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]

概要：現在, 囲碁プログラムの多くはモンテカルロ木探索 (MCTS) を採用している．また，MCTS と Rapid Action Value Estimation (RAVE) を組み合わせることも一般的である．本研究では，RAVE に局所的な類似性の適用による改善を試みた．局所的な類似の指標として，着手の周囲 8 近傍の状態の一致の有無を採用した．局所的な類似の指標に基づいて，RAVE の更新時の重みのパラメータ調整を行った．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，有望であると考えられる．

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

Abstract: Monte carlo tree search (MCTS) has been used in many Go programs. Rapid action value estimation (RAVE) that is one of enhancements in MCTS. We tried to improve RAVE using local state near moves. We selected eight neighboring point as the index of local similarity. We adjusted the weight which is updated in RAVE. It is effective to using local state near moves in RAVE.

1. はじめに

コンピュータ囲碁は研究対象として，注目を集めている分野である．チェスやオセロではコンピュータプログラムの方がトッププロにも勝つことができる．しかし，囲碁では未だに強いプログラムでもアマ 5 段程度の棋力であり，トッププロには及ばない棋力である．

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，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，探索木のリーフでのプレイアウト (ランダムな着手の繰り返し) 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 が用いられている．UCT はプレイアウトを行う価値のある局面を選ぶ (ノード選択) 為に，UCB1 値を用いる．UCB1 値は勝率とノードの信頼性をもとに計算される．UCT ではプレイアウト回数が増えれば，UCB1 値が min-max 値に収束することが知られている [3]．UCT に RAVE (Rapid Action Value Estimation)[1] を利用することで，有望な着手をより早くみつけることが知られている．その効果は特に，探索の初期に表れる．囲碁では合法手が多いため，有力な着手であっても UCB1 値が安定するのは, 他のゲーム比べると多くのプレイアウトを必要とする．RAVE はプレイアウトの結果の利用範囲を拡張している．そのために，評価値の見積りが，より少ないプレイアウト数で安定しやすい．

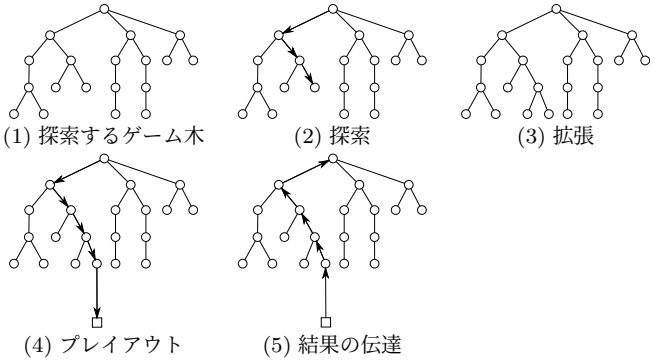
本研究では，囲碁プログラムの棋力の更なる向上を目的として，局面の類似性に基づく RAVE の拡張を提案する．局所的な類似に関するもっとも簡単な指標として，着手の周囲 8 近傍の状態の一致の有無を採用した．この指標は，囲碁では，既に RAVE 以外のパラメータ調整に用いられている．そのため，RAVE でも効果があると期待できる．提案手法を実装し，様々な条件での対局を通じて提案手法の効果を検証した．

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

2. 関連研究

2.1 モンテカルロ木探索

モンテカルロ木探索では局面の有利不利を，お互いにランダムな着手 (プレイアウト）を行った場合の勝率を元に判断する．なぜなら，もしある局面がどちらかに有利であれば, プレイアウトで勝利する可能性も高いと考えられるためである．評価関数を使った評価と比べると，プレイアウトを用いた評価は囲碁に適している．その理由は以下の二つの性質にある．囲碁では，コンピュータは対局途中の形勢判断は難しいが，終局時においては結果を計算する事が容易であるという点と，自分の眼に打たないという制約を設ければ，ほぼ一定の手数で終局する点である．



(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一つ選ぶ．リーフの選択はルートから有力なノードを選び続けることで行う．
(3) 訪問回数が閾値に達したノードがあれば展開する．
(4) プレイアウトを行う．
(5) プレイアウトの結果をノードに返す．

図 1 モンテカルロ木探索

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のが，(2) の手順で行われるノードの選択である．このノードの選択をどのように行うかによって，探索の特徴が表れる．

2.2 Fuego でのノード選択

Fuego では UCT と RAVE を組み合わせて，以下で定義する value の高い着手 a をノード s で選択している．

$$\mathrm{value}(s,a) = \beta(s,a)\mathrm{RAVE}(s,a) + (1-\beta(s,a))\mathrm{UCB1}(s,a)$$

$$\beta(s,a) = \frac{m(s,a)}{n_s \cdot (1.0/0.9 + 1.0/20000 \cdot m(s,a) + m(s,a))}$$

$$m(s,a) = \sum_j \alpha_j$$

$$\alpha_j = 2 - \frac{\mathrm{first} - i}{\mathrm{len} - i}$$

$$\mathrm{UCB1(s,a)} = \overline{X_s} + c_1\sqrt{\frac{2\log n}{n_s}}$$

$$\mathrm{RAVE}(s,a) = \overline{X}(s,a) + c_2\sqrt{\frac{\log m(s)}{m(s,a)}}$$

$$m(s) = \sum_a m(s,a)$$

ただし，s:ノード，a:着手，$m(s,a)$:ノード s もしくはその子孫のノードで着手 a を選んだ回数，$\overline{X_s}$: ノード s を通ったプレイアウトの平均報酬，c_1:定数，c_2:定数，n : ノード s の親ノードを通った回数，$\overline{X}(s,a)$:子孫のノードで着手 a を選んだプレイアウトの平均報酬，n_s:ノード s を通った回数，$m(s)$:着手 a を合法手に持つノードを通った回数，first:RAVE 値を更新する着手がプレイアウトの中で初めて打たれた時に達するノードのルートからの深さ，len:プレイアウトの長さである．

Fuego では各プレイアウトの結果に重みをつけて，RAVE での勝率は重み付き平均として計算している．すなわち，重み α_j はそのプレイアウトの勝敗を α_j 回分のプレイアウトとして扱う効果を持つ．この α_j を変えることによって，各プレイアウトごとの重要さに差をつけることができる．

2.3 囲碁における局所的なパターンの利用

囲碁では局所的なパターンを利用することが，一般的である．これは，「ツケにはハネよ」「ハネには伸びよ」等の格言や，局所的な石の並びに「一間飛び」「桂馬」「タケフ」等の名前がある．これらを参考に人間は候補手を考えている．このことから，局所的なパターンの利用は，コンピュータ囲碁に限った話ではないことが分かる．

コンピュータ囲碁においても，以下のように局所的なパターンが利用されている．”モンテ・カルロ碁における UCT のパターンによる改良”[2] では UCT に 8 近傍のパターンを用いている．このようにコンピュータ囲碁では，パターンマッチングを行うことは様々な指標に用いられている ([4]).

本研究では局所的なパターンを局面の類似性の指標として用いることで，RAVE の精度を向上させることを目的としている．

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

本研究の目的は着手の類似性を詳細に検討し，RAVE の効果をより高めることである．そのために，着手の周囲の状態として着手点の 8 近傍を考える．つまり，ある 2 つの着手の 8 近傍が一致していれば，一致していない場合よりも類似しているとして扱う．

RAVE ではある点への着手の効果は着手の順序にあまり影響を受けないために，順序の違うプレイアウトの結果に

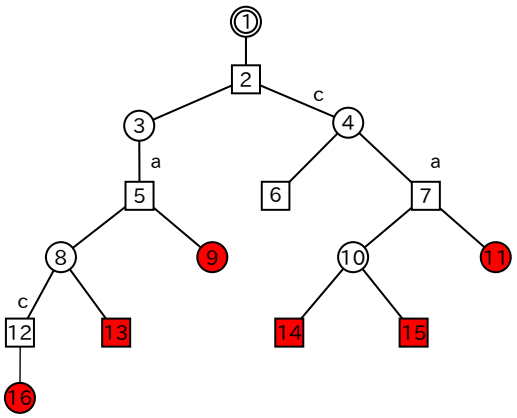


図 2 RAVE でのプレイアウトの結果の利用

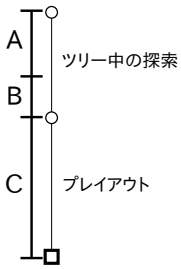


図 3 実装

も用いることができると考える．図 2 の探索木を例に説明すると，従来の RAVE では，どこかで a が着手されていれば，ルートからノード 1 を通って行われるプレイアウトの結果をノード 1 の着手 a に対応する RAVE 値の更新に用いる．着手 a はノード 3 から 5 と 4 から 7 で打たれるので，ノード 1 の着手 a の RAVE 値の更新にはノード 9，11，13，14，15，16 からのプレイアウトの結果を用いる．

一方，提案手法では，ノード 1 の着手点 a の周囲 8 近傍の状態と周囲 8 近傍の状態が一致する時に a へ着手したプレイアウトのみを考える．着手 a の 8 近傍の着手 c が a より先に打たれた場合は 8 近傍の状況が異なる．ノード 9，13，16 のプレイアウトはノード 1 での着手点 a への着手の効果を，類似した局面で測定していると期待される．そのために，本手法では従来の手法で用いていたリーフの中で着手点 a への着手の効果と同等と考えられるプレイアウト，つまり，ノード 9，13，16 でのプレイアウトの結果による更新の際の α_j を相対的に大きくする．このことによってノード 9，13，16 でのプレイアウトの結果が重視される．

3.2 局所的なパターンの Fuego への実装

提案手法の実装について説明する．プレイアウトの簡略図である**図 3** に示した区間 A，B，C について，A，B は UCT が管理する木を降りる区間，C はランダムにプレイアウトを行う区間である．本手法では，事前に決めた深さに基づいて A の部分で 8 近傍の一致を考慮した RAVE 値の管理を行い，B の部分では従来手法を用いる．そのために区間 A，と区間 B，C で，異なる種類の追加の計算を行い計算結果の一部を各節点毎に保存する．全ての区間で共通して行う計算は，直前の着手の周囲 8 近傍が根から変化しているかどうかを表現する 0 から 3^8 の範囲の整数を求めることである．この整数は各節点に保存される．ただし，実装の簡略化のため，一度でも変化した格子点は以降，異なるままとして扱った．つまり，着手の後に打ち上げられて，偶然もとの状況に戻った場合には不正確な判定を行っている．さて，整数を計算するためには，着手の前にその場所の 8 近傍がどの程度変化していたかを知っている必要がある．そのために，根から葉に降りる間の作業変数として整数の配列を用意し，配列内の各整数が対応する格子点に対して上記の 8 近傍の変化具合を表すように維持する．加えて A の区間では，その配列自身も節点に保存する．Fuego の場合は 9 路盤のゲームであっても 19 路をプレイ可能なデータ構造を用いるため，各格子点を表現可能な配列は要素数が大きなものとなり，その保存はオーバーヘッドが大きい．特に断らない限り，以降の実験では区間 B を設けず，探索木の範囲を区間 A とする．区間 A の長さを制限すると，正確さは一部犠牲になるものの，動作を高速にできると期待される．

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．そして，提案手法を導入した Fuego とオリジナルの Fuego の対局を行ない，勝率を測定した．9 路，13 路，19 路の三種類で行なったが，本稿では 9 路の結果を紹介する．

実験には以下のようなプログラムを用いた．

Fuego オリジナルの Fuego (リビジョン 1603)

倍率変更 k α_j を一律 k 倍したプログラム

パターン優先 (k，l) 提案手法のようにある着手の周囲 8 近傍の状態と一致した状態の時にある着手を行った時の α_j を k 倍，一致しなかった時の α_j を l 倍したプログラム

これらを以下のような条件でオリジナルの fuego と対戦させた．9 路，中国ルール，コミは 6 目半，対局数 10000 局．使用した CPU は ”AMD Opteron(TM) Processor 6274” である．

まず 1 手あたりのプレイアウト数を揃えた条件での結果を示す．ただし，指定のプレイアウト数に達する前のプレイアウト打ちきりの判断は Fuego の設定による．この条件では，提案手法のオーバーヘッドを切り離してパターンを使う効果の有無を図ることが目的である．9 路で，パターン優先 (2.25，1.5) は黒番・白番ともに最も高い勝率をあげた．オリジナルの Fuego 同士での対戦では黒番の勝率 56%となった．パターン優先 (2.25，1.5) が黒番，オリジナルの fuego が白番で黒番パターン優先 (2.25，1.5) の勝率が 59.4%となった [5]．また，19 路でも同様の実験を行った． 試合数は 3000 で，オリジナルの Fuego 同士での黒番の勝率が 50.6． パターン優先 (1.75,1) がが黒番, オリジナルの Fuego が白番での黒番の勝率が 55.6%であった．

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，指定した時間に達する前のプレイアウト打ちきりの判断は Fuego の設定による．

(9 路)	勝率	
プレイヤ	黒番	白番
倍率変更 0	2.9%	1.2%
倍率変更 0.5	49.8%	38.4%
Fuego (倍率変更 1	*56.0%	*44.0%
パターン優先 (1.5, 1.5)	#57.6%	#45.6%
倍率変更 2	56.4%	44.9%
倍率変更 2.5	56.0%	#45.7%
倍率変更 3	56.5%	45.2%
倍率変更 3.5	56.1%	44.0%
倍率変更 4	55.1%	43.6%
パターン優先 (1，0)	33.2%	24.4%
パターン優先 (1，0.25)	49.9%	36.0%
パターン優先 (1，0.5)	54.2%	42.4%
Fuego (パターン優先 (1，1))	*56.0%	*44.0%
パターン優先 (1.5 1)	$58.6%	$47.1%
パターン優先 (2，1)	#57.5%	$46.6%
パターン優先 (4，1)	52.5%	42.1%
Fuego (パターン優先 (1，1))	*56.0%	*44.0%
パターン優先 (3，1.5)	$58.4%	$47.0%
パターン優先 (2.25，1.5)	$59.4%	$48.2%
パターン優先 (4，2)	$58.0%	$46.6%

表 1 プレイアウト数 10000(9 路)

(9 路)	勝率	
プレイヤ	黒番	白番
Fuego	*56.0%	*44.0%
倍率変更 1.5	$58.4%	#45.4%
パターン優先 (1.5 1)	#54.3%	#42.3%
パターン優先 (2，1)	$53.9%	$41.5%
パターン優先 (2.5，1)	$52.9%	
パターン優先 (3，1.5)	55.5%	43.6%
パターン優先 (2.25，1.5)	54.8%	

表 2 1 手 2 秒 (9 路)

プレイアウト数を 10000 回に揃えて行った実験で，オリジナルの Fuego に比べて有意に勝率を向上させたプログラムを対象に実験を行った．時間固定ではどのパラメータもオリジナルの Fuego より強くなることがなかった．プレイアウトの指し手周囲の局所パターンを保持する為，1 プレイアウトにかかる時間がオリジナルの Fuego より遅くなったためと考えられる．

そこで，実行速度を向上させるために，部分的に提案手法を導入したバージョンの対局を行った．ツリーの中で深さ 10 までのノードの RAVE 値の更新時のみパターンを用いることにした．図 3 の図の A の区間をルートから深さ 10 までのノードに対応する区間とした．他のノードでは，パターンが一致していないものとして扱った．

勝率の上がるパラメータもあったが有意ではない．パターン優先 (3，1.5)，パターン優先 (2.25，1.5) はともにパターン優先 (1.5，1.5) を改良したものと考えられる．またパターン優先 (2，1) は勝率向上することができたが，倍率変更 1.5 の勝率の方が高いため，ので，この実装でも良い結果は得られていないものと考えられる．

(9 路)	勝率
プレイヤ	黒番
Fuego	*56.0%
パターン優先 (1.5 1)	55.3%
パターン優先 (2，1)	57.3%
パターン優先 (3，1.5)	56.8%
パターン優先 (2.25，1.5)	56.4%

表 3 部分 (9 路)

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 という手法の効率改善に取り組んだ．RAVE とは，局面の勝率を推定する情報が少ない場合には，他の局面でのプレイアウトの結果を補うことで，モンテカルロ木探索の性能を高める手法である．この RAVE で類似性としては，着手の周囲 8 近傍の状態を用いた．着手の周囲 8 近傍の石の状況が一致した状況での試行 (プレイアウト) で得られた結果を，一致しなかった状況での試行による結果より重視する様に，RAVE の枠組みを拡張した．

着手の周囲の 8 近傍が一致した状況をどの程度重視すると良いかについては，適用対象の問題ごとに決める必要があると思われる．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パターンを利用した RAVE の重みの調整は，効果があったと考えられる．

モンテカルロ木探索は，囲碁のような二人零和完全情報ゲーム以外にも，多数応用されている．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 のような考え方は有効である対象は多いと考えられ，提案手法も応用可能であると期待される．それらの具体化は，今後の研究課題である．

参考文献

[1] Gelly, S. and Silver, D.: Combining Online and Offline Knowledge in UCT, *the 24th international conference on Machine learning* (2007).

[2] Gelly, S., Wang, Y., Munos, R. and Teytaud, O.: モンテ・カルロ碁における UCT のパターンによる改良.

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

[4] 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).

[5] 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).