geq 1$. We shall need the Hoeffding-Azuma inequality:

Lemma 8 (Hoeffding-Azuma inequality) *If Y_n is a martingale difference with $|Y_i| \leq C$, a.s., $i = 1, 2, \ldots,$, where C is a positive real number, then*

$$\mathbb{P}\left(S_n \geq \epsilon n\right) \leq \exp\left(-\frac{2n\epsilon^2}{C^2}\right).$$

Similarly,

$$\mathbb{P}\left(S_n \leq -\epsilon n\right) \leq \exp\left(-\frac{2n\epsilon^2}{C^2}\right).$$

We shall need tail inequalities for stopped martingales. Interestingly, except one recent paper due to Limnios and Ting-Lee [16] we have not been able to found any tail inequality results for stopped martingales. Unfortunately, the proof of [16] is incorrect.[9] Hence, we give some very simple bounds here.

We start with a simple observation:

Lemma 9 *Let N be an integer-valued random variable and let S_t be an $\mathcal{F}_t$-adapted real-valued process (not necessarily a martingale) ($t = 0, 1, 2, \ldots$), which is centered: $\mathbb{E}\left[S_t\right] = 0$. Pick any integers $0 \leq a < b$ and $\epsilon > 0$. Then*

$$\mathbb{P}\left(S_N \geq \epsilon N\right) \leq (b - a + 1) \max_{a \leq t \leq b} \mathbb{P}\left(S_t \geq \epsilon t\right) + \mathbb{P}\left(N \notin [a, b]\right), \tag{10}$$

$$\mathbb{P}\left(S_N \leq -\epsilon N\right) \leq (b - a + 1) \max_{a \leq t \leq b} \mathbb{P}\left(S_t \leq -\epsilon t\right) + \mathbb{P}\left(N \notin [a, b]\right), \tag{11}$$

Proof. We have the following inequalities:

$$\begin{aligned}\mathbb{P}\left(S_N \geq \epsilon N\right) &\leq \mathbb{P}\left(S_N \geq \epsilon N, a \leq N \leq b\right) + \mathbb{P}\left(N \notin [a, b]\right) \\ &= \mathbb{P}\left(S_N \geq \epsilon N | a \leq N \leq b\right) \mathbb{P}\left(a \leq N \leq b\right) + \mathbb{P}\left(N \notin [a, b]\right).\end{aligned}$$

Let us denote the first term on the right hand side of the last line by p. Since

$$\begin{aligned}\mathbb{P}\left(S_N \geq \epsilon N | a \leq N \leq b\right) &= \mathbb{E}\left[\mathbb{I}(S_N \geq \epsilon N) | a \leq N \leq b\right] \\ &\leq \mathbb{E}\left[\sum_{i=a}^{b} \mathbb{I}(S_i \geq \epsilon i) | a \leq N \leq b\right] \\ &= \sum_{i=a}^{b} \mathbb{P}\left(S_i \geq \epsilon i | a \leq N \leq b\right).\end{aligned}$$

Hence, we can bound p by $\sum_{i=a}^{b} \mathbb{P}\left(S_i \geq \epsilon i\right)$. Bounding the sum by the maxima of its terms and multiplied by the number of terms, we get the desired inequality.

The lower-tail inequality can be obtained in an entirely analogous manner.

The next result is a simple corollary of this lemma and the Hoeffding-Azuma inequality. Let N be an integer-valued random variable. The following lemma generalizes the H-A inequality for S_N.

[9] The error in the proof becomes obvious e.g. for a stopping time like $N(t) = \max\{0 \leq i \leq t | S_i > 0\}$.

Lemma 10 (Hoeffding-Azuma inequality for Stopped Martingales) *Assume that S_t is a centered martingale such that the corresponding martingale difference process is uniformly bounded by C. Then, for any fixed $\epsilon > 0$, integers $0 \leq a < b$, the following inequalities hold:*

$$\mathbb{P}\left(S_N > \epsilon N\right) \leq (b-a+1)\exp(-2a^2\epsilon^2/C^2) + \mathbb{P}\left(N \notin [a,b]\right), \tag{12}$$

$$\mathbb{P}\left(S_N < -\epsilon N\right) \leq (b-a+1)\exp(-2a^2\epsilon^2/C^2) + \mathbb{P}\left(N \notin [a,b]\right). \tag{13}$$

Proof. The result follows by combining Lemma 9 and Lemma 8.

A very essential part of the proof is to bound the deviation of counting processes from their mean. For this purpose we use the bounded differences method and Doob's martingales.

We start by citing a well-known result, often used in the analysis of randomized algorithms (see e.g. [20, 19]). We need the following definition: Let f be a function of n variables. We say that f is C-Lipschitz if $|f(z_1,\ldots,z_n) - f(z_1,\ldots,z_{i-1},z_i',z_i,\ldots,z_n)| \leq C$ holds for any $z_1,\ldots,z_n,z_i'$ in $\mathrm{Dom}(f)$.

Lemma 11 *Let (Z_i), $i = 1,\ldots,n$ be a sequence of random variables such that Z_i is conditionally independent of $Z_{i+1},\ldots,Z_n$ given $Z_1,\ldots,Z_{i-1}$. Then the Doob martingale $X_i = \mathbb{E}\left[f(Z_1,\ldots,Z_n)|Z_1,\ldots,Z_i\right]$ has bounded differences, in particular*

$$|X_{i+1} - X_i| \leq 2C.$$

Proof. The proof is omitted as this is a well known result.

Now let $N = \sum_{i=1}^n Z_i$, where Z_i are $0-1$-valued random variables. We assume that Z_i is adapted to the filtration $\{\mathcal{F}_i\}_t$ and that Z_{i+1} is conditionally independent of $Z_{i+2}, Z_{i+3},\ldots,Z_n$ given $\mathcal{F}_i$. Our aim now is to obtain upper and lower tail bounds for the counting process N.

Lemma 12 *We have*

$$\mathbb{P}\left(N - \mathbb{E}\left[N\right] > u\right) \leq \exp(-u^2/(2n)).$$

Similarly,

$$\mathbb{P}\left(N - \mathbb{E}\left[N\right] < -u\right) \leq \exp(-u^2/(2n)).$$

Proof. Note that the function $f(z_1,\ldots,z_n) = z_1 + \ldots + z_n$ is 1-Lipschitz. Hence, the Doob martingale $X_i = \mathbb{E}\left[N|Z_1,\ldots,Z_i\right]$ is a bounded difference martingale with bound 2. The Hoeffding-Azuma inequality applied to the centered martingale $X_i - \mathbb{E}\left[N\right]$ yields the desired result.

The next result gives an upper tail bound on N when $\mathbb{E}\left[\sum_{i=1}^n Z_i\right]$ is slowly growing:

Lemma 13 *Let Z_i be as in Lemma 12, $N_n = \sum_{i=1}^n Z_i$. Assume that a_n is an upper bound on $\mathbb{E}\left[N_n\right]$. Then for all $\Delta > 0$, if n is such that $a_n \leq \Delta/2$ then*

$$\mathbb{P}\left(N_n \geq \Delta\right) \leq \exp(-\Delta^2/(8n)).$$

Proof. We have

$$\mathbb{P}\left(N_n \geq \Delta\right) = \mathbb{P}\left(N_n > \mathbb{E}\left[N_n\right] + \Delta - \mathbb{E}\left[N_n\right]\right) \leq \mathbb{P}\left(N_n > \mathbb{E}\left[N_n\right] + \Delta/2\right)$$

since by assumption $\mathbb{E}\left[N_n\right] \leq a_n \leq \Delta/2$. Using Lemma 12 we obtain the required bound.

For sums of (not necessarily identically distributed) independent Bernoulli variables Janson derived significantly his bounds then the one just derived (see e.g. [10–12]). In particular, if the sum is slowly growing then the above bounds are very weak. Janson's inequalities depend on the variance of the counting process N_n, which can be expected to grow slowly if the sum is growing slowly. With some strong restrictions, these results are known to apply to dependent variables, too. Unfortunately, bounding the variance of N_n is not trivial and the conditions of the variables being "positively or negatively related" do not hold in our applications. Hence, we use the above simpler, but cruder bounds. For corresponding lower bounds, we will exploit that for our special counting processes deterministic lower bounds can be given.

The following technical lemma is at the core of our results for propagating confidence bounds "upward in the tree":

Lemma 14 *Let $Z_i, \mathcal{F}_i, a_i$ be as in Lemma 13. Let $\{X_i\}$ be an i.i.d. sequence with mean μ, and $\{Y_i\}$ and $\mathcal{F}_i$-adapted process. We assume that both X_i and Y_i lie in the $[0,1]$ interval. Consider the partial sums*

$$S_n = \sum_{i=1}^{n} (1 - Z_i) X_i + Z_i Y_i.$$

Fix an arbitrary $\delta > 0$, let $\Delta = 9\sqrt{2n \ln(2/\delta)}$ and let

$$R_n = \mathbb{E}\left[\sum_i X_i\right] - \mathbb{E}\left[S_n\right].$$

Then for n such that $a_n \leq (1/9)\Delta$ and $|R_n| \leq (4/9)\Delta/2$,

$$\mathbb{P}\left(S_n \geq \mathbb{E}\left[S_n\right] + \Delta\right) \leq \delta \tag{14}$$

and

$$\mathbb{P}\left(S_n \leq \mathbb{E}\left[S_n\right] - \Delta\right) \leq \delta. \tag{15}$$

Proof. Let $p = \mathbb{P}\left(S_n \geq \mathbb{E}\left[S_n\right] + \Delta\right)$. We have $S_n = \sum_{i=1}^{n} X_i + \sum_{i=1}^{n} Z_i(Y_i - X_i) \leq \sum_{i=1}^{n} X_i + 2\sum_{i=1}^{n} Z_i$. Therefore,

$$p \leq \mathbb{P}\left(\sum_{i=1}^{n} X_i + 2\sum_{i=1}^{n} Z_i \geq \mathbb{E}\left[\sum_{i=1}^{n} X_i\right] - R_n + \Delta\right).$$

Using the elementary inequality $\mathbb{I}(A+B \geq \Delta) \leq \mathbb{I}(A \geq \alpha\Delta) + \mathbb{I}(B \geq (1-\alpha)\Delta)$ that holds for any $A, B \geq 0$, $0 \leq \alpha \leq 1$, we get

$$p \leq \mathbb{P}\left(\sum_{i=1}^{n} X_i \geq \mathbb{E}\left[\sum_{i=1}^{n} X_i\right] + \Delta/9\right) + \mathbb{P}\left(2\sum_{i=1}^{n} Z_i \geq 8/9\Delta - R_n\right).$$

Using the Hoeffding-Azuma inequality, the first term can be bounded by $\delta/2$.[10] Since by assumption $|R_n| \leq 4/9\Delta$, the second term can be upper bounded by

$$\mathbb{P}\left(2\sum_{i=1}^{n} Z_i \geq 4/9\Delta\right) = \mathbb{P}\left(\sum_{i=1}^{n} Z_i \geq 2/9\Delta\right).$$

Finally, by Lemma 13 and thanks to our assumptions on a_n, this term is also bounded by $\delta/2$, thus, finishing the proof of the first part. The second part can be proved in an analogous manner.

References

1. M.H. Albert, J.P. Grossman, R.J. Nowakowski, and D. Wolfe. An introduction to Clobber. *INTEGERS: The Electronic Journal of Combinatorial Number Theory*, 5(2), 2005.
2. P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.
3. P. Auer, N. Cesa-Bianchi, Y. Freund, and R.E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32:48–77, 2002.
4. D. Billings, A. Davidson, J. Schaeffer, and D. Szafron. The challenge of poker. *Artificial Intelligence*, 134:201–240, 2002.
5. B. Bouzy and B. Helmstetter. Monte Carlo Go developments. In H.J. van den Herik, H. Iida, and E.A. Heinz, editors, *Advances in Computer Games 10*, pages 159–174, 2004.
6. D. Breuker, J.W.H.M. Uiterwijk, and H.J. van den Herik. Replacement schemes for transposition tables. *ICCA Journal*, 17(4):183–193, 1994.
7. H.S. Chang, M. Fu, J. Hu, and S.I. Marcus. An adaptive sampling algorithm for solving Markov decision processes. *Operations Research*, 53(1):126–139, 2005.
8. M. Chung, M. Buro, and J. Schaeffer. Monte Carlo planning in RTS games. In *CIG 2005, Colchester, UK*, 2005.
9. Y. Higashiuchi and R. Grimbergen. Enhancing search efficiency by using move categorization based on game progress in amazons. In *Advances in Computer Games 11 (to appear)*, 2006.
10. S. Janson. Large deviation inequalities for sums of indicator variables. Technical report, Uppsala, 1994.
11. S. Janson, T. Luczak, and A. Ruciǹski. *Random Graphs*. John Wiley & Sons, New York, 2000.
12. S. Janson and A. Ruciǹski. The infamous upper tail. *Random Structures and Algorithms*, 20(3):317–342, 2002.

[10] Note that the result applies when X_i is a martingale difference process shifted by some constant.

13. M. Kearns, Y. Mansour, and A.Y. Ng. A sparse sampling algorithm for near-optimal planning in large Markovian decision processes. In *Proceedings of IJCAI'99*, pages 1324–1331, 1999.
14. T. Ali Khan and R. Neininger. Probabilistic analysis for randomized game tree evaluation. In M. Drmota, P. Flajolet, D. Gardy, and B. Gittenberger, editors, *Mathematics and Computer Science III (Vienna 2004)*, Trends in Mathematics. Birkhäuser, 2004.
15. T.L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.
16. N. Limnios and M.L. Ting-Lee. Hoeffding's inequality for stopped martingales and semi-Markov processes. *Communications on Statistics – Theory and Methods*, 34:713–720, 2005.
17. M. Müller and T. Tegos. Experiments in computer amazons. In R. Nowakowski, editor, *More Games of No Chance*, pages 243–260, 2002.
18. L. Péret and F. Garcia. On-line search for solving Markov decision processes via heuristic sampling. In R.L. de Mántaras and L. Saitta, editors, *ECAI*, pages 530–534, 2004.
19. W. Rhee and M. Talagrand. Martingale inequalities and NP-complete problems. *Mathematics of Operations Research*, 12:177–181, 1987.
20. E. Shamir and J. Spencer. Sharp concentration of the chromatic number on random graphs $g_{n,p}$. *Combinatorica*, 7:121–129, 1987.
21. B. Sheppard. World-championship-caliber Scrabble. *Artificial Intelligence*, 134(1–2):241–275, 2002.
22. S.J.J. Smith and D.S. Nau. An analysis of forward pruning. In *AAAI*, pages 1386–1391, 1994.
23. M. Snir. Lower bounds for probabilistic linear decision trees. *Theoretical Computer Science*, 38:69–82, 1985.
24. G. Tesauro and G.R. Galperin. On-line policy improvement using Monte-Carlo search. In M.C. Mozer, M.I. Jordan, and T. Petsche, editors, *NIPS 9*, pages 1068–1074, 1997.
25. T. Wang, D. Lizotte, M. Bowling, and D. Schuurmans. Bayesian sparse sampling for on-line reward optimization. In *ICML-2005*, 2005.
26. J. Willemson and M.H.M. Winands. MILA wins Clobber tournament. *ICGA Journal*, 28(3):188–190, September 2005.
27. M.H.M. Winands. *Informed Search in Complex Games*. PhD thesis, Universiteit Maastricht, Maastricht, The Netherlands, 2004.