情報処理学会論文誌 Vol.55 No.11 2362–2369 (Nov. 2014)

局面の局所的な類似性を利用したモンテカルロ木探索の効率化

志水 翔[1,a)] 金子 知適[1]

受付日 2014年2月21日, 採録日 2014年9月12日

概要：囲碁は，2 人零和有限確定完全情報ゲームの 1 つであるが，min-max 探索での探索が効果的でないことが知られている．代わりに現在，囲碁プログラムの多くにモンテカルロ木探索（MCTS）が採用され，MCTS の精力的な研究が行われている．また，Rapid Action Value Estimation（RAVE）は MCTS の有力な拡張の 1 つである．本研究では，RAVE において局所的な類似性を活用することによる性能向上を提案する．局所的な類似の指標としては，着手の周囲 8 近傍の状態の一致の有無を用いる．着手の周囲 8 近傍の状態は囲碁の様々なパラメータ調整に用いられているため，RAVE での利用も効果的であると期待できる．実験では有力なプログラムの 1 つである Fuego を対象に，局所的な類似性に基づいて，プレイアウト結果を RAVE 値に反映させる際のパラメータを変更した．9，13，19 路盤ではパラメータ次第で，プレイアウト数を揃えた条件で，オリジナルの Fuego と比較して，勝率が有意に向上した．類似性の計算コストを加味するために 1 手あたりの思考時間を揃えた対戦でも，提案手法はほぼ五分の勝率となった．総合して，着手の周囲 8 近傍の状態の一致の有無に着目する提案手法は，今後も研究に値する有望な手法であると考えられる．

キーワード：囲碁，モンテカルロ木探索，RAVE

Optimization for Monte-carlo Tree Search with Similality Local Similarity

Sho Shimizu[1,a)] Tomoyuki Kaneko[1]

Received: February 21, 2014, Accepted: September 12, 2014

Abstract: While Go is a two-person zero-sum game with perfect information, Minmax searches have a little effect on computer Go. Monte carlo tree search (MCTS) has been used in many Go programs and intensively researched. RAVE is a major enhancement to MCTS and widely used in Go programs. In this paper, we present a method to improve RAVE by taking local similarity of moves. We used eight neighboring points as the index of local similarity. The eight neighboring points are known to be effective features in computer Go. So, they are expected to be effective to work with RAVE. In the experiments, we modified Fuego so that the similar moves considering eight neighbors have more influence than others when updating RAVE values by playout results. Actually, it was observed in self-play with 9 by 9 board and with 19 by 19 board, that the winning probability was significantly improved by presented method in some sets of parameters. Therefore, we conclude that the incorporation of local similarities based on eight neighbors is effective and promising for further researches.

Keywords: Go, Monte-carlo tree search, RAVE

1. はじめに

コンピュータ囲碁は人工知能の研究対象として，注目を集めている分野である．これまでに 2 人ゲームをコン

[1] 東京大学総合文化研究科
Department of General Systems Studies, Graduate School of Arts and Sciences, The University of Tokyo, Meguro, Tokyo 153–8902, Japan
[a)] shimizu@graco.c.u-tokyo.ac.jp

ピュータがプレイする場合においては，min-max 探索を基本にした手法を用いることが一般的であった．実際にそのアプローチにより，チェスやオセロでは，コンピュータプログラムがトッププロに勝つレベルに到達している．しかし，囲碁では，コンピュータプログラムがいまだにトッププロには及ばない棋力にとどまっている．この理由の 1 つとして，チェスやオセロのような 2 人ゲームと比較すると，囲碁は探索空間が広く，有効な評価関数の作成方法が分かっていないことが原因と考えられる．

現在，トップレベルの囲碁プログラムの多くはモンテカルロ木探索を採用している．モンテカルロ木探索は，min-max 探索に代表される従来のゲーム木探索とは異なり，評価関数を用いない．その代わりに，探索木のリーフからのランダムな着手の繰り返しであるプレイアウトによる勝率を参考に，探索木を成長させる．特に，囲碁プログラムには，モンテカルロ木探索の 1 つである UCT を基本とした手法が用いられている．UCT はプレイアウトを行うノード選択の基準として，UCB1 値を用いる．UCB1 値はプレイアウトの勝率とノードの訪問回数をもとに計算される．UCT では十分にプレイアウトを行えば，最善手が選択されるという証明を持つ [6]．しかし，囲碁のようなゲームでは探索空間が広いので，最善手を解明することは現実には不可能であり，限られた時間でなるべく良い手を選ぶことが目標となる．現在までに，囲碁プログラムは，アマ 5 段程度の棋力に到達している．

UCT に RAVE（Rapid Action Value Estimation）[4] を組み合わせることで，有望な着手をより早く見つけることができることが知られている．囲碁のような合法手の多いゲームでは，有力な着手であっても UCB1 値が大きくなるまでに，多くのプレイアウトを必要とする場合がある．RAVE は，プレイアウトの結果を，その局面だけでなく類似局面でも利用する手法である．そのために，特にプレイアウト数が少ない探索の初期に，有力な着手を見つけやすいという効果がある．

本研究では，囲碁プログラムの棋力のさらなる向上を目的として，局面の局所的な類似性に基づく RAVE の拡張を提案する．局所的な類似に関する簡単な指標として，着手の周囲 8 近傍の状態の一致の有無を用いる．着手の周囲 8 近傍の状態は，囲碁では，すでに様々なパラメータ調整に用いられているため，RAVE でも効果的であると期待できる．実際に提案手法を実装し，様々な条件での対戦を通じて提案手法の効果を検証した結果を報告する．この論文は過去の論文 [14], [15] の拡充版である．

2. 関連研究

囲碁におけるゲーム木探索と囲碁における局所的なパターンの利用について述べる．

2.1 モンテカルロ木探索

モンテカルロ木探索は局面の有利不利を，プレイアウトと呼ばれるランダムな着手を行った場合の勝率を元に推定する手法である．局面の有利不利と，プレイアウトの勝率は関連が深いと考えられるためである．以下の 2 つの囲碁の性質から，評価関数による評価と比べると，プレイアウトによる評価は囲碁に適している．対局途中の形勢判断は難しいが，終局時の結果の計算はコンピュータにも容易であるという性質と，自分の眼に打たないという制約を設ければ，どのように打ってもほぼ一定の手数で終局するという性質である．

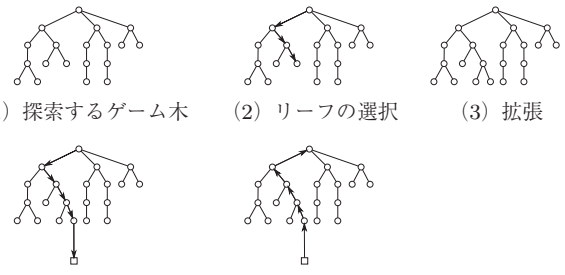
モンテカルロ木探索の動作の概略を**図 1** に示す．この一連の流れの中でモンテカルロ木探索の性能に大きく関わる手順が，(2) のリーフの選択のために，繰り返し行われるノードの選択である．このノードの選択の具体的な決め方として，UCB1 と RAVE を用いる方法を順に説明する．

2.1.1 UCT

UCT は，モンテカルロ木探索の代表的なアルゴリズムである．この手法はゲーム情報学で広く扱われていて，様々なゲームの探索に有効であることが分かっている．UCT では図 1 の手順 (2) で，以下のように定義される UCB1 値を用いる．

$$\mathrm{UCB1}(s,a) = \overline{X}(s,a) + c\sqrt{\frac{2\log n}{n_a}} \qquad (1)$$

ここで s はノードを表す．a はノード s に対応する局面の合法手の 1 つを表す．$\mathrm{UCB1}(s,a)$ はノード s での着手 a の UCB1 値を表す．$\overline{X}(s,a)$ はノード s で着手 a を選択したプレイアウトでの平均勝率を表す．c は探索と活用の割合を決める定数，n はノード s の訪問回数，n_a はノード s で着手 a を選択した回数を表す [6]．UCB1 値の第一項はノード s で着手 a が行われたプレイアウト全体の勝率を表す．第 2 項はノード s で着手 a が行われることの目新しさ



(1) 現時点で探索木がここまで成長していたとする．
(2) 情報を得るリーフを 1 つ選ぶ．リーフの選択はルートからノードを選び続けることで行う．
(3) リーフの訪問回数が閾値に達していれば展開する．
(4) プレイアウトを行う．
(5) プレイアウトの結果をリーフからルートに順に反映される．

図 1 モンテカルロ木探索の動作
Fig. 1 An outline of a Monte-Carlo tree search.

を表す．

2.1.2 RAVE

UCB1 値の計算で使われる勝率は，対象のノード s で対象の着手 a を選択したプレイアウトのみを考慮したものである．RAVE は，着手 a 以外を選択したプレイアウトの情報も，局面の類似性を利用して，活用しようとする方法の 1 つである．具体的に囲碁では，着手の効果は，着手の順序によって変化しにくい．RAVE では，手順を変えたプレイアウトの情報を活用する．そのため，プレイアウト数の少ない段階で，有力な着手を早く発見できる可能性がある．

RAVE を用いたモンテカルロ木探索では，以下のように UCB1 値と RAVE 値を組み合わせた値 value(s,a) の最大のノードを，図 1 の手順（2）で繰り返し選択する [4]．

$$\begin{aligned}\mathrm{value}(s,a) &= \beta(s,a)\cdot \mathrm{RAVE}(s,a)\\ &\quad + (1-\beta(s,a))\cdot \mathrm{UCB}(s,a) \qquad (2)\\ \beta(s,a) &= \frac{k}{(3n+k)} \qquad (3)\end{aligned}$$

ここで RAVE(s,a) は着手 a のノード s での RAVE 値を表す．k は RAVE と UCB1 の重視具合を表す定数．UCB1(s,a) は着手 a のノード s での UCB1 値を表す．k は定数を表す．

また，RAVE 値は以下のようにして計算される．

$$\begin{aligned}\mathrm{RAVE}(s,a) &= \overline{X'}(s,a) + c'\sqrt{\frac{\log m(s)}{m(s,a)}} \qquad (4)\\ m(s) &= \sum_a m(s,a) \qquad (5)\end{aligned}$$

ここで $\overline{X'}(s,a)$ はノード s を通ったプレイアウトの中で s 以降に着手 a が行われたものの勝率を表す．c' は探索と活用の割合を決める定数を表す．$m(s,a)$ はノード s を通ったプレイアウトの中で s 以降に着手 a が行われた回数を表す．

RAVE 値の式は UCB1 値の式と似ているが，第 1 項は手順（2）でノード s を通り，さらにその後手順（2）または手順（4）で着手 a を一度でも着手したプレイアウト全体の勝率を表す．プレイアウトの結果を反映させる際に，UCB1 値は探索で通ったノードと着手の組のみ更新される．一方，RAVE 値は，探索で通ったノードと着手の組だけでなく，探索で通ったノードとそれ以後に打たれた着手の組についても更新される．このように，RAVE ではプレイアウトの各着手につき，多くのノードと着手の組に情報を与える．式 (2) で，value(s,a) はプレイアウト数が少ないときは RAVE 値に近づき，プレイアウト数が多いときは UCB1 値の信頼性が増すので，UCB1 値に近づく．

2.1.3 Fuego でのノード選択

本研究では有力な囲碁プログラムの 1 つであり，オープンソースで開発されている Fuego [*1] を用いた [3]．Fuego を例に UCT と RAVE が強い囲碁プログラムで実際にどのように使われているかを説明する．

Fuego では UCT と RAVE を組み合わせて，以下で定義する value' の最大の着手 α を図 1 の手順（2）で選択する．

$$\begin{aligned}\mathrm{value'}(s,a) &= \beta'(s,a)\cdot \overline{X''}(s,a)\\ &\quad + (1-\beta'(s,a))\cdot \overline{X}(s,a) \qquad (6)\\ \beta'(s,a) &= \frac{m'(s,a)}{n\cdot(\frac{1}{0.9} + \frac{m'(s,a)}{20000} + m'(s,a))} \qquad (7)\\ \overline{X''}(s,a) &= \frac{\sum_p \alpha_p W_p}{m'(s,a)} \qquad (8)\\ m'(s,a) &= \sum_p \alpha_p \qquad (9)\end{aligned}$$

ここで p はノード s を通ったプレイアウトの中で s 以降に着手 a が行われたものを表す．α_p はプレイアウト p の結果の更新の重みを表す．

Fuego では各プレイアウトの結果に重みをつけて，RAVE での勝率 $\overline{X'}$ の代わりに式 (8) の $\overline{X''}$ を用いる．重み α_p の場合に，そのプレイアウト p の勝敗を α_p 回分のプレイアウトの勝敗として扱うことに相当する．たとえば，$\alpha_p = 1.4$ の場合にプレイアウト p に勝利したとすると，そのプレイアウト p の結果は 1.4 回試合をして 1.4 回勝ったものとして扱われる．$\alpha_p = 1$ であれば，重みをつけていない場合と同じである．このように，この α_p を変化させることによって，プレイアウト結果の各ノードに対する重要さに差をつけることができる．

$$\alpha_p = 2 - \frac{\mathrm{first}_p - i_p}{\mathrm{len}_p - i_p} \qquad (10)$$

i_p はプレイアウト p が行われたときのツリーでのノード s のルートからの深さを表す．first_p は着手 a がプレイアウト p の中で初めて打たれたノードのルートからの深さを表す．len_p はルートからプレイアウト p の最後のノードの深さを表す．W_p はプレイアウト p の結果を表す．

Fuego では，重み α_p を式 (10) のように，更新対象のノードとプレイアウト中の着手 a の近さを利用して計算している．本研究では，着手の周囲の類似性を加味して，この重みを調整する．

2.2 囲碁における局所的なパターンの利用

人間が囲碁を打つ際には，局所的なパターンに相当する概念がしばしば用いられる．その例として，「ツケにはハネよ」「ハネには伸びよ」等の格言があることや局所的な石の並びに「一間飛び」「桂馬」「タケフ」等の名前がつけられていることがあげられる．

コンピュータプレイヤが囲碁を打つ際にも局所的なパターンが利用されている．Mogo [*2] には，プレイアウト中

[*1] http://fuego.sourceforge.net/

[*2] http://senseis.xmp.net/?MoGo

の着手の選択に局所的なパターンが用いられている [5]．そのほかにも局所的なパターンは，様々な利用が行われている [1], [2], [7], [10], [11]．

本研究では局所的な局面の類似性を用いることで，RAVE の精度を向上させることを目的としている．そして，局所的な局面の類似性の指標として局所的なパターンの一致の有無を用いる．

3. 局所的な類似性の利用

3.1 RAVE への局所パターンの利用

本節では局面の類似性を検討し，RAVE の効果をより高める手法を提案する．ある局面での着手の評価に異なる局面を利用する場合，より類似した局面を用いることで，正確な評価が行えると期待される．本研究では，着手点の周囲の状態を局面の類似性の指標として用いる．着手点の周囲の状態として着手点の周囲 8 近傍を考える．ある着手について，その周囲 8 近傍の状態が一致している局面は，一致していない局面よりも類似性が高いと判断する．

図 2 の探索木を例に，本手法で局面の類似性を考慮する手続きを説明する．まず従来の RAVE では，探索木中もしくは，プレイアウト中で，着手 A が選択されていれば，そのプレイアウトの結果をルートノード 1 の着手 A に対する RAVE 値の更新に用いる．図 2 の中で着手 A はノード 1 から 2 に遷移する枝，ノード 5 から 7 に遷移する枝，ノード 6 から 10 に遷移する枝，プレイアウト c，e，j 中で選択されるので，ノード 1 の着手 A の RAVE 値の更新にはプレイアウト a，b，c，e，f，g，h，j，l の結果を用いる．

提案手法では，ノード 1 の着手点 A の周囲 8 近傍の状態と着手 A が打たれたときの周囲 8 近傍の状態が一致するかどうかに着目する．ここで，着手 B を着手点 A の隣の格子点への着手とする．着手 B が着手 A より先に打たれた場合には着手点 A の周囲 8 近傍の状態は着手 B の着手前と異なる．そのため，プレイアウト a，b，c，f，g，h は，プレイアウト e，j，l よりもノード 1 での着手 A の効果を正確に予測していると期待される．このような考えで，本手法では従来の RAVE では区別なく用いていたプレイアウトの結果を着手の 8 近傍の状態で区別し，RAVE 値を更新するノードでの着手点 A と同じ 8 近傍の状態で着手 A が行われたプレイアウトの結果，つまり，プレイアウト a，b，c，f，g，h の結果についての更新の際の重み α_p を相対的に大きくする．このことによってプレイアウト a，b，c，f，g，h の結果がプレイアウト e，j，l に比べて相対的に重視される．

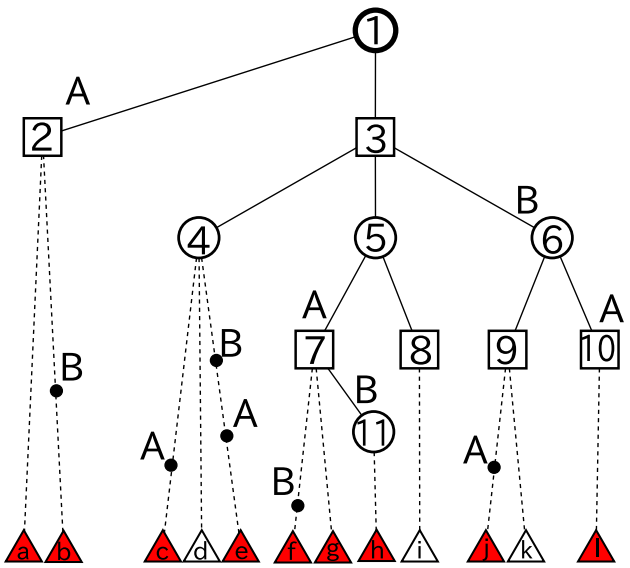


図 2 RAVE でのプレイアウトの結果の利用
Fig. 2 Utilization of result of random simulation in RAVE.

3.2 局所的なパターンの実装の詳細

提案手法の Fuego への実装方法について説明する．各着手点の 8 近傍の状態を判定するために，2 種類のデータを追加した．まず現局面がルートの局面からどのように異なるかについて，各着手点ごとの変化を {0, 1, 2} の数値で表わす配列を用意した．0 はルートの局面から変化がないことを表し，1 はルートの手番の石が着手されたことを表し，2 は相手の石が着手されたことを表す．ただし，実装の簡略化のため，着手による最初の変化のみ記録した．石が打ち上げられた場合や，着手の後に打ち上げられて，偶然もとの状態に戻った場合等には不正確な判定を行っている．このデータは，着手ごとに更新する．この配列を基に，各格子点の 8 近傍の状態の一致を簡単に比較するためのパターン ID を着手ごとに計算する．パターン ID は，ルートからの 8 近傍の変化を，各格子点ごとに $[0, 3^8]$ の範囲の整数で表現したものである．パターン ID は，図 1 の手順（3）で，新たなノードを作成した際に各ノードに合法手に対するパターン ID をすべて保存するほか，着手が行われるたびに，着手の履歴とともに対応するパターン ID を保存する．Fuego における実装では，SgUctSearch.h 内の構造体 SgUctGameInfo がルートからの木の探索とプレイアウトの情報を持つ．変化を表す配列と着手に対応するパターン ID はこの SgUctGameInfo に追加した．また，SgUctTree.h 内のクラス SgUctNode は探索木のノードに対応するクラスである．SgUctNode に，ノードでの各合法手に対応するパターン ID に相当する変数を追加した．このノードに保存するパターン ID の計算回数は，ノードの合法手の数に比例する．なお，パターンを 8 近傍から拡張した場合には，パターンのサイズに比例してパターン ID の計算にかかる時間が増えると考えられる．

プレイアウト終了時には，ルート局面から終局までの各着手に対応する RAVE 値を更新する．その際に，各着手に対応するパターン ID と，各ノードに保存されたパターン ID を比較し，一致していれば式 (8)，(9) の重み α_p を相対

的に大きくする．あるノードのRAVE値の更新時のパターンIDの比較回数は，そのノードの合法手のうちプレイアウトが終わるまでに打たれた着手の数に等しい．一般に，合法手が多いとプレイアウトが終わるまでの着手回数が多くなる．そのため，合法手が多いノードのRAVE値の更新ほどパターンIDの比較が多くなると考えられる．Fuegoではクラス SgUctSearch のメンバ関数 UpdateRaveValuesでRAVE値の更新が行われる．そのため，この関数内でパターンIDの比較を行う．

4. 対戦実験

提案手法の効果を測定するために，提案手法をオープンソースプログラムであるFuegoに実装した．Fuegoは最新版に近いリビジョン1603を用いた[3]．そして，提案手法を導入したFuegoとオリジナルのFuegoの対戦や，別の囲碁プログラムであるPachiとの対戦を行い，手法を評価した．

本実験では，提案手法を導入したFuegoとして，以下の2種類のプログラムを用いた．

パターン重視 (k,l)　パターンIDが一致した場合の α_p を k 倍，一致しなかった場合の α_p を l 倍したプログラム．$k>1, l=1$ ならば周囲が一致した結果を重視することに，$k=1, 0<l<1$ ならば一致しなかった場合に重みを下げる動作となる．

パターン重視（$\beta'(s,a)$ 変更なし） (k,l)　RAVEの式(8)の $X''(s,a)$ の計算には，パターン重視 (k,l) と同様の方法で重みを調整した α_p を用い，式(7)の $\beta'(s,a)$ の計算にはオリジナルのFuegoと同様の α_p を用いるプログラム．

4.1 Fuegoを用いた自己対戦

提案手法を導入したFuegoをオリジナルのFuegoと対戦させて，勝率を測定した．重み α_p の変更方法として，いくつかのパラメータを設定した．対戦条件は，中国ルールで，コミは6目半，試合数は特に断らなければ10,000局である．

オリジナルのFuego どうしの対戦では定めた試合数で対戦した結果から黒番，白番の勝率を計算した．提案手法を導入したプログラムの勝率は，定めた試合数を黒番を持ってオリジナルのFuegoと対戦して測定した．また，白番についても同様に行った．

4.1.1 プレイアウト数固定の対戦

まず，類似性に基づく重みの調整そのものの有効性を調べるために，1手あたりのプレイアウト数を固定して対戦を行った．この対戦では，1手あたりのプレイアウト数の上限を10,000回として，時間による制限は行わなかった．9路盤，13路盤，19路盤で実験を行った．

表 1 に9路盤での各プログラムの勝率を示す．右肩に

表 1　自己対戦の勝率 9 路盤（プレイアウト数固定）
Table 1　Wining rate of programs against default Fuego (9×9 board). Number of simulations per move was fixed.

プログラム名	黒番	白番
パターン重視（1，0）	33.2%	24.4%
パターン重視（1，0.25）	49.9%	36.0%
パターン重視（1，0.5）	54.2%	42.4%
Fuego（パターン重視（1，1））	56.0%	44.0%
パターン重視（1.5，1）	58.6%**	47.1%**
パターン重視（2，1）	57.5%*	46.6%**
パターン重視（4，1）	52.5%	42.1%
パターン重視（$\beta'(s,a)$ 変更なし）（1.5，1）	57.1%	47.2%**
パターン重視（$\beta'(s,a)$ 変更なし）（2，1）	57.6%*	46.8%**

* のついたものは有意水準5%で有意に向上した結果，** のついたものは有意水準1%で有意に向上した結果である．まずオリジナルのFuegoは黒番と白番の勝率が5分ではないことが分かっているため，オリジナルのFuego どうしの対戦における勝率を測定した．表1中の “Fuego” の欄にあるとおり黒番が56%と勝ち越している．

表1の各行に対応する結果から，パターン優先（1.5，1），パターン優先（2，1）は勝率が有意に向上している．

パターン重視（1，0.5）とパターン重視（2，1）では，$RAVE(s,a), \overline{X''}(s,a)$ の値は本質的に同じであるが，実験結果には差がある．その理由は以下のように考えられる．Fuegoでは，α_p を大きくすると，$\beta'(s,a)$ も大きくなるので，パターン重視（1，0.5）とパターン重視（2，1）での $value(s,a)$ は本質的に異なる値となる．そのために，パターン重視（1，0.5）とパターン重視（2，1）では同じ局面でも異なる指し手を選択する可能性がある．実際に本実験でも勝率に差が生じている．

また，α_p をパターンの一致の有無に関係なく，一律に大きくすることによってFuegoが強くなることがある[14], [15]．このことから，パターン重視（2，1）等は，$\beta'(s,a)$ が偶然に上手く調整されて勝率が向上した可能性もある．そこで，パターン重視（$\beta'(s,a)$ 変更なし）（2，1）について検討すると，オリジナルのFuegoと同様の $\beta'(s,a)$ の値を用いていても，有意に勝率が向上していることが分かる．よって，勝率の向上は $\beta'(s,a)$ の変化にのみに因るのではなく，局所的なパターンの利用によるRAVEの勝率 X'' の調整による効果が存在すると考えられる．

また，13路盤でも同様の実験を行った．実験時間の都合で黒番のみ実験を行ったが，他の条件は9路盤での実験と同じである．

結果を**表 2** に示す．13路盤ではオリジナルのFuegoに対してパターン重視（$\beta'(s,a)$ 変更なし）（1.5，1）は有意に勝率を向上した．また，その他にも勝率を向上したパラメータもあった．

19路盤の対戦にはさらに時間がかかるので，試合数は

表 2 自己対戦の勝率 13 路盤（プレイアウト数固定）
Table 2 Wining rate of programs against default Fuego (13×13 board). Number of simulations per move was fixed.

プログラム名	黒番
Fuego（パターン重視（1，1））	51.4%
パターン重視（1.25，1）	52.0%
パターン重視（1.5，1）	52.7%
パターン重視（1.75，1）	51.3%
パターン重視（2，1）	50.5%
パターン重視（$\beta'(s,a)$ 変更なし）（1.5，1）	53.0%*

表 3 自己対戦の勝率 19 路盤（プレイアウト数固定）
Table 3 Wining rate of programs against default Fuego (19×19 board). Number of simulations per move was fixed.

プログラム名	黒番
パターン重視（1，0.75）	50.4%
Fuego（パターン重視（1，1））	50.6%
パターン重視（1.25，1）	54.6%
パターン重視（1.5，1）	55.7%*
パターン重視（1.75，1）	55.6%*
パターン重視（2，1）	49.1%
パターン重視（4，1）	52.5%
パターン重視（$\beta'(s,a)$ 変更なし）（1.5，1）	53.3%
パターン重視（$\beta'(s,a)$ 変更なし）（1.75，1）	52.1%

表 4 各プログラムの 1 秒あたりのプレイアウト回数
Table 4 Average number of simulations per secondo.

プログラム名	9 路盤	13 路盤	19 路盤
Fuego	7,522.7 g/s	3,913.2 g/s	1,787.3 g/s
パターン重視	6,949.6 g/s	3,632.1 g/s	1,627.9 g/s

1,000 局，黒番のみで実験を行った．その他の条件は 9 路盤と同じである．ただし，オリジナルの Fuego どうしの対戦は 3,000 局行った結果である．

結果を**表 3** に示す．19 路盤ではパターン重視（1.5，1）とパターン重視（1.75，1）がオリジナルの Fuego に対して黒番で有意に勝率が向上した．また，提案手法を導入したパターン重視（1.5，1）とパターン重視（1.75，1）の結果から，19 路盤では提案手法は周囲 8 近傍の状態が一致した場合の α_p を 1.5 倍または 1.75 倍することで効果を発揮することが分かった．

以上の結果から，提案手法は 9 路盤，13 路盤，19 路盤おいて有効に働くパラメータが存在し，有望であると考えられる．ただし，9 路盤，13 路盤，19 路盤ではそれぞれ効果的なパラメータが異なっていることから，ゲームに合わせた調整が必要である．

4.1.2 計算速度の測定

表 4 は Fuego と提案手法を実装した Fuego であるパターン重視（1.5，1）の 1 秒あたりのプレイアウト回数に関する計算速度を表している．CPU は AMD Opteron™ Processor 6274 を用いて，1 スレッドで Fuego 4.1.1 項の

表 5 自己対戦の勝率 9 路盤（時間固定）
Table 5 Wining rate of programs against default Fuego (9 × 9 board). Time per move was fixed.

プレイヤ	黒番	白番
Fuego	56.0%	44.0%
パターン重視（1.5，1）	55.9%	43.3%
パターン重視（2，1）	56.1%	44.5%

表 6 自己対戦の勝率 19 路盤（時間固定）
Table 6 Wining rate of programs against default Fuego (19×19 board). Time per move was fixed.

プレイヤ	黒番
Fuego	50.9%
パターン重視（1.5，1）	49.1%
パターン重視（2，1）	46.7%

実験を行った際の平均を表 4 にまとめている．提案手法の現在の実装は，7%程度のオーバヘッドを持っていることが分かる．

4.1.3 時間固定の対局

続いて実用的な条件での効果を測るために，思考時間を 1 手 2 秒で揃えた対戦を行った．ただし，Fuego の判断で指定した時間に達する前にプレイアウトを打ち切って着手する場合もある．また，本稿では実験条件を単純に保つために 1 スレッドで行った．実験条件は基本的にこれまでと同じで，プレイアウト数の上限を指定せず，代わりに，時間制限を 1 手 2 秒と変更した．9 路盤と 19 路盤で実験を行った．

結果はそれぞれ**表 5**，**表 6** に示す．前節の実験で，オリジナルの Fuego に比べて有意に勝率を向上させたパラメータを対象として実験を行った．1 手の思考時間を揃えた条件ではオリジナルの Fuego より有意に勝率を向上させるパラメータはなかった．これらは現状では，提案手法の実装による速度のオーバヘッドと提案手法の効果がほぼ釣り合っている状況にあると考えられる．なお表 1 と表 5 の Fuego の勝率は表示の範囲内では同じ値だが，実際の勝率は異なる．

4.2 他の囲碁プログラムを用いた対戦

提案手法を導入した Fuego と Pachi（version 10.00（Satsugen））の対戦を行った．実験条件は以下のとおりである．実験条件は 9 路盤，中国ルールで，コミは 6 目半，試合数は 10,000 局，時間制限はなしとして，Fuego の 1 手あたりのプレイアウト数の上限を 10,000 回，Pachi の 1 手あたりのプレイアウト数の上限を 30,000 回とした．オリジナルの Fuego と 4.1.1 項で 9 路盤で有意に勝率の向上が見られたパターン重視（1.5，1）を比較．

結果は**表 7** に示す．提案手法により，勝率は向上しているが，有意ではない．対戦相手によってパラメータの調整

表 7 Pachi との対局の勝率（9 路盤）
Table 7 Wining rate of programs against pachi (9 × 9 board).

プログラム名	黒番
Fuego	66.6%
パターン重視（1.5，1）	67.7%

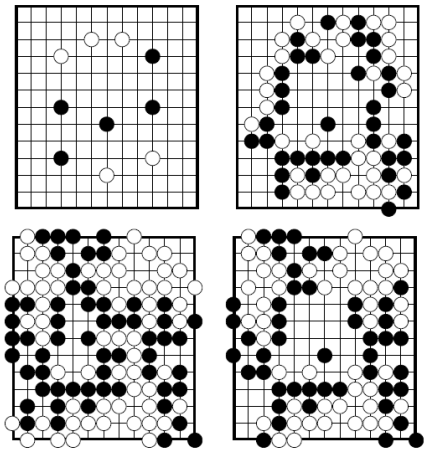


図 3 パターン ID の一致率を調べるために用いた局面（9 路盤）
Fig. 3 The 9 × 9 positions which were used to measure concordance rate of pattern ID.

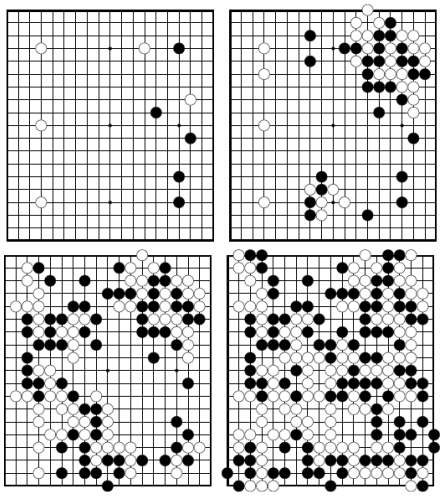


図 4 パターン ID の一致率を調べるために用いた局面（9 路盤）
Fig. 4 The 19 × 19 positions which were used to measure concordance rate of pattern ID.

を行う必要がある可能性がある．

4.3 局面による提案手法の効果の差

提案手法がどのような局面で特に効果があるかを把握する材料として，提案手法の実装により RAVE の重み α_p が変化する割合を，複数の局面で調査した結果を報告する．具体的には RAVE の更新を行う際に，パターン ID が一致する割合を調査した．

黒番で 1 手着手する探索を，**図 3**，**図 4** に示した局面を対象に行った．これらは定石を用いないで Fuego どうしを対戦させた棋譜から序盤，中盤，終盤，終局間際の局面を

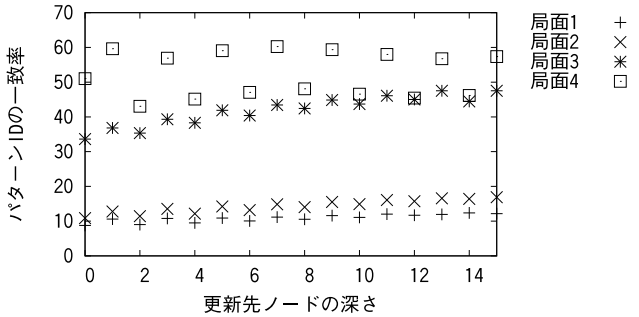


図 5 RAVE の更新を行う際のパターン ID の一致率（9 路盤）
Fig. 5 The concordance rate of pattern ID when RAVE values were updated (9 × 9 board).

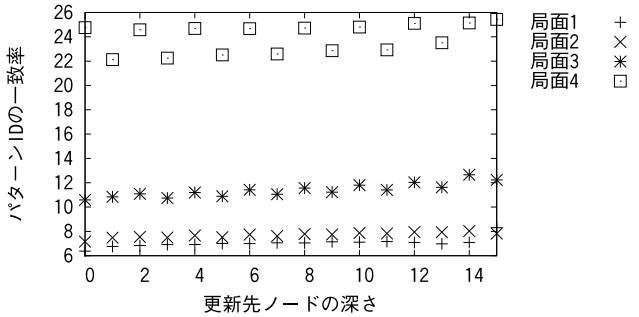


図 6 RAVE の更新を行う際のパターン ID の一致率（19 路盤）
Fig. 6 The concordance rate of pattern ID when RAVE values were updated (19 × 19 board).

抽出したものである．9 路盤，19 路盤ともに左上，右上，左下，右下の順に局面の番号をそれぞれ 1，2，3，4 と表記する．

結果を**図 5**，**図 6** に示す．横軸が RAVE 値を更新するノードのルートからの深さ，縦軸はパターン ID の一致率である．プレイアウト回数は 10,000 回である．局面 1，2，3，4 の順に一致率が高くなっていることから，局面が進むほど一致率が高くなる傾向が予想される．理由としては，試合が進むほど着手可能な点が少なくなるので，ある点の周囲の状態が変化しにくいからと考えられる．この予想は 9 路盤での一致率が 19 路盤での一致率より高くなっていることとも一致する．

ノードの深さを表す横軸に着目すると，手番が一致率に影響することが観察された．また 9 路盤の局面 3 の結果に強く表れているように，更新先のノードが深くなるほど一致率が増加する傾向が読み取れる．

一致率が 0%もしくは，100%の局面があると仮定すると，このような局面では，提案手法による計算コストが無駄になってしまう．そのため，提案手法の計算時間に対する効果の大きさは局面ごとに異なると考えられる．このようなパターン ID の一致率と提案手法の効果については今後の研究課題である．

5. おわりに

本研究では，探索問題を解くためのアルゴリズムであるUCT に着目し，UCT と組み合わせて用いられる RAVE の性能改善に取り組んだ．RAVE とは，類似する局面を利用することで，モンテカルロ木探索の性能を高める手法である．そこで，類似性をより精度良く判断するために，囲碁を題材とした研究では標準的な，着手の周囲 8 近傍の状態に注目し，RAVE の枠組みを拡張した．提案手法では，RAVE において，着手の周囲 8 近傍の石の状態が一致した状態での試行（プレイアウト）で得られた結果を，一致しなかった状態での試行（プレイアウト）で得られた結果より重視する．

着手の周囲 8 近傍が一致した状態をどの程度重視すると良いかについては，盤面の大きさ（9 路盤，13 路盤，19 路盤）によって異なることが，実験結果から分かっている．したがって，このパラメータは，適用対象の問題ごとに決める必要があると思われる．なお，実験の過程で，RAVE 自体をどのように重視するかを決めるパラメータについても，Fuego には調整の余地があることが分かった．今回の実験の結果からその効果を除いても，提案手法の効果があることが示された．本稿では実験条件を単純に保つために 1 スレッドで行ったが，本手法は MCTS を並列化した場合にも，RAVE が動作することと同様に動作する．局所的な局面近似の指標として 3×3 のパターン以外のもの（5×5 のパターン等）を用いた手法や，その他の指標の利用は今後の研究課題である．

モンテカルロ木探索は，囲碁のような 2 人零和完全情報ゲーム以外にも，多数応用されている [8], [9], [12], [13]．UCT が使われる問題であれば，プレイアウトの類似性を利用して効率的に情報を収集する RAVE のような考え方が有効である問題も多いと考えられ，提案手法も応用可能であると期待される．それらへの応用は，今後の研究課題である．

参考文献

[1] Chou, C.-W., Yen, S.-J. and Yang, J.-K.: Using Global Corresponding Move Bias on MCTS, *The 17th Game Programing Workshop 2012*, pp.63–67 (2012).

[2] Coulom, R.: Computing elo ratings of move patterns in the game of Go, *ICGA Journal*, Vol.30, No.4, pp.198–208 (2007).

[3] Enzenberger, M., Muller, M., Arneson, B. and Segal, R.: Fuego –An Open-Source Framework for Board Games and Go Engine Based on Monte Carlo Tree Search, *IEEE Trans. Computational Intelligence and AI in Games*, Vol.2, No.4, pp.259–270 (online), DOI: 10.1109/TCIAIG.2010.2083662 (2010).

[4] Gelly, S. and Silver, D.: Monte-Carlo tree search and rapid action value estimation in computer Go, *Artificial Intelligence*, Vol.175, No.11, pp.1856–1875 (2011).

[5] Gelly, S., Wang, Y., Munos, R., Teytaud, O., et al.: Modification of UCT with patterns in Monte-Carlo Go (2006).

[6] Kocsis, L. and Szepesvari, C.: Bandit Based Monte-Carlo Planning, *Machine Learning: ECML 2006*, Vol.4212, Springer, pp.282–293 (2006).

[7] Stern, D., Herbrich, R. and Graepel, T.: Bayesian pattern ranking for move prediction in the game of Go, *Proc. 23rd International Conference on Machine Learning*, pp.873–880, ACM (2006).

[8] 関　栄二，三輪　誠，鶴岡慶雅，近山　隆：将棋におけるモンテカルロ木探索の特性の解明，*The 17th Game Programing Workshop 2012*, pp.68–75 (2012).

[9] 横山大作：モンテカルロ木探索アルゴリズムの将棋への適用，*The 17th Game Programing Workshop 2012*, pp.76–83 (2012).

[10] 本田拓郎，Viennot, S., 池田　心：囲碁における大局観を実現する広域パターンマッチング，*The 17th Game Programing Workshop 2012*, pp.17–21 (2012).

[11] 中西　惇，中村貞吾：局面変化とパターンによる着手予測を用いた囲碁の好手の判別，*The 16th Game Programing Workshop 2011*, pp.108–111 (2011).

[12] 万代悠作，橋本　剛：UCB+を用いた Big Two AI の研究，*The 17th Game Programing Workshop 2012*, pp.205–210 (2012).

[13] 大町　洋，池田　心：同時進行ゲームのためのモンテカルロ木探索，*The 17th Game Programing Workshop 2012*, pp.197–204 (2012).

[14] 志水　翔，金子知適：局面の局所的な類似性を利用したモンテカルロ木探索の効率化，ゲームプログラミングワークショップ 2013 論文集，pp.130–133 (2013).

[15] 志水　翔，金子知適：局面の局所的な類似性を利用したモンテカルロ木探索の効率化，情報処理学会研究報告．GI［ゲーム情報学］，No.1, pp.1–7 (2013).

志水 翔　（学生会員）

2013 年東京大学教養学部卒業．2013 年より東京大学総合文化研究科修士課程在学中．人工知能に興味を持つ．

金子 知適　（正会員）

1997 年東京大学教養学部卒業．2002 年東京大学院総合文化研究科博士課程修了．博士（学術）．2002 年東京大学院総合文化研究科助手．2007 年助教を経て 2012 年より准教授．ACM，人工知能学会，日本ソフトウェア科学会各会員．思考ゲーム，機械学習に興味を持つ．