Vol.2010-GI-23 No.2
2010/3/8

囲碁におけるシミュレーション結果の継承を用いたモンテカルロ法の改良

豊田 琢磨[†1] 松本 祐輔[†2]
佐々木 健太[†2] 小谷 善行[†3]

近年，モンテカルロ法を用いたコンピュータ囲碁に関する研究が広く行われている．通常，モンテカルロ法をコンピュータ囲碁に用いた場合，着手後の探索木はメモリに保存されていなかった．本論文ではモンテカルロ法におけるシミュレーション結果の継承という新しい概念を提案し，以前行われたシミュレーション結果を再利用することでアルゴリズムの効率化を図った．原始モンテカルロ法，UCB1 アルゴリズム，UCT アルゴリズムに提案手法を適用し，適用していないプログラムとの比較実験を行ったところ，原始モンテカルロ法に 62.5（± 3.9）%，UCB1 アルゴリズムに 59.2（± 4.0）%，UCT アルゴリズムに 54.5（± 4.1）%の勝率を得た．これらの結果より，モンテカルロ法におけるシミュレーション結果の継承という新しい概念の有効性を示すことができた．

The Improvement of Monte-Carlo Method using Inherited Results of Simulation in Go

Takuma TOYODA,[†1] Yusuke MATSUMOTO ,[†2]
Kenta SASAKI [†2] and Yoshiyuki KOTANI[†3]

The researches on Go using Monte-Carlo method are treated as hot topics in these years. When Monte-Carlo method was used for Go, the previous results of simulation weren't usually memorized. In this paper, we suggest a new idea of using previous results of simulation and try to improve the method. When we applied this idea to Crude Monte Carlo, UCB1 algorithm, UCT algorithm, we got winning percentage of 62.5(± 3.9)%, 59.2(± 4.0)%, 54.5(± 4.1)%. Therefore, it was able to be shown that our idea was effectivity.

1. はじめに

囲碁は，二人零和有限確定完全情報ゲームのひとつであり，近年注目されているゲーム研究の対象のひとつである．簡単に言えば陣取りゲームであり，黒と白の石を交互に置いていき，囲った陣地の広さによって勝ち負けを判断する．また，先手の優位性をなくすためにコミと呼ばれるハンデ（後手は終局時にいくらかのポイントを足す）があり，手番による有利不利はほとんどない．

近年，コンピュータ囲碁において着手の評価方法にモンテカルロ法を用いることで大きな進歩があった．この手法は着手を決定する際に，終局までランダムシミュレーションを大量に行い，その中で勝率が最もよかった着手を選ぶ手法である．特に UCT アルゴリズム[1] と呼ばれる UCB1 値を再帰的に用いて木構造に適応させたアルゴリズムは，CrazyStone[2] や MoGo[3] などトップクラスのプログラムに使用され，9 路盤においてはプロレベルの強さを示すことに成功した．

通常，モンテカルロ法をコンピュータ囲碁に用いた場合，着手後の探索木はメモリに保存されていなかった．本論文ではモンテカルロ法におけるシミュレーション結果の継承という新しい概念を提案し，以前行われたシミュレーション結果を再利用することでアルゴリズムの効率化を図った．

以降，2 章では基礎理論について述べ，3 章では提案手法を説明する．そして 4 章では実験方法ならびに実験結果について示し，最後 5 章ではまとめと今後の展望について述べる．

2. 基礎理論

本章では，基本となる原始モンテカルロ法と UCB1 アルゴリズム[4]，UCT アルゴリズムについて述べる．

†1 東京農工大学 工学部 情報工学科
Dept. of Computer, Information and Sciences, Tokyo University of Agriculture and Technology
toyoda@fairy.ei.tuat.ac.jp
†2 東京農工大学大学院 工学府 情報工学専攻
Dept. of Computer and Information Sciences, Graduate School of Technology,
Tokyo University of Agriculture and Technology
†3 東京農工大学大学院 工学府
Dept. of Computer and Information Sciences, Tokyo University of Agriculture and Technology

2.1 原始モンテカルロ法

モンテカルロ法をコンピュータ囲碁に適用することを考えた場合，最も単純な方法は各合法手に対して同数のプレイアウトを行い，一番勝率の高い手を着手とすることである．例えば，図 1 に示すように各合法手に対して 100 回プレイアウトが行われたとすると，最も勝率の高い手 b が着手となる．

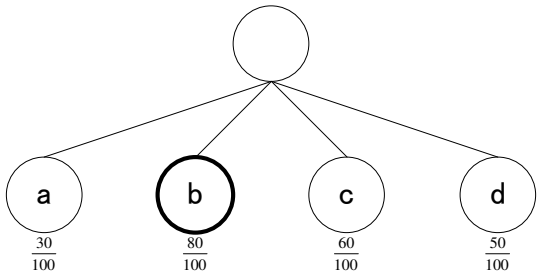

図 1 原始モンテカルロ法の着手例

2.2 UCB1 アルゴリズム

原始モンテカルロ法では，各合法手に対して同じ回数のプレイアウトを行っていた．一定時間内に行えるプレイアウト数は限られているため，これはあまり効率的であるとは言えない．シミュレーションを効率化するためには，有望そうな手に多くのプレイアウトを割り当てることが必要となってくる．そこで考えられたのが UCB1 値を用いてシミュレーションを行う手法である．UCB1 値は式 (1) で定義される（初期値として∞が与えられる）．

$$UCB1(i) = \bar{X}_i + c\sqrt{\frac{\log n}{s_i}} \tag{1}$$

$\bar{X}_i$ は着手 i の勝率，s_i は着手 i の総プレイアウト数，n は兄弟ノードの総プレイアウト数，c は定数を表す．式 (1) からわかるように,「勝率」と「不確定性（プレイアウト数が少ないと高くなる）」の総和によって UCB1 値は決まる．また，c の値を小さくすると勝率の良い手ばかりシミュレーションを行い，c の値を大きくすると原始モンテカルロ法と同じような動きになる．UCB1 値を用いてシミュレーションを行うことで，勝率の高そうな手とそうでない手の探索頻度をうまく制御することが可能となる．

2.3 UCT アルゴリズム

UCB1 値を再帰的に用いて木構造に適応させたものが UCT アルゴリズムである．これは CrazyStone や MoGo などトップクラスのプログラムに使用されている．シミュレーションを元に探索木が非対称的に生長していくので，良い手をより深く探索することができる．また，UCT アルゴリズムはゲームの知識を必要としないため，囲碁以外の評価関数の作りにくいゲームにも応用されている[5]．UCT アルゴリズムの簡単な流れを次に示す．

（1） ルートノードから展開
（2） UCB1 値の最も高い子ノードへ進んでいき，末端ノードへ．このときノードのプレイアウト数が閾値を超えているならば展開，そうでないならばプレイアウトしてホーム
（3） (2) へ戻る
（4） 時間がきたらホームして勝率の最も高い子ノードを着手とする

ただし，展開とはそのノードから派生する子ノードを全て生成すること，ホームとは各ノードの UCB1 値を更新しルートノードに戻ることを言う

3. シミュレーション結果の継承を用いたモンテカルロ法の提案

本章では，シミュレーション結果の継承を用いたモンテカルロ法を提案し，提案手法の適用方法について説明する．

3.1 提案手法の概要

原始モンテカルロ法，UCB1 アルゴリズム，UCT アルゴリズムにおいて通常は初期値として各ノードにプレイアウト数，勝数がそれぞれ 0 として与えられている．この初期値を 2 手前のシミュレーション結果を元に与えることによって，アルゴリズムを効率化できると考えた（好悪の判断に使われるシミュレーションの時間を節約することができる）．囲碁は将棋やオセロなどと比べ，着手の評価が一手進んでも変化しにくいという特徴がある（将棋では駒の利きや取り合い，オセロでは石の反転などで局面の状態が変化しやすいため，2 手前の好手が現在においても好手であることは少ない）．そのため 2 手前のシミュレーション結果の継承は有効であると考えた．

3.2 原始モンテカルロ法における継承方法

原始モンテカルロ法において，2 手前のシミュレーション結果を継承することを考えた場合図 2 のようになる．

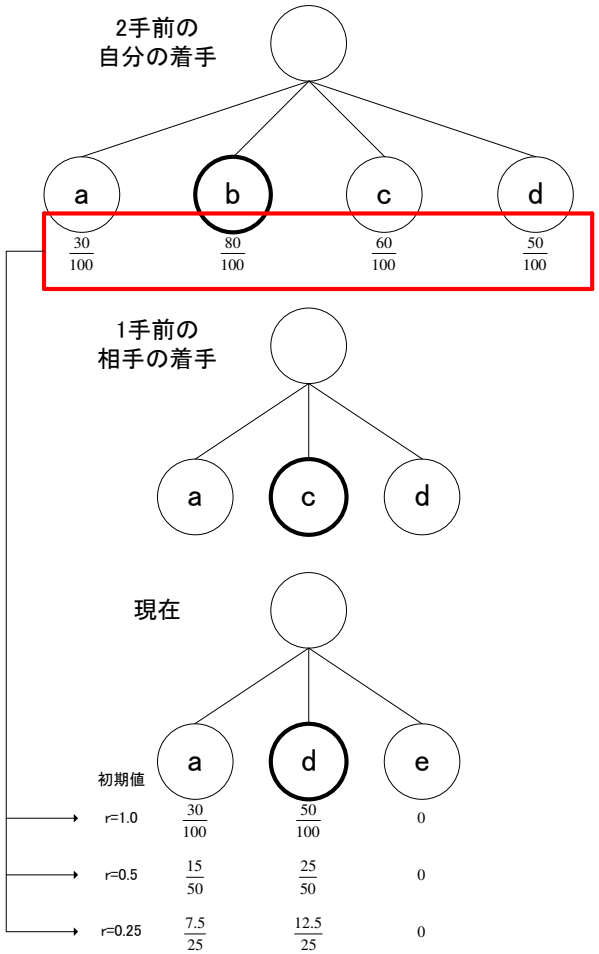

図 2 原始モンテカルロ法における継承方法

例えば，図 2 に示すように 2 手前の局面において，a，b，c，d という四つの合法手が存在していたとする．原始モンテカルロ法によるシミュレーションを行った結果，b が着手となったとする．相手の手番になり c が着手されたとすると，残る合法手は a と d である．しかし囲碁において，1 手前の相手の着手が石を取る手だったとき新たに合法手が生まれる場合がある．新たに生まれた合法手を e とする．このとき，a と d において，2 手前のシミュレーション結果が存在するため，2 手前のシミュレーション結果（プレイアウト数，勝数）を初期値として継承することができる．e は 2 手前のシミュレーション結果が存在しないため，通常通り初期値としてプレイアウト数，勝数をそれぞれ 0 と設定する．こうしてそれぞれの合法手に対して初期値を与え，原始モンテカルロ法によるシミュレーションを行い着手を決定する．

しかしモンテカルロ法をコンピュータ囲碁に適用したとき，着手の決定に勝率を用いるため，2 手前のシミュレーション結果を初期値としてそのまま継承すると，ある着手が現在の局面のシミュレーションで全勝したとしても，初期値で高い勝率を持つ着手を逆転することができないという事態が発生してしまう．そこで 2 手前のシミュレーション結果を割引いて継承することで，勝率の逆転が起こるようにする．この 2 手前のシミュレーション結果を割引く割合を本論文では割引率 r（$0 \leqq r \leqq 1$）と定義する．つまり，現在の局面において与えられる初期値は式 (2)(3) で表すことができる．

$$s_i = rs'_i \tag{2}$$
$$w_i = rw'_i \tag{3}$$

s_i は着手 i のプレイアウト数，s'_i は 2 手前の着手 i のプレイアウト数，w_i は着手 i の勝数，w'_i は 2 手前の着手 i の勝数を表す．

3.3 UCB1 アルゴリズムにおける継承方法

UCB1 アルゴリズムにおいて，2 手前のシミュレーション結果を継承することを考えた場合，原始モンテカルロ法と変わらない．式 (2)(3) で継承されるプレイアウト数，勝数を求め，初期の UCB1 値を求めればよい．ただし，初期の UCB1 値を求める際に兄弟ノードの総プレイアウト数が必要となるため，現在の局面における各合法手の継承したプレイアウト数をそれぞれ足して兄弟ノードの総プレイアウト数を求めることが必要となる．また，2 手前のシミュレーション結果が存在しない場合は初期の UCB1 値に∞を与える．

3.4 UCT アルゴリズムにおける継承方法

UCT アルゴリズムにおいて，2 手前のシミュレーション結果を継承することを考えた場合，いくつかの手法が考えられる．本論文では，祖父ノードのシミュレーション結果をノードを展開したときの初期値として用いた．

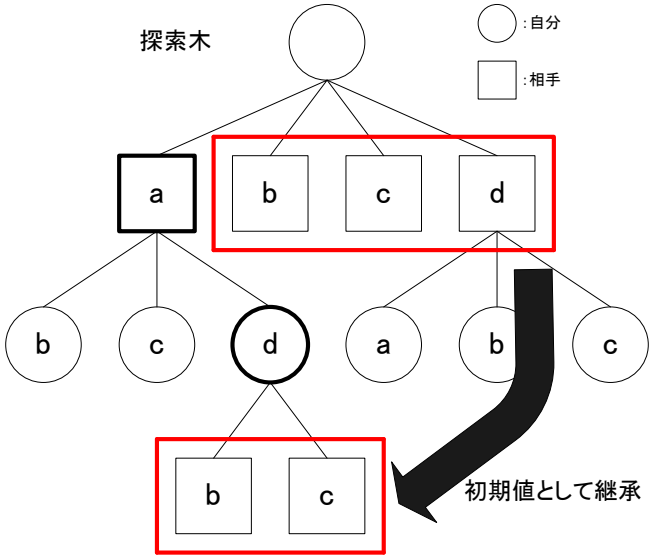

図 3 UCT アルゴリズムにおける継承方法

例えば，図 3 に示すような探索木ができたとする．このとき各ノードはそれぞれ，プレイアウト数，勝数，UCB 値などを保持している．いまルートノードから a，d と進み子ノードを展開するとしたとき，通常ならば UCB 値に∞を与えるが，祖父ノードが存在する場合は座標が一致する祖父ノード（の兄弟ノード）の情報を初期値とする．このとき継承されるプレイアウト数，勝数は割引率 r による．また，祖父ノード（の兄弟ノード）の中に座標が一致するものがなければ初期値として UCB 値に∞を与える．

ただし，祖父ノードのプレイアウト数を単純に割引いて継承すると，閾値（そのノードのプレイアウト数が閾値を超えたら子ノードを展開する）を超えたプレイアウト数が初期値として与えられてしまい，シミュレーションが行われないままノードの展開がされてしまう可能性がある（祖父ノードのプレイアウト数は最低でも閾値の 2 倍である）．そのため継承されるプレイアウト数は閾値を超えないようにしなければならない．継承されるプレイアウト数を式 (4) と定めた．

$$s_i = rThN\frac{s'_i}{n} \tag{4}$$

Th は閾値，N は着手 i の兄弟ノード数，s'_i は着手 i と座標が一致する祖父ノード（の兄弟ノード）のプレイアウト数，n は祖父ノードの兄弟ノードの総プレイアウト数を表す．

式 (4) により，祖父ノード世代におけるプレイアウト数の比を変えずに継承することができる．着手 i の兄弟ノード数をかけているのは，継承される値が小さくなりすぎないようにするためである．また，式 (4) で得たプレイアウト数を元に継承される勝数を計算する（勝率を覚えておき，プレイアウト数にかける）．実際の継承の例を図 4 に示す．

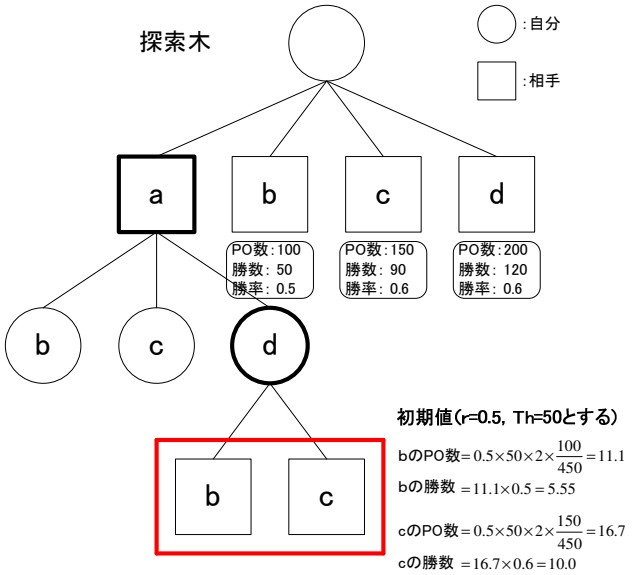

図 4 シミュレーション結果の継承を用いた UCT アルゴリズムにおける初期値の与え方

Vol.2010-GI-23 No.2
2010/3/8

4. 実　　験

本章では，シミュレーション結果の継承を用いたモンテカルロ法の性能を評価するための実験結果を示す．

4.1 実験概要

3 章で提案した各手法の性能を評価する．原始モンテカルロ法，UCB1 アルゴリズム，UCT アルゴリズムに提案手法を適用し，適用していないプログラムとの比較実験を行う．前者のプログラムは割引率 r を変化させて実験を行った．実験環境を次に示す．

- 9 路盤（コミは 7 目半）
- 特に断りがない限り思考時間は一手 5 秒（一手につき約 15000 回プレイアウト）
- 対局数は先後入れ替えで計 600 回

4.2 原始モンテカルロ法の対局実験

着手決定に原始モンテカルロ法を用いたプログラムと原始モンテカルロ法に提案手法を適用したプログラムを対局させた．実験結果を表 1 に，グラフにまとめたものを図 5 に示す．

表 1 原始モンテカルロ法の対局実験結果

r	継承を用いた原始モンテカルロ法	原始モンテカルロ法	提案手法の勝率
1.0	95 勝（黒 25 勝：白 70 勝）	505 勝（黒 230 勝：白 275 勝）	0.158
0.9	189 勝（黒 62 勝：白 127 勝）	411 勝（黒 173 勝：白 238 勝）	0.315
0.8	248 勝（黒 104 勝：白 144 勝）	352 勝（黒 156 勝：白 196 勝）	0.413
0.7	304 勝（黒 124 勝：白 180 勝）	295 勝（黒 119 勝：白 176 勝）	0.507
0.6	342 勝（黒 139 勝：白 203 勝）	258 勝（黒 97 勝：白 161 勝）	0.570
0.5	354 勝（黒 148 勝：白 206 勝）	244 勝（黒 92 勝：白 152 勝）	0.590
0.25	375 勝（黒 156 勝：白 219 勝）	225 勝（黒 81 勝：白 144 勝）	0.625
0.125	332 勝（黒 151 勝：白 181 勝）	267 勝（黒 118 勝：白 149 勝）	0.554

図 5 ではバーの上部が 95%信頼区間における上限値を表しており，下部が下限値を表している．つまり，$r \leqq 0.6$ のとき有意に勝ち越していることになる．また，$r = 0.25$ のとき最大の勝率 0.625 を得ている．r の値が大きいときは継承する情報が多すぎて，勝率が低くなっていると考えられる．例えば 2 手前の局面においては好手だったものの現在の局面においては悪手である着手があったとする．このとき継承されたプレイアウト数，勝数が大きいと新たにシミュレーションを行っても勝率の逆転が起きなくなってしまい，結果悪手が選択されることとなる．逆に r の値が小さくなると，継承される情報が少ないため通常の原始モンテカルロ法に近い着手となり，勝率が 0.5 に収束していく．

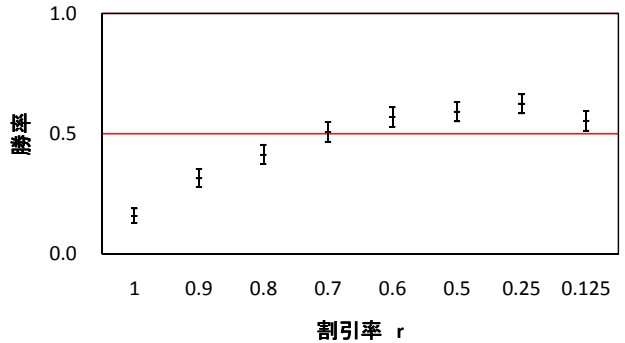

図 5 シミュレーション結果の継承を用いた原始モンテカルロ法の勝率の信頼区間

4.3 UCB1 アルゴリズムの対局実験

着手決定に UCB1 アルゴリズムを用いたプログラムと UCB1 アルゴリズムに提案手法を適用したプログラムを対局させた．実験結果を表 2 に，グラフにまとめたものを図 6 に示す．

表 2 UCB1 アルゴリズムの対局実験結果

r	継承を用いた UCB1 アルゴリズム	UCB1 アルゴリズム	提案手法の勝率
1.0	36 勝（黒 11 勝：白 25 勝）	564 勝（黒 275 勝：白 289 勝）	0.060
0.9	106 勝（黒 30 勝：白 76 勝）	494 勝（黒 224 勝：白 270 勝）	0.177
0.8	254 勝（黒 103 勝：白 151 勝）	346 勝（黒 149 勝：白 197 勝）	0.423
0.7	354 勝（黒 164 勝：白 190 勝）	244 勝（黒 109 勝：白 135 勝）	0.592
0.6	346 勝（黒 156 勝：白 190 勝）	253 勝（黒 110 勝：白 143 勝）	0.578
0.5	353 勝（黒 164 勝：白 189 勝）	247 勝（黒 111 勝：白 136 勝）	0.588
0.25	340 勝（黒 140 勝：白 200 勝）	260 勝（黒 100 勝：白 160 勝）	0.567
0.125	337 勝（黒 142 勝：白 195 勝）	262 勝（黒 104 勝：白 158 勝）	0.563

図 6 より，$r \leqq 0.7$ のとき有意に勝ち越していることがわかる．また，$r = 0.7$ のときに最大の勝率 0.592 を得ている．原始モンテカルロ法に提案手法を適用したときと比べ，r の値が大きいときの勝率が低くなっている．これは UCB1 値を求めるときに勝率を用いるため，継承する情報量が大きいと 2 手前に勝率が良かった手を原始モンテカルロ法のときと比べて多くプレイアウトしてしまうからだと考えられる（つまり原始モンテカルロ法のときと比べ勝率の逆転が起きにくくなっている）．r の値が小さくなると，原始モンテカルロ法のときと同様に，継承される情報が少ないため勝率が 0.5 に収束していく．

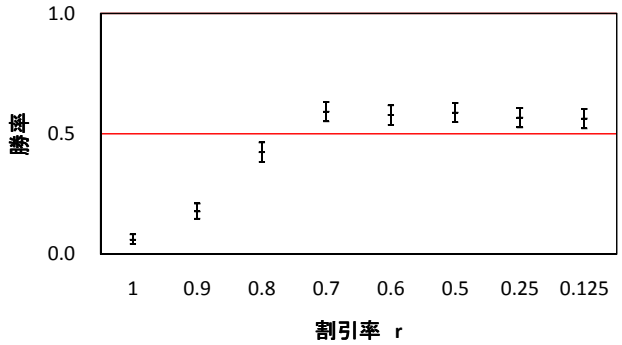

図 6 シミュレーション結果の継承を用いた UCB1 アルゴリズムの勝率の信頼区間

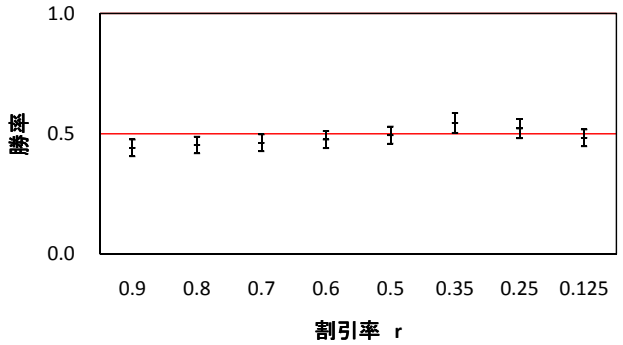

図 7 シミュレーション結果の継承を用いた UCT アルゴリズムの勝率の信頼区間

4.4 UCT アルゴリズムの対局実験

着手決定に UCT アルゴリズムを用いたプログラムと UCT アルゴリズムに提案手法を適用したプログラムを対局させた．UCT アルゴリズムの対局実験は一手 10 秒で行い，ノードを展開する閾値を 10 とした．実験結果を表 3 に，グラフにまとめたものを図 7 に示す．また，UCT アルゴリズムに提案手法を適用した場合と通常の UCT アルゴリズムの探索ノード数を調べた．手番は黒番とし，手数（49 手まで）の平均（計 10 回）を取った．図 8 に示す．

表 3 UCT アルゴリズムの対局実験結果（思考時間一手 10 秒，閾値 10）

r	継承を用いた UCT アルゴリズム	UCT アルゴリズム	提案手法の勝率
0.9	265 勝（黒 120 勝：白 145 勝）	335 勝（黒 155 勝：白 145 勝）	0.441
0.8	272 勝（黒 122 勝：白 150 勝）	328 勝（黒 150 勝：白 150 勝）	0.453
0.7	277 勝（黒 115 勝：白 162 勝）	323 勝（黒 138 勝：白 162 勝）	0.462
0.6	286 勝（黒 131 勝：白 155 勝）	314 勝（黒 145 勝：白 155 勝）	0.476
0.5	296 勝（黒 130 勝：白 166 勝）	304 勝（黒 134 勝：白 166 勝）	0.494
0.35	327 勝（黒 145 勝：白 182 勝）	273 勝（黒 118 勝：白 155 勝）	0.545
0.25	313 勝（黒 126 勝：白 187 勝）	287 勝（黒 113 勝：白 174 勝）	0.522
0.125	290 勝（黒 110 勝：白 180 勝）	310 勝（黒 120 勝：白 180 勝）	0.483

図 7 より，$r = 0.35$ のとき有意に勝ち越し，最大の勝率 0.545 を得ていることがわかる．UCT アルゴリズムは木が生長することで，悪そうな手を選択しないようになるので，前節で述べたような勝率の逆転が起きやすくなっている．そのため r の値が大きくとも今回の継承方法ではある程度の勝率を保つことができたと考えられる．

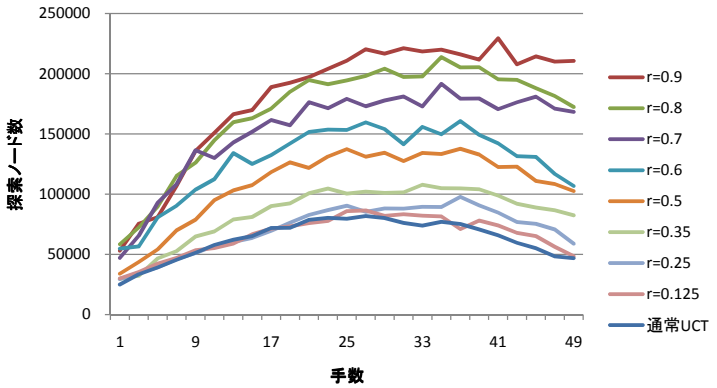

図 8 UCT アルゴリズムにおける探索ノード数の変化

図 8 より，通常の UCT アルゴリズムの探索ノード数が一番少なく，提案手法を適用した UCT アルゴリズムは r の値が大きくなるにつれ探索ノード数が増えていることがわかる．これは r の値が大きいと継承されるプレイアウト数が多くなり，ノードが展開されやすくなるためである．しかし，実際のプレイアウトがあまりされずにノードが展開されてしまうと，悪手であっても深く読んでしまう可能性があり．効率的でない．そのため，継承されるプレイアウト数と実際に行われるプレイアウト数のバランスを取ることが重要である．実験では $r = 0.35$ のときに最適なプレイアウト数を継承することができたと言える．

5. おわりに

本論文ではモンテカルロ法におけるシミュレーション結果の継承という新しい概念を提案し，以前行われたシミュレーション結果を再利用することでアルゴリズムの効率化を図った．原始モンテカルロ法，UCB1 アルゴリズム，UCT アルゴリズムに提案手法を適用し，適用していないプログラムとの比較実験を行ったところ，原始モンテカルロ法に 62.5（± 3.9）%，UCB1 アルゴリズムに 59.2（± 4.0）%，UCT アルゴリズムに 54.5（± 4.1）%の勝率を得た．これらの結果より，モンテカルロ法におけるシミュレーション結果の継承という新しい概念の有効性を示すことができた．

今後の展望として，UCT アルゴリズムにおける他の継承方法の適用が挙げられる．本論文では，UCT アルゴリズムにおいて，祖父ノードのシミュレーション結果をノードを展開したときの初期値として用いたが，他にも 2 手前の UCT 木全体を継承するなどさまざまな継承方法が考えられる．今後は UCT アルゴリズムにおいてさまざまな継承方法を実装し，実験を行うつもりである．

参 考 文 献

1) L.Kocsis and C.Szepesvari : Bandit based monte-carlo planning, European Conference on Machine Learning, pp282–293 (2006).
2) R.Coulom : Computing elo ratings of move patterns in the game of Go, In Computer Games Workshop (2007).
3) S.Gelly, Y.Wang, R.Munos, and O.Teytaud : Modification of UCT with Pattens in Monte-Carlo Go, RR-6062-INRIA, pp1–19 (2006).
4) P.Auer, N.Cesa-Bianchi, and P.Fischer : Finite-time analysis of the multiarmed bandit problem, Machine Learning, Vol47, pp235–256 (2002).
5) 佐々木健太，小谷善行 : ブロックスデュオにおけるモンテカルロ木探索，第 14 回ゲームプログラミングワークショップ，pp103–106 (2009).