2008－GI－20 (4)
2008／6／27

UCT アルゴリズムにおける確率的な試行回数削減方法

但馬 康宏，小谷 善行

東京農工大学 工学部 情報工学科

あらまし 近年，囲碁や将棋などのゲームにおける着手の決定にモンテカルロシミュレーションを用いた手法が注目されている．このシミュレーションの制御には UCT アルゴリズムが効果的であることが知られているが，これは k-armed バンデット問題に対する UCB1 アルゴリズムをゲーム木の各節点に適用したアルゴリズムである．ところが，UCB1 アルゴリズムは k-armed バンデット問題に対して，任意の試行回数のもとでの効果を保証しているためゲームの着手決定へ適用する場合には改善の余地がある．本研究では，UCB1 アルゴリズムの実行中にもっとも高い勝率とならない着手を推定する手法を提案し，評価実験を行った．その結果，UCB1 および UCT の両アルゴリズムと同程度の強さを保ちつつ，シミュレーションの試行回数を削減することができた．

A probabilistic reduction of playouts in UCT algorithm

Yasuhiro TAJIMA and Yoshiyuki KOTANI

Department of Computer and Information Sciences,
Tokyo University of Agriculture and Technology

Abstract Monte-carlo simulation in games is a hot topic for selecting the best move. Especially, computer GO is growing rapidly with this method. UCT algorithm is one of the most effective algorithm to control random similations and UCT is based on UCB1 algorithm which is an effective algorithm for k-armed bandit problem. Since, this algorithm is made for the k-armed bandit problem, we can sophisticate it for games. In this paper, we propose a probability estimation method to find moves which may not be selected in the limited times of simulations and we apply this method to UCT algorithm. In experimets, we have made matches between UCT , UCB1 and our algorithm, then we have confirmed that our estimation method works.

1 はじめに

モンテカルロシミュレーションによるゲームシステムは，評価関数の作成が難しいゲームにおいて近年大きな成果をあげている．一般にゲームの着手決定アルゴリズムは，ヒューリスティックや機械学習により獲得された評価関数を用いて，ゲーム木を min-max 探索することにより実現されることが多い．しかし，ランダムシミュレーションの結果による手法はそのアルゴリズムの簡潔さに比べ強い手を選択する場合が多く，ゲームに対する新たなアルゴリズムとして注目を集めている．

数年前までは，ランダムシミュレーションの制御にヒューリスティック [8] や正規分布とみなした勝率の分布 [6]，並列化やヒューリスティックとの組合せ [5]，さらに局所戦略をランダムシミュレー

ションで評価づけし，着手選択を行う方法 [7] など様々な試みが行われていた．しかし近年になって，UCT アルゴリズムを中心にヒューリスティックの適用 [4] や機械学習の利用 [3] を行う研究が一定の成果を上げている．

モンテカルロシミュレーションは，可能手すべてに対して同じ回数だけシミュレーションを行っていたのでは得られる勝率のばらつきや，明らかに強くない手に対するシミュレーションなど無駄が多くなる．そこでシミュレーションの制御を行うアルゴリズムが必要となる．この制御に対して，k-armed バンデット問題に対する解法の応用が広く知られている．k-armed バンデット問題とは，0 から 1 の値をとる k 個の確率変数 $X_i(i = 1, 2, \cdots, k)$ から一つを選択する操作 (試行) を n 回行い，そのとき得られる値の総和をできるだけ大きくする問題である．ここで，各確率変数は互いに独立で，それぞれの期待値は全試行を通じて固定されているものとする．この問題に対するアルゴリズムとして，UCB1 アルゴリズムが知られている [1]．UCB1 アルゴリズムは，任意の n 回の試行において，もっとも期待値の高い確率変数を選びつづけた場合との差 (取りこぼし) の上界を保証している．このアルゴリズムを思考ゲームに適用する場合，ゲームの各局面における可能手を k-arm とみなし，一つの可能手でゲームを進めた局面からランダムシミュレーションを行った場合の勝率を各確率変数の値として利用する．この適用の発展として，ゲーム木探索と組み合わせたアルゴリズム UCT [2] も多くの研究で実装されている．

以上のように UCB1 アルゴリズムとそのゲーム木への応用である UCT アルゴリズムはゲームの着手決定における有益なアルゴリズムであるが，特に UCB1 アルゴリズムはシミュレーションの総試行回数に関する仮定を持っていない．すなわち，任意の試行回数において獲得値の最大化を目指すアルゴリズムとなっている．しかし，ゲームの着手選択においては，一定の時間内にもっとも獲得値の高い arm，すなわち勝率の高い着手を見つけることが重要である点が k-armed バンデット問題との相違となっている．

本研究では，獲得期待値が最大となる着手を効率よく発見するために，UCB1 アルゴリズムにおいて獲得期待値が最大とはなり得ない可能手を推定する方法を提案し，その評価を行う．UCB1 アルゴリズムでは，獲得期待値が低い確率変数の試行回数は，もっとも期待値の高い変数との期待値の差の二乗に反比例する点に着目し，各変数がもっとも高い勝率となるのに必要な試行回数を推定し，ランダムシミュレーションを行うべきか否かを決定する．

さらに，本手法を UCT アルゴリズムに適用することにより，その効果を確かめた．その結果，UCB1, UCT 両アルゴリズムと比較して，同程度の強さを保ちつつシミュレーションの試行回数を削減することができた．

2 UCB1 アルゴリズムと思考ゲームへの応用

k-armed バンデット問題は以下のように定義される問題である．

- k 個の確率変数 $X_1, X_2, \cdots, X_k$ は $0-1$ の範囲の値をとる．それぞれの期待値はあらかじめ定まっているが未知であり，アルゴリズム実行中に変化しないものとする．これらを arm と呼ぶ．X_i の期待値を μ_i で表す．X_i の値は分布 P_i にしたがうものとする．
- 1 回の試行とは，X_i $(i = 1, 2, \cdots, k)$ を 1 つ選択し，その値を得ることにより終了する．
- n 回の試行を繰り替えした後の総獲得値 $\sum_{j=1}^{n} X_{\Lambda(j)}$ を最大化することが目的である．ここで，$\Lambda(j)$ は，j 回目の試行における選択を表す．すなわち，j 回目の試行で X_i を選択した場合，$\Lambda(j) = i$ である．

μ_i $(i = 1, 2, \cdots, k)$ の中で最大のものを μ^* と表す．同様に $\mu^* = \mu_i$ である i について，X_i を X^* と表す．あるアルゴリズムにおいて，合計 n 回の試行の中で X_i を試行した回数を $T_i(n)$ と表す．また，$\bar{x}_i = (1/T_i(n)) \sum_{1 \le j \le n, \Lambda(j)=i} X_i$ とする．さらに，$\Delta_i = \mu^* - \mu_i$ とする．

UCB1 アルゴリズムは，以下のとおりである．

1. すべての X_i $(i = 1, 2, \cdots, k)$ について 1 度づつ試行する．

2. 以下の値がもっとも大きな j について試行を行う．

$$\bar{x_j} + \sqrt{\frac{2\ln n}{n_j}}$$

ここで n は現時点での総試行回数であり，n_j は現時点までに X_j を試行した回数である．

3. 前ステップを繰り返す．

このとき，以下の定理が成り立つ [1]．

定理 1 UCB1 アルゴリズムを用いて n 回の試行を行ったときの選択の系列を $\Lambda_u(k)$ $(k = 1, 2, \cdots, n)$ とする．このとき，$n \cdot \mu^* - \sum_{k=1,\cdots,n} X_{\Lambda_u(k)}$ は以下の値で抑えられる．

$$\left(8 \sum_{i:\mu_i<\mu^*} \frac{\ln n}{\Delta_i}\right) + \left(1 + \frac{\pi^2}{3}\right)\left(\sum_{j=1}^{k} \Delta_j\right)$$

□

さらに，任意の $i = 1, 2, \cdots, k$ について，

$$E(T_i(n)) \leq \frac{8\ln n}{\Delta_i^2} + 1 + \frac{\pi^2}{3}$$

が成り立つ．すなわち，ある arm i が試行される回数の期待値は，Δ_i の二乗に反比例する．

囲碁や将棋などの二人確定完全情報零和ゲームの過程は，初期局面を根，可能手を枝に持ち，その可能手により進められた局面を子節点にもつゲーム木で表現することができる．上記 UCB1 アルゴリズムは，以下のようにゲームにおける着手選択に適用される．

- 現在の局面に対応するゲーム木の節点から，直接枝が張られているすべての子節点を k-armed バンデット問題の arm に対応させる．
- UCB1 アルゴリズムにおける試行は，対応する子節点からランダムに着手を選択し対戦結果をシミュレートし，その結果が勝ちならば $X_i = 1$ 負けならば $X_i = 0$ とする (図 1)．

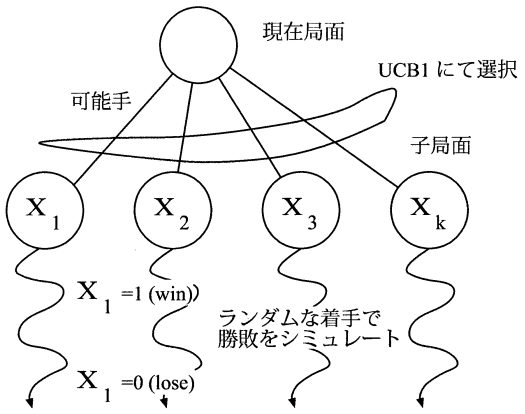

図 1: ゲーム木における UCB1 アルゴリズム

3 UCB1 アルゴリズムに対する試行回数削減手法

UCB1 アルゴリズムは，任意の試行回数においてその性能が保証されたアルゴリズムである．これは，探索と獲得 (exploration-exploitation) ジレンマを上手にさばいていることとなり，試行全体での獲得値の最大化を目指している．しかし，ゲームへの適用においては，子節点の中でもっとも勝率が高いものを選び出せればよく，試行全体での勝率の上昇は求めていない．この点に改善の余地がある．

UCB1 アルゴリズムにおいて j を $\mu^* = \mu_j$ であるとし，i は $\mu^* \neq \mu_i$ であるとする．さらに，$\bar{x_i} < \bar{x_j}$ であるとする．このとき，i に対する試行を重ねて $\bar{x_j} \leq \bar{x_i}$ となるために必要な試行回数を考える．現時点で n 回の試行がなされ，i に対する試行回数を $T_i(n) = n_i$ とする．m_i を現時点での i における勝ちの数とすれば，$\bar{x_i} = m_i/n_i$ と表せる．ここで，新たに i を試行し勝ちを得た場合，$\bar{x_i}$ の増加量は，

$$\frac{m_i + 1}{n_i + 1} - \frac{m_i}{n_i} = \frac{1 - \bar{x_i}}{n_i + 1}$$

である．この後さらに勝ちの試行を得た場合の $\bar{x_i}$ の増加量は，

$$\left(\frac{1 - \bar{x_i}}{n_i + 1}\right)\left(\frac{n_i}{n_i + 2}\right) \leq \frac{1 - \bar{x_i}}{n_i + 1}$$

である．したがって，$\bar{x_j}$ に変化がない場合，少なくとも以下の回数は勝ちの試行を得なければならない (図 2)．

$$\frac{\bar{x_j} - \bar{x_i}}{\frac{1-\bar{x_i}}{n_i+1}} = \frac{\bar{x_j} - \bar{x_i}}{1 - \bar{x_i}}(n_i + 1)$$

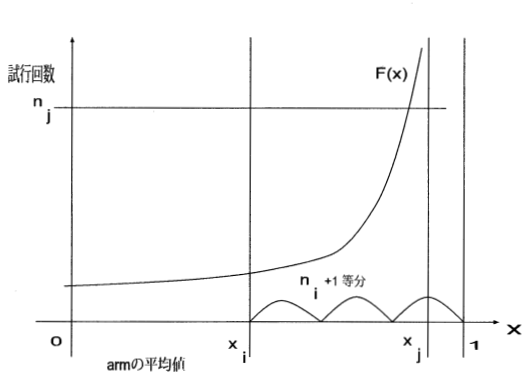

図 2: arm の平均値と試行回数

次に i が 1 回試行される間に j が試行される回数を考える．定理 1 と付随する結果より，おおよそ

$$E(T_i(n)) \leq \frac{8\ln n}{(\bar{x_j}-\bar{x_i})^2}+1+\frac{\pi^2}{3}$$

が成り立つ．ここで，

$$F(x)=\frac{8\ln n}{(\bar{x_j}-\bar{x})^2}+1+\frac{\pi^2}{3}$$

と置く．ところで，UCB1 アルゴリズムは，試行回数が同じ二つの arm に対しては，より勝率の高い arm を選択するので，j に対して，$T_j(n)=n_j$ とし，m_j を $m_j=\bar{x_j}n_j$ とすれば，i が 1 回試行されると j は

$$\max\left(1,\frac{n_j}{F(\bar{x_i})}\right)$$

回試行されることとなる．

以上より，$\bar{x_j}\leq\bar{x_i}$ まで変化する間に j は，少なくとも以下の z 回試行されることとなる．

$$z=\sum_{k=1}^{N}\max\left(1,\frac{n_j}{F(\bar{x_i}+\frac{1-\bar{x_i}}{n_i+1}k)}\right)$$

ここで，

$$N=\lfloor\frac{\bar{x_j}-\bar{x_i}}{1-\bar{x_i}}(n_i+1)\rfloor$$

である．

一般にゲームにおける着手選択は，時間制限のある中で決定しなければならず，UCB1 アルゴリズムにおいても最も試行回数の多い arm がある一定回数となった時点で着手が決定される場合が一般的である．その回数を s とすると，

$$z>s-\bar{x_j}$$

であれば，着手が決定される前に $\bar{x_j}\leq\bar{x_i}$ となることはない．すなわち，上記条件を満たす i に関する試行は，やらなくてよいこととなる．

4 UCT アルゴリズムでの試行回数削減手法

前節にて示した UCB1 アルゴリズムに対する削減手法を UCT アルゴリズムに適用する．本研究では，UCT で構成される根ノードからの着手選択において，前記アルゴリズムを用いて試行回数削減を行う (図 3)．

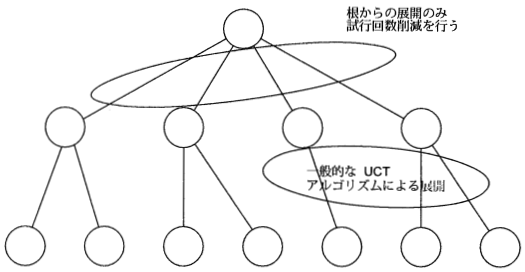

図 3: UCT アルゴリズムにおける試行回数削減方法の適用

5 評価実験

提案手法による arm の選択の効果を実験を通して確かめる．本研究では，ブロックスデュオというボードゲームにおいて提案手法の有効性を確かめた．ブロックスデュオは，以下のようなルールで行われる二人有限確定完全情報零和ゲームである．

- プレイヤーは，1 から 5 個の正方形でできたパズルピース (ポリオミノ) 21 種類を各 1 つずつ持つ．

- ボードは，14x14 のマス目である．
- ゲームは互いにピースをボードに置くことにより進められる．
- ボードの一つの対角線の、両端から 5 番目の位置 (2 箇所) に印がついている．それぞれ 1 手目のピースはこのマスを覆うように置く．
- 2 手目以降のピースは，自分のピースの角同士が接し，辺同士は接してはいけない．相手のピースとはどのように接してもよい．

実験では，UCB1 および UCT アルゴリズムそれぞれについて，その試行回数削減手法 (提案手法) との対戦で，以下の項目を測定した．

- 選択した着手の違い
- 試行の削減回数
- 計算時間の変化
- 対戦結果

5.1 選択着手の違い

本手法によるプログラムと UCB1 アルゴリズムによるプログラムで，同一局面で着手に違いがあるかどうかを示す．これは，一般の UCB1 (もしくは UCT) アルゴリズムどうしで対戦を行い，対戦中の各局面において，実際に打たれた手と，提案手法がその局面から打つ手との一致を計測したものである．

表 1 に結果を示す．UCB1 アルゴリズムにおいてはおよそ 75% の一致率となり，これは，同一局面から UCB1 アルゴリズムで複数回着手を決定したときの一致率とほぼ同じである．したがって，UCB1 アルゴリズムについては，提案手法は着手決定に変化を与えていないと言える．

UCT アルゴリズムについては，一致率は極めて低く，同一の着手となることは少なかった．この変化が強さに影響を与えているかどうかより調査が必要である．

5.2 試行の削減回数

表 2 に提案手法によって削減された試行回数を示す．これは，提案手法と削減を行っていない UCB1, UCT 各アルゴリズムとの対戦において，本来のアルゴリズムでは試行を行うため選択された候補手が，提案手法の基準により試行されなかった場合の数を表している．

UCB1 アルゴリズムでは，全体の 5% ほどの削減率であるが，UCT アルゴリズムでは 40% を越えた値となっている．これは，UCT での削減手法が根に限られている点，および UCT アルゴリズムが良い候補手をより高い勝率として判断する点が原因と思われる．より多くの実験が必要だが，本手法の有効性を示している．

5.3 計算時間の変化

表 3 に本手法と UCB1 の対戦でそれぞれが利用した計算時時間を示す．UCB1 アルゴリズムでは，先手後手とも 101 対戦での平均であり，UCT アルゴリズムでは，提案手法が先手 3 対戦，提案手法が後手 4 対戦である．

UCB1 における比較では，試行回数の削減率より低い値となっている．これは，削減対象について提案手法による計算を行っている時間と見ることができる．ブロックスデュオでは，UCB1 アルゴリズムでは，後手の方が計算時間が少なく住む場合が多いことが読み取れる．

UCT アルゴリズムについては，対戦数が少ないが，試行の削減率に比べ計算時間は減っていない．これも，削減対象となるか否かの判定に計算時間が取られているものと思われる．さらに，試行の削減の計測は根での選択のみを計測対象としているのに対して，計算時間は根以外での計算時間も原因として挙げられる．

5.4 対戦結果

表 4 に対戦結果を示す．UCB1 アルゴリズムとその削減方法との対戦は，先手後手それぞれ 101 対戦ずつである．この結果より，本手法による試行回数削減は，強さに影響を与えていないことが分かる．これは，本来強い手がサンプリングの結

表 1: 着手の一致率

	一致した手	局面数	一致率
UCB1 (202 対戦)	2137	2858	74.8%
UCT (3 対戦)	6	54	11.1%

表 2: 試行の削減回数

	削減基準を満たした回数		全試行回数		削減率
UCB1 アルゴリズム (202 対戦)	15321443	/	298166620	=	5.14%
UCT アルゴリズム (5 対戦)	880270	/	1964721	=	44.80%

表 3: 1 ゲームあたりの計算時間

対戦内容 (先手 vs 後手)	先手計算時間 (秒)	後手計算時間 (秒)	削減率
UCB1 提案手法 vs UCB1	1026.57	585.126	
UCB1 vs UCB1 提案手法	1056.23	594.505	
UCB1 全体			98.76%
UCT 提案手法 vs UCT	1096.89	741.05	
UCT vs UCT 提案手法	2169.99	1173.59	
UCT 全体			78.00%

果偶然に弱いと見なされ，削減対象となっていることが少ないことを表している．

UCT アルゴリズムとその削減方法との対戦は，実験時間の都合で対戦数が少ないが，UCB1 における結果と同様の傾向が見える．

6 おわりに

k-armed バンデット問題に対するアルゴリズム UCB1 をゲームの着手選択に応用した場合に，着手が選択されなくなる条件を示し，試行回数の削減を行う手法を示した．さらに，本手法を UCT アルゴリズムに適用する手法を示した．評価実験として，ボードゲームであるブロックスデュオの対戦プログラムを作成し，本手法による効果を確認した．しかし，計算時間については大きな削減はできておらず，今後の課題である．

本研究で示した条件は，着手選択が一定の試行回数を上限とする前提で組み立てられているため，その前提の緩和が今後の課題として挙げられる．さらに，UCT の根以外のノードにおける削減手法も今後の課題である．

参考文献

[1] P.Auer, N.Cesa-Bianchi and P. Fischer, Finite-time analysis of the multiarmed bandit problem, Machine Learning, vol.47, nos.2,3, p.235–256, 2002.

[2] Levente Kocsis and Csaba Szepesvari, Bandit based monte-carlo planning, Lecture Notes in Computer Science, vol. 4212, pp.282-293, 2006.

[3] Sylvain Gelly and David Silver, Combining online and offline knowledge in UCT, Proc. of the 24th International Conference on Machine Learning, pp.273-280, 2007.

[4] Guillaume Chaslot, Mark Winands, H. Jaap van den Herik, Jos Uiterwijk, Bruno Bouzy, Progressive strategies for Monte-Carlo tree search,

[5] Haruhiro Yoshimoto, Kazuki Yoshizoe, Tomoyuki Kaneko, Akihiro Kishimoto, and Kenjiro Taura, Monte Carlo Go Has a Way to Go, Proc. of the 21st National Conference on Artificial Intelligence (AAAI-06), pp. 1070-1075, 2006.

[6] Remi Coulom, Efficient selectivity and backup operators in Monte-Carlo tree search, Proc. of the 5th International Conference on Computer and Games, CG2006, pp.72–83, 2006.

[7] Tristan Cazenave, Bernard Helmstetter, Combining tactical search and Monte-Carlo in the game of Go, Proc. of IEEE conference on Computational Intelligence and Games, CIG2005, 2005.

[8] Bruno Bouzy and Bernard Helmstetter, Monte Carlo go developments, Advances in Computer Games conference (ACG-10), pp. 159-174, 2003

表 4: 対戦結果

対戦内容 (先手 vs 後手)	提案手法の勝ち	負け	分け
UCB1 削減法 (提案手法) vs UCB1	33	64	4
UCB1 vs UCB1 削減法 (提案手法)	63	33	5
UCT 削減法 (提案手法) vs UCT	2	0	0
UCT vs UCT 削減法 (提案手法)	0	2	0