UCTのシミュレーションに対する重みづけに関する研究

野口陽来，松井利樹，橋本隼一
北陸先端科学技術大学院大学

概要

現在，コンピュータ囲碁ではUCTを用いた探索法が主流となっている．その流れの中で強いプログラムを作るために様々な研究が行われている．大きく二つに分けるとUCTのような木探索アルゴリズムの改良と，パターンの使用のようなプレイアウトの改良に分けられる．本研究ではUCT部分の改良としてプレイアウトの重みを変更することでUCTの性能向上を図った．その結果勝率を目的の値に近づけることができたが，プログラムの強さを向上させることはできなかった．

A Study about Weighted Simulations on UCT

Haruki Noguchi, Toshiki Matsui, Junichi Hashimoto
Japan Advanced Institute of Science Technology

Abstract

Today, Monte-Carlo method is mainly used in the domain of computer go. There are many studies that make the program stronger. There are two types of study: one is for improvement of search efficiency of tree search algorithms, the other is for improvement of accuracy of the Monte-Carlo playout. In this paper, we change the importance of each playout by use of weight parameter. The experimental result shows that winning ratio comes close to purpose value by using weight parameter. But we cannot make the program stronger.

1．はじめに

従来の方法で囲碁プログラムを作る際，大きな障害となるのが適切な評価関数の設計である．しかし1993年に評価関数を設計せずにプログラム作成を可能とする新しい方法，モンテカルロ法を囲碁に適用する方法が提案された[1]．モンテカルロ法では評価関数は使用せず，ランダムなゲームを多数行いそれらの結果を局面の評価とする．

単純なモンテカルロ法だけでは強いプログラムを作ることはできなかった．しかし現在は9路盤でプロと対等に渡り合うだけの実力を有している．この飛躍的な性能向上の理由の一つにモンテカルロ法と探索アルゴリズムを組み合わせたことが挙げられる．これによりCrazyStoneは2006年のコンピュータオリンピアドで優勝という成果を上げた[2]．現在では木探索がアルゴリズムとして数学的な裏付が明確なUCTが主に使われている[3]．

UCTではある局面から末端局面まで木を下り，必要があれば局面を新たに展開し，そこからランダムなゲームを行い，その結果を通過した全ての局面に繰り上げる．この作業を繰り返すことで探索木を成長させていく．この，ある局面から末端局面まで下りその局面まで値を繰り上げる処理を本論文ではシミュレーショ

ンと呼ぶ.

UCTでは式(1)に示す指標を用いて着手を選択し木を降りていく. p_iはシミュレーションで勝利すれば1, 負ければ0となる. 右辺の第一項は評価する局面を通ったシミュレーション結果の平均値を表す項であり, 第二項は『探検と収穫』のバランスをとるための項である. 平均値を用いるということは全てのシミュレーションを同等に扱うことを意味する.

$$UCB_s = \sum_{i=1}^{n}[p_i] \Big/ n + C\sqrt{\ln n / n_s} \cdots(1)$$

s : UCB値を求めたい局面
p_i : シミュレーション結果
n : sの親局面を通過したシミュレーションの回数
n_s : sを通過したシミュレーションの回数

UCTでは通常後に行われるシミュレーションほど信頼できるため, 全てのシミュレーションを同等に扱うのは問題がある. 例えば図1で一番上の局面を通るシミュレーションを二つ示したが, 右のシミュレーションの結果の方を左の結果より重要視して扱うべきである.

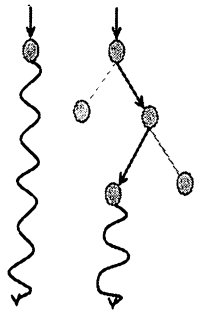

図1. 異なる時期に行ったシミュレーション

図1は木の成長によってシミュレーションがどのように変化するかを示したものである. 左の木では一番上の局面からランダムなゲームを行っているように初期のシミュレーションではある局面で選択される手はほとんどランダムに選択される. 対して右の木では探索木を最善手である可能性が高い手を選びながら末端まで降りてからランダムなゲームを行っている. 結果として後に行われたシミュレーションほど信頼性が向上する.

本研究では後に行われるシミュレーションの結果を重要視するために, シミュレーションに対する重みを変化させる. これによりプログラムが強くなるか検証する.

2. シミュレーションに対する重みづけ

通常のUCTではsの勝率E_sは(2)式で表される. ここでCは子供の数, μ_cは子供の勝率の期待値を表す. また$I(c)$はcを通過したプレイアウトの集合とする.

$$E_s = \sum_{i=1}^{n}[p_i] \Big/ n = \sum_{c=1}^{C}\left[\mu_c \sum_{i \in I(c)}[p_i] \Big/ n\mu_c \right] \cdots(2)$$

(2)を用いると通常$E_s < \max_c(\mu_c)$となる. ゲーム木がMin-Max的に生長することを考えると$E_s = \max_c(\mu_c)$が望ましいと考えられる.

そこでE_sに重みw_iを導入し, (3)式のように変形した. (3)式を(2)と同様の表現に変換すると(4)式が得られる. $W = \sum_{i=1}^{n}[w_i]$である.

$$E_s = \sum_{i=1}^{n}[w_i p_i] \Big/ \sum_{i=1}^{n}[w_i] \cdots(3)$$

$$E_s = \sum_{c=1}^{C}\left[\mu_c \sum_{i \in I(c)}[w_i p_i] \Big/ W\mu_c \right] \cdots(4)$$

(4)式で$w_i = 1$とすれば(2)式と等しくなる. 適当なw_iを選択することでE_sを$\max_c(\mu_c)$に近づけることができると考えられる.

UCBで勝率が最大の子供を選択する問題を解くときに, (4)式を用いることでE_sがどのよ

うに変化するかを実験した．w_iとして(5)式を用いた．

$$w_i = \begin{cases} i \\ \log(i+1) \\ \exp(i/100) \end{cases} \cdots(5)$$

実験条件として：

- $C = 15$,
- $\mu_c = \begin{cases} 0.37278 & 0.3958 & 0.41965 & 0.39566 & 0.42928 \\ 0.4383 & 0.402375 & 0.43515 & 0.44111 & 0.45242 \\ 0.40368 & 0.4338 & 0.44672 & 0.45395 & 0.45897 \end{cases}$,
- $\max(\mu_c) = 0.45897$,
- プレイアウトの結果はベルヌーイ分布に従う，

を用いた．この勝率は単純モンテカルロ法を用いて初期局面の各合法手の勝率を計算した値である．

図1(A)に各w_iを用いたときのE_sの変化を示した．グラフはそれぞれ上から$w_i = i$，$w_i = \log(i+1)$，$w_i = 1$の重みにおけるE_sの値を示している．重みを変えることでE_sが$\max(\mu_c)$に近づいていることがわかる．

また$\exp(i/100)$で重みを付けたグラフは振動して一定値に収束しなかった．

3．重みづけによる強さの変化

$w_i = i$，$w_i = \log(i+1)$としたプログラムと$w_i = 1$としたプログラムを戦わせ性能を評価した．実験で用いたプログラムの性能を表1に示す．

表1．実験に用いたプログラムの性能

シミュレーション速度	4000回／sec
選択方式	UCB1-TUNED
プレイアウトにおける合法手の確率分布	機械学習を用いて確率を生成
思考時間	5 sec／move

実験結果を表1(B)に示す．logを用いたプログラムは$w_i = 1$のプログラムと同じ強さを示している．iを用いたプログラムは$w_i = 1$のプログラムより弱くなっている．

表2．自己対戦の結果

$\log(i+1)$	499勝501敗
i	477勝523敗

図1(B)に図1(A)のシミュレーション回数が60000回以下のグラフを拡大したものを示す．$w_i = \log(i+1)$は$w_i = 1$に追従するように変化している．またE_sもほとんど変化していないため勝率が変わらなかったと考えられる．

$w_i = i$では追従はしているが，変化の幅が大きくなっていることがわかる．そのためE_sがばらつき探索が安定しなくなり，結果プログラムが弱くなったと考えられる．

4．まとめと今後の課題

UCTのシミュレーションは後半ほどより重要となると考え，重みw_iを導入した．その結果ノードの評価値を目的の値に近づけることができた．

しかし自己対戦によって効果を確認したところ，プログラムを強くすることはできなかった．

参考文献

[1] B. Bruegmann (1993), Monte Carlo Go, *ftp://www.joy.ne.jp/welcome/igs/Go/computer/mcgo.tex.Z.*

[2] R. Coulom (2006), Efficient Selectivity and Backup operatorsin Monte-Carlo Tree-Search, *Proceedings of the 5th International Conference on Computers and Games,* Turin, Italy.

[3] L. Kocsis and C. Szepesvari (2006), Bandit based Monte-Carlo planning, *5th European Conference on Machine Learning (ECML),* Pisa, Italy, Pages 282-293.

[4] L. Kocsis and C. Szepesvari (2006), Discounted UCB, *2nd PASCAL Challenges Workshop,* Venice, Italy.

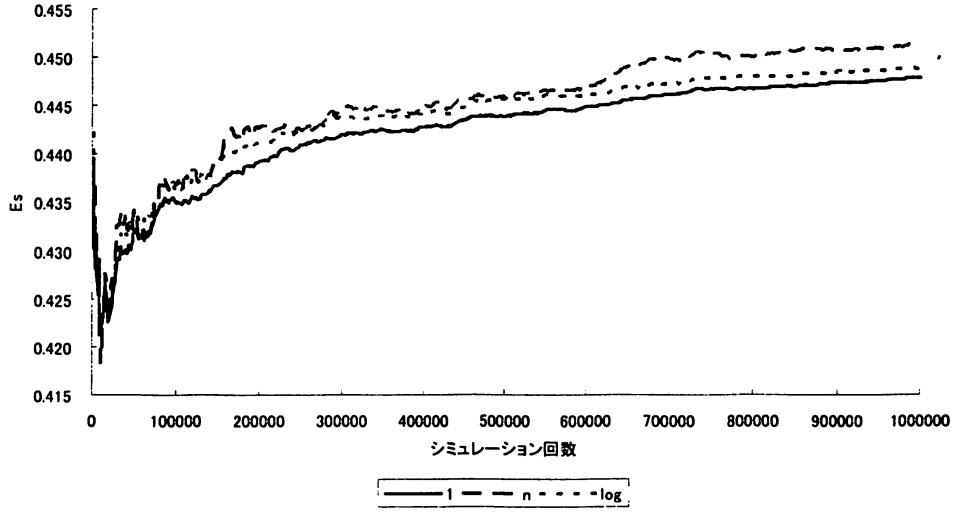

図 1 (A). 各 w_i に対する E_s の変化

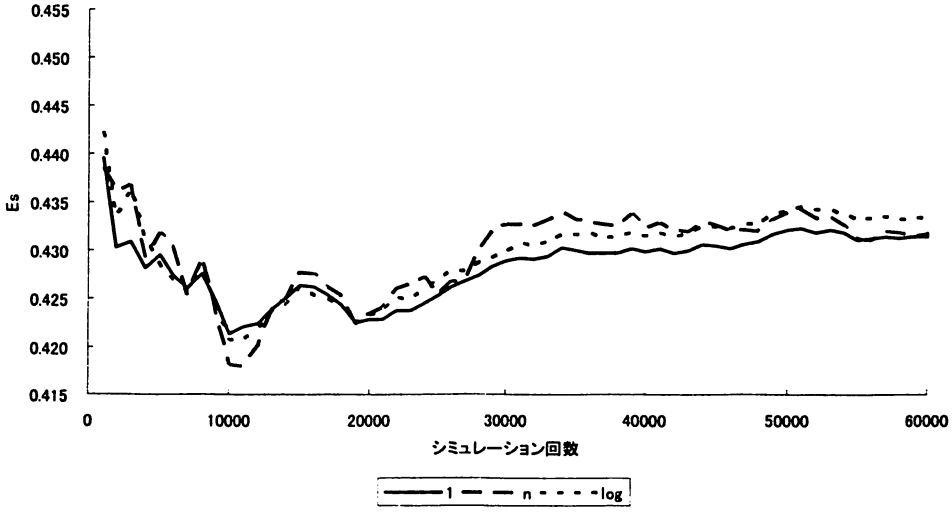

図 1(B). 各 w_i に対する E_s の変化