プレイアウト局面の乱雑さを考慮した学習用棋譜作成

長谷川　敦史[†1]　池田　心[†1]
飯田弘之[†1]　橋本隼一[†1]

近年，コンピュータ囲碁の世界ではモンテカルロ木探索（MCTS）が主流のアルゴリズムとなっており，木探索部・プレイアウト部ともに多くの研究がなされている．プレイアウト部は，人間の設計したルールや棋譜を元に学習した重みに基づき，終局まで着手が確率的に選択されるランダムシミュレーションが行われるが，その質は MCTS の性能に直結する．本研究では，シミュレーション中に現れる局面は通常のゲーム中に現れるものにくらべて**乱雑**になる傾向があるということに着目する．乱雑な局面では，指し手の評価は学習時とは異なる状況下で行われることになり，これがプレイアウトの質を落とす原因となっており，この問題は棋譜枚数を増やすことでは解決できないというのが我々の仮説である．この問題に対し，本稿では，乱雑な局面を**意図的に**生成し，その局面で良い着手を探索して学習用棋譜として用いるというアプローチを提案し，その有効性を示す．

Game Record Generation for Machine Learning that Considers Disorder of Board in Playout Phase

Atsushi Hasegawa,[†1] Kokolo Ikeda,[†1] Hiroyuki Iida[†1]
and Junichi Hashimoto[†1]

Monte-Carlo Tree Search (MCTS) is a representative algorithm in the domain of computer Go today, and its two major phases, tree search and playout, have been intensively studied.In the playout phase, which directly affects to the performance of MCTS, random game simulations are performed according to a certain policy which is pre-designed by hand or learned from game records. We focus on the fact that a position in a playout tends to be more disordered than a position in actual games. As the consequence of the fact, moves are evaluated in different situation where from the evaluation function is trained, and this gap of the situation degrades the quality of playouts. This problem cannot be solved only by the number of game records in the learning process. We propose to create disordered game records using playout function and MCTS, and to use these records as a part of the training data.

1. はじめに

駒の価値を用いることが可能な将棋やチェスに比べ，囲碁では局面の静的状態評価関数を設計することが困難であるとされ，ある局面から終局までランダムに着手してその勝敗を状態評価関数として用いるモンテカルロ碁が提案された[1]．モンテカルロ木探索 (MCTS) を用いた囲碁プログラム CrazyStone が成功し，探索資源を適切に分配するための UCT(UCB applied to trees)[5] が提案されるなど，現在のコンピュータ囲碁の主流となっている．MCTS は大きく木探索部とランダムシミュレーションを行うプレイアウト部に分かれるが，本稿ではプレイアウト部に着目しその精度を高めるための手法を提案する．

†1 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology

原始的なモンテカルロ碁では自分の眼を潰す以外の合法手を全て等しい確率で着手するが，囲碁に関する知識を援用し，良さそうな手を高い確率で着手させる試みは多くなされている[4]．着手の選択確率を定める**行動**評価関数は，着目点の周囲の状況を表す“パターン”や，囲碁独自の概念であるアタリ・ヌキなどの特徴を入力とし，手作業や学習などで定められた値を出力とする．なかでも，高段者の棋譜を用いて頻出パターンを抽出し，その着手を再現するように評価関数を調整する教師あり学習はしばしば用いられ，Bradley-Terry モデルによる手法[3] や，シグモイド損失関数による勾配法[9] などが提案されている．

一般に，棋譜を用いた学習が成功するためには，評価したい局面の特徴（パターン等）が棋譜中に十分登場していることが重要である．着目点周囲の石配置パターンが棋譜中にも頻繁に登場するものであれば，そのパターンにおける着手の好ましさは学習しやすく

信頼できるものになるが，棋譜中にめったに登場しないものであった場合には頑健な学習は困難になる．このことは，プレイアウト部で行動評価関数を用いる場合には特に大きな問題となりうる．なぜなら，プレイアウト部では比較的低性能な着手選択を行うため，葉ノードから何十手もこれを繰り返すと高段者の棋譜には現れないような**乱雑な局面となりやすい**からである．参考にする棋譜枚数を多くするだけではこの問題は解決できず，**乱雑な局面を多く含みかつその中での良い着手が行われている棋譜**が必要となる．そこで本研究では，通常の学習を行った囲碁プログラムを用いて，数手～数十手プレイアウトを行って意図的に乱雑な局面を生成し，その局面で MCTS を行うことで良い着手を生成，これらを学習棋譜に加えて用いることを提案する．

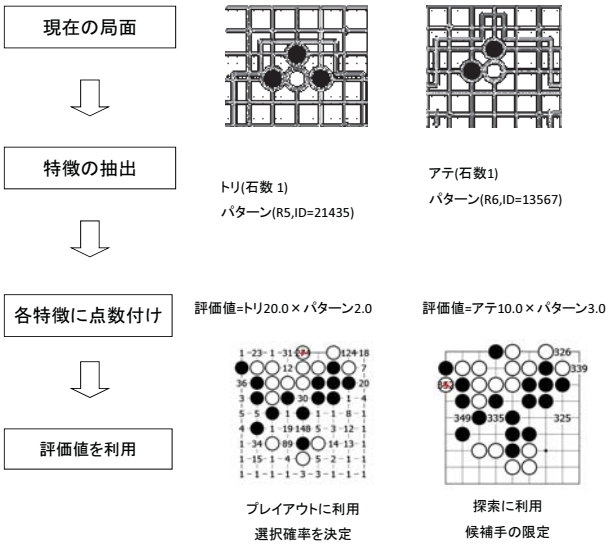

図 1　行動評価関数の計算と利用

2. パターンマッチの課題

MCTS の探索性能を向上させる主な方法は指し手を評価する関数（行動評価関数）の精緻化を図ることである．すなわち，探索やプレイアウトで現れた局面（状態）の全合法手（行動）に対し，そこが打ちたい場所であるかそうでないかを何らかの方法で点数付けすることができれば，その行動評価関数はプレイアウト時の手の選択確率の決定や，木探索時の着手制限[2]に利用することができ，探索の効率化，プレイアウト，ひいては局面評価の精緻化ができる．

図 1 は行動評価関数の計算法，利用法の概念図である．多くの場合，各合法手には囲碁の要素に関係する「アテ」「ノビ」「トリ」「盤端からの距離」「直前手からの距離」「周囲のパターン」などの “特徴量” が割り当てられ，それぞれの善悪を合算することで手の評価値が定められる[3]．

その中でも，周囲のパターン，つまり着目点近傍の石の並びは重要なものの一つである．良く用いられるのは着目点を中心とした正方形，ひし形，円形のパターンで，3×3 から 9×9 程度の範囲を対象とする[4][6][8]．3×3 などの小さなパターンでは着手や局面の状態を正確に評価できない恐れがあり，一方 9×9 などの大きなパターンでは棋譜中に登場する頻度が下がり，また組み合わせの数が増えるため，機械学習による評価値の決定が難しくなるというトレードオフがある．

Stern ら[6] は，パターンの自動生成を行う過程で複数のサイズのパターンを用意しておき，マッチした最大のパターンを利用することを提案している．本研究で利用する囲碁プログラム NOMITAN[9] も Stern らの方法と似た手法を採用している．図 2 に NOMITAN が利用するパターンを示す．着目点（中心）から (x, y) だけ離れた点に対し距離を $d(x,y)=|x|+|y|+max(|x|,|y|)$ とし，Rn で $d(x,y)\le n$ であるような点からなるパターンを表す．たとえば，R3 とは 3×3 の範囲ということである．

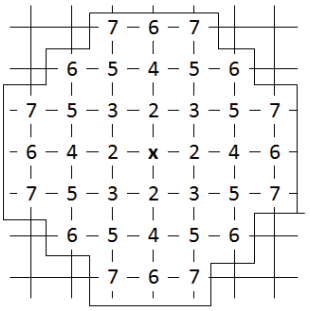

図 2　x, y は 2 つの座標の x 成分，y 成分の差であり，着手点の位置（×印のある点）が距離 0 となるパターン範囲

Stern らの手法同様，NOMITAN でも学習棋譜中に一定回数（例えば 100 回）以上登場したパターンのみ抽出して評価時に利用する．そのためある局面が棋譜中に現れるような比較的整った局面であれば，平均的に大きなパターン（R6, R7）がマッチしやすく，逆にプレイアウト中にしか現れないような比較的乱雑な局面であれば，小さなパターン（R2, R3）のみがマッチすると予想できる．大きなサイズのパターンほど善悪の

判断に用いることができる情報が多く，手を正確に評価できると期待できるため，与えられた局面の全合法手に対してどのサイズのパターンがマッチするかは，重要な情報である．

我々の仮説では，**棋譜に登場する局面は比較的整っており大きなパターンがマッチし，プレイアウトに登場する局面は比較的乱雑であり小さなパターンしかマッチしない場合がある** としている．このことを確認するため，NOMITAN の評価関数を通常の対局棋譜を元に学習した上で両者を比較する．学習対象に用いたのは，CGOS(Computer GO Server, http://cgos.boardspace.net/) から採取した，９路盤におけるレーティング 2200 以上のコンピュータプログラム同士の対戦棋譜である（以降，通常棋譜と呼ぶ）．

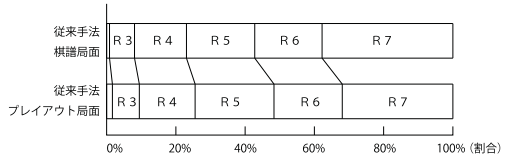

図 3 30 手目におけるパターンサイズの割合．上：棋譜局面，下：プレイアウト局面

図 3 は，棋譜の 30 手目に現れる局面（上）および初期局面からのプレイアウト 30 手目に現れる局面（下）でマッチするパターンのサイズの比率を帯グラフにしたものである．係数は通常棋譜 4 万枚で学習し，サンプル局面数はそれぞれ 5000 とした．最も大きい R7 から最も小さい R2 までマッチしているが，全体にプレイアウト局面のほうが小さいパターンにしかマッチしていないことがわかる．これは，プレイアウト局面のほうが乱雑な局面であり，“よくある形” が少ないためと推測される．

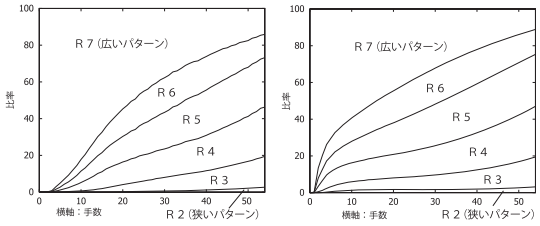

図 4 マッチするパターンのサイズの比率の推移．左：棋譜局面，右：プレイアウト局面

次に，図 4 は，初手から 55 手目まで（横軸），この比率（縦軸）がどのように変化するかを示したものである．棋譜局面およびプレイアウト局面どちらでも，手数が進むほど，小さいパターンにしかマッチしなくなっていくことがわかる．盤上に石のない同一の局面から，着手（やプレイアウト）によって徐々に局面が乱雑になっていくことを考えればこれは自然なことである．

図 4 の左右を比較すると，プレイアウト局面のほうがより急激に小さいパターンにしかマッチしなくなっていくことが見て取れる．これをより分かりやすく比較するために，図 5 は，マッチしたサイズ（R2 なら 2，R7 なら 7）を平均して表示したものである．最も差が出ているのは中盤 20 手目前後であるが，いずれにせよ全体にプレイアウト局面のほうが小さいサイズとなっている．

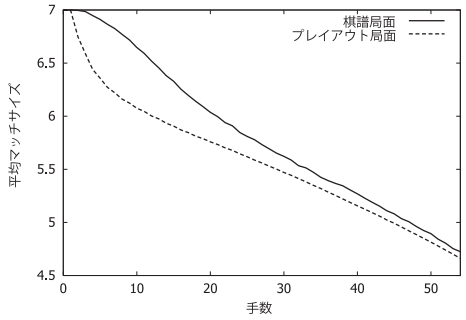

図 5 平均マッチサイズの推移，棋譜局面とプレイアウト局面の比較

以上の結果から，従来の方法で学習した評価関数はプレイアウト中に現れる比較的乱雑な局面になるにつれ小さなパターンでしかマッチできず，おそらく評価関数の質も低下しているであろう様子が確認できた．実際に評価関数が利用されるのは，棋譜として現れるような局面ばかりではない．つまり，確率的な着手の連続であるプレイアウト中の局面は勿論のこと，木探索中も，悪い手を悪い手と判断するためには，それが打たれた「普通ではありえない局面」において適切な行動評価関数が学習されている必要がある．次章では乱雑な局面を意図的に生成し，それを学習対象に加えることでこの問題の解決を図る．

3. 乱雑棋譜の生成

前章の問題点を解決するために，本研究では **“プレイアウトで生じるような乱雑な局面において，十分学習に値する良い手が打たれている棋譜”**（以降，乱雑棋譜と呼ぶ）を生成することを目標とする．学習のために棋譜を自己生成する試みは 5 五将棋でも行われているが[7]，プレイアウト中に登場するような乱雑な局面

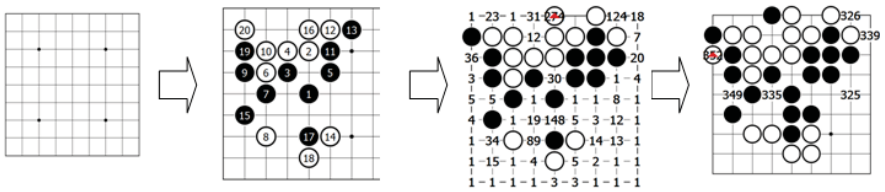

図 6 提案手法の生成と学習

を意図的にあえて作ることが本提案の主旨である．その方法は，図 6 および下記に示すように 3 つのステップからなる．

提案：乱雑局面の生成法（図 6 参照）

(1) 【通常棋譜をなぞる】
通常の学習に用いている棋譜を n_1 手目まで進める．この手順を含めた理由は，初手からプレイアウトのみを行うと，実際に頻出する定石的な局面から派生する乱雑局面を経験できないという懸念があるからである．

(2) 【確率的に着手する】
続いて，プレイアウトする際に用いるのと同じ着手選択確率に従って n_2 手だけ進める．このとき，n_2 を大きくするほど局面はより乱雑になると予想される．

(3) 【良い手を棋譜に残す】
通常の着手決定アルゴリズム（MCTS による探索）に従い着手を定め，一手分局面を進める．これを n_3 回繰り返すことで，一枚の棋譜に n_3 手の良い手を含めて学習用に保存する．なおこの際，MCTS では有望な手の勝率（厳密には UCB 値）が推定できるため，それも記録する．

この 3 ステップを，複数のパラメータ (n_1, n_2, n_3)・通常棋譜・乱数シードで行うことで，多様な特徴を持った乱雑棋譜を生成することができる．現実の対戦では，プレイアウトは序盤から終盤まであらゆる局面から行われるため，常に $n_1 = 0$ として初期局面からのプレイアウトに特化させたり，$n_1 = 40$ などとして終盤からのプレイアウトに特化させることは好ましくない．

本手法を行うためには，ある程度の精度でプレイアウトできる行動評価関数，良い手を選べるアルゴリズム，が既に存在しなければならない．本論文では NOMITAN（CGOS のレートで 2300 前後）を用いたが，より高速で強いプログラムのほうが望ましいことは確かだろう．

4. 学習方法と乱雑棋譜の扱い

NOMITAN では，パターンの抽出には Stern[6] と同様に Zobrist hashing を用いて一定回数以上登場したものを登録する手法，特徴量ごとの評価値（係数）の最適化には Coulom[3] と同様に平均対数尤度の最大化（ただし minorization-maximization アルゴリズムではなくて勾配法）を用いている．新しく生成した乱雑棋譜についても，通常棋譜と全く同じ扱いをしてパターン抽出・最適化を行うことは勿論可能である．ただし，乱雑棋譜が通常棋譜に比べて少数しか生成できないことを考慮すれば，**乱雑棋譜を重視するような重み付け**を行うことで，より効率的に提案手法を利用することができる．

4.1 パターンの抽出

NOMITAN では，学習の前段階として頻出パターンの抽出を行う．そこでは，全ての利用可能な棋譜について，初手から適当なところまで（本論文の実験では 55 手まで）の全ての局面に対し，全ての合法手を抜き出し，その周囲 R2 から R7 までの全てのパターンのハッシュ値（周辺の石の有無と，着目点の位置の分類から求める）を計算し，メモリ上に記録する．そのうえで，一定回数（本論文の実験では 100 回）以上登場したものについてのみ，頻出パターンとしてそのハッシュ値をファイルに出力することにする．例えば，通常棋譜 4 万枚を用いた場合は全部で約 21 万のパターンが抽出された．

乱雑棋譜が通常棋譜に比べて手数の意味で少ない場合，せっかく珍しい形を経験したとしても登場回数が

閾値を越えることができずにそのパターンが抽出されない，ということがありうる．そこで，乱雑棋譜が 1 回登場した際には，それを重み w で倍する，つまり **w 回登場したことにする**，という重み付けの方法を提案する．極端な例として $w = 100$ なら，乱雑棋譜に 1 回でも登場したパターンは全て登録されるということになる．本論文の実験では $w = 2$ を用いたが細かいチューニングは行っていない．

また，乱雑棋譜中の，n_1 手の通常棋譜をなぞった部分，n_2 手の確率的選択部分を，パターン抽出に用いるかというのも利用者のオプションである．本論文の実験では通常棋譜をなぞった部分だけを除いてあるが，次節で示す学習との整合性を考えると，確率的選択部分も除いたほうが良いと考える．

4.2 係数の最適化

NOMITAN では，用いる特徴量（パターンを含む）と学習棋譜が定まると，棋譜で打たれた手の選択確率についての平均対数尤度[3] が最大となるような勾配法によって係数を最適化する．用いた特徴量は，パターン（盤上の位置も含む），直前手からの距離，Liberty に関するもの（トリ，アテ，ヌキ，自殺手など）であり[9]，その総数はほぼパターン数の 2 倍である．学習には，通常の計算サーバを用いて 10 分程度要する．

通常，インターネット等で入手できる棋譜には，「何手目に（どの局面で）どこに着手したか」しか記載されていない．つまり，その手が良い手なのか悪い手なのか，また良い手だとしても絶対の一手なのか，複数ある良い手の 1 つにすぎないのかは分からない．もし絶対の一手ならばそれは必ず再現できるように学習したいし，複数ある良い手の 1 つにすぎないならばそれは仮に再現できなくとも大きな問題ではないかもしれないため，このような情報は手に入るのならばぜひ活用すべきであろう．

本論文で提案する手法では，棋譜は自分で作成するため，より豊かな情報を残すことが可能である．具体的には，例えば MCTS の探索結果として得られる「訪問回数」「勝率」「UCB 値」などを記録することが可能であり，これを参照することで強く学習すべきものとそうでもないものを区別することができる．

本論文の実験では，乱雑棋譜生成時に UCB 値上位 6 位までの手を抜き出し，その勝率を記録した．その上で，次の二つの工夫を行った．

- 【勝勢・敗勢時の着手を学習しない】
  局面がすでにどうやっても勝ちの場合，MCTS は「より多くの地を得る」という意味で最善とは言えない手を打つことが多い．逆にどうやっても負けそうな場合，単純なアタリなど自暴自棄な手を打つことが多い．これらはいずれも学習対象としては不適切であるため，勝率が 10% 以下，90% 以上のときはその着手を学習しないことにした．
- 【複数の手を学習する】
  最善手の勝率との差が一定値（例えば 1%）未満であるような手は，場合によっては良い手でありうる．このような場合，2 位の手や 3 位の手も「打たれた」とみなし，そのような手の数 *goodMoveCount* を用いて式 1 の重みで学習することにした．もしそのような手がない場合は，実際に打たれた 1 位の手がいわゆる絶対の一手であったとみなし，重みは大きくなる．本論文では *totalWeight* は 6 とした．

$$weight = \frac{totalWeight}{goodMovesCount} \tag{1}$$

なお学習は，n_3 手の，探索を伴った着手についてのみ行われる．n_2 手のプレイアウト部を学習することは，用いているプログラムが同じである以上は無意味である．

5. 評価実験

前章までででも出てきたとおり，本論文では 9 路盤を対象とした実験を行っている．提案手法は 19 路盤でも，あるいは他のゲームであっても汎用的に使えるものではあるが，19 路盤ではプロ棋士によるものを含む高品質の棋譜が多く手に入る一方，十分強いコンピュータプログラムはまだ少ないことを考え，対象を 9 路盤にしている．9 路盤では逆にプロ棋士を含む人間による高品質の棋譜は手に入りにくいため，十分に強いプログラム (CGOS でレーティング 2200 以上) 同士の対局の棋譜を用いることにする．このレーティングは概ね人間の有段者レベルであるが，ある種のコンピュータプログラムの特徴として実質的な終局後も無意味に打ち続けることがあるため，棋譜サンプルは 5 手から 55 手までとした．

提案手法で作成した乱雑棋譜は n_1，n_2 については偶数を用いることとし，$n_1 = 4 \sim 20$，$n_2 = 6 \sim 16$，の範囲で一様ランダムに選択した．探索する手数 $n_3 = 6$ で，一手につき 10 秒探索することとした．1 枚の棋譜を生成するのに約 1 分必要とする計算である．乱雑棋譜は全部で 85000 枚（うち 5000 枚はテスト用）生

成したので，約 10 台のマシンで 6 日ほど要した．

表 1 学習に用いた棋譜セット

識別記号	通常棋譜	乱雑棋譜	総手数	乱雑棋譜の割合
(A)	2 万枚	-	100 万	0
(B)	4 万枚	-	200 万	0
(C)	2 万枚	2 万枚	112 万	10.7%
(D)	4 万枚	2 万枚	212 万	5.7%
(E)	4 万枚	8 万枚	248 万	19.4%

表 1 に，実験でさまざまな比較をするための「学習に用いた棋譜セット」の組み合わせを示す．例えば (A) と (C) の比較により，乱雑棋譜を加えたことの効果を知ることができ，(B)(D)(E) の比較ではさらに乱雑棋譜の枚数が変化した場合の効果を知ることができる．ここで，学習対象となるのが通常棋譜では 1 枚あたり 50 手，乱雑棋譜では $n_3 = 6$ 手に過ぎないことに注意されたい．従って，それぞれ同じ “枚数” を用いている (C) であっても “手数” という観点では乱雑棋譜は 1 割ほどしか占めていない．このような理由から，前章で説明した重み付けを導入している．

棋譜生成等に使用した実験機のスペックは以下の通りである．

- OS: Linux
- CPU： 2× Intel(R) Xeon(R) L5520 @ 2.27GHz
- メモリ： 24.0GB

続いて，5.1 節ではパターンサイズについて第 2 章で行ったのと同じ実験を提案手法に対しても行う．5.2 節では学習時の一致率等，複数の指標を比較する．

5.1 マッチするパターンサイズの比較

第 2 章では，通常棋譜 4 万枚を使って学習したものについて，そのマッチサイズを棋譜局面とプレイアウト局面で比較した（条件 B．図 3，4，5）．本節では，そこに乱雑棋譜 8 万枚を加えた場合（条件 E）のマッチしたパターンサイズについて調べる．

図 7 は，棋譜の 30 手目およびプレイアウト 30 手目における，従来手法 (B) と提案手法 (E) の，マッチしたパターンサイズの割合を帯グラフで示したものである．提案手法は従来手法に比べ大きなパターンでマッチしており，プレイアウト局面であっても従来手法の棋譜局面程度またはそれ以上のパターンにマッチしていることが分かる．なお，学習した局面とテストに用いた局面は独立である．

図 8 は，マッチしたパターンの平均サイズの 1 手

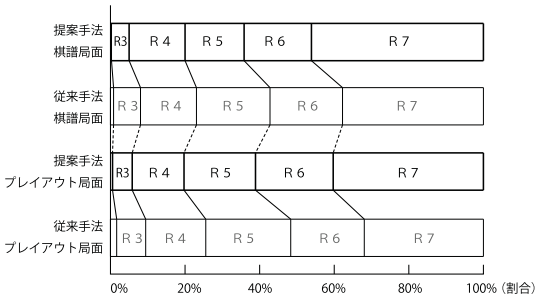

図 7 従来手法と提案手法の比較：30 手目におけるパターンサイズの割合．

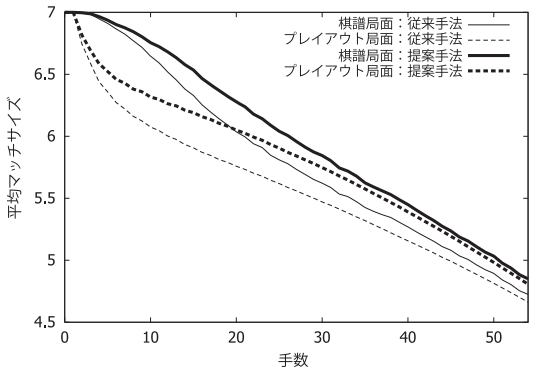

図 8 従来手法（細線）と提案手法（太線）の比較：平均マッチサイズの推移．

目から 55 手目までの推移を示したものである．細線（従来手法，B）に比べ，太線（提案手法，E）は全体に大きなパターンにマッチしており，さらに，**プレイアウト局面でそれはより顕著である**ことが分かる．多くの棋譜を使えばマッチするパターンサイズが大きくなることは自然であるが，プレイアウト局面を生成・学習したことで，それを得意とする（未知のプレイアウト局面でも大きなパターンでマッチする）ように学習できていることは期待通りであるといえる．

5.2 学習時の各指標

前節では，乱雑棋譜を学習棋譜に含めることで，確かにマッチするパターンを大きくすることができることを確認した．マッチするパターンが大きくなれば，係数もより適切に定めることができるはずである．本節では，勾配法を用いて学習した係数を，4 つの指標で比較することにする．まずその指標の定義と意味合いを説明する．

5.2.1 一 致 率

教師信号すなわち棋譜に示されている手が，評価値

表 2 学習時の複数の評価指標による評価結果

	通常棋譜でテスト				乱雑棋譜でテスト			
組み合わせ	一致率	平均ランク	MLE	$Not_{0.9}$	一致率	平均ランク	MLE	$Not_{0.9}$
(A) 通常 2 万	0.386	4.63	-2.25	0.107	0.317	5.08	-2.46	0.199
(B) 通常 4 万	0.392	4.58	-2.21	0.100	0.334	4.87	-2.38	0.177
(C) 通常 2 万+乱雑 2 万	0.380	4.86	-2.37	0.128	0.352	4.42	-2.26	0.093
(D) 通常 4 万+乱雑 2 万	0.387	4.61	-2.22	0.150	0.355	4.39	-2.23	0.117
(E) 通常 4 万+乱雑 8 万	0.398	4.40	-2.27	0.165	0.382	3.96	-2.10	0.090

最大となった割合を一致率（一位率）と呼ぶ．H を全局面，$h \in H$ を一つの局面，γ_h をその局面での教師信号の評価値順位（ランク）とすると，一致率 $MatchRate$ は式 (2) で定義できる．

$$MatchRate = \frac{|\{h \in H | \gamma_h = 1\}|}{|H|} \tag{2}$$

5.2.2 平均ランク

通常，プレイアウトでは評価値一位の手だけが選択されるわけではなく，評価値に応じた割合や順位に応じた割合で確率的に選択される．従って，仮に一位でなくとも，教師信号の順位は若いほうが良く，選択確率は高いほうが良い．そこで，順位の単純平均を式 (3) で求め，評価に用いる．

$$avRank = \frac{\sum_{h \in H} \gamma_h}{|H|} \tag{3}$$

5.2.3 平均対数尤度

NOMITAN では，Bradley-Terry モデルに基づき，手の評価値はその手の持つ特徴量係数の積で計算し，その選択確率は全ての合法手の評価値の和に対する割合で計算している．このような方法はしばしば用いられる[6][3] が，その場合，教師信号の選択確率の対数を取り全棋譜について平均を取ったものを**平均対数尤度**（Mean Log Evidence）と呼び，評価に用いることが多い．局面 $h \in H$ での教師手の選択確率を $p(a_h^*)$ とすると，平均対数尤度 MLE は式 (4) で定義できる．

$$MLE = \frac{\sum_{h \in H} \log p(a_h^*)}{|H|} \tag{4}$$

全ての教師信号の選択確率が 100% であれば 0 となるが通常負値であり，0 に近いほど良い．

5.2.4 α 非着手率（Not_α）

行動評価関数では，「良い手が高い評価値にならない」ことも問題であるが，それ以上に「悪い手なのに高い評価値である」ことが大きな問題である．そこで，選択確率が α 以上であるにもかかわらず教師信号と一致しなかった割合を α **非着手率**（Not_α）と呼び式 (5) で定義する[10]．A_h は局面 $h \in H$ における合法手の集合，$a_h^* \in A_h$ は局面 h の教師手，$p(a)$ は手 a の選択確率である．

$$Not_\alpha = 1 - \frac{|\{h \in H | p(a_h^*) \geq \alpha\}|}{|\{a \in A_h, h \in H | p(a) \geq \alpha\}|} \tag{5}$$

例えば $Not_{0.9}$=0.7 である場合，選択確率が 90% を越える手，つまりコンピュータがほぼ絶対の一手と思うような手が，実際には棋譜中では 30% 程度の割合でしか選ばれていないことを意味し，深刻な問題である．この値は $1-\alpha$ 程度には小さいほうが好ましい．

5.3 学 習 性 能

前節の 4 つの指標を用いて，表 1 の 5 つの場合における学習時性能を評価する．学習に用いる棋譜とは別に，汎化性能を調べるため，テスト用に通常棋譜 5000 枚，乱雑棋譜 5000 枚を用いる．

表 2 に，5 つの場合（縦軸）の，通常棋譜に対する 4 指標，乱雑棋譜に対する 4 指標の結果（横軸）を示す．ここから分かることは以下の通りである．

- まず (A) を単独で見ると，乱雑棋譜での性能が，通常棋譜での性能に大きく劣る（一致率 7%，$Not_{0.9}$ も約 2 倍）ことが分かる．これは，比較的整った通常棋譜だけでは，プレイアウトに登場するような乱雑な局面に十分対応できていないことを示しており，本論文の前提となる仮説を支持している．
- 続いて (A) と (C) を比較する．手数にして約 1 割にすぎない乱雑棋譜を加えることで，乱雑棋譜でのテスト結果は一致率 3.5%, ランク 0.66，$Not_{0.9}$ 半減などの大きな向上が見られる．実際には，4.2 節最後で述べた $totalWeight = 6$ の重み付けの工夫も効いていると予想される．
- さらに (B) を (C)(A) と比較する．(B) は (A) の棋譜を 2 倍にしたもので，手数としては (C) より遥かに多い．実際，通常棋譜における性能は (A) はもちろん (C) よりも高い．しかしながら，乱雑棋譜に対する性能については，(A) よりは向上して

いるものの，(C) には遠く及ばないものとなっている．このことから，プレイアウトに登場するような局面での良い評価関数を作るには，単に用いる棋譜枚数を増やすだけでは非効率であり，本論文で提案した手法が有効であるということが主張できる．

- 最後に (B)(D)(E) を比較する．通常棋譜 4 万枚に，乱雑棋譜を 2 万および 8 万枚加えることで，徐々に乱雑棋譜の性能が向上していることが分かる．特に (E) は，一致率 0.382 等，通常棋譜の場合と比肩する性能を出せている．また興味深いこととして，(A) から (C)，(B) から (D) など，乱雑棋譜を加えると通常棋譜での性能は若干ながら低下していたにも関わらず，(E) は通常棋譜でも相当良い性能となっていることが挙げられる．図 8 でも示すようにマッチしたパターンのサイズは棋譜局面でも向上しているため，乱雑棋譜と通常棋譜は必ずしも相容れない関係ではなく，補いあう関係であると考えることができるかもしれない．

以上のことから，少なくとも学習性能においては，本論文の仮説および提案は十分説得性を持つものであると考える．現状では，乱雑棋譜を用いた場合の，用いない場合に対する対戦成績は勝率 5〜6 割に留まっており，必ずしも学習性能が対戦性能に結びついていないが，n_1, n_2 などのパラメータを含む乱雑棋譜の作り方，4.2 節で述べた工夫などの改善によっては，対戦成績も向上するものと考えている．

6. ま と め

本稿では，囲碁のモンテカルロ木探索において，プレイアウト中は比較的質の低い着手選択を繰り返すため局面が乱雑になり，それが通常の棋譜学習では十分に対応できなくなるという仮説を提唱し，その根拠を示した．そのうえで，意図的に乱雑な局面を生成し学習用棋譜として用いる枠組みと，学習の際の工夫を提案し，複数の指標を用いた評価でその効果を実証した．さらに，通常の棋譜を多く用いるだけでは乱雑な局面での性能は効率的に向上せず，提案手法を用いた場合は（手数にして 1 割程度の乱雑棋譜を加えるだけで）一致率で 7% の向上が見られるなど，期待通りの性能向上が図れることが分かった．今後は，乱雑棋譜の適切な生成法・利用法について検討を行い，対戦性能も含めたより一層の改善を目指す．

参 考 文 献

1) Brügmann, B.: Monte Carlo Go, Technical Report (online) (1993).
2) Bouzy, B.: Move-Pruning Techniques for Monte-Carlo Go, *In Proc. of Computers and games: the 5th international conference (CG2006)*, Lecture Notes in Computer Science, No.4250, pp.104–119 (2006).
3) Coulom, R.: Computing Elo Ratings of Move Patterns in the Game of Go, *ICGA Journal*, Vol.30, No.4, pp.198–208 (2007).
4) Gelly, G., Wang, W., Munos, R. and Teytaud, O.: Modification of UCT with Patterns in Monte-Carlo Go, Technical Report, No.6062, INRIA (2006).
5) Kocsis, L. and Szepesvári, C.: Bandit-based Monte-Carlo Planning, *In Proc. of the 17th European Conference on Machine Learning (ECML2006)*, Lecture Notes in Computer Science, No.4212, pp.282–293 (2006).
6) Stern, D., Herbrich, R. and Graepel, T.: Bayesian Pattern Ranking for Move Prediction in the Game of Go, *In Proc. of the 23rd International Conference on Machine Learning (ICML2006)*, ACM International Conference Proceeding Series, No.148, pp.873–880 (2006).
7) 柿木義一：5 五将棋における評価関数の自動学習，エンターテイメントと認知科学研究ステーション第 5 回 招待講演会 (2008).
8) 清愼一, 川嶋俊明：「局所パターン」知識主導型の囲碁プログラムの試み，第 1 回ゲームプログラミングワークショップ予稿集 (GPW'94), pp.97–104 (1994).
9) 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).
10) 土井祐紀：局所化と汎化を両立させる囲碁パターンマッチング, 北陸先端科学技術大学院大学 修士論文 (2011).