Vol.2012-GI-27 No.10
2012/3/2

モンテカルロ探索における動的閾値調整によるスコア利用

秋山晴彦[†]　　小谷善行[††]

ゲームにおけるモンテカルロベースの着手選択手法において，勝率が偏る局面で適切な着手選択ができない問題がある．この問題の対処策として，スコアから擬似勝率を計算する手法を提案する．勝敗を，0 を閾値としてスコアから算出される値とみなし，この閾値を動的調整する．勝敗を評価とする通常のモンテカルロ手法では勝率を，スコアを評価とするモンテカルロ手法では平均や最大スコアを最大化するが，提案手法では局面同士の差がつく最低限の閾値変更を行った後の勝率を最大化する．実験の結果，本手法を導入した UCB は通常の UCB に対して勝率 0.58 で勝ち越した．また深さ 2 のαβ探索との対戦の勝率が通常 UCB の 0.37 から 0.48 に向上し，本手法の有効性を確認した．

Dynamic Threshold Adjustment for using Score in Monte-Carlo Based Game Player

Haruhiko Akiyama[†]　and　Yoshiyuki Kotani[††]

Monte-Carlo based methods in games have a problem that they cannot select the best move in the position with biased winning percentage. In order to solve this problem, we propose the method to calculate the pseudo winning percentage from a score. We regarded the victory and the defeat as the value computed from a score when a threshold value is 0, and dynamic adjustment of this threshold value is carried out. The normal Monte-Carlo method maximizes the winning percentage, and the Monte-Carlo method which uses score of the game for evaluation maximizes the average score. In contrast, the winning percentage with the minimum change of the threshold value that is comparable each other is maximized in the proposal method. As a result, the winning percentage of UCB player with this method against normal UCB player was 0.58, and the winning percentage against αβ search player of the depth 2 was also improved by 0.48 from 0.37.

1. 概要

乱数シミュレーションの結果から確率的に局面の良さを推定するモンテカルロ法，これを改良した UCB[13]や UCT[2]などの探索手法は，囲碁などの多くのゲーム[1][15][10][11]で成果を上げている．対戦ゲームにおけるモンテカルロ法では，基本的に局面の良さの判断基準には勝率を用いるのが最適である．しかし，局面によっては，子局面のシミュレーション結果が勝利や敗北に極端に偏り，勝率差を基準とした最良優先探索や着手選択が困難となる場合がある[7]-[9]．二人対戦ゲームのコリドール[3][11]では，この問題が特に顕著である[7]．この問題を，ゲームの最終スコアを利用して解消する手法として，勝敗をゲームの最終スコアから 0 を閾値として算出される値と見て，この閾値を動的に調整する手法を提案する．スコアを評価とするモンテカルロ法では平均スコアを最大化する局面を最善と推定するが，提案手法では局面同士の差のつく最低限のスコアを得られる確率の最大化を行う．閾値を適切に調整することで，深さ 2 のαβ探索との対戦勝率が 0.37 から 0.48 に向上したほか，対 UCB で勝率が 0.58 となり，勝ち越した．

2. コリドールとモンテカルロ法

2.1 コリドール（Quoridor）のルールと概要

コリドール[3]は 9×9 マスの盤面と各プレイヤに一つずつの「駒（pawn)」および 10 枚ずつの「板（fence)」を用いて行う二人対戦ゲームである．各プレイヤは自分の手番において，自分の駒を，相手の駒がなく板で遮られていない四近傍のマスに移動するか，手持ちの板を既存の板と重ならずに 2 マスにかかるように盤面に設置するかのいずれか一方を一度だけ行う．これを交互に繰り返し，自分の駒を相手側の端列のマスに移動することに成功したプレイヤの勝利となる．コリドールの初期盤面を図 1 に示す．また，お互いに 10 枚ずつの板を打ち終えた盤面の例を図 2 に示す．

図 1 では e9 にある「x」が先手の駒，e1 にある「o」が後手の駒を表す．お互いの駒は，初期局面で盤上のこの位置に置かれており，以後，この盤面から取り除かれることはない．盤面の上にある「|」は後手の持っている板，下にあるのは先手の持っている板を表す．これらの板は，初期状態では各プレイヤの手持ちとしてそれぞれ 10 枚ずつ盤外に保持する．2 マス分の長さを持ち，盤面からはみ出さずマスの位置に沿うように，マスとマスの間のラインに設置する．この板は一度置いたら取り除くことはできない．また，互いの板は，手持ちの時点では区別するが，盤上では区別はない．

[†] 東京農工大学工学府情報工学専攻
Department of Computer and Information Sciences, Tokyo University of Agriculture and Technology
[††] 東京農工大学工学研究院
Institute of Engineering, Tokyo University of Agriculture and Technology

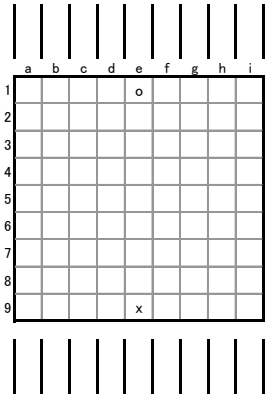

図 1. コリドールの初期局面

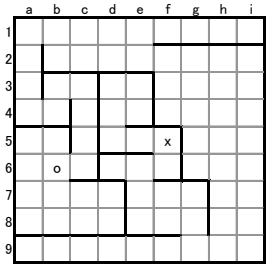

図 2. お互いに板を設置し終えた局面

[コリドールの基本ルール]
- 手番にいずれかを行う：「駒を四近傍に一歩動かす」「板を 1 枚設置する」
- 駒は盤外に出ることはできない. また, 板を超えて移動することはできない. 相手の駒と隣り合っている場合，相手の駒のあるマスに入ることはできず，代わりに相手のマスを超えた 1 歩先のマスに移動できる（ジャンプ）.
- 板は盤上の板に重なるようには置けない. 盤外に出るようには置けない. 板によりいずれかの駒のすべてのゴールへの道を完全封鎖してはいけない.

これらより，駒の移動によりゴールを目指しつつ，板の設置により相手の駒のゴールを妨害する，ということが，基本のゲームの流れとなる．四近傍にいる相手の背後に板がある場合のジャンプルールなどの詳細は公式ルールページ[3]を参照されたい.

コリドールの着手選択アルゴリズムの研究はあまり行われていないが，既存研究としては囲碁でモンテカルロ法が活躍する以前に行われた，評価関数と探索による α β 法の研究が存在する[4]-[6]．しかし，駒の種類なしの評価関数の難しさもあってか，あまり成果は出ていない.

2.2 モンテカルロ法

多数のシミュレーションにより確率的に推定を行うモンテカルロ法は，バックギャモンなどのランダム性を含むゲームにおいては早くから成功を収めていた[1]．局面の変化に確率が絡まない確定ゲームである囲碁におけるモンテカルロ法は，1993 年に Brügmann がモンテカルロ碁を提案した[12]時点では，山登り法をベースとして考えられた，シンプルなシミュレーション手法であった．このときの性能は非常に低く，このような確率的な手法が確定ゲームにおいて有効であるとは考えられていなかった.

最も単純なモンテカルロベース手法を，原始モンテカルロ法と呼ぶ．原始モンテカルロ法では，現在局面の各子局面に関して，終局までの乱数着手によるゲームをそれぞれ多数回行う．それぞれの乱数による 1 ゲームをプレイアウトと呼ぶ．一定回数のプレイアウトを行った後，最も高い勝率を持つ局面を最善と推定し，この局面を導く手を最善手として選択する．2002 年に，原始モンテカルロ法を，正確には有限時間の多腕バンディット問題を改善する手法として，シミュレーション中に勝率とプレイアウト回数による確からしさを含めて最善と推定される局面を計算し，より良いと推定される局面により多くプレイアウトする手法が，Auer らによって提案された[13]．この判断基準となる値は，UCB (Upper Confidence Bound) と名付けられている．ノード i の UCB 値は以下の式で求められる.

$$UCB_i = x_i + \sqrt{\frac{C*n_{all}}{n_i}}$$

ここで，x_i はノード i の勝率，n_i はノード i のプレイアウト回数，n_{all} はノード i の全兄弟ノードのプレイアウト回数の合計，C は勝率とプレイアウト回数のスケール調整用の定数である．C=2.0 が基本的な最適値とされている．この手法により，最善手が指数的に多くプレイアウトされることが証明されている.

本稿では，この UCB を基本のモンテカルロ着手選択手法として用いる.

2.3 モンテカルロ法の問題点

モンテカルロ着手選択は実装が容易であり，人間の知識や棋譜がない問題にも適用できるが，シミュレーション結果が偏る局面ではうまくいかない問題がある[7]．具体的にはすべての手の勝率が高すぎるなどの局面ではどの手がより高いかをうまく比較できない．モンテカルロ着手選択では，子局面の勝率が 5 割程度であり，最高勝率が一つ存在することが勝率比較の上で理想であると考えられるが，勝率が 1.0 や 0.0 付近に偏った場合など，最高勝率の手が多数ある場合，どれがより良いのかを選択することが不可能となる．最悪の場合，すべての子局面の勝率が等しくなることも考えられる．基本的に最高勝率の手からランダムに選択するため，この場合ランダム着手と同様になってしまう．勝率が等しく最高である手，つまり可能手は違っても同等の効果を及ぼす手が複数あることは考えられるが，あまりに多くの手がこのような状態になった場合，それらの間に実際の着手の良さの違いがないとは考え辛い．すなわち，本来は異なる性質を持つ手であるにも関わらず，ランダムシミュレーションの勝率に差異が現れていないだけであると考えられる．このようなとき，スコアをうまく用いることで，より良い着手を選択することが可能になると考えられる.

3. 勝率計算閾値の動的調整による擬似勝率

3.1 勝敗とスコアの解釈

通常，ゲームにおけるモンテカルロ法では，スコアではなく勝率を最大化する局面を最善とする．これは，ゲームの目的が勝利であるため，とても高いスコアを得られる可能性がわずかにある手より，勝率のより高い手のほうが望ましいからである．すなわち，スコアと勝率では目的が異なり，平均スコアを最大化することは勝利を目的とするゲームでは望ましくない可能性がある．本節では，スコアの値を最大化するのではなく，より高いスコアが得られる確率も含めた，勝率とスコアの合成方法を示す．囲碁の目数差のように，より勝っているほどプラス，負けているほどマイナスとなる値をスコアと考えると，通常用いる勝敗（勝利＝1，敗北＝0）の値は，±0 を閾値として

勝敗 ＝ 1（スコア>0）
0（スコア<0）

として算出される値であると考えられる．図 3 は，ある局面から，可能手 A と可能手 B を着手することにより得られる子局面の，10 回のシミュレーション結果のスコアである．閾値を 0 として閾値を上回れば勝利，下回れば敗北となっている．可能手 A は勝率 0.7，可能手 B は勝率 1.0 と算出され，このような場合，勝率を比較すればよい．

しかし，多くの可能手について，着手後の局面の勝率が可能手 B のようになった場合，またはすべての手が敗北側になった場合は，勝敗の値を見て勝率を算出しても，局面ごとの差がつかず，より良い手を判別することはできない．また，単純にスコアにより比較すると，より平均スコアが高い手を最善と判断することになるが，これはスコアの期待値である．極端に高いスコアの手があれば期待値は高くなるが，その高いスコアが「得られる確率」が高いとは限らない．スコアが報酬となるようなゲームでは期待値を最大化することは適切であると考えられるが，勝利を目的とするゲームでは目的と異なると考えられる．

そこで，勝敗を算出する際に用いた閾値を 0 から変更することを考える．すなわち，

勝敗 ＝ 1（スコア＞th）
0（スコア＜th）

として，閾値 th を適切に調整する手法を提案する．この閾値をプラス側またはマイナス側に適切に動かすことで，勝率に応じて要求する条件を高くまたは低く設定することができ，ある範囲のスコアを得られる確率を最大化する手を最善手と推定できる．図 4 は図 3 の閾値を 0 からプラス側に動かした例である．

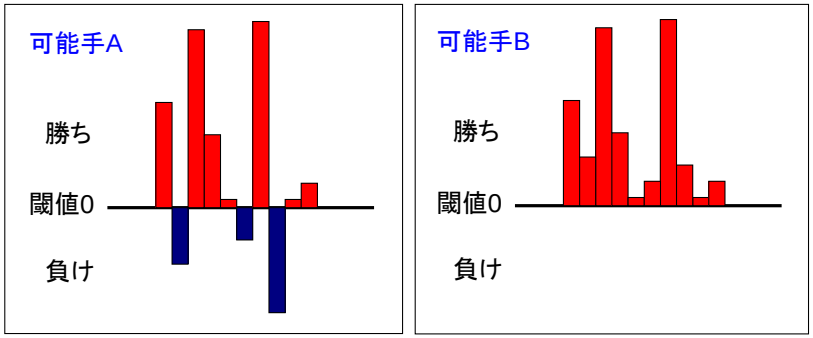

図 3. 各子局面で得られたスコアと勝敗の対応

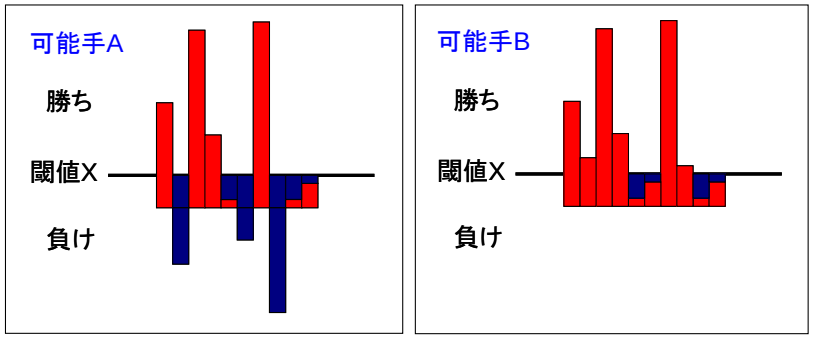

図 4. 前図からの勝敗判定閾値の変更

このようにプラス側に動かすと，勝利条件がより厳しくなったものと考えられる．この手法では，今のシミュレーション結果から計算だけで直接適切な値を算出できるため，適用のためのオーバーヘッドはわずかである．

閾値 X として勝敗を判定し，勝率を求めることは，閾値 X 以上のスコアを得られる確率と等しい．これは，スコアの平均値を最大化する，すなわちスコアの期待値を最大化する手法と比べて，よりそのスコアが得られる確率を重視するという点で優れていると考えられる．例えば，1/10 の確率で 50 のスコアが得られるが 9/10 の確率で-1 のスコアが得られる場合と，10/10 で 2 のスコアが得られる場合を考えると，期待値は明らかに前者が高いが，目的が勝利，すなわち「プラスのスコアを得ること」ならば，後者を選択するべきである．このように，閾値を最低限調整したときの擬似的な勝率を最大化することで，勝率とスコアを合成する．

3.2 動的閾値調整

前節のように，勝率による推定が適切に行えない局面で，スコアから閾値を変更して算出した擬似的勝率を用いることを考える．図 3 から図 4 のように，閾値を大幅に変更するのではなく，実際には勝率を計算した後に「閾値を調整すべきであるか」を判定し，調整すべきであれば最低限だけ閾値を変更し，再度勝率を算出する．単に記録しておいたスコアから計算を行うだけであるため，ランダムシミュレーションの計算量と比較すると，ある程度再計算を繰り返しても計算時間には問題はないと考えられる．

再計算を行う条件としては，勝率が高い場合および低い場合という条件が考えられるが，本手法では勝率の差別化を重視し，最大スコアの手が複数あった場合に再計算を行うものとする．ここで，多くの子局面の勝率が 1.0 の場合や 0.0 の場合には，最大スコアが完全に等しいときに問題であると考えれば良いが，そうでない場合でも，実際には勝率にはわずかな差があるが乱数から生まれる誤差に過ぎないという可能性が考えられる．そこで，最大勝率からの差 d を誤差として考え，d 以内に収まる場合は最大勝率の局面に含めることを考える．各勝率帯の局面数と誤差 d の様子を図 5.に示す．実装としては，ある局面の勝率について，d を用いて

if(勝率＞最大勝率+d) 最大勝率のリストをリセット，最大勝率を勝率で更新
else if(勝率＞最大勝率-d) 最大勝率のリストに勝率の局面を追加

とする．最大勝率の局面から d 以上大きい勝率が得られれば，誤差の範囲を超えたと考え，これまでに得た最大勝率の局面を破棄し，この手を最大勝率とする．また最大勝率から d 以内であるが最大勝率より小さい勝率が得られた場合，誤差の範囲内で，最大勝率に含まれると考え，この局面を最大勝率の局面のひとつとして記憶する．これによって得られた最大勝率の手が n 個以上のとき，勝率＞0.5 であれば閾値をプラスに，勝率＜0.5 であれば閾値をマイナスに最小値変動させる．

今回は対象がコリドールであるため，スコアを

スコア＝（相手の最短歩数）－（自分の最短歩数）

として算出する．最短歩数とは，勝利条件を満たすゴール，すなわち相手側の端列までの，駒をルール通り 1 マスずつ妨害されることなく移動し続けた場合の移動回数である．最短歩数が 0 である場合，勝利局面である．このスコアを利用することを考えると，盤面の広さから最大でも 72 より多くの差はつかないため，閾値は 1 ずつ変動させて再計算を行っていく．

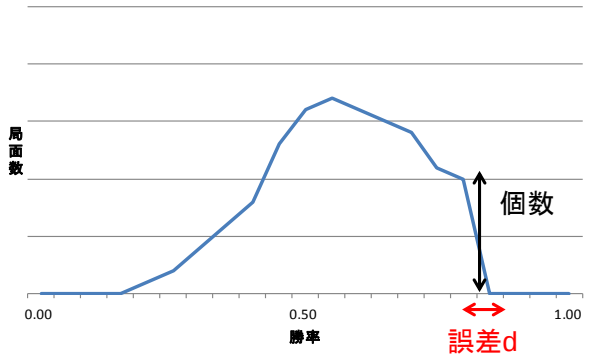

図 5. 擬似勝率再計算の適用条件

3.3 実験と考察

前節で述べたパラメータ d を変更しながら，動的閾値調整手法の性能実験を行う．今回は n=10 で固定とした．提案手法は対局における 1 手につき 50000 プレイアウトを行う UCB プレイヤを基本とし，閾値による調整を導入した．対戦相手として，同様に 50000 プレイアウトの UCB プレイヤ，および深さ 2 で前節のスコアを評価値としたαβプレイヤを用いた．ランダム性を得るために初手から 3 手ずつ，合計 6 手はランダムとし，先手番および後手番それぞれについて 500 対戦，合計 1000 対戦を行った．勝ちは 1 ポイント，負けは 0 ポイントとし，300 手を超えた試合は引き分けとして，引き分けはごく少数であるため，簡易的に両者に 0.5 ポイントを加算した．

まず，αβプレイヤとの対戦結果を示す．勝率のグラフは図 6 のようになった．

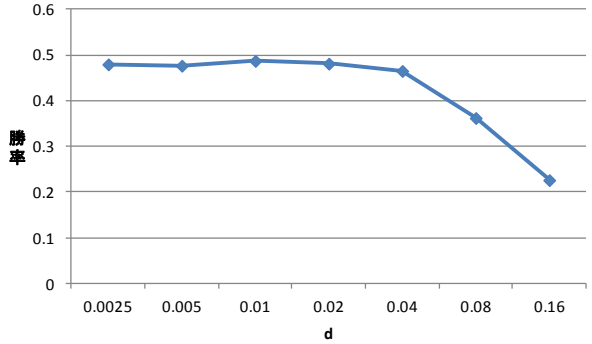

図 6. 実験結果：パラメータ d の値によるαβに対する勝率

実験した最大値である d=0.16 のとき最低勝率となり，d を小さくするほど勝率が上がり，d=0.01 のとき最高勝率となって，以後はあまり変化が見られなかった．最高勝率でも勝ち越していないが，適切なパラメータ設定で性能が向上すること，また表 1 のように，適切なパラメータでは通常の UCB プレイヤからは大きく勝率が向上していることを確認した．

表 1　実験結果：パラメータ d の値による α β に対する勝数と勝率

d	先手番勝数	後手番勝数	平均勝率
0.0025	225.5	254.0	0.480
0.005	239.5	237.0	0.477
0.01	229.5	257.5	0.487
0.02	198.0	283.0	0.481
0.04	213.0	252.0	0.465
0.08	169.0	193.0	0.362
0.16	93.0	133.0	0.226

	先手番勝数	後手番勝数	平均勝率
通常UCB	176.0	191.5	0.368

次に，通常の UCB プレイヤとの対戦実験を行った．両者ともに 50000 プレイアウトの UCB プレイヤであり，初手からお互いに 3 手ずつランダムとし，先手番および後手番それぞれについて 500 対戦，合計 1000 対戦を行った．勝率の推移は図 7 のようになった．

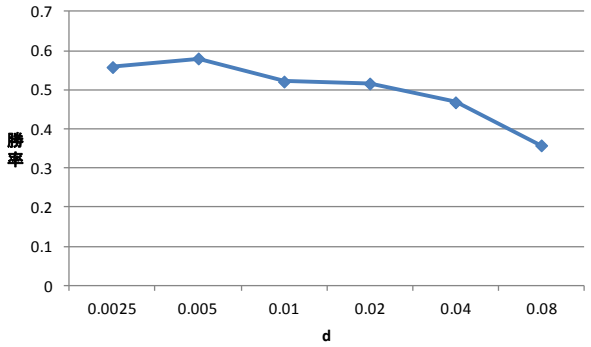

図 7.　実験結果：パラメータ d の値による通常 UCB に対する勝率

α β プレイヤとの対戦と同じく，d が最大のとき勝率が最小となり，d を小さくするほど勝率が上がっている．最適な設定では UCB プレイヤに勝ち越すことを確認した．UCB ベース同士では，表 2 のように，後手番での勝ち越しが大きくなっている．

表 2　実験結果：パラメータ d の値による UCB に対する勝数と勝率

d	先手番勝数	後手番勝数	平均勝率
0.0025	280	278	0.558
0.005	276	304	0.580
0.01	260	261	0.521
0.02	246	270	0.516
0.04	231	237	0.468
0.08	159	199	0.358

d が 0.005～0.01 程度であると勝率が高かった結果から，最高勝率とカウントする手は，誤差 1%以内と見て，ごくわずかな差しかないときだけ数えるべきであると考えられる．すなわち，勝率差をできる限り考慮し，最低限の調整を行うことが良いと考えられる．d=0.16 では勝率 1.0 と 0.84 を同等に見なすため，閾値調整を行いすぎ，勝率が悪化していると考えられる．α β プレイヤ相手の場合および UCB プレイヤとの直接対戦ともに，各パラメータ設定で同様の勝率推移を見せ，UCB プレイヤより性能が上回った．結果をまとめると，最適な設定では，深さ 2 の α β との対戦では勝率が 0.37 から 0.48 と 0.11 ポイント向上した．また，UCB との直接対決では勝率 0.58 となり勝ち越した．このことから，本手法により，UCB プレイヤの性能が改善されること，また適切なパラメータを確認した．また，深さ 2，つまり 2 手先読みを行う α β プレイヤとの対戦結果が互角であったことから，先読みのない UCB でも，勝率比較の問題が緩和されれば先読みをする手法と同等の強さを発揮できると考えられる．

4.　結論と展望

勝率とスコアの合成方法として，スコアから動的に設定した閾値により擬似的勝率を得る手法を提案し，適用するための制御パラメータを適切に設定することで，通常 UCB プレイヤより高い性能を得られることを確認した．本手法では，あらかじめ保存したスコアから工夫した勝率を計算することで，予備シミュレーションや再シミュレーションを必要とせず，直接適切に調整した値を得られる．しかし，各スコアの出た回数をそれぞれの局面ごとに保持しなければならないこと，子局面同士の最終比較に使用する方法しか示せていないことが課題である．現状，囲碁などの多くのゲームでは，UCB の改善手法として，UCB を木探索に適用した UCT が用いられる．本手法を UCT に適用する場合，UCB の場合と同様に最終比較には使用できるが，木探索の展開の補助のためにプレイアウトごとに再計算を行おうと考えると，計算コストが大きくなる．このため，ある程度のプレイアウトを行うごとに再計算する，プレイアウトを同時に多数回行って回数を減らすなどの工夫が必要となると考えられる．今回は木探索を行わない UCB プレイヤのみの適用方法を提案した．プレイアウトから得たス

コアからより良い結果を得られる確率を計算するこのような手法を UCT にも活かす方法は，今後の課題である．

参考文献

1) Gerald Tesauro and Gregory 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.
2) Levente Kocsis and Csaba Szepesvári, "Bandit based Monte-Carlo Planning", The 15th European Conference on Machine Learning (ECML), pp. 282–293, 2006.
3) Quoridor - The Amazing Maze Board Game, http://www.quoridor.net/, 2011.
4) Lisa Glendenning, "Mastering Quoridor", Bachelor Thesis, Department of Computer Science, The University of New Mexico, May 2005.
5) Peter J.C. Mertens, "A Quoridor-playing Agent", Bachelor Thesis, Department of Knowledge Engineering, Maastricht University, June 2006.
6) Quinn McDermid, Anand Patil and Touran Raguimov, "Applying Genetic Algorithms to Quoridor Game Search Trees for Next-Move Selection", CS486 Final Group Project Report, August 14, 2003.
7) 秋山晴彦, 是川空, 小谷善行, "ゲームにおけるモンテカルロ着手選択の動的勝率調整", 第 52 回プログラミング・シンポジウム予稿集, pp. 111-118, January 2011.
8) Petr Baudiš, "Balancing MCTS by Dynamically Adjusting Komi Value",ICGA Journal, In review, 2011.
9) Petr Baudiš and Jean-loup Gailly, "Pachi State of Art Open Source Go Program", the 13th Advances in Computer Games Conference (ACG13), Novenber, 2011.
10) Kazutomo Shibahara and Yoshiyuki Kotani, "Combining Final Score with Winning Percentage by Sigmoid Function in Monte-Carlo Simulations", 2008 IEEE Symposium on Computational Intelligence and Games (CIG'08), December 2008.
11) Haruhiko AKIYAMA, "Qai wins Quoridor Tournament", ICGA Journal, Volume 34, Number 1, pp.37-39, March 2011.
12) Bernd Brügmann, "Monte Carlo Go", Technical Report, Physics Department, Syracuse University, 1993.
13) Peter Auer, Nicolò Cesa-Bianchi, Paul Fischer, "Finite time analysis of the multiarmed bandit problem", Machine Learning, 47(2-3), pp. 235-256, 2002.
14) Sylvain Gelly, Yizao Wang, Rémi Munos, Olivier Teytaud, "Modification of UCT with Patterns in Monte-Carlo Go", Technical Report 6062, INRIA, 2006.
15) Sylvain Gelly, David Silver, "Combining Online and Offline Knowledge in UCT", Proceedings of the 24th International Conference of Machine Learning (ICML 2007), pp. 273–280, June 2007.
16) Haruhiko AKIYAMA, Kanako KOMIYA, Yoshiyuki KOTANI, "Nested Monte-Carlo Search with AMAF Heuristic", 2010 International Conference on Technologies and Applications of Artificial Intelligence (TAAI 2010), pp. 172-176, Nov. 2010.

付録

コリドールの「手」の記述および棋譜の表現方法

ICGA Journal で発表した[11]コリドールのユニークに判別できる「手」の表現の仕方および，ゲームの棋譜のルールについて記載する．

盤面の座標は，横軸を左端から右に向けて，a〜i で表記する．また，縦軸は上から下に向けて 1〜9 とする．この向きは一般的な表の順序であり，オセロ（日本オセロ連盟の規定）と同様である．初期の駒は図 1 のように先手が e9，後手が e1 に配置されることになる．

駒を移動する手の記述は，お互いに駒は一種類しかなくどの駒を移動したかは自明であるため，単に座標を記す．すなわち，初期局面から先手が一歩前に進む手は，単に

e8

と記載する．板を置く手に関しては，板はマスとマスの間に置くため，どの箇所に板を設置する場合にどのマスの座標を用いるかを決める必要がある．提案する記述は，長さ 2 ある板の中央部分を，マスの右下の座標に合わせるものである．例えば，図 A.2 では c5 のマスの右下の角を中心とする板を設置しているため，座標は c5 と表記する．また，板を縦向きに置いたか横向きに置いたかを，縦なら「|」，横なら「-」で現す．これは見た目通りの縦向きと横向きの線を意図している．つまり，図 8 の最後の手は

c5|

となる．棋譜は，単にこれをカンマで区切って表記する．例えばこの局面に至る棋譜は，

e8, e2, e7, e3, e6, e4, e5, e6, e4, e7, e3, e8, e8-, d8, c8-, d2-, a8-, e3|, d7|, c3|, c6-, c8, c5|

であった．これは実際にコンピュータオリンピアードで指された棋譜である[11].

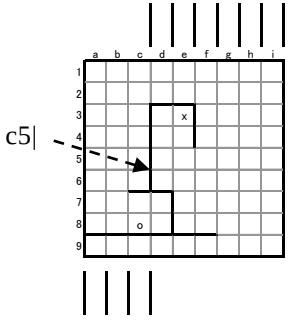

図 8. 着手の棋譜記述法