モンテカルロ法における勝率近似関数の組み込み方法

但馬 康宏†，小谷善行†

†東京農工大学大学院 工学府

あらまし 本稿では，UCB1 アルゴリズムを用いたモンテカルロシミュレーションにおいて，評価関数を効果的に取り込む方法を提案する．過去の研究においても UCT アルゴリズムに対してヒューリスティックな評価関数を用いて着手の制限を行い，効率を高める提案がなされているが，本手法は UCB1 アルゴリズムの一部に評価関数をスムーズに取り込む方法である．評価実験として，ブロックスデュオにおいて，UCB1 アルゴリズムおよび評価関数のみによるアルゴリズムと対戦し，計算時間と勝敗を計測した．その結果，一定の成果があることが確認できた．

A combination method between Monte-Carlo simulations and a win-rate approximation function

Yasuhiro Tajima†, Yoshiyuki Kotani†

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

Abstract We show a combination method between UCB1 algorithm and an win-rate approximation function. Eventhough there are some studies which uses a huristic evaluation function in UCT algorithm, our method takes the evaluation function into UCB1 algorithm smoothly. For evaluation to confirm our method, we made some matches between our algorithm and UCB1 algorithm or a huristic search algorithm.

1 はじめに

コンピュータ囲碁などにおけるモンテカルロシミュレーションによる着手選択は，その手軽さと強さから有望な着手選択手法として注目されている．これは，ランダムに着手を決定し勝敗を求めたゲーム (ランダムサンプリング) を多数繰り返し，その勝率を基に着手を決定する方法である．

ランダムサンプリングの制御には，UCB1 アルゴリズム [1] やそのゲーム木への応用である UCT アルゴリズム [2] を用いる方法が大きな成果を挙げている．ここで，UCB1 アルゴリズムは，k-armed バンデット問題に対するアルゴリズムであり，より勝率の大きな着手を重点的に調査するアルゴリズムである．また，UCT アルゴリズムは，ゲーム木を作成し，その中間ノードにおいて UCB1 アルゴリズムに沿った選択を行うことによりランダムシミュレーションを行う葉を決定するアルゴリズムである．

ところが，UCT アルゴリズムにおいて可能手をすべて等しく扱うと，作成されるゲーム木が巨大なものとなってしまうことが知られており，これはすなわちシミュレーション時間の無駄遣いにつながってしまう．そこで，UCT アルゴリズムにおける中間ノードでの選択に評価関数を取り込む方法が多く試されている．これは，ゲームの局面を評価する評価関数を用いてあらかじめ中間ノー

ドにおける選択に優先順位を付けておき，その優先度の低いものはその先の木が展開されにくく制御することによりゲーム木を効率よく展開するものである．ここで，評価関数はヒューリスティックに作られたもの [4][6] や機械学習により獲得されたもの [3]，また，局面の特徴に基づいた評価を用いるもの [5] など様々な手法が試されている．

2 ランダムサンプリングと勝率近似関数

本研究では，UCB1 アルゴリズムにおいて以下のような関数の利用方法を考える．まず，与えられた局面において，ランダムサンプリングによる勝率に近い評価値を返す関数を考え，それを勝率近似関数と呼ぶ．すなわち，与えられた局面から特徴量を抽出し，それに基づき局面の勝率をランダムサンプリングを用いずに出力する関数である．UCB1 アルゴリズムでは，可能手の勝率とその可能手が選択される回数の上界値が理論的に明確であるので，勝率近似関数の出力値をその可能手の勝率と見なしたときに，可能手ごとにランダムサンプリングが試行された回数を定めることができる (図 1)．

したがって，すべての可能手に対してその勝率と試行された回数を決定し，その状態からランダムサンプリングを行う本来の UCB1 アルゴリズムを実行することにより，着手決定までの計算量を短縮させることができる．

3 UCB1 アルゴリズムにおける勝率近似関数による初期値設定

本研究における提案手法を具体的に説明する．ゲームにおけるある局面の可能手が k 個あると仮定する．k 個の確率変数 $X_1, X_2, \cdots, X_k$ はそれぞれの可能手のランダムサンプリングによる勝率を表し，X_i の期待値を μ_i で表す．ある可能手に対するランダムサンプリングを 1 回の試行と呼ぶ．n 回の試行を繰り替えした後の総獲得値 $\sum_{j=1}^{n} X_{\Lambda(j)}$ を最大化するためのアルゴリズムが UCB1 アルゴリズムである．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，$\bar{x_i} = (1/T_i(n)) \sum_{1 \leq j \leq n, \Lambda(j)=i} X_i$ 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，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，ある可能手 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 (図 2)．

機械学習などで勝率近似関数 $f()$ をあらかじめ求めておき，以下のアルゴリズムで着手を決定する．

1. 着手を求めたい局面のすべての可能手について局面を展開し，子局面の集合 $\{A_1, A_2, \cdots, A_k\}$ を求める．
2. 各子局面 A_i について，勝率 X_i と試行回数 n_i を $f()$ と $E(T_i())$ を用いて算出する．
3. この状態から UCB1 アルゴリズムを実行し着手を選択する．

以上の操作により，すべてランダムサンプリングにより着手を決定した場合よりも計算量を削減することが期待できる．すなわち，評価関数の出力値と同等の精度を得るまでのランダムサンプリングにかかる時間が，評価関数の計算時間よりも長い場合に本手法は有効となる．

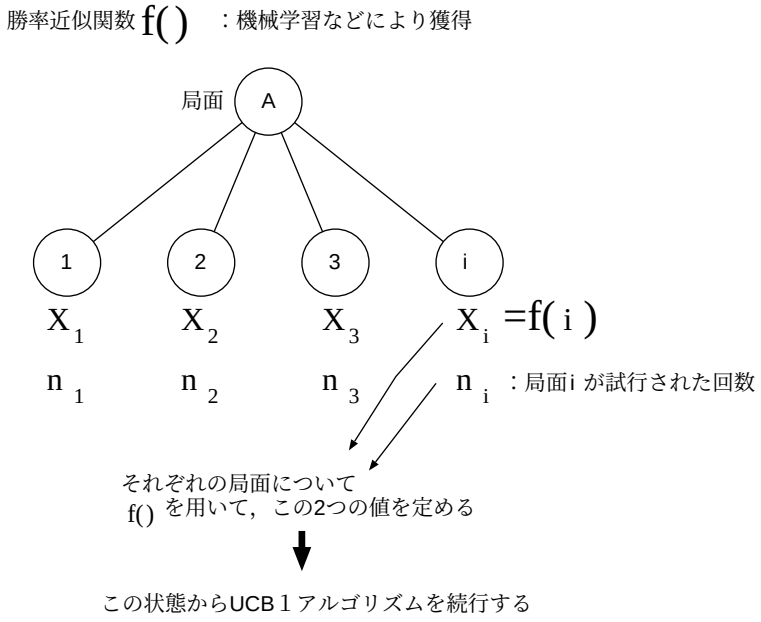

図 1: 勝率近似関数の利用方法

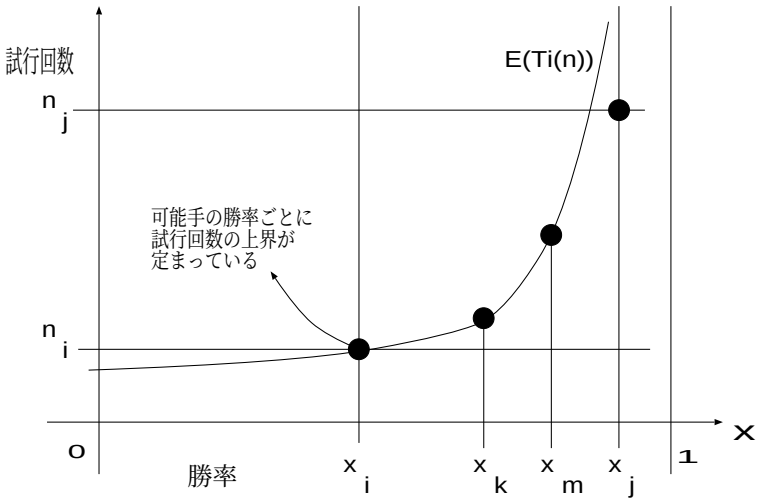

図 2: UCB1 における勝率と試行回数の関係

表 1: 本手法との対戦結果

本手法の	勝	敗	分
本手法 (先)vsUCB1	5	6	0
UCB1(先)vs 本手法	3	7	1
本手法 (先)vs 評価関数	4	0	0
評価関数 (先)vs 本手法	4	0	0

表 2: 平均時間計算量

(秒)	先手	後手
本手法	33027.2	10334.4
UCB1	55295.5	36567.8

4 評価実験

本手法の効果確認として，本手法によるアルゴリズムと一般の UCB1 アルゴリズムによる対戦および，本手法とヒューリスティックな評価関数によるアルゴリズムとの対戦をおこなった．対象とするゲームはブロックスデュオとし，本手法，UCB1，評価関数いずれのアルゴリズムについても対戦時間がおおよそ 15 分程度となるように調整した．本手法によるアルゴリズムでは，以下の手順で着手を決定する．

1. 与えられた局面に対して各可能手の評価値を評価関数を用いて決定する．ここで，評価関数は昨年度ブロックスデュオ大会優勝者のページ [7] を参考に作成した．
2. 評価値の最も高い着手について，ランダムシミュレーションを 100 回行い，おおよその勝率を求めた．
3. 各着手について，その見込みの勝率を上記勝率と評価値から求めた．
4. 以上を用いて本手法を実行する．

また，評価関数によるアルゴリズムは，本手法と同じ評価関数を用い，候補手の局面の評価値を利用したため，探索は行っていない．

対戦結果を表 1 に示す．

また，各アルゴリズムにの 1 対戦あたりの平均時間計算量を表 2 に示す．対戦結果と合わせて，本手法がそれほど弱くなくかつ時間計算量を半減させられたことが判る．

以上より本手法の有効性が確認できた．今後の課題として，UCT アルゴリズムへの本手法の適用が挙げられる．

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, Proc. of the 10th Joint Conference on Information Science, paper no. CSI-48, 2007.

[5] Remi Coulom, Computing Elo ratings of move patterns in the game of Go, Proc. of the Computer Games Workshop 2007.

[6] Peter Drake and Steve Uurtamo, Move ordering vs heavy playouts: Where should heuristics be applied in Monte-Carlo Go?, Proc. of GAMEON North America 2007, paper no. AI_04, 2007

[7] irori の日記 - ブロックスデュオプログラム対戦結果
http://d.hatena.ne.jp/Irori/20071104/1194151812