情報処理学会論文誌 Vol.55 No.11 2377–2388 (Nov. 2014)

行動評価関数を用いたモンテカルロ木探索の重点化と見落としの抑制

池田 心[1,a)] ビエノ シモン[1,b)]

受付日 2014年2月21日, 採録日 2014年9月12日

概要：モンテカルロ木探索は現在囲碁プログラムの主流であり，基本となるアルゴリズムにさまざまに工夫が加えられ用いられている．シミュレーション部分において，行動評価関数などを用いて良い手を高い確率で打つことは，全合法手を等確率で選ぶ場合に比べ効果的であることはよく知られる．この評価関数は木探索部分への利用も可能であり，有望な着手に探索を重点化したり，あるいはそれらのみに探索を着手限定したりといったことも行われる．本論文では，行動評価関数を木探索部分で活用する着手限定と重点化の 2 つの方法の効果やパラメータの影響，組合せた場合の性能を，囲碁プログラム Nomitan，Fuego を用いた実験により示す．そのうえで，着手限定で生じる “見落とし” を抑制するための 3 つの方法を提案し，Nomitan の Fuego に対する勝率が 4,000 試合ずつの実験で 57.7%から 64.5%に向上したことを示す．

キーワード：モンテカルロ木探索，行動評価関数，バイアス，Progressive Widening，見落とし

Knowledge Bias in Monte-Carlo Tree Search and Techniques for Reducing Oversight Mistakes

Kokolo Ikeda[1,a)] Simon Viennot[1,b)]

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

Abstract: Monte-Carlo Tree Search is now the most popular method for the game of Go, and many techniques and variations are used to improve the strength. It is already known that biased Monte-Carlo simulations using a probability model containing static knowledge are more efficient than random simulations. Such probability models can be also used in the tree search policy to bias the search or limit the search to a subset of the legal moves. In this article, first we describe more precisely how static knowledge can be used to improve the tree search policy. Then, we show how to reduce the oversight mistakes caused by the limitation of the number of searched moves. We confirm experimentally the efficiency of the proposed methods, with a large number of games using our Go program Nomitan, against Fuego, an open source program. The winning ratio of our program is increased from 57.7% to 64.5% (4,000 games for each).

Keywords: Monte-Carlo Tree Search, action evaluation function, bias, Progressive Widening, oversight

1. はじめに

チェスや将棋などのボードゲームのコンピュータプログラムでは，盤面（状態）を評価する**状態**評価関数を何らかの方法で設計したうえで，$\alpha\beta$ 法などの木探索を行うことで着手選択をすることが一般的である．囲碁ではこの状態評価関数を作成することの困難さからプログラムの強さ（棋力）の向上が遅れていたが，モンテカルロ法によって局面の良さを評価する試みが 1993 年 Brügmann らにより行われた [3]．さらに 2006 年 Kocsis らによって UCT（Upper Confidence bounds applied to Trees）[17] が開発され，モンテカルロ法と UCT の組合せであるモンテカルロ木探索（Monte Carlo Tree Search，MCTS と略す）の枠組みが有効であることが明らかになった．今日ほぼすべての強豪プログラムが MCTS を用いており，またその棋力も年々向

[1] 北陸先端科学技術大学院大学
JAIST, Nomi, Ishikawa 923–1292, Japan
[a)] kokolo@jaist.ac.jp
[b)] sviennot@jaist.ac.jp

上している．

MCTS は大きく木探索部分とシミュレーション部分に分けられる．シミュレーション部分での着手（打ち手，行動）は “ランダムに” 選ぶこともできるが，シミュレーションで得られた勝率が状態の評価として適切なものとなるためには，何らかのゲーム固有の知識や方策を導入して “現実的な” シミュレーションを行うほうが良い場合が多いことが次第に知られるようになった．その代表例は，強豪プログラム CrazyStone の作者 Coulom による Bradley-Terry Minorization Maximization（本論文では BTMM と略す）[6] である．BTMM は各着手の選択確率を求めるための枠組みで，着手の打たれやすさを決める**特徴量**の重みを棋譜から機械学習する．この選択確率はその着手（行動）の良さを表現するものとしても利用できるため，本論文ではこれを**行動**評価関数とも呼ぶことにする．CrazyStone のシミュレーションはこの選択確率に従って行われるが，現在多くの強豪プログラムがこれに類するシミュレーションを行っている．

ゲームに固有の知識や行動評価関数は，シミュレーション部分にだけでなく，木探索部分にも利用される．Coulom は，BTMM 法により求めた着手の評価関数をノード（局面）から**探索する着手の限定**に用い，形の悪い着手などの探索を後回しにするという **Progressive Widening** を提案した [6]．Chaslot ら [4] および Huang [12] は，UCT の着手選択指標である UCB 値に行動評価関数を用いた補正値（バイアスとも呼ぶ）を加えることで良さそうな手への**探索の重点化**を試みている．

しかしながら，これら Progressive Widening や UCB 値補正については，比較的広く使われ有効であることも知られている一方で [26]，学術的文献や技術公開は限定的であり，パラメータが性能に与える影響など詳細なデータは数少ない状態である [4], [15]．たとえば上記 Chaslot の論文では補正の式は書かれているものの行動評価関数の中身が十分記述されていない．また上記 Coulom の論文や Huang の論文も含め，手法を利用した場合と利用しない場合の比較しか行われておらず，パラメータを変更した場合に強さがどのように変わるのか，つまり敏感さに関する実験が示されていない．さらに二手法を使わなかった場合・単独で使った場合・両方使った場合すべてでの比較データも示されていない．

そこで我々は，これらの手法を改めて記述したうえで，囲碁の対戦実験を通じてその効果を確認し，パラメータを変化させることで勝率がどのように変化するかを調べること，2 つの手法を両方とも使った場合に最も効果的であることを示すことを第 1 の目的とする．実験には，これまで頻繁に用いられてきた九路盤ではなく，標準サイズである十九路盤により近い性質を持つと思われる十三路盤を用い，対戦相手には十分な強さを持つ公開プログラム Fuego を用いることで，その効果をより明確に示す．

さらに我々は，着手限定を行う Progressive Widening にとって宿命的な “良い手の見落とし” に着目する．一般に，静的な（探索なしの）評価関数の精度には限界があり，本当は打つべき手が何らかの理由で悪いと評価され探索から外れてしまうことは頻繁にある．そこで我々は，その理由を 3 つに分類し，それぞれについて，動的な情報などを利用して外れた手を探索範囲に含めなおす方法を提案し，見落としを抑制することを試みる．この効果も対戦実験により検証され，有意な勝率向上が確認された．本論文により，囲碁以外のゲームも含め，比較的原始的な MCTS の性能向上を図りたい多くの開発者にとって有益な情報が提供できると考える．

本論文の構成は以下のとおりである．まず背景として，2 章ではモンテカルロ木探索，3 章では Bradley-Terry モデルによる行動評価関数の設計とそのシミュレーションへの利用法，および行動評価関数を探索部に用いる既存手法を簡単に解説する．4 章は我々が用いるプログラム Nomitan と実験設定の説明である．続いて 5 章では，本論文での Progressive Widening の設定と，その有効性・パラメータの影響を示す実験を行う．さらに 6 章では，本論文での UCB 値補正の設定と，同様に有効性・パラメータの影響，さらに Progressive Widening と組み合わせた場合の性能を示す．そのうえで 7 章では，Progressive Widening で生じうる見落としに言及し，3 つの分類と対策を示し，有効性を確認する．8 章はまとめである．

2. モンテカルロ木探索

MCTS は，現在局面を表すルートノードから探索を開始し，ある基準によって “最も調べるべき” であるような行動（着手，枝）をたどり，遷移先に子ノードを拡張し，徐々に探索範囲を広げながら木を成長させる探索手法である．その最大の特徴は，葉ノードを評価するために状態評価関数ではなく，終局までの乱数を用いたシミュレーションを行い，その結果（通常は勝敗）を評価値として用いる点である．シミュレーション結果はルートノードからたどられたすべてのノードに蓄積されてゆき，各ノードのプレイヤにとっての勝ちやすさの推測に用いられる．

2.1 UCT algorithm

最も調べるべき行動（あるいは子ノード．着手と局面どちらに統計量を保存するかは実装により異なる）をどのように定めるかについては，さまざまな手法が提案・比較されている [19]．多くの場合，現在勝率が高い行動，または多少勝率が低くともまだ調査が不十分で過小評価されている可能性がある行動が，最も調べるべき行動であるとされる．これらを 1 つに表したのが UCB 値 [17] であり，多くの手法の土台となっている．着手 a_j の UCB 値 u_j は式 (1)

のように定義され，最大の u_j を持つ着手が選択される．

$$u_j = \frac{w_j}{n_j} + C \cdot \sqrt{\frac{\ln n}{n_j}} \quad (1)$$

ここで，n は親ノードの訪問回数（この局面をたどって探索が行われ，勝敗が蓄積された回数），n_j が着手 a_j の訪問回数，w_j が着手 a_j の勝利回数であり，第 1 項がその手を打った場合の推定勝率，第 2 項が訪問回数がまだ少ないことによる推定の不確かさ，いいかえれば勝率の上昇の余地を表す式となっている．C はその 2 つを調整する可変パラメータであり，このような調整を必要とする関係はしばしば収穫と探索のジレンマと呼ばれる．良いことが分かりつつある手をより深く読むのが収穫で第 1 項，より広い可能性を調べるのが探索で第 2 項と関係する．

UCT の普及から間もないにもかかわらず，木探索部分にも多くの工夫が試みられている．最も成功し頻繁に用いられるものの 1 つは Rapid Action Value Estimation（RAVE）と呼ばれる手法 [10] で，これは初期のモンテカルロ囲碁プログラムで用いられていた all-moves-as-first（AMAF）と呼ばれる手法 [3] の発展形である．基本的な考え方は，“多少異なる局面であっても，ある箇所への着手の価値はそう変わらないだろう” という信念に基づき，1 回のシミュレーションの結果（勝敗）を，そのたどってきたノードだけでなく，一部の兄弟ノードでも利用しようとするものである．

局面 s から見て「s に似た局面での手 a の良さ」を表す項は RAVE 項と呼ばれ，s における a の訪問回数が増えると影響が小さくなるように調整されることが多い．これは，訪問回数が小さいうちは RAVE 項によって手のだいたいの有望さを早く推定し，訪問回数が大きくなるほど “その局面 s における” その手 a の良さを高精度で推定することができるようになるためである．異なった局面での着手の価値をどの程度信用するかは状況による．志水らは，着手付近の類似度が低ければ RAVE 値に利用しないようなヒューリスティックを用いて性能向上を試みている [24]．

3. 行動評価関数とその利用

1 章でも述べたように，探索やシミュレーションを行わずに配石のパターンなどから着手の良さを評価する行動評価関数は囲碁プログラムで頻繁に用いられる．その代表が Coulom の用いた Bradley-Terry モデル（BT モデルと略す）であり [6]，なんらかの形で Zen [18]，Erica [12]，Aya [26] などの強豪プログラムでも使われている．本章ではその定義と利用例を紹介する．

3.1 一般論としての BT モデル

BT モデルは一般には，異なる強さ（γ 値と呼ぶ）を持つ複数のプレイヤからなるチームの総合的な強さを計算するモデルとして用いられる．まず単純に 2 人のプレイヤ 1，2 を考え，その強さをそれぞれ γ_1，γ_2 とするとき，プレイヤ 1 の勝率は $\gamma_1/(\gamma_1+\gamma_2)$ で表される．いいかえれば，勝率がそうなるように “強さ” の値 γ_1，γ_2 を定める．

続いて n 人のプレイヤからなるゲームに拡張し，その強さをそれぞれ $\gamma_1,\ldots,\gamma_n$ とするとき，プレイヤ i の勝率は $\gamma_i/(\sum_{j=1}^{n}\gamma_j)$ で表される．

さらに BT モデルでは，複数のプレイヤからなるチームの総合的な強さを，それぞれの強さの積で表す．たとえば，プレイヤ 1，2（それぞれ強さ γ_1，γ_2）のチームが，プレイヤ 3，4 のチーム，およびプレイヤ 5，6，7 のチームの中で優勝する確率は，$\gamma_1\gamma_2/(\gamma_1\gamma_2+\gamma_3\gamma_4+\gamma_5\gamma_6\gamma_7)$ で表せる．

3.2 着手選択確率決定のための BT モデル

Coulom は BT モデルを囲碁の着手の選択確率の決定，および順序付けのために用いた．ある局面において，すべての合法手はそれぞれ良さ（強さ）を持つチームで，“次に打たれることを競っている” ととらえることができる．さらに各合法手は，それぞれ良さを持つ**特徴量**（feature，前節の書き方ならプレイヤ）からなっていると考えることができる．囲碁の場合は特徴量として，周囲の配石パターン，アタリやヌキ[*1]など石の呼吸点に関するもの，最終着手の近くかどうかなど，さまざまなものが用いられ，その γ 値もそれぞれである．なお Coulom の論文では占有率（ownership）が特徴量の 1 つとして用いられているが，これは通常探索中に得られる動的な量であり，探索やシミュレーションなしで用いることはできない．

ある局面 s で全合法手 A_s の中から着手 $a^* \in A_s$ が選択される確率 $p(a^*)$ は，次の式で表される．

$$p(a^*) = \frac{\prod_{feature\ i\ \in\ a^*} \gamma_i}{\sum_{a\in A_s}\left(\prod_{feature\ i\ \in\ a} \gamma_i\right)} \quad (2)$$

図 1 は囲碁十三路盤における選択確率の例（次章で述べる我々のプログラム Nomitan のもの）である．ある程度囲碁の知識が必要だが，C9 ハネ，C12 下ハネ，D9 ノビなど有力な手が高い確率となっている．

3.3 BT モデルにおける γ 値の機械学習

各特徴量にどのような γ 値が適しているかは目的によって異なり，またその調整の方法もさまざまである．囲碁や将棋など強いプレイヤ同士の対戦記録（棋譜）が入手しやすいゲームでは，機械学習のアルゴリズムを用いて，棋譜に登場した局面で実際に打たれた手が良く評価されるように調整を行うことが多い．

1 つの局面 s に対して，実際に打たれた手 a^* の着手選択

[*1] 囲碁用語．次に相手の石を取る（通常得をする）という手をアタリ，実際に取る手をヌキと呼ぶ．

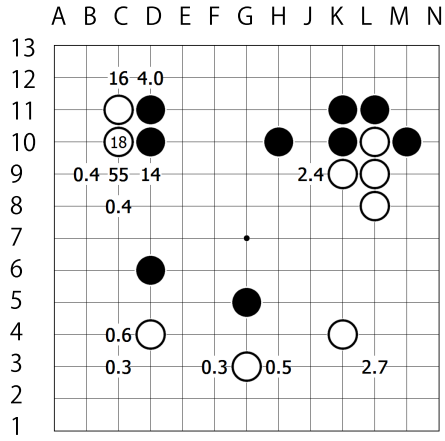

図 1 選択確率の例（黒番，単位%）．C10 が白の最終着手．
Fig. 1 Example of selection probability (Black, in %), after White C10. Low probability moves are not shown.

確率 $p(a^*)$ を最大化するように γ 値を設定することは容易である．しかし，n 個の局面と着手の集合 $\{(s_i, a_i^*)\}_{i=1..n}$ からなる棋譜を学習する場合には，その局面すべてについて実際の着手の選択確率を最大化することは通常不可能である（n が特徴量の数に比べ相対的に小さければ可能な場合もあるが，その場合は過学習が懸念される）．そのため，目的に応じてモデルに果たしてほしい役割を定め，それを反映したなんらかの総合的な指標を導入して最適化を行うことが一般的である．

Coulom は，“確率の大小” を重視した，尤度を用いた指標 $L(\{\gamma_i\}) = \prod_{i=1}^{n} p(a_i^*)$ を用いており，モンテカルロシミュレーション中に，大石のヌキ[*2]など必然の着手が高い確率で選択されるようになっている．

そのうえで Coulom は Minorization Maximization 法によって変数 $\{\gamma_i\}$ を最適化している．最適化法としては勾配法も用いられる [25] が，いずれにせよ，微分可能なように指標を設計しておくことが標準的である．

3.4 行動評価関数の利用

BT モデルには，囲碁に限っても，Coulom の論文 [6] を始めとしてさまざまな利用法が提案されている．

まず，着手 a の選択確率 $p(a)$ はそのままに MCTS のシミュレーション部分での着手選択に用いることができる．強いプレイヤの棋譜を機械学習した選択確率に従ったシミュレーションはランダムシミュレーションよりも現実的な経過をたどって終局に到達し，それゆえにより正確な勝率推定が可能になる．たとえば，高段者同士が打った場合には死んだものとして扱われる石はシミュレーション結果でも死に（取られ），地になりそうな模様（勢力）は最終的にも高確率で地になるために，葉ノードの局面の状況をより良く反映するということである．なお，一方で，現実的な経過をたどることに固執せずに，最終的な勝率がより正確であるように選択確率を調整する Simulation Balancing という手法 [14] も注目されている．

本論文では主に，着手の選択確率を行動評価関数としてとらえ，木探索部分の効率化に用いる 2 つの手法に注目する．1 つは各局面から探索する着手を有望なものに限定する Progressive Widening，1 つは UCB 値に補正値を加えることで形の良い手を優先的に探索する手法である．これらの手法の説明はそれぞれ次節および 3.6 節で，Nomitan における実験結果はそれぞれ 5 章および 6 章で述べる．

BT モデルの別の利用法としては，$p(a)$ が着手の一見した自然さを表すことを用いて，プレイヤを楽しませるために「見破られにくい手加減をする」というものもある [16]．

3.5 Progressive Widening による着手限定

ある局面での合法手数はゲームにより幅が広く，たとえば麻雀や大貧民では十数個，チェスやぷよぷよでは数十，将棋になると持ち駒の存在により 100 を超えることも珍しくなく，囲碁十九路盤では数百に及び，もちろんそれ以上のゲームもある．MCTS を囲碁に適用した場合，たとえば深さ 2 までの手を読んで 100 回ずつシミュレーションすればそれだけで 1,000 万回程度のシミュレーションが必要になるが，これは現在の一般的なコンピュータでのシミュレーション可能回数（4.2 節で詳述する我々の実験環境では数万回）を考えれば効率が良いとはいえない．

将棋では，行動選択確率 $p(a)$ を掛け合わせてある手順 $a_1 \to a_2 \to a_3 \ldots$ が実現する確率 $p(a_1)p(a_2)p(a_3)\ldots$ を求め，起こりにくいと予想される手順は後回しにする（あるいは読まない）実現確率探索 [21] がしばしば用いられる．

同様に Coulom は BT モデルで表現した行動評価関数 $p(a)$ を用いて，探索する手の数を制限する Progressive Widening を提案した [6]．Progressive Widening では，ある局面（ノード）の訪問回数 n に応じて，BT モデルの選択確率 $p(a)$ の上位の着手だけが探索され，結果として囲碁プログラムの強さが向上することが知られている．独立に Chaslot らも Progressive Unpruning を提案している [5] が基本的な考え方は同じである．

3.6 UCB 値の補正

2.1 節で述べた UCB 値の式には，さまざまな工夫が加えられてきた．その代表例は動的な情報を導入して探索序盤の推定を正確にしようとする RAVE 項であるが，行動評価関数値による補正を用いて，良いと思われる着手を優先的に探索することも行われる [23]．

Chaslot は手動および一部自動で調整した行動評価関数を用いて UCB 値を補正することで囲碁プログラムが強く

[*2] 大石とは多数の同色の石が上下左右につながったものを指す．それを取ることは通常大きな得になるので，ヌキは打ちたいし，防ぎたい．なお単に石といった場合，1 個の石の場合と，同色の石のつながったものの場合がある．後者は連とも呼ばれる．

なることを示している [4]．実際の式は以下のとおりで，通常の UCB 値の第 2 項が用いられない代わりに RAVE 項と行動評価関数 H による補正項が加わっているが，$H(a_j)$ の設計の詳細は示されていない．

$$u_j = \alpha \cdot \frac{w_j}{n_j} + \beta \cdot p_{rave}(a_j) + \left(\gamma + \frac{C}{\log(2+n_j)}\right) \cdot H(a_j) \tag{3}$$

論文の中で，行動評価関数による補正項と動的な情報を用いる RAVE 項とをうまくバランスさせることは難しく，十分な試行錯誤が必要であることが述べられているものの，パラメータを変化させた実験結果は示されていない．

囲碁では他に Erica [12]，Zen [18] などでこれに類する補正が行われていることが知られているものの，たとえば評価関数に BT モデルが用いられているのかなど，その詳細および効果は発表されていない．

多腕バンディット問題に対しては，行動評価関数にあたる予測値（predictor）を用いて UCB 値を補正する PUCB が提案され，木探索における実験はないものの，predictor の精度が収束に与える影響が理論的に解析されている [20]．

4. 用いるプログラム Nomitan と実験設定

本論文では，囲碁プログラム Nomitan を用いてさまざまな手法やパラメータの効果を調べる．Nomitan は Fuego [8] や Pachi [1], [2] などと異なりソースコードが公開されていないため，追試可能性といった論文としての客観性には欠けるが，比較的高精度な行動評価関数を持つため，本論文の趣旨からこれを用いることにする．本章では Nomitan の基本設計と，本論文で用いる実験設定についてまとめる．

4.1 実験設定

本論文ではほとんどの実験を，十三路盤を用いて，公開プログラム Fuego のバージョン 1.1 を対戦相手として行う．

九路盤を用いた実験はコストや解析の容易さの点で優れるが，最終的な目標である十九路盤に比べると合法手の数がかなり少ない．十三路盤は実際に著者らが対戦している感覚としても九路盤よりは十九路盤に近く，実験コストとのバランスも取れていると判断した．

また，提案手法あり/なしの自分のプログラムを対戦させること（自己対戦）もしばしば行われるが [4]，これは手法の良さを過大に評価してしまう傾向があり，本論文の実験としては適さないと判断した．Fuego は単純で高速なシミュレーションと，RAVE を利用した MCTS プログラムであり，Zen や CrazyStone などには及ばないが，それでもアマチュア有段レベルの十分強いプログラムである．

実験には 8 コア 16 スレッド（Intel Xeon L5520, 2.26 GHz, Dual CPU）のサーバを用い，一手の思考時間を 4 秒に固定した．Nomitan は探索木を共有する Tree 並列を行う．定石の利用や待ち時間の先読み（pondering）は行わない．試合数はほとんどの場合 600 以上，一部詳細な比較を行いたい実験では 4,000 試合まで行った．図 4 などでは，グラフ上に 95%信頼区間を表すエラーバーを示しているが，一部のエラーバーに長短があるのは主にこの試合数の差による．

4.2 Nomitan の基本構造

Nomitan での着手決定は MCTS を基本とし，機械学習によって得られた BT モデルの選択確率 $p(a)$ に従って着手を進めるシミュレーション部と，$p(a)$ を用いて Progressive Widening や UCB 値補正を行う木探索部からなる．UCB 値の補正は比較的最近に追加されたもので，それまでアマチュア級位者レベルであったものが現在では KGS（Kiseido Go Server）で 3d つまり平均的なアマチュア四段程度にランクされるまでになっている．

他の強豪プログラムと比較した際の Nomitan の特徴は，

(1) シミュレーションに大きな配石パターンを用いているため遅い，
(2) RAVE を利用していない，

という点にあると考えているが，これが良い選択だと主張するつもりはない．

他のプログラムではシミュレーションに 3×3 のパターンを用いることが多いようだが，Nomitan では周囲 36 マス（7×7 からカドの一部を取り除いたもの）のパターンを用いている．これにより選択確率 $p(a)$ の精度が向上する一方で，速度が犠牲になる．前節の設定のもとでの序盤のシミュレーション回数は約 2 万回であり，これは Fuego の約 14 万回や Pachi の約 17 万回に比べると相当に劣る．

RAVE を利用しない理由は主に “多少異なる局面であっても，ある箇所への着手の価値はそう変わらないだろう” という信念への疑いである．探索の挙動に複雑な影響を与えうる RAVE をあえて採用せず，勝率推定の補助には，本論文で詳述する UCB 値の補正を用いたいと考えている．

4.3 Nomitan の行動評価関数

Nomitan では BT モデルによる行動評価関数を 3 つの方法で利用しているが，それらはほぼ同じ特徴量セットと学習法・学習データを用いて最適化されている．主な要素は以下のとおりである．

- **学習データ**：Aya の作者山下氏から提供された 4 万枚（うち 1 万枚はテスト用）のプロ棋士の棋譜と，KGS から入手した約 27 万枚の 1d 以上のプレイヤ同士の対戦棋譜を用いる．これらは十九路盤の棋譜であり，十九路盤用の行動評価関数が学習されるが，十三路盤でも同じものを用いる．
- **目的関数**：3.3 節で述べた尤度の対数平均（Mean Log Evidence, MLE）を目的関数として最大化する．

Nomitan の設計を記した昔の論文 [25] ではボナンザ型の目的関数が用いられているが，性能比較などを経て現在は MLE を用いている．

- **学習法**：勾配法の一種を用いる．偏微分値が正の特徴量では γ 値を一定倍し，負の特徴量ではその逆を行う [25]．出現頻度の低い特徴量に極端な γ 値が割り当てられないよう，正則化項 [22] を導入している．
- **特徴量**：特徴量の多くは，Coulom の論文 [6] と同様あるいはその変種である．具体的には，盤上の位置と周囲の配石パターン，1 手前および 2 手前の着手からの距離，隣接する石の呼吸点の変化（アタリ，逃げ，ヌキなど）である．Coulom のものと大きく異なる点として，取得コストの高い ownership は用いていない．

1 万枚のテスト用棋譜を用いて評価した汎化性能は，一致率つまり BT モデルの最良手と棋譜の手が一致した割合が 37.2%，平均順位つまり棋譜の手が BT モデルの評価値で何位であるかの平均値が 10.8，平均選択確率つまり棋譜の手 a^* の選択確率 $p(a^*)$ の平均値が 23.1%となっている．

次章で述べる Progressive Widening では，良い手が BT モデルの上位たとえば 15 位以内に入っていることも重要である．**図 2** は横軸が順位 T，縦軸がその順位以内までに棋譜で打たれた手が入っている確率（割合）を示したもので，実線が Coulom のデータ [6]，破線が Nomitan のものである．対象とした棋譜の種類，数，学習条件などは異なるが，ownership を使わずにほぼ同程度の性能が得られている．

点線は，“1 手前および 2 手前の着手からの距離” に関する特徴量を用いずに学習した場合の結果である．T 位以内率は 10%程度あるいはそれ以上悪化しており，この特徴量が精度の高いモデルのために必要であることが見て取れる．これは，囲碁ではほとんどの場合 “現在の局面だけを見れば次の良い手が決まる” という Markov 性が成り立つことからすると意外な結果であり，逆にいえば，配石パターンや呼吸点の変化だけを見ている現在の行動評価関数の限界を示唆している．

5. Progressive Widening の効果

3.5 節で述べたように，行動評価関数を用いて着手を限定する Progressive Widening が有効であることは良く知られている．Progressive Widening では，ある局面（ノード）の訪問回数 n に応じて，BT モデルの選択確率 $p(a)$ の上位 $T(n)$ 位までの着手だけが探索される．$T(n)$ の定め方はさまざまであるが，Nomitan では Aya のもの [26] と近い，$T(n) = 1 + \log(n/10)/\log\mu$ という式を用いている．

μ はパラメータで，これを変更した場合に，どのくらいの訪問回数でどのくらいの数の手だけに限定するのかを **図 3** に示す．たとえば $\mu = 1.5$（図中の縦線）を用いると，200 回訪問時点では上位 8 手のみ，2 万回では上位 20 手ほどを探索することになる．4.1 節の実験設定では Nomitan は 2 万回ほどしかシミュレーションを行わないので，この上位 20 手の中に含まれなかった手の中に良い手があれば “見落とし” が生じる．詳細は 7 章で述べる．

前のページになるが，図 1 は実戦で現れた局面で，$\mu = 2.0$ の場合に探索された 12 手を表したものである．この場合はあえていえば他には右辺 L6 打ち込み，L4 ツケくらいがあるかもしれないが，おおむね調べるべき手は含まれているといってよいと考える．

5.1 実験結果

Progressive Widening による着手限定は，The Many Faces of Go [9] や Aya [26] など多くのプログラムで利用され，かつ性能向上がもたらされていると想像されるにもかかわらず，その効果の程度やパラメータ μ の与える影響について実験した論文は少ない [15]．本論文でこれを示すことは囲碁プログラマ，MCTS 利用者にとって意義が大きいことであると考える．

図 4 は，パラメータ μ の値を変えたときに Nomitan の性能がどう変化するかを Fuego に対する勝率を用いて調べ

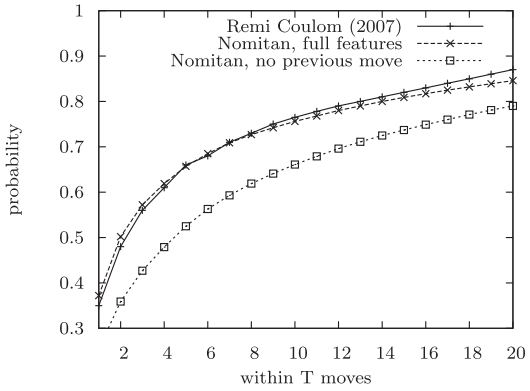

図 2 棋譜の手が，BT モデルの T 位以内に入っている確率
Fig. 2 Probability of finding a move from the game records in the top T moves of BT model.

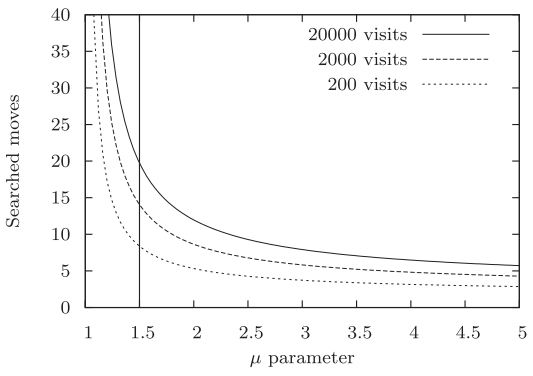

図 3 μ の値を変えたときの探索着手数．訪問回数 200 の場合（点線），2,000 の場合（破線），2 万の場合（実線）
Fig. 3 Searched moves in function of parameter μ.

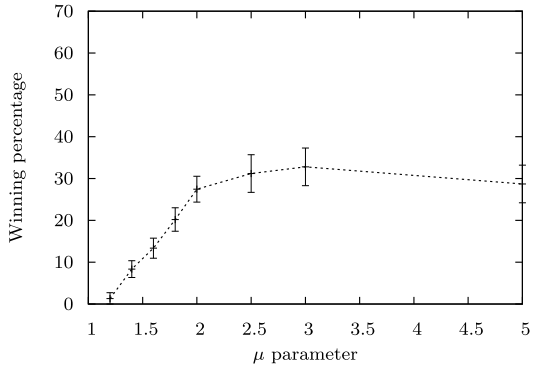

図 4 μ の値を変えたときの Fuego に対する勝率．エラーバーは 95%信頼区間を表す（以降の図も同じ）．
Fig. 4 Winning rate against Fuego in function of μ.

たものである．横軸は図 3 とも対応している．次章以降で述べる UCB 値の補正などの手法は使っていない．

μ が小さすぎる場合たとえば $\mu = 1.2$ の場合には，訪問回数が 200 でも 40 程度の手が調べられてしまい，深い探索ができずに，Fuego に対して数%程度しか勝てない．μ を大きくするに従い，つまり探索を限定するに従い，その勝率は向上し，$\mu = 3.0$ のときに勝率 33%で最良となっている．さらに $\mu = 5.0$ と大きくすると，訪問回数 200 では 3 手，2 万でも 6 手までに探索を限定し，見落としはより頻繁に発生するが，探索もより深くなるため，勝率は 29%までしか落ちていないと解釈できる．

探索を限定することは，単にその分の探索資源を有望な手に集中できるということにとどまらず，勝率のより正確な推定にも役立つことを強調したい．たとえば，大石に対するアタリが打てる局面を考える．アタリに相手が正しく対応すれば勝率は 50%，相手が対応を誤れば勝率が 75%になるとする．UCB 値を考慮すると，正しい対応手が 1 万回訪問されるとき，それ以外の手もそれぞれ 100 回ほど探索されることになる[*3]．もし探索の限定を十分に行わずに，それ以外の手が 100 手探索されるとすると，アタリを打った場合の勝率は 62.5%と著しく過大評価されてしまう．このような過大評価の抑制にも，探索の限定は効果がある．

6. BT モデルを用いた UCB 値の補正

3.6 節で述べたように，行動評価関数を用いて UCB 値を補正する試みはいくつか提案されている．本論文では，Nomitan を用いて，BT モデルを用いた場合の UCB 値の補正の効果，さらには Progressive Widening と組み合わせた場合の影響を実験を通じて示す．Nomitan の木探索部でノード選択に用いている式は UCB を変形した以下のものであり，RAVE 項がないこと以外は Chaslot のもの [4]，Huang のもの [12] と似ており，C は 0.9 で固定する．

[*3] 総訪問回数が 2 万とすると，正しい対応手の UCB $= 0.5 + 0.9\sqrt{\ln(20000)/10000} \fallingdotseq 0.528$，それ以外の対応手の UCB $= 0.25 + 0.9\sqrt{\ln(20000)/100} \fallingdotseq 0.533$.

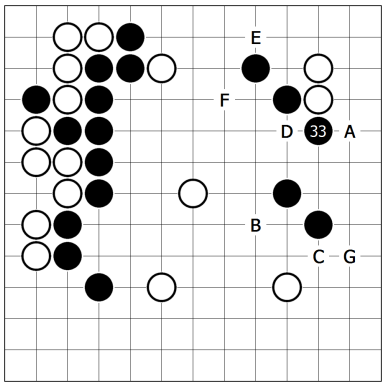

図 5 右上をハネられた局面の，$p(a)$ 上位の手．
Fig. 5 Top $p(a)$ moves after Hane 33.

$$u_j = \frac{w_j}{n_j} + C \cdot \sqrt{\frac{\ln n}{n_j}} + C_{BT} \cdot \sqrt{\frac{K}{n+K}} \cdot p(a_j) \quad (4)$$

第 3 項が補正項である．C_{BT} は補正の大きさを調整するパラメータ，$p(a_j)$ は 4.3 節で述べた，BT モデルによる着手 a_j の評価関数値である．K は訪問回数によって補正量を減衰させるためのパラメータで，本論文では $K = 600$ を固定値として用いる．これは訪問回数 $n = 1{,}800$ になれば補正量が $n = 0$ のときの半分になるということである．

6.1 UCB 値補正の効果

式 (4) のように $p(a)$ を用いて UCB 値に補正を加えることで，形の良い手，ヌキやツギなど急がれる手の探索を優先することができる．これにより期待できる恩恵には一般的に次のようなものがある．

- 訪問回数が少ない探索序盤でも，比較的起こりうる変化に探索資源が集中し，勝率の精度が高まる．
- 勝率が同程度なら，形の良い手が優先的に探索され，結果として最終的にも選ばれる．特に対局序盤で少々形の悪い手でも大きな勝率差が出ない局面では，形の良い手を優先しておくほうが良いことが多い．
- 大石のアタリに対するツギなど必然の手に，勝率差によるもの以上に探索が集中する．これにより探索が深くなり，また 5.1 節最後にあげた問題をも低減できる．

2 番目の恩恵について，Fuego との実戦例を**図 5** に示す．右上を黒にハネられた局面で，自然な着手は白 A のハネである．**表 1** にこのときの Nomitan の上位 7 つの手の評価・探索状況をまとめる（$C_{BT} = 0.6$）．最も勝率の高い手は B カドに打つ手だが，上位の手の勝率に大きな差はなく，乱数の揺らぎを考えればどの手が選ばれてもおかしくない．たとえば訪問回数が 2,500 回だとすれば，確率変動や騙し構造 [13] がなかったとしても 2 ポイントくらいの揺らぎは見込まなければならない．

ところが，$p(a) = 0.627$ という大きな選択確率による補正が行われた結果，A のハネは 1 ポイント勝率が悪いにもかかわらず B の 7 倍以上も訪問されて，Nomitan により

表 1 ルートノードで探索された手の一部，勝率，訪問回数，$p(a)$ 値
Table 1 Moves searched in root node and their statistics.

着手	勝率（%）	訪問回数	$p(a)$（%）
A ハネ	55.8	18,102	62.7
B カド	56.8	2,525	1.1
C コスミツケ	55.7	1,849	1.4
D キリ	55.1	1,714	4.9
E ツケ	53.9	1,460	11.1
F カタ	54.4	1,362	1.1
G ケイマ	54.0	1,150	0.8

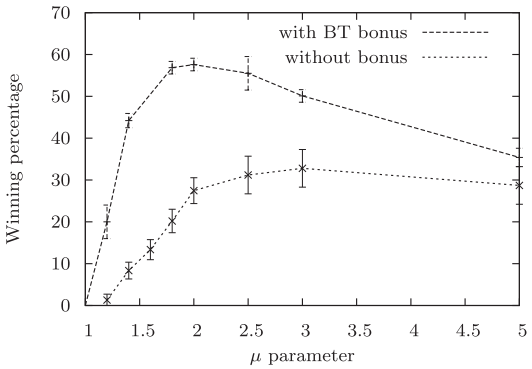

図 6 μ の値を変えたときの Fuego に対する勝率．補正なし（点線），補正あり（破線）
Fig. 6 Winning rate with and without bonus.

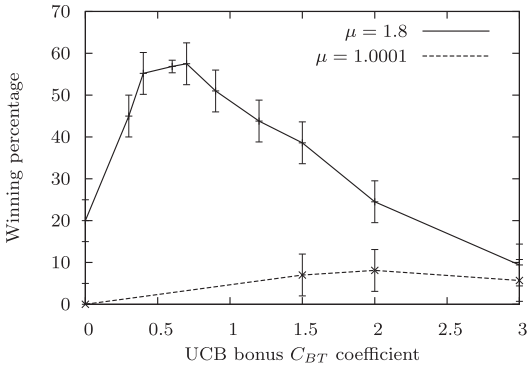

図 7 C_{BT} の値を変えたときの Fuego に対する勝率．Progressive Widening なし（点線），あり（実線）
Fig. 7 Winning rate in function of bonus coefficient.

打たれている．このように，自然に見える手を打つことは多くの場合良い方向の偏りとなっている．

図 4 の実験結果に加えて，UCB 値の補正 $C_{BT} = 0.6$ を加えた場合の Fuego との対戦結果を**図 6** に示す．$\mu = 3.0$ で最大約 33%だった勝率は $\mu = 1.8$, 2.0 で約 58%まで向上し，UCB 値の補正の効果は非常に大きいことが分かる．

図 7 は，横軸を C_{BT} に取り，UCB 値の補正の大きさを変化させた場合の性能を調べたものである．実線は Progressive Widening あり（$\mu = 1.8$）の場合で，C_{BT} が 0.4 から 0.7 の比較的広い範囲で勝率 60%近い良い性能が得られている．C_{BT} が小さすぎると恩恵を十分に得られず，大きすぎると “一見良いだけ” の手への強すぎる偏りにより性能が劣化する．破線は Progressive Widening なしの場合で，非常に C_{BT} を大きくした場合に若干性能向上するが，それでも勝率は 10%未満である．

この 2 つの図から分かることは，Progressive Widening と UCB 値補正はどちらか 1 つよりも組み合わせた場合に良い性能が出て，また良いパラメータの範囲も比較的広い，緩やかな mountain curve を描くということである．組み合わせて使った場合に，好ましい μ と C_{BT} の値が単独の場合より小さくなるのも，両者の効果が探索の集中という意味で似ていることから，自然な結果であると考える．

7. Progressive Widening による見落としとその抑制

前章の実験では，$\mu = 2.0$ 程度のパラメータによって Progressive Widening を用いて着手限定することが有効であることを示した．これは 2 万回の訪問ならば 12 手しか探索しないということである（図 3 および図 1 参照）．図 2 によれば，棋譜の手が BT モデルの上位 12 位以内に入る割合は十九路盤では 8 割ほどに過ぎず，多くの良い手・必要な手が探索されていないことになる．このような Progressive Widening の副作用・デメリットを，本論文では**見落とし**（oversight）と呼んだうえで，いくつかに分類し，それぞれに問題を軽減するための手段を提案する．

7.1 見落としの分類

Nomitan の敗因を分析すると，多くは死活や石の強弱の見誤りか，見落としに由来することが分かった．前者は主にシミュレーション部の問題として，後者の中にも，おおむね以下の 3 つの見落とし方があることが分かっている．

- **偽りの急所**．石が多く接触すると，仮にその周辺がもはや重要な領域でなくとも，局所的には良い形に見える箇所が多くなる．その場合，本来着手を急ぐべき箇所（大場や急所と呼ばれる）が見落とされる．
- **攻め合い**．攻め合い[*4]の呼吸点に打つ手は形だけ見ると良くない場合が多く，しばしば見落とされる．勝てる攻め合いに負ければ，それだけで容易に試合全体の負けにつながる．
- **偽りの先手**．4.3 節最後に述べたように，直前手からの距離は重要な特徴量で大きな γ 値を持つ．このため，自分の着手の周囲に相手が応手することを期待しすぎ，それと離れた相手の良い手をしばしば見落とす．

次節以降，それぞれについて具体例と対応策を示していく．対応策はすべて，“Progressive Widening によって探索から外してしまった着手を条件に従って**再導入**（reintroduction）する” という形式で行われる．

[*4] 囲碁用語．黒白の石の連なりが接しており，どちらかが取られどちらかが助かり，それを部分的に争っている状態．どちらも助かりうる場合は攻め合いと呼ばない．

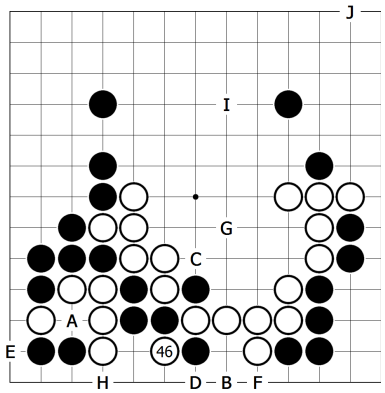

図 8 黒番．下辺に価値がなく，上辺の価値が高い局面．$p(a)$ 上位の手 A〜H と，criticality の高い手 I，J．

Fig. 8 Black turn. Board top has high value. A–F: high $p(a)$, I, J: high criticality.

7.2 偽りの急所

図 8 に，Fuego との実戦から偽りの急所の例を示す．左下方面で戦いがあり石が込み合っているが，白 46 に打たれた時点でこの周辺に価値の高い手はなくなっており，上辺に着目すべきである．ところが，$p(a)$ 最良の手は A で，以下順に H までの 8 手はすべて下辺に集中しており，これらはどれも実際には価値の低い手である．

そこで我々は，「各地点を支配することが勝敗に影響するか」を表す criticality [7] という動的な統計量を各ノードに保存し，それを用いて以下の手順で探索着手の再導入をすることにする．図 8 では，上辺の criticality は 0.15 程度と高く，下辺中央はほぼ 0 である．

(1) 終局時に，criticality を計算するための情報（勝者側が支配していた地点）を計算し，各ノードに保存する．
(2) 探索中各ノードを 200 回訪問するごとに，すべての着手に対し $g(a) = f(criticality(a)) \times p(a)$ を計算し，ソートする．$f(x)$ は**図 9** で表す，criticality が大きいほど大きくなる補正用関数である．
(3) $g(a)$ の上位一定数の手を（探索範囲に含まれていなければ）再導入する．具体的には，Progressive Widening による制限数と同じ $T(n) = 1 + \log(n/10)/\log\mu$ 個の手を再導入する．$p(a)$ 上位の手と $g(a)$ 上位の手は重複していることが多いので，実際には数個程度しか探索範囲が増えないことも多い．

$f(criticality(a))$ だけを用いず $p(a)$ と掛け算している理由は，前者だけだと，たとえば図 8 の J のような部分的にあり得ない手まで含まれてしまうからである．$p(a)$ もそこそこ高く，criticality も高い手のみを再導入すべきである．再導入の頻度や個数はパラメータであるが本論文では上記のように固定した．この例では，I のシマリが $p(a)$ の順位 20 位ながら勝率 50.8%で最終的に選ばれた．一方 A から H の中では A の勝率 45.6%が最善であり，$p(a)$ が良いからといって良い手とは限らないことが分かる．

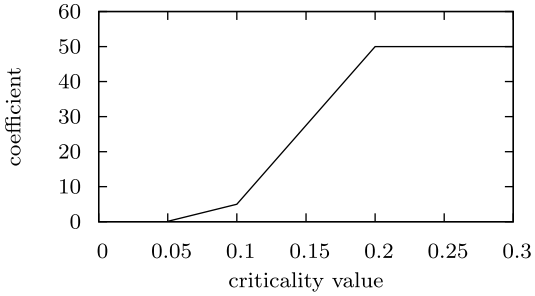

図 9 criticality による評価値の補正を行う関数 f

Fig. 9 Correction coefficient in function of criticality.

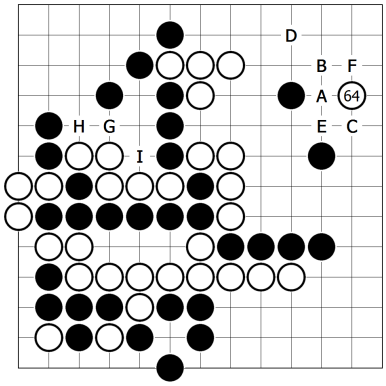

図 10 中央左側，黒 8 子と白 5 子の攻め合い．$p(a)$ 上位の手 A〜F と，再導入された手 G，H，I．

Fig. 10 Semeai between 8 black stones and 5 white stones. A–F: high $p(a)$, G, H, I: semeai moves.

表 2 各着手の $p(a)$ 値，その順位，探索した場合の勝率，訪問回数

Table 2 For each move, $p(a)$ value, rank, winning ratio, visits.

着手	$p(a)$（%）	$p(a)$ 順位	勝率（%）	訪問回数
A ブツカリ	24.2	1	70.3	2,266
B コスミ	23.4	2	70.7	2,497
C コスミツケ	14.2	3	69.0	1,309
D トビサガリ	8.4	4	69.0	1,115
E ナラビ	6.9	5	70.6	1,634
F ツケ	4.5	6	71.1	1,754
G ブツカリ	0.10	28	74.9	6,870
H マガリ	0.13	27	73.5	3,534
I アテコミ	0.04	42	72.3	2,440

7.3 攻め合い

図 10 に，Fuego との実戦から攻め合いの例を示す．左辺黒 8 子（このように上下左右につながった同色の石の集まりを連と呼ぶ），白 5 子が攻め合いの状態にあるが，白番の Fuego は右上 64 と別方面に着手した．部分的には，A のブツカリなどで応える形だが，攻め合いに勝つ G などの手も大きい．**表 2** に各手の行動評価関数値 $p(a)$ とその順位を示す．$p(a)$ が上位の手はほとんどが右上隅に集まっており，G，H，I など攻め合いに勝つ手は最良でも 27 位と，探索範囲から除外されてしまう．

そこで我々は，動的な統計量から攻め合い状態にある黒

白の石の連のペアを検出し，その呼吸点を探索範囲に再導入することにする．

(1) 終局時に，「各地点を黒が支配するか白が支配するか」を調べ，ownership として各ノードに保持する．

(2) さらに，「各地点とその上下左右それぞれが同じ色の支配下にあるか」を調べ，仮に運命共同値と呼んで各ノードに保持する．
- 具体的には，“右と同じ色であった回数”，“下と同じ色であった回数” などが保存され，これをシミュレーション回数で割ることで，運命共同値が計算される．
- たとえば 200 シミュレーション中 20 回しか同じ色の支配下にならなかったペアは運命共同値は 0.1 であり，共に強い（生き残りそうな）白黒の石が接している場合にはこのような小さい値になる．
- 逆に 200 シミュレーション中 180 回同じ色の支配下になった場合は運命共同値は 0.9 であり，白黒のペアが高い運命共同値を持つ場合はどちらかが取られる可能性が高いことを意味する．

(3) 探索中各ノードを 1,000 回訪問するごとに，次の条件を満たす隣接する白黒の石の連のペアがないか調べる．
- 運命共同値が 0.9 以上．つまりどちらかが取られる可能性が高い．
- ownership が 0.15 以上．つまりどちらかの石の連が弱すぎない．

(4) 条件を満たすペアがあったら，黒番のノードではその白石の連の呼吸点をすべて探索範囲に再導入する．白黒逆の場合も同じ．

アルゴリズム中の再導入頻度，運命共同値や ownership の閾値はパラメータであるが，本論文の実験では固定した．

たとえば図 10 では，白 5 子と黒 8 子の ownership は 0.33 と 0.73，運命共同値は 0.94 で条件を満たし，G，H，I が再導入される．表 2 の右側には，再導入して探索した場合の勝率・訪問回数を示す．G，H，I は探索さえされれば勝率が高いことが分かる．一般に呼吸点 3 以上の攻め合いを探索なしで検出することは困難で，他にも検出法が提案されている [11]．

7.4 偽りの先手

図 11 に，Fuego との実戦から偽りの先手の例を示す．今黒 A にツがれた場面であり，右下白を安全にする B ツギ（$p(a) = 21.7\%$で 1 位）あるいはその右のカケツギ（$p(a) = 13.8\%$）が普通の手である．ところが，Nomitan はしばしば C ツケや K ブツカリを打ってしまう．これは，それらの手以降の探索で黒 B に打たれる手が見落とされているためである．実際，白が C ツケを打った局面では，黒の $p(a)$ 上位の手は D から順に Q までで，B は 15 位と低い．

そこで我々は，“敵の急所は我が急所” という囲碁の格言に基づき，親ノードで最良の $p(a)$ 値を持つ地点は，子ノードでも探索すべきという仮説を立て，探索範囲に必ず再導入することにした．この場合なら，黒 A を打った局面では白 B が最良の $p(a)$ を持つので，白 C の後でも，黒 B を探索範囲に含めるということである．この局面では，再導入を行わない場合 50 回中 50 回，C など悪い手を打ってしまうが，再導入を行った場合は B ツギが 3 回，右にカケツギが 47 回で，すべて右下を守ることができた．

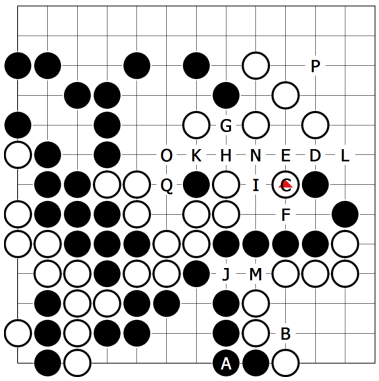

図 11 相手の手を見落とす例．A は直前手，B が必要な手，C が打ってしまった手，D〜Q は黒の $p(a)$ 上位の手．

Fig. 11 Oversight after Black A. C is played instead of B because D–Q have high $p(a)$.

7.5 再導入の効果

前節まで，Progressive Widening による見落としを防ぐ 3 つの再導入方法を提案してきた．**図 12** には，図 6 に加え，この 3 つの再導入手法を加えた場合の Fuego に対する性能を示す．最良の勝率はどちらも $\mu = 2.0$ の場合で，4,000 試合の勝率は 57.7%から 64.5%に 6.8 ポイント向上し，有意に良い結果となった．

3 つの導入手法それぞれについて，その有無がどの程度性能に影響を与えるのかも限定的な試合数ではあるが調査した．各実験は 4,000 試合行い，この場合の 95%信頼区間は約 ±1.6%である．結果を**表 3** にまとめる．再導入をまったく行わない場合（番号 1）から，偽りの急所（番号 2），攻め合い（番号 3），偽りの先手（番号 4）一種類のみの再導入を行った場合の勝率の向上はそれぞれ 5.0 ポイント，0.6 ポイント，1.3 ポイントであった．逆にすべてを使う場合（番号 8）から，一種類のみを使わない場合（番号 5，6，7）の勝率の低下はそれぞれ 4.0，1.3，2.2 ポイントであった．これらの結果から，偽りの急所に関する再導入が最も効果が大きく有意に勝率向上の効果があることが分かり，攻め合いと偽りの先手についても，両方を使った場合には有意に性能が向上し（番号 1 と 5 の比較），単独でも有意ではないながらも効果がある可能性が高いことが推測できる．

攻め合いに関しては番号 3 の場合 4,000 試合中 990 試合（のべ 1,228 回）で再導入した手が着手されている．勝率が

表 3 偽りの急所，攻め合い，偽りの先手それぞれについて，再導入ありの場合となしの場合の，Fuego に対する勝率

Table 3 Winning rate against Fuego with and without re-introducing moves for criticality, semeai, false sente.

番号	偽りの急所	攻め合い	偽りの先手	勝率
1	なし	なし	なし	57.7%
2	あり	なし	なし	62.7%
3	なし	あり	なし	58.3%
4	なし	なし	あり	59.0%
5	なし	あり	あり	60.5%
6	あり	なし	あり	63.2%
7	あり	あり	なし	62.3%
8	あり	あり	あり	64.5%

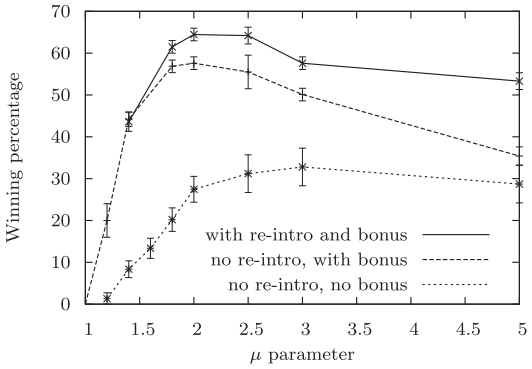

図 12 μ の値を変えたときの Fuego に対する勝率．補正・再導入なし（点線），補正のみあり（破線），補正・再導入あり（実線）

Fig. 12 Winning rate against Fuego with re-introduction of moves (top line).

大きく向上していないのは，提案手法で攻め合いの見落としが減らないということではなく，攻め合いと関係なく勝負がついている試合が多かったり，再導入のためのコストや探索資源の分散による全体的な性能低下があったりするためだと考えている．図 10 のように級位者でも分かるような攻め合いを正しく処理できないことは，単に強さの問題だけでなく人間プレイヤを楽しませる・教えるといった目的にも有害であり，再導入には価値があると考える．

十九路盤でも，Progressive Widening のパラメータ $\mu = 2.0$，UCB 値補正のパラメータ $C_{BT} = 0.8$，$K = 400$ に固定してではあるが実験を行った．一手 10 秒で Fuego と 1,600 戦以上対戦させたところ，再導入ありは勝率 66.7%，再導入なしは勝率 63.1%と，再導入ありのほうが強くなるという結果となった．なお十九路盤では序盤に数手〜十数手の opening book を用いているが結果に与える影響は小さいと考える．

7.6 見落としを抑制する方法に関する考察

本論文では，Progressive Widening による見落としを抑制する方法として，行動評価関数そのものは変えずに，特定の条件を満たす着手を再導入するというアプローチを取った．しかしそもそも，行動評価関数にこれらの着手が含まれるように特徴量の決定と学習を行えば，このような 2 段階の方法は必要ない可能性がある．本論文は再導入が唯一のあるいは最善のアプローチであることを主張するものではなく，少なくとも再導入という比較的簡単な方法によって大きな効果が得られることを主張するものである．

評価関数の学習に動的な情報を用いることは，BT モデルにおいて ownership 情報の利用がすでに行われており [6]，同様の方法で，criticality や攻め合いの情報を特徴量として用いることは原理的には可能である．しかしそのためには学習の前あるいは途中でこれらの値を得るための十分な回数のシミュレーションが必要になり，プログラムが複雑になるほか，シミュレーションのコストや棋譜枚数によっては非常に長い学習時間を要する点も問題になる．

もう 1 点，BT モデルを使った場合に限れば，新しい特徴量に大きな重みが学習されても，配石パターンなど他の特徴量の重みが非常に小さい手については上位に来にくいという問題がある．たとえば図 10 で，I のような手は攻め合い以外ではまず打たれないため，重みも非常に小さくなる（我々が用いた約 30 万枚の棋譜中，同じ配石パターンは 2,658 回登場し，1 回も打たれていない．重みは 0.07）．この重みと “乗算して” F と並ぶまで上位に持ってくるためには，攻め合い特徴量には 100 を超える重みが求められる．しかし，実際 Nomitan にはヌキなども含めても 100 以上の重みは学習されておらず，攻め合い特徴量を仮に導入しても見落としを防ぐのに十分ではないと予測される．

学習時間に関する問題，モデルの限界に関する問題は，どちらも解決不可能なものではないはずで，結果として再導入を用いるよりも特徴量に組み込んで学習するほうが性能が良くなることは十分ありうるし，今後そのような研究・開発は我々も行っていく予定である．しかしながら，そのような方法を用いずとも，比較的簡単な再導入という方法によっていくつかの見落としが防げ，性能がはっきりと向上することを示したことには，意義があると考えている．

8. おわりに

本論文では，MCTS の木探索部を効率化するための行動評価関数の 2 つの利用法 Progressive Widening と UCB 値補正を説明したうえで，公開囲碁プログラム Fuego を対戦相手に，十三路盤による詳細な実験を行った．その結果，比較的広い範囲のパラメータでそれらが有効であること，かつ両方用いた場合に最も効果的であることを明確に示した．その上で，Progressive Widening で着手を限定しすぎることによる見落としを分類しての対策を提案し，Fuego に対する勝率がそれぞれ 4,000 試合ずつで 57.7%から 64.5%に向上したことを報告した．

囲碁のトッププログラム群はその内容や実験結果をあまり公表してこなかったため，それらよりは弱いとはいえ

KGS 3d の強さがある Nomitan でこれら手法の効果を検証し詳細を発表したことには，大きな価値があると考える．7 章で述べた再導入の基準は囲碁特有のものが多いが，着手を限定したうえで必要なものを動的に再導入するという基本的な考え方は他のゲームにも応用できると考える．

参考文献

[1] Baudis, P.: MCTS with information sharing, Master Thesis, Charles University in Prague (2011).
[2] Baudis, P. and Gailly, J.: Pachi: State of the Art Open Source Go Program, *Advances in Computer Games, LNCS*, Vol.7168, pp.24–38 (2012).
[3] Brügmann, B.: Monte Carlo Go, Technical Report (1993).
[4] Chaslot, G., Fiter, C., Hoock, J.P., Rimmel, A. and Teytaud, O.: Adding expert knowledge and exploration in Monte-Carlo Tree Search, *Advances in Computer Games, LNCS*, Vol.6048, pp.1–13 (2010).
[5] Chaslot, G., Winands, M., Uiterwijk, J., Herik, H.V.D. and Bouzy, B.: Progressive Strategies for Monte-Carlo Tree Search, *New Mathematics and Natural Computation*, Vol.4, No.3, pp.343–357 (2008).
[6] Coulom, R.: Computing Elo Ratings of Move Patterns in the Game of Go, *ICGA Journal*, Vol.30, No.4, pp.198–208 (2007).
[7] Coulom, R.: Criticality: A Monte-Carlo Heuristic for Go Programs, Invited talk at the University of Electro-Communications, Tokyo, Japan (2009).
[8] Enzenberger, M., Müller, 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 (2010).
[9] Fotland, D.: Computer Go mailing list への投稿 (2009). 入手先 ⟨http://www.mail-archive.com/computer-go@computer-go.org/msg12628.html⟩
[10] 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).
[11] Graf, T., Schaefers, L. and Platzner, M.: On Semeai Detection in Monte-Carlo Go, *Conference on Computer and Games* (2013).
[12] Huang, S.C.: New Heuristics for Monte Carlo Tree Search applied to the Game of Go, Ph.D. Thesis, National Taiwan Normal University (2011).
[13] Hashimoto, J., Kishimoto, A., Yoshizoe, K. and Ikeda, K.: Accelerated UCT and Its Application to Two-Player Games, *Advances in Computer Games, LNCS*, Vol.7168, pp.1–12 (2012).
[14] Huang, S.C., Coulom, R. and Lin, S.S.: Monte-Carlo Simulation Balancing in Practice, *Advances in Computer Games, LNCS*, Vol.6515, pp.81–92 (2011).
[15] Ikeda, K. and Viennot, S.: Efficiency of Static Knowledge Bias in Monte-Carlo Tree Search, *Conference on Computer and Games* (2013).
[16] Ikeda, K. and Viennot, S.: Production of Various Strategies and Position Control for Monte-Carlo Go - Entertaining human players, *IEEE Conference on Computational Intelligence in Games*, pp.145–152 (2013).
[17] Kocsis, L. and Szepesvari, C.: Bandit based Monte-Carlo Planning. *17th European Conference on Machine Learning*, pp.282–293 (2006).
[18] Ojima, Y.: Computer Go mailing list への投稿 (2009). 入手先 ⟨http://www.mail-archive.com/computer-go@computer-go.org/msg10969.html⟩
[19] Perick, P., St-Pierre, D.L., Maes, F. and Ernst, D.: Comparison of different selection strategies in monte-carlo tree search for the game of Tron, *IEEE Conference on Computational Intelligence and Games*, pp.242–249 (2012).
[20] Rosin, C.D.: Multi-Armed Bandits with Episode Context, *Annals of Mathematics and Artificial Intelligence, LNCS*, pp.203–230 (2011).
[21] Tsuruoka, Y., Yokoyama, D. and Chikayama, T.: Game-tree Search Algorithm based on Realization Probability, *ICGA Journal*, Vol.25, No.3, pp.145–152 (2002).
[22] 保木邦仁：局面評価の学習を目指した探索結果の最適制御，第 11 回ゲームプログラミングワークショップ，pp.78–83 (2006).
[23] 前原彰太，橋本 剛，小林康幸：局面評価関数を使う新たな UCT 探索法の提案とオセロによる評価，情報処理学会 ゲーム情報学研究会，Vol.2010-GI-24, No.5, pp.1–5 (2010).
[24] 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，第 18 回ゲームプログラミングワークショップ，pp.130–133 (2013).
[25] 松井利樹，野口陽来，土井祐紀，橋本 剛：囲碁における勾配法を用いた確率関数の学習，情報処理学会論文誌，Vol.51, No.11, pp.2031–2039 (2010).
[26] 松原 仁（編），美添一樹，山下 宏：コンピュータ囲碁モンテカルロ法の理論と実践，共立出版 (2012).

池田 心 （正会員）

1975 年生．1999 年東京大学理学部数学科卒業．2000 年東京工業大学総合理工学研究科知能システム専攻修士課程修了．2003 年同博士課程修了，博士（工学）．同年京都大学メディアセンター助手，2010 年北陸先端科学技術大学院大学情報科学研究科准教授．ゲーム，進化計算，パズル等の研究に従事．計測自動制御学会，進化計算学会等会員．コンピュータ囲碁フォーラム理事．

ビエノ シモン （正会員）

1980 年生．2004 年グランゼコール Ecole Centrale de Nantes 大学大学院ロボット制御理論修士課程修了，2011 年 Lille 大学計算機科学博士後期課程修了，博士（Computer Science）．2012 年北陸先端科学技術大学院大学研究員，2013 年同情報科学研究科助教．