修 士 論 文

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
情報科学研究科情報科学専攻

土井 佑紀

2011 年 3 月

修 士 論 文

局所化と汎化を両立させる囲碁パターンマッチング

指導教員　飯田弘之 教授

審査委員主査　飯田弘之 教授
審査委員　池田心 准教授
審査委員　鶴岡慶雅 准教授

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

0810041 土井 佑紀

提出年月: 2011 年 2 月

概 要

近年提案されたモンテカルロ囲碁はコンピュータ囲碁の世界に急激な進展をもたらし，その棋力は今までにない勢いで向上している．モンテカルロ囲碁ではプレイアウトと呼ばれるランダムシミュレーションをある局面からゲームの終局まで行い，これを何度も繰り返しその勝敗数を用いて局面の評価を行う．現在はこれに木探索を併用するモンテカルロ木探索アルゴリズムが主に用いられている．

モンテカルロ囲碁は発展途上であり，性能向上のための多くの課題があるが，本論文では盤面のパターンマッチングの精度を向上させてプレイアウトの質を高めることに着目する．これまでは着手点を中心とするパターンが一般的であったが，3×3 や 5×5 といった狭いパターンでは囲碁の着手や局面の良さを正確に評価することが困難である一方で，7×7，9×9 などとパターンのサイズを単純に大きくすると同じパターンが登場する頻度が下がり，学習に必要な棋譜枚数が膨大になるというトレードオフが生じる．特にプレイアウト中には普通の棋譜に現れないような局面が多く現れるためマッチングの困難さはさらに大きくなる．

そこで本論文では，着手点を中心とする従来のパターンではなく着手点の周囲を分割するコラージュパターンを提案することで，マッチングし易くかつ広い範囲をカバーできるようにした．その上で，パターンマッチングの精度が向上しているかを多様な視点から実験，評価した．プレイアウトの質の向上と棋力の向上のみならず，人間の相手をする自然な着手の生成などへの応用も期待できる．

目 次

図 目 次

表 目 次

第1章　はじめに

1.1　背景

1.1.1　コンピュータ囲碁

ゲーム情報学の分野において人間より強いプログラムを作ることは大きなテーマであるが，中でも近年コンピュータ囲碁の分野が世界的に盛り上がりを見せている．これまでオセロやチェス，将棋においては静的評価関数による Minmax 探索 [1] が成功を収めているが，囲碁において同様の手法を用いても一向に棋力は伸びなかった．その理由は大きく分けて 2 つで，第 1 に囲碁の局面の評価の難しさが挙げられる．Minmax 探索にはある程度正確な評価関数が必要であり，例えば将棋であれば駒の損得，駒の利き，駒組みなどで評価することができる．しかし囲碁では石には機能的な差は無く，また基本的に盤上のどこにでも置くことができる．また局所的には良い手でもより広い範囲を見るとあまり良くない手であることがあるなど手の良さを評価することが難しい．第 2 に探索空間の大きさが挙げられる．表 1.1 にいくつかのボードゲームの状態空間と探索空間のオーダーを示す [1][2][3][4]．なお表中の 9 路と 13 路の囲碁の値については以下の式で求めた．N は交点の数，M は平均合法手数，L は平均終了手数である．

$$\text{状態空間} = 3^N \tag{1.1}$$

$$\text{探索空間} = M^L \tag{1.2}$$

9 路と 13 路の各値は表 1.2 の値として計算した．表 1.1 より囲碁は他のゲームに比べ複雑であるといえる．このためチェスや将棋と同様の Minmax 探索では深い探索をすることが難しく，棋力も伸びなかった．

しかし 1993 年に提案されたモンテカルロ法を用いた囲碁（以下モンテカルロ囲碁，詳しくは第 2.1.1 章で述べる）によって状況は大きく変わった．モンテカルロ囲碁ではプレイアウトと呼ばれるランダムシミュレーションをある局面からゲームの終局まで複数回行い，その勝敗を手（局面）の評価値の替わりに用いる．

この時プレイアウトの質が強さに大きく関係していることが知られている [5]．例えば最初の提案は単純に全ての手を同じ確率で打つという方法であったが，これを良さそうな手ほど高い確率で打つように変更すると 1 回のプレイアウトにかかる時間は長くなるが強くなることが知られている．このプレイアウトの質がモンテカルロ囲碁において重要な要素の 1 つであり，本論文でもこの向上を目的とする．

表 1.1 主なボードゲームの状態空間と探索空間のオーダー

	状態空間	探索空間
三目並べ	10^3	10^5
リバーシ	10^{28}	10^{58}
囲碁 (9x9)	10^{38}	10^{67}
五目並べ (15x15)	10^{105}	10^{70}
チェス	10^{47}	10^{123}
囲碁 (13x13)	10^{80}	10^{180}
将棋	10^{71}	10^{226}
囲碁 (19x19)	10^{172}	10^{360}

表 1.2 計算に用いた値

	N	M	L
9 路	81	50	40
13 路	169	100	90

1.1.2 囲碁のルール・用語

ここでは本論文の読みやすさのため囲碁について説明する．囲碁は二人零和有限確定完全情報ゲームに分類されるある意味最も単純なゲームである．囲碁は国によって多少ルールが異なるので，ここでは日本ルールと囲碁プログラムでよく用いられる中国ルールについて説明する．

まず用語について説明する．

碁石
: 標準的には黒と白の石を用いる．単純に石とも呼ぶ．

碁盤
: 板に格子状に線を引いたものでこれに碁石を置く．線の数によって呼び名が変わり，縦横それぞれ 19 本であれば 19 路盤と呼ぶ．線の数は 19 路が標準であるが，初心者向けの 9 路盤，13 路盤などがある．囲碁は複雑なゲームであるためコンピュータ囲碁においても 9 路盤のように小さい碁盤を用いることがある．単純に盤とも呼ぶ．

交点
: 盤上の線と線が交差した点のこと．単純に点とも呼ぶ．碁石は格子内ではなく交点上に置く（打つ）．

座標
: 囲碁では左上の交点を (1, 1)，その右隣を (2, 1) のように呼ぶ．

空点
: 石の置かれていない交点のこと．

第何線

盤上の一番外側の線上の交点を第一線と呼ぶ．その 1 つ内側を第二線，同様にその内側は第三線，第四線と呼ぶ．

石（連）

盤上に置かれた石，あるいは石に繋がった一連の石の集団のこと．繋がるとは上下左右に同じ色の石があることを示す．コンピュータ囲碁では連と呼ぶことが多い．単に石と言った時 1 つの交点上の石のことを指しているのか，一連の石のことを言っているのか注意が必要である．本論文では 1 つの石は単に石と，一連の石は連と呼ぶことで区別する．

呼吸点

ある連に隣り合う空点の数のこと．駄目（ダメ），リバティーとも呼ぶが本論文では呼吸点で統一する．

死と活き

自分が先にどう打っても相手に正しく応手されると取られてしまう場合その連は死んでいると呼ぶ．その逆であれば活きていると呼ぶ．死んだ連を死に石や死石，活きた連を活き石と呼ぶ．

地

黒白どちらの活き石で囲まれた領域のこと．黒のものなら黒地，白のものなら白地と呼ぶ．単位は目（モク）．地の計算方法はルールによって異なる．

ダメ（駄目）

黒白どちらにも囲まれていない点．打つ価値の無い点（ただしルールによっては意味がある）．駄目は上記のように呼吸点の意味もあるが，本論文ではこの意味でしか使わないことにする．

ハマ（アゲハマ）

対局中に取った相手の連のこと．日本ルールでは対局終了時にこれで相手の地を減らすことができる．

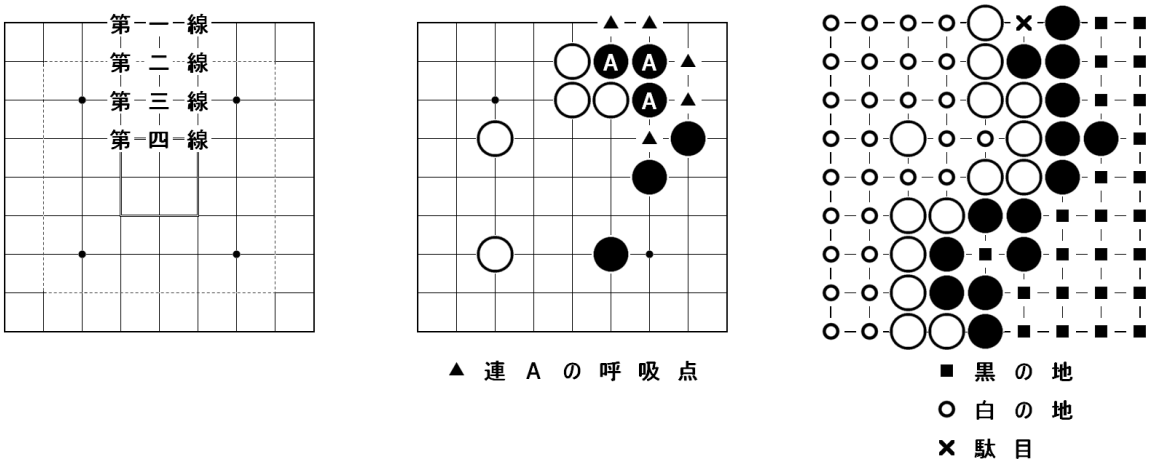
コミ

囲碁は先手（黒）が有利なため地の計算時に黒地から引かれるハンディキャップのこと．現在 19 路では 6.5 目，9 路では 7.5 目が一般的である．

合法手

ルールに従って打つことのできる手のこと．

図 1.1 に用語に関する図を示す．図 1.1(a) は第一線から第四線を図示したものである．図 1.1(b) の A と書かれた一連の石が連でその周りの三角印のついた点が呼吸点である．図 1.1(c) は終局の例で，四角印が黒の地，丸印が白の地である．またバツ印はダメである．



(a) 第何線　　(b) 連と呼吸点　　(c) 地と駄目

図 1.1 用語の説明

囲碁の基本的なルールは以下の 5 つである．

1. 2 人で交互に黒石，白石を碁盤の交点に打つ（基本的に黒が先）．

2. 呼吸点の無くなった連は盤上から取り除かれる．

3. 呼吸点のなくなるところへは打てない（ただし相手の連を取る場合は打てる）．

4. 同形反復の禁止（コウ）．

5. 地の多い方の勝ち．

ルールの 2 つ目はトリと呼ばれる．例を図 1.2 に示す．四角印のついた白の連の呼吸点は点 A のみなので黒が A に置くことでこの連を取ることができる．このようなあと一手でとれる状態をアタリという．取った連は（日本ルールでは）アゲハマとしてとっておく．

ルールの 3 つ目は自殺手と呼ばれる禁止手である．例を図 1.3 に示す．1.3(a) は三角印の付いた白の連が呼吸点 1 つであるが，白が A に打つと呼吸点が 0 となるのでここには打てない．しかし 1.3(b) のように着手によって相手の連を取ることができるときはトリを優先するため禁止手ではない．

ルールの 4 つ目のコウとは図 1.4 のような形を言う．1.4(a) から黒が A に打つと 1.4(b) の状態になる．次に白が B に打つと 1.4(a) に戻る．このように同じ形を延々と繰り返してしまうため同形反復は禁止されている．この場合黒が A に打った後白は他の個所に打ってからであれば B に打つことができる．

最後にルールの 5 つ目であるが日本ルールと中国ルールでは地の計算方法が異なる．日本ルールでは活き石に囲まれた空点の数を数える．この際アゲハマで相手の地を減らす

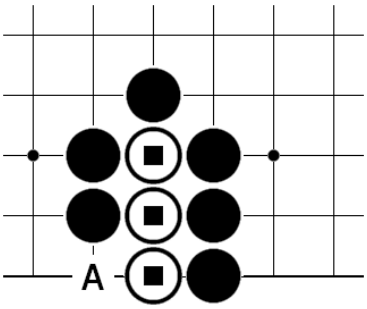
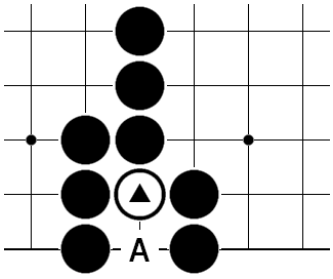
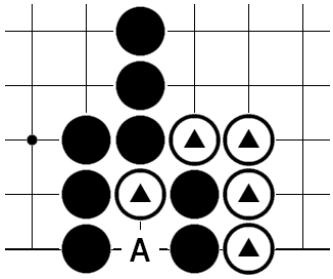


図 1.2 トリの例

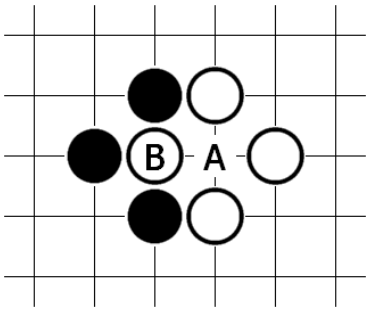


(a) 自殺手となる例

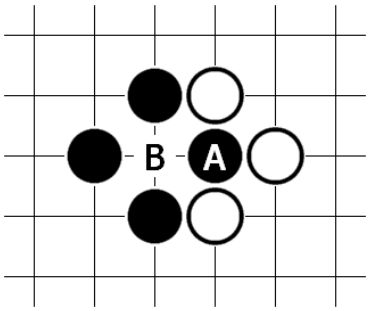


(b) 自殺手とならない例

図 1.3 自殺手の例



(a) 局面 1



(b) 局面 2

図 1.4 コウの例

ことができる．中国ルールでは活き石に囲まれた空点の数と活きている石の数を数える．中国ルールではアゲハマは用いない．中国ルールでは手入れなどの問題が生じにくく，地の計算がしやすいためコンピュータ囲碁で良く用いられる．

1.1.3 Nomitan プロジェクト

Nomitan とは 2008 年から本学，飯田研究室・池田研究室において研究・開発されている囲碁プログラムである [6][7]．プログラムの名称は変化しており，最初は DMC（Direct Monte Carlo の略）という仮の名前であり次に誤碁能美譚（ごーごーのみたん）となった．現在はよりわかりやすいように Nomitan としている．大まかな特徴としてモンテカルロ木探索を行い，勾配法による特徴の係数の学習を行っている．詳しくは第 3 章で述べる．本論文ではこの Nomitan を用いて実験・評価を行う．

1.2 論文の構成

本論文の構成は以下のようになっている．

第 2 章 関連研究
まずモンテカルロ囲碁について説明し，次に囲碁プログラムの重要な技術であるパターンマッチングについてこれまでどのような方法があったか課題と共に紹介する．

第 3 章 Nomitan プロジェクトの概要
我々が用いる囲碁プログラム Nomitan について本論文に関係の深い部分を重点的に述べる．

第 4 章 コラージュパターンの提案と評価
提案手法であるコラージュパターンについて目的と実装を説明し，実験と評価を行う．

第 5 章 考察
第 4 章の結果について考察を行う．

第 6 章 おわりに
全体のまとめと今後の課題，展望を述べる．

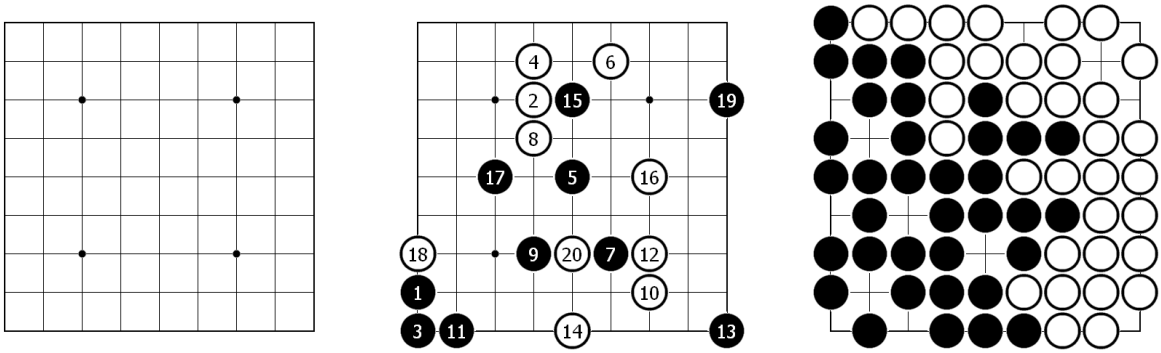
第2章　関連研究

ここでは 2 つの事柄について関連研究をまとめる．1 つ目はモンテカルロ囲碁について定義と歴史，特徴と課題について述べる．2 つはパターンマッチングに関してモンテカルロ囲碁以前のものも含めて複数の手法とその利用法・課題について紹介する．

2.1　モンテカルロ囲碁

2.1.1　モンテカルロ囲碁とは

モンテカルロ囲碁は 1993 年に Brügmann[8] によって提案された．初期のプレイアウトは単純に終局までランダムに手を進めて行くというもので，探索の深さも 1 であった．ただしこのとき「自分の眼には打たない」というルールを加えることでゲームが終了するようにしている．このようなモンテカルロ囲碁を原始モンテカルロ囲碁と呼ぶ．図 2.1 にランダムプレイアウトの例を示す．ランダムな手順ではあるが眼に打たないというルールを加えているため終局している．終局した状態であれば勝ち・負けは容易に判定できる．このように囲碁ではランダムに打ってもゲームは収束し，終局を迎えるためモンテカルロ法を適用することができる．



(a) 初期局面　(b) 20 手まで　(c) 終局

図 2.1 ランダムプレイアウトの例

図 2.2 にモンテカルロ囲碁での評価の例を示す．これは黒の手番を考えており，黒がどの手を選択すると良いかを調べようとしている．まず全合法手に対して一手先に局面を進める（一手先の局面を深さ 1 の局面という）．ここでは合法手は 3 つのみとする．深さ 1 の各局面に対し一定回数のプレイアウトを行う．図では 5 回としてあるが，本来はもっと多くするべきである．そして自分の勝率の高い手を選択するというのが基本的なモンテカルロ囲碁の考え方である．図の場合真ん中の手が 5 回のプレイアウトの内 4 回勝っており勝率が最大なのでこの手を選ぶことになる．

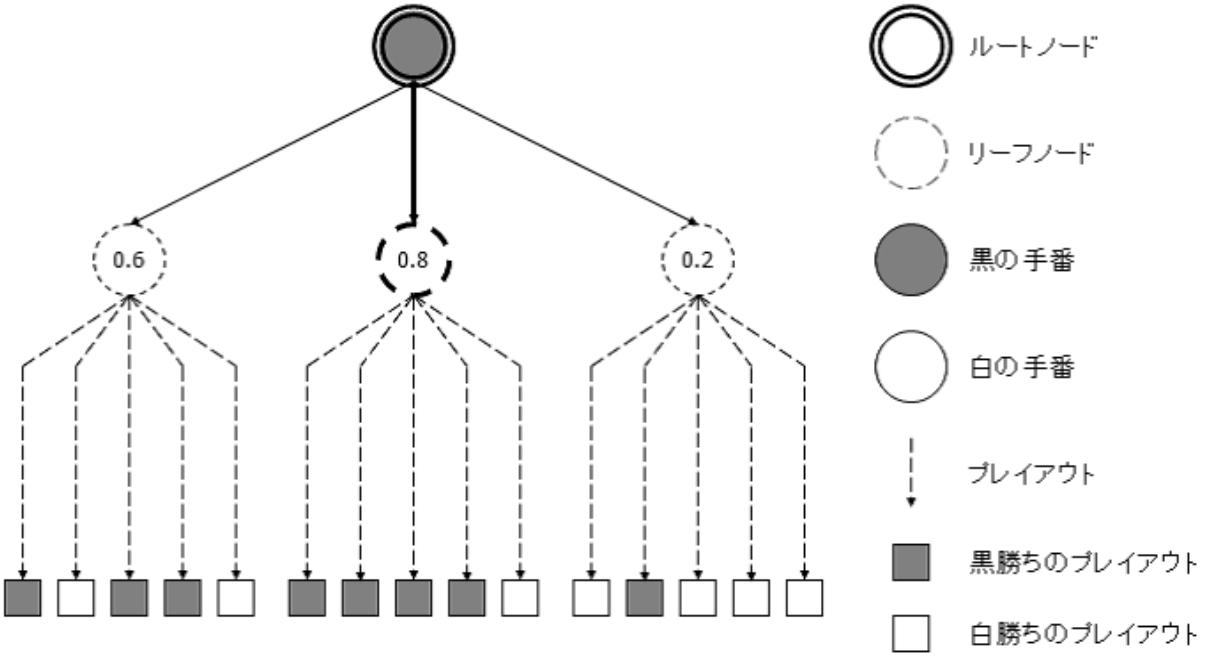


図 2.2 原始モンテカルロ囲碁の評価の例

一手先の局面がその手の善悪を反映して若干の有利不利の差が生じているはずである．プレイアウトはランダムな着手で行われるため，その差が終局まで正確に保存されることは期待できない．従って多くのプレイアウトを行うことで，少なくともそのばらつきの影響（ノイズ）を減じる必要がある．しかしこの方法ではプレイアウトをどれだけ増やしても最善手を返す保証はない．これは良い手も悪い手もランダムに選択し，相手の最善応手を考慮しないためで，これにより深く探索すれば損と分かるような手や相手のミスを期待するような手を打ち易くなるという問題点があった．

しかしその後木探索アルゴリズムを使用することによって飛躍的に棋力は向上した．またプレイアウトに評価関数を用いて良い手ほど確率を上げるようにしたことで棋力が向上した．この木探索アルゴリズムと評価関数によるプレイアウトの質の向上が現在のモンテカルロ囲碁において重要な要素である．次節からはそれぞれについて詳しく説明する．

2.1.2　モンテカルロ囲碁における木探索アルゴリズム

木探索アルゴリズムとモンテカルロを組み合わせた方法はモンテカルロ木探索（MCTS : Monte-Carlo Tree Search）と呼ばれ，現在囲碁プログラムでも良く用いられている．木探索アルゴリズムとはチェスや将棋で用いられてきた Minmax 探索のように木構造によって探索を行うアルゴリズムである．Minmax アルゴリズムは自分は自分にとって最も有利な手を，相手は自分にとって最も悪い手を選ぶとして最善の手を探すというものである．Minmax アルゴリズムの探索例を図 2.3 に示す．一つの局面は丸や四角のノードで表され，あるノードから一手進んだ局面を子ノードと呼ぶ．現在局面から n 手進んだ局面を深さ n の局面と呼び，深さは深ければ深いほど良いが，実際には時間などの制約があるためある程度の深さで打ち切ることになる．図では深さ 2 で打ち切っている．末端に当たる局面で評価関数を呼び出してその局面の評価値を与える．全ての子ノードに評価値が与えられたノードは子ノードの中から最も良いノードを選び自分の評価値とする．この際 Max ノード（自分の手番）と Min ノード（相手の手番）で評価が異なり，Max ノードでは評価値が最も大きい手を，Min ノードでは評価値が最も小さい手を選ぶ．そして最終的にルートノードで選ばれた手を打つ．実際にはα βカットや探索延長などのテクニックを用いることが多いがここでは省略する．

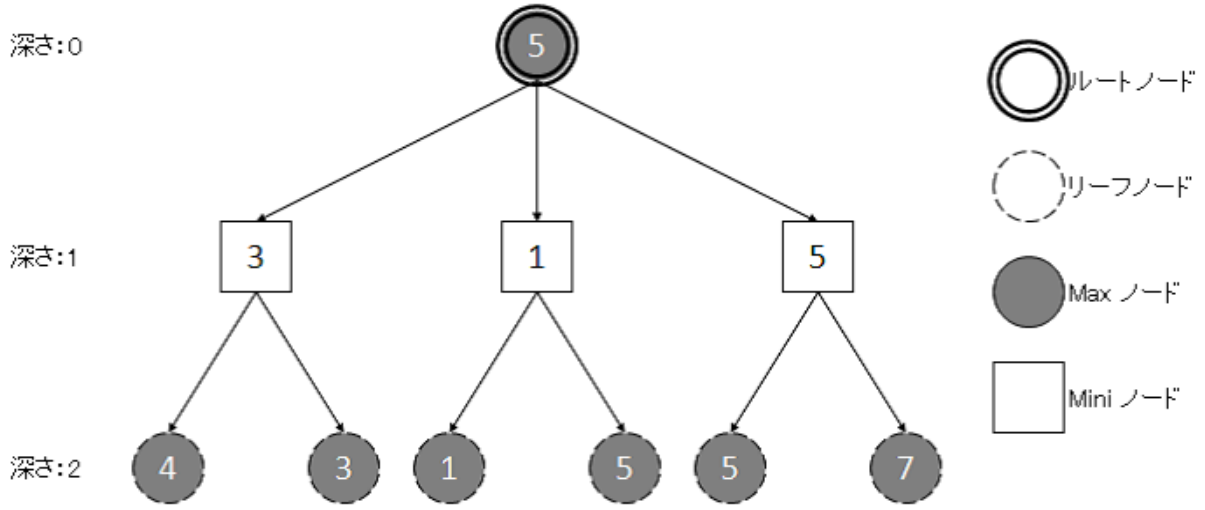


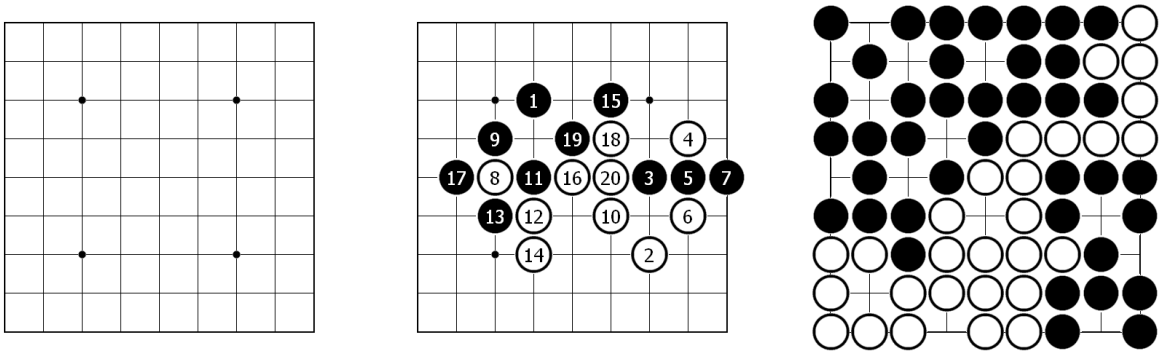
図 2.3 Minmax アルゴリズムの評価の例

Minmax アルゴリズムではある程度精度の良い評価関数が必要であり，囲碁では分岐数の大きさもあいまって強いプログラムを作ることの障害となっていた．モンテカルロ囲碁ではプレイアウトによる局面評価を導入すること，また有望な手や有望な手順に大きな探索資源を割き木の成長を制御することでこの障害に対処しようとしている．

2.1.3　モンテカルロ囲碁における評価関数

モンテカルロ囲碁において評価関数は主に 2 つのことに利用されている．

1 つ目はプレイアウトに用いてプレイアウトの質，ひいては局面評価の精度を向上させるという利用法である．図 2.4 に評価関数を用いて確率に偏りを持たせたプレイアウトの例を示す．図 2.1 と比べると囲碁らしい進行となっている．



(a) 初期局面　　(b) 20 手まで　　(c) 終局

図 2.4 評価関数によって確率に偏りを持たせたプレイアウトの例

単純にランダムに手を打つのではなく，良さそうな手ほど高い確率で打つように確率に偏りを持たせることで，より実戦に近いプレイアウトを行うようにすることができる．これによってプレイアウトが局面の良さに乗せるノイズを減らすことができ，良い手ほどプレイアウトの勝率も高くなると期待できる．すると木探索においても良い手ほど深く探索することができ，信頼性の高い結果が得られる．

2 つは目は木探索に用いるというものである．ここで Progressive Widening[9] という手法を用いる．これは全ての合法手を探索の対象とせず，手に評価値を与えて順位の高いものから順次探索対象として行く方法である．これによって良さそうな手は先に深く探索することができる．この評価値を与える際に評価関数を用いる．

2.2　パターンマッチング

囲碁プログラムにおけるパターンマッチングはある点の周囲の石の状態をパターンとして捉え，そのパターン毎に係数を与え利用するものである．このとき用いる範囲はある点を中心とした 3×3，5×5 などがある．囲碁プログラムにおいてパターンマッチングはモンテカルロ囲碁が主流となる前から広く行われていた．以前は Minmax 探索用の局面の静的評価関数に用いていた他，囲碁は合法手が多いため考える手を減らす目的でも用いられていた．モンテカルロ囲碁ではプレイアウトに用いてプレイアウトの精度を向上させ

たり木探索に用いて手を制限する目的で使用している．ここではモンテカルロ囲碁以前も含めてパターンマッチングの手法についてまとめる．

2.2.1 局所パターンの利用

清ら（1994）[10] は局所パターン知識を用いた次の着手生成を提案している．これは囲碁用語で「形」と呼ばれる狭い範囲の石の配置をプロの棋譜から取り出して次の着手に利用している．この際パターンは図 2.5 のように着手点からマンハッタン距離で 4 の範囲を用いている．

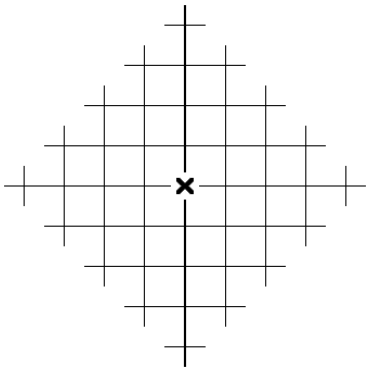


図 2.5 清 の用いたパターンの範囲

またこの際類似度を定義し類似度の高い点を次の着手点にしている．これによって実際に出現しなかったが似ているパターンについても評価している．類似度の式は以下の通りである．

$$\text{類似度} = \sum(\text{距離係数} \times \text{石の違い度}) \tag{2.1}$$

なお距離係数と石の違い度は図 2.1，2.2 の通りである．

表 2.1 距離係数

着手位置からの距離	1	2	3	4
距離係数	16	9	4	1

表 2.2 石の違い度

石が異なる（白 ↔ 黒）	-3
石の存在（空点 ↔ 石）	-1

プロの対局 73 局，第 1 手から 150 手までの約 10,000 局面についての出現頻度毎のパターン数は表 2.3 のようになっていた．

約 10,000 の局所パターンを用いたが実際のプロ棋譜に現れた 3 つの局面に対してプロの打った着手位置とプログラムが求めた着手位置は一致しなかった．その原因の 1 つとして手数が進むにつれて完全に一致するパターンが激減したためと考察している．また表

表 2.3 距離 3 と距離 4 の局所パターンの出現頻度

	50 以上	10 以上	5 以上	4 以上	3 以上	2 以上	1 以上	0
距離 3	8	61	146	215	350	751	7552	約 3^{25}
距離 4	6	28	82	107	186	425	9095	約 3^{41}

2.3 から一間トビのような典型的なパターンは範囲が狭いため，統計データでは多くのパターンに分散され，また 1 度しか現れないパターンが多いため多数の局所パターンを知識として扱わなければ，強い囲碁プログラムを作るのには有効でないと書いている．このように均等にパターンの範囲を拡大するだけではパターンが分散するという問題点がある．

2.2.2 MoGo の 3 × 3 パターンマッチング

MoGo（2006）[5] は着手点を中心とする 3 × 3 のパターンを用いている．大きなパターンではないがこれだけでもプレイアウトの質は大きく向上している．Mogo では 13 パターンを手動で生成している．また Don't Care（ワイルドカード，その点の状態を考慮しない）を用いており，これによってパターンのマッチ率が増加する効果がある．パターンの一部（ハネについて）を図 2.6 に示す．バツが着手点，? は Don't Care ，黒石とバツが重なっているのものは黒番時のみ考える．

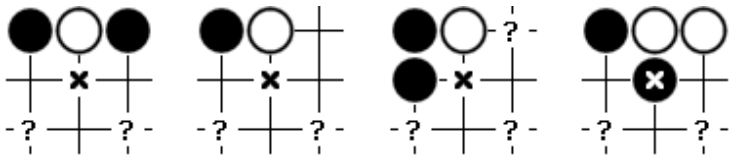


図 2.6 MoGo のハネ・パターン

しかし手動生成であるため大量のパターンを生成するのは難しく，また打って欲しいもしくは打って欲しくないパターンを定めるためには高度な囲碁の知識も必要となるという問題もあった．

2.2.3 パターンテンプレート

Stern ら（2006）[11] は図 2.7 の様な着手点を中央としてひし形のように広がって行くパターンテンプレートを用いている．パターンはプロの棋譜 181,000 局を用いて抽出し，BPM 分類機と ADF によって係数を学習している．これによって良さそうなパターンのみを用いることができる．またパターンは Zobrist hashing[12] によってハッシュ化して利

用している． そして学習時にはテンプレートでマッチングした最大のパターンを用いている．現在 Nomitan も似たようなパターンを用いているが，手数が進むと大きなパターンほどマッチングし難くなる．特にプレイアウト中では棋譜に表れない局面になりがちなのでその時にうまくマッチングしないことが考えられる．

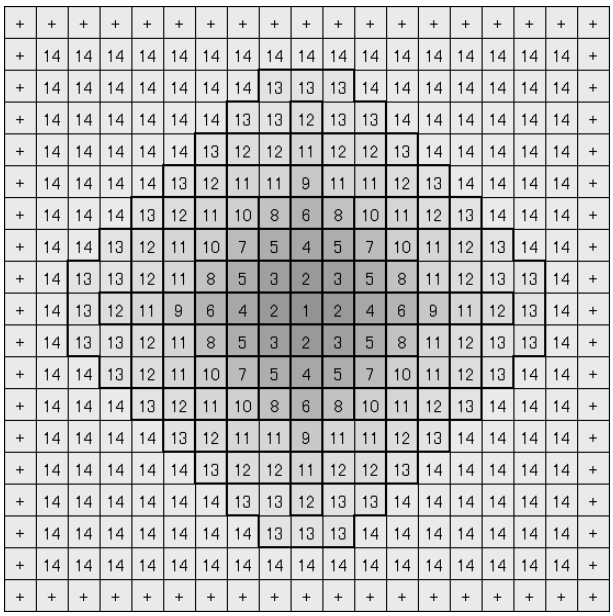


図 2.7 Stern の用いたパターンテンプレート

2.2.4 ファジーパターンマッチング

荒木ら（2006）[13] はファジーパターンマッチングという手法を提案している．これは着手点を中心とするパターンの周りの石の情報を「ファジー（曖昧）に」取り入れることで汎化性能の向上を試みている．汎化性能の向上はプレイアウト中でのパターンマッチングでも重要な要素となる．この際図 2.7 のパターンテンプレートは単純パターンとして用い，加えて図 2.8 のファジーパターンテンプレートを用いている．単純パターン内の石については正確に取り出し，その外については「ファジーに」取り出している．このとき単純パターンについてはマッチングした最大サイズを用いる．これは図 2.8 のある領域について黒石なら 1，白石なら -1 とし，その値を距離で割ったものの和をその領域の値とし，その値によって黒，白，中間の 3 つに分類した．つまり通常のパターンに周囲の各領域が黒っぽいか，白っぽいか，どちらでもないかという情報を付与するというマッチングを試みた．実際のファジーパターンマッチングの例を図 2.9 に示す．図 2.9(b) のように着手点の周囲は正確にマッチングを行い，その周りは黒っぽいか，白っぽいかを見る．

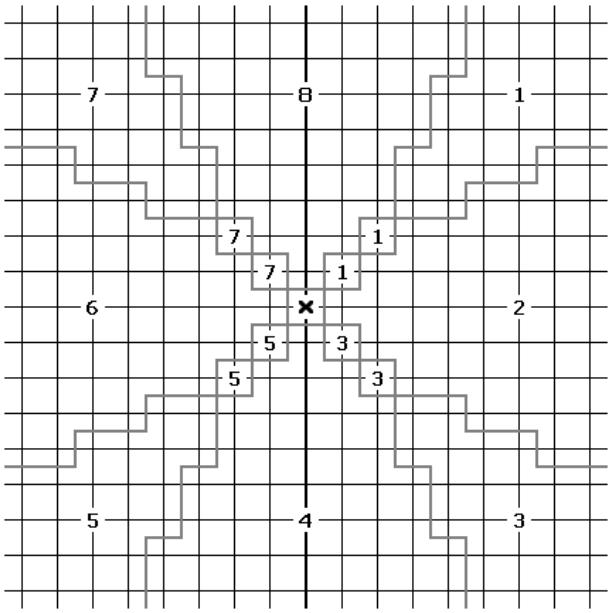
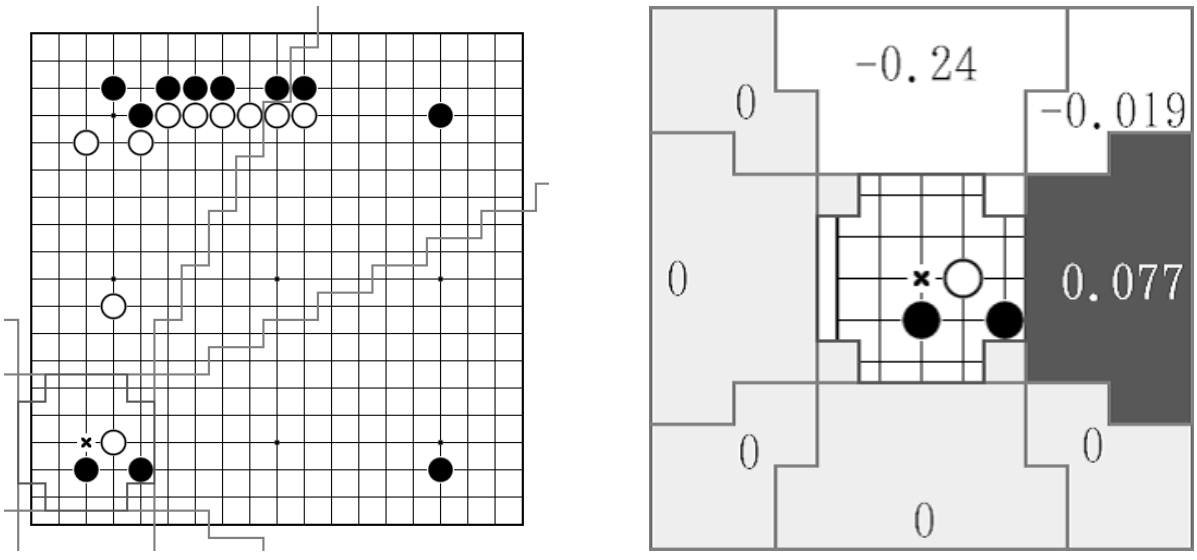


図 2.8 ファジーパターンテンプレート



(a) 局面の例　　(b) マッチング例

図 2.9 ファジーパターンのマッチング例

第3章　Nomitan プロジェクトの概要

3.1　探索法：UCT

Nomitan では探索に UCT（Upper Confidence bounds applied to Trees）[14] を用いている．UCT は代表的なモンテカルロ木探索アルゴリズムで，UCB（Upper Confidence bounds）[15] に従ってプレイアウトや木の成長をさせるアルゴリズムである．UCT では勝率の高い手ほど木が成長するため良い手ほど深く読むことができる．さらに Nomitan では UCT に Progressive Widening を用いており，最初から全ての手をプレイアウトの対象とせず，ある評価関数値に基づいて着手の順序付けをした上でプレイアウト数が増えるほど対象とする手を増やして行く．本論文は新しいパターンマッチングを提案するもので Nomitan の概要の詳細については他論文 [6][7] に譲る．

3.2　評価関数と特徴量

特徴数 K のときある手 m_i のパラメータベクトル $\vec{x}$（x_i は整数），特徴があるかないか判定する特徴関数 $l_k(m_i)$，評価関数 $g_{\vec{x}}(m_i)$ を以下のように定義する．

$$\vec{x} := [x_1, x_2, \ldots, x_K]^T \tag{3.1}$$

$$l_k(m_i) := \begin{cases} x_k & (\text{特徴がある}) \\ 1 & (\text{特徴がない}) \end{cases} \tag{3.2}$$

$$g_{\vec{x}}(m_i) := \prod_{k=1}^{K} l_k(m_i) \tag{3.3}$$

式 3.3 のようにある手の評価値は特徴量の積で表される．以上を用いてプレイアウトのある局面から合法手集合 M のある手 m_i を打つ確率 $f(m_i)$ を以下のように定義する．

$$f(m_i) = \frac{g_{\vec{x}}(m_i)}{\sum_{j=1}^{M} g_{\vec{x}}(m_j)} \tag{3.4}$$

ここで確率分布により偏りを持たせるためにフィルタを用いる．フィルタの特性としてフィルタリング後も順位が変わらず，値が正であることが必要であるため Nomitan では

パラメータベクトルを累乗している．フィルタリング後のパラメータベクトル $\vec{z}$ を以下のように定義する．

$$\vec{z} := [z_1, z_2, \ldots, z_K]^T \tag{3.5}$$

$$z_k = (x_k)^{\xi} \tag{3.6}$$

ξ の値は学習の結果や自己対戦によって適当に決定することとする．

特徴量として現在以下の 9 個を用いている．

I 着手点の位置どり

II 直前手との距離

III 2 手前の手との距離

IV トリ

V ノビ・逃げ

VI アテ

VII 救出トリ

VIII 自殺手

IX パターン

またいくつかの特徴には以下の式で表す距離 d[9] を用いている．δ_x, δ_y は 2 つの座標の x 成分，y 成分の差である．区別のため本論文ではこれを Remi 距離と呼ぶことにする．

$$d(\delta_x, \delta_y) = |\delta_x| + |\delta_y| + \max(|\delta_x|, |\delta_y|) \tag{3.7}$$

この距離の例を図 3.1 に示す．図のようにこの距離には 1 は存在しない．また着手点の位置（×印のある点）が距離 0 となる．

次に各特徴量について詳しく説明する．この特徴量を評価関数に用いて局面の各合法手の評価を行う．Nomitan ではこの評価関数をプレイアウト中の確率の生成と木探索（UCT）での手の制限（Progressive Widening）に用いている．プレイアウトに用いる方を MC 用評価関数，木探索に用いるものを UCT 用評価関数と呼ぶことにする．MC 用評価関数はプレイアウト中で用いるため基本的に高速である方が良いが精度の良いプレイアウトを行えるのであれば速度を犠牲にしても良い結果となる場合がある．また MC 用評価関数は確率を与えるものなのでその比率が重要となる．UCT 用評価関数は MC 用評価関数に比べて呼ばれる回数は少ないため速度が遅くても精度が良いことが望ましい．また UCT 用評価関数では各合法手の（評価値の絶対的な大小ではなくて）相対的な順位が重要という大きな違いがある．

なお以下に $C_{feature,element}$ のような表記があるが，これは特徴毎の要素の最大値である．この値は MC 用評価関数と UCT 用評価関数で異なる（同じものもある）．最後に MC, UCT 用での各値を表にまとめる．

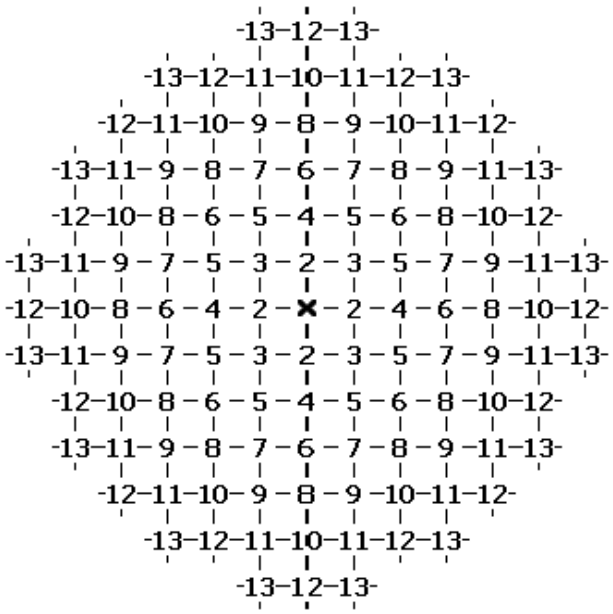


図 3.1 距離テンプレート

I. 着手点の位置どり

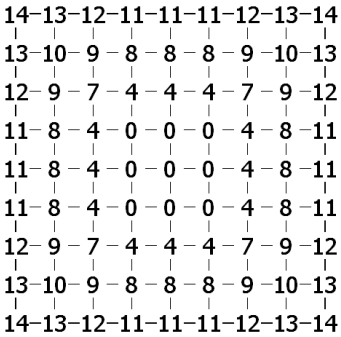
着手点が盤上のどこかを分類する．図 3.2 にこの分類を示す．図のように盤上を 0 から 14 の 15 通りに分類している．これは角や辺などは同じ値となるようにしている．また 9 路の場合は小さいためいくつかの数が存在せず，全部で 10 通りに分類している．0 から 9 にしていないのは 19 路での値に合わせているためで，例えば角は両方 14 となるようにしている．通常第三，四線の手（1 から 7）は他より優先度が高く，第一線の手（11 から 14）は低い．

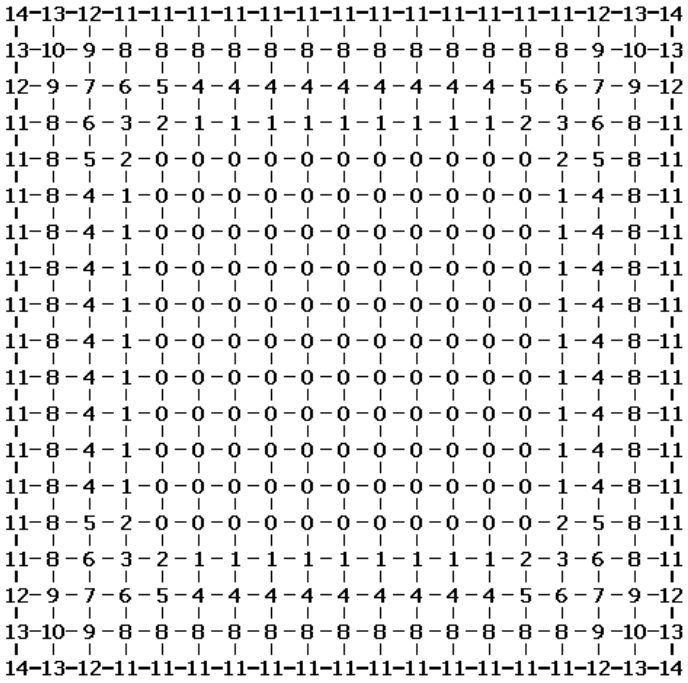
II. 直前手との距離

着手点と直前手との Remi 距離によって分類する．$distance1(1, 2, \ldots, C_{distance1}$以上) で分類される．囲碁は Markov 性を持つゲームではあるが，最近の着手の付近は落ち着いておらずに急ぐべき手が多くあることを鑑み II，III の特徴量を導入している．

III. 2 手前の手との距離

着手点と 2 手前の手との Remi 距離によって分類する．$distance2(1, 2, \ldots, C_{distance2}$以上) で分類される．





14	13	12	11	11	11	12	13	14
13	10	9	8	8	8	9	10	13
12	9	7	4	4	4	7	9	12
11	8	4	0	0	0	4	8	11
11	8	4	0	0	0	4	8	11
11	8	4	0	0	0	4	8	11
12	9	7	4	4	4	7	9	12
13	10	9	8	8	8	9	10	13
14	13	12	11	11	11	12	13	14

(a) 9 路の場合

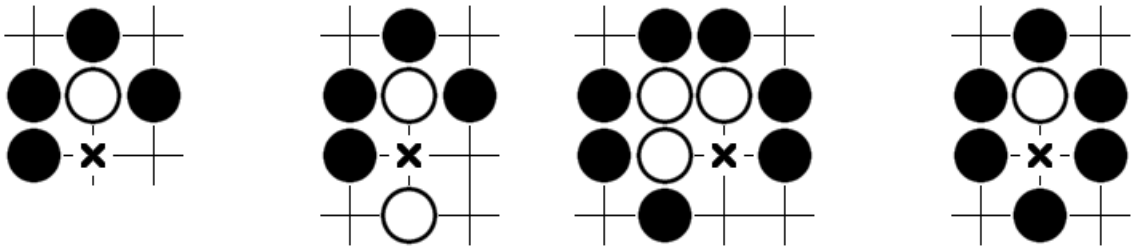
14	13	12	11	11	11	11	11	11	11	11	11	11	11	11	11	12	13	14
13	10	9	8	8	8	8	8	8	8	8	8	8	8	8	8	9	10	13
12	9	7	6	5	4	4	4	4	4	4	4	4	4	5	6	7	9	12
11	8	6	3	2	1	1	1	1	1	1	1	1	1	2	3	6	8	11
11	8	5	2	0	0	0	0	0	0	0	0	0	0	0	2	5	8	11
11	8	4	1	0	0	0	0	0	0	0	0	0	0	0	1	4	8	11
11	8	4	1	0	0	0	0	0	0	0	0	0	0	0	1	4	8	11
11	8	4	1	0	0	0	0	0	0	0	0	0	0	0	1	4	8	11
11	8	4	1	0	0	0	0	0	0	0	0	0	0	0	1	4	8	11
11	8	4	1	0	0	0	0	0	0	0	0	0	0	0	1	4	8	11
11	8	4	1	0	0	0	0	0	0	0	0	0	0	0	1	4	8	11
11	8	4	1	0	0	0	0	0	0	0	0	0	0	0	1	4	8	11
11	8	4	1	0	0	0	0	0	0	0	0	0	0	0	1	4	8	11
11	8	4	1	0	0	0	0	0	0	0	0	0	0	0	1	4	8	11
11	8	5	2	0	0	0	0	0	0	0	0	0	0	0	2	5	8	11
11	8	6	3	2	1	1	1	1	1	1	1	1	1	2	3	6	8	11
12	9	7	6	5	4	4	4	4	4	4	4	4	4	5	6	7	9	12
13	10	9	8	8	8	8	8	8	8	8	8	8	8	8	8	9	10	13
14	13	12	11	11	11	11	11	11	11	11	11	11	11	11	11	12	13	14

(b) 19 路の場合

図 3.2 着手点の位置どりの割当て表

IV. トリ

トリは相手の連を取るという特徴である．つまり着手点に隣接する相手の連の呼吸点が 1 の時である．取る連の石の数 $stone(1, 2, \ldots, C_{tori,stone}$以上$)$, 相手が着手点に着手したときに増える相手の呼吸点の数 $increase(1, 2, \ldots, C_{tori,increase})$ によって分類される．図 3.3 に例を示す．囲碁では石を取ることが直接の勝利条件ではないが，相手の石を取り除くことは相手の地を減らす，自分の地を増やす，自分の石を強化するなどの利点を持つため係数は高くなる傾向にある．



(a) stone = 1, increase = 1　(b) stone = 1, increase = 3　(c) stone = 3, increase = 0　(d) stone = 1, increase = -1

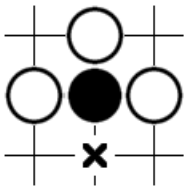
図 3.3 トリの特徴の例

V. ノビ・逃げ

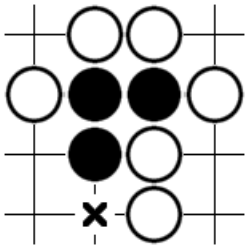
ここではノビは隣接する自分の連の呼吸点が 1 か 2 の状態とする．隣接する自分の連の石の数 $stone(1, 2, \ldots, C_{nobi,stone}$以上$)$ とその呼吸点の数 $liberty(1, 2, \ldots, C_{nobi,liberty}$以上$)$，着手によって呼吸点がいくつ増えるか $increase(1, 2, \ldots, C_{nobi,increase}$以上$)$，隣接する連が直前に更新した連かどうか $update(0, 1)$ によって分類される．$update$ は UCT 用評価関数にのみ使用される． 図 3.4 に例を示す．この図では $update$ は考慮していない．この特徴量を用いることで自石を助ける手を優先することができる．ノビはトリ，アテを防ぐ特徴量であるといえる．

VI. アテ

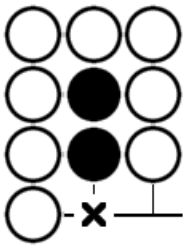
アテは着手点に石を置くことで相手の連をアタリの状態にするような特徴である．これは隣接する相手の連が 2 のときである．対象となる相手の石の数 $stone(1, 2, \ldots, C_{ate,stone}$以上$)$，隣接する連が直前に更新した連かどうか $update(0, 1)$ によって分類される．V と同じく $update$ は UCT にしか用いられない．図 3.5 に例を示す（$update$ は考慮していない）．アテはトリの前段階にある特徴量であると言える．



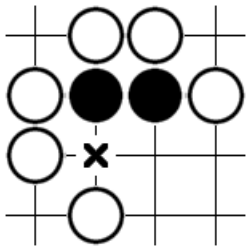
(a) stone = 1,
liberty = 1,
increase = 3



(b) stone = 3,
liberty = 2,
increase = 1

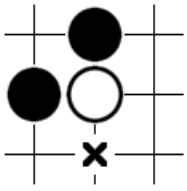


(c) stone = 2,
liberty = 1,
increase = 0

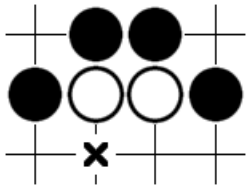


(d) stone = 2,
liberty = 2,
increase = -1

図 3.4 ノビの特徴の例



(a) stone = 1

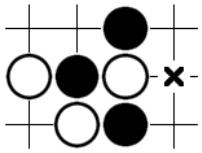


(b) stone = 2

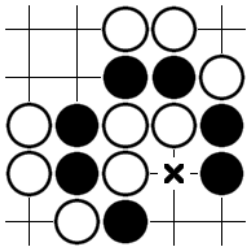
図 3.5 アテの特徴の例

VII. 救出トリ

救出トリは隣接する相手の連の呼吸点が 1（取れる時）でかつその連が自分の連を当てているときに現れる特徴である．隣接する相手の連が当てている自分の連の石の数 $stone(1, 2, \ldots, C_{nuki,stone})$ によって分類される．また複数の連を当てている場合はそれぞれ計算される．図 3.6 に例を示す．救出トリは一石二鳥の手であるためトリよりさらに価値が高い場合が多い．なお救出トリの特徴量が適用される場合，トリの特徴量も別途適用される．



(a) stone = 1



(b) stone = 2, stone = 2

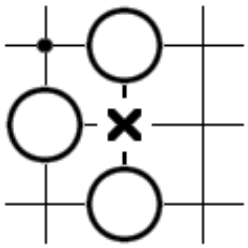
図 3.6 救出トリの特徴の例

VIII. 自殺手

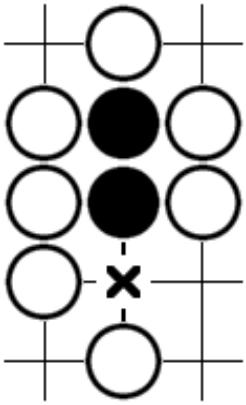
これは自ら取られに行くような手を打つという特徴である．着手点に置いたときに自分の連の呼吸点が 1 の時に表れる．石を置いたとき置いた石を含む自分の連の石の数 $stone(1, 2, \ldots, C_{suicide,stone})$ によって分類される．図 3.7 に例を示す．自殺手は忌避すべき手であるが，しばしばアテのために自殺手を打つことがあり，それを防ぐための特徴量である．一方でナカデの場合は自殺手を打たなければ相手の連を取りに行けないため自殺手を一律に禁止するわけにはいかない．

IX. パターン

パターンは着手点近傍の石の配置による特徴である．黒石，白石，その他（空点，盤外）を情報として取り込む．これまでは着手点を中心として Remi 距離 d で $C_{pattern,distance}$ の範囲をパターンとして取り込んでいた．この際着手点の位置どりによって同じパターンであっても異なる特徴とした．これは図 3.2 と同じように 15 個所に分類している．また回転を考慮しており，場所が異なっていても同様のパターンは同一視する．例を図 3.8 に示す．A を基準としたとき，B, C は位置が異なるため違う特徴となる．D は位置が同じで



(a) stone = 1



(b) stone = 3

図 3.7 自殺手の特徴の例

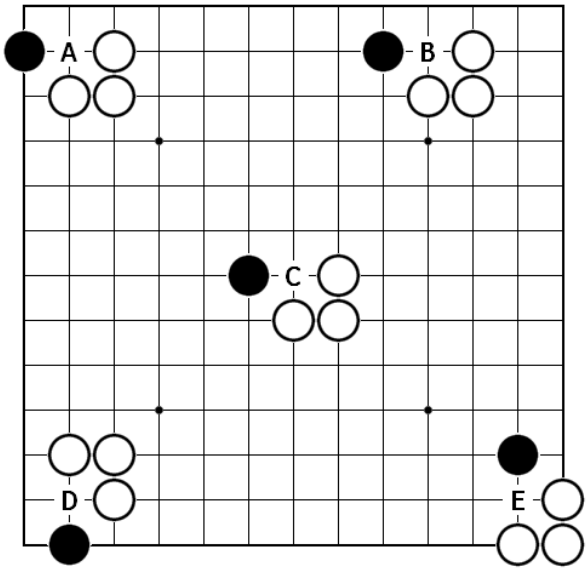


図 3.8 パターンの例

回転させると同じパターンなので同じ特徴となる．E は位置が同じでパターンも似ているが A で第一線にある黒石が E は第二線にあるため異なるパターンである．

パターンは Stern と同様に Zobrist hashing によってハッシュ化して利用している．パターンの抽出方法は次の通り．

1. 大きさ $Size$ のハッシュテーブルを用意する．
2. 棋譜を読み込む．
3. 各局面の各合法手について周囲のパターンのハッシュ値を計算しハッシュテーブルに加える（開番地法を用いる）．この際出現回数をカウントする．
4. ハッシュテーブルの $1/Limit$ が埋まったら出現回数が $Border$ 以下のパターンを削除する．
5. 2 から 4 を繰り返す．
6. 全棋譜を読み込んだら最後に $Sample$ 回以上表れたパターンのハッシュ値をファイルに出力する．

各定数の値は表 3.1 の通り．19 路の方がパターン数が多いため $Border$ と $Sample$ を大きくしている．取ることのできるハッシュテーブルのサイズは学習用マシンのメモリやプログラム言語の使用によって有限であるため大きいパターンの場合うまくパターンを抽出できない可能性がある．また少数回しか登場しないパターンでは十分な学習が行えないため，手順 6 のように最低登場回数を定めて足切りを行っている．

表 3.1 パターンのハッシュ化に関する定数

	9 路	19 路
$Size$	40000000	40000000
$Limit$	2	2
$Border$	50	100
$Sample$	100	200

MC，UCT 用評価関数のでの各特徴の要素の最大値を表 3.2 にまとめる．ほとんどの項目で同じであるが，II，III の計算する距離の範囲，V，VI の $update$ ついては計算量の関係で MC 用評価関数を軽くしている．VIII はナカデのような手を候補から外さないよう UCT では用いていない．

表 3.2 各特徴の要素

	UCT	MC
I. 着手点の位置どり	$C_{position} = 15$	$C_{position} = 15$
II. 直前手との距離	$C_{distance1} = 13$	$C_{distance1} = 5$
III. 2 手前の手との距離	$C_{distance2} = 13$	不使用
IV. トリ	$C_{tori,stone} = 3$ $C_{tori,increase} = 2$	$C_{tori,stone} = 3$ $C_{tori,increase} = 2$
V. ノビ・逃げ	$C_{nobi,stone} = 3$ $C_{nobi,liberty} = 3$ $C_{nobi,increase} = 2$ $update$ 使用	$C_{nobi,stone} = 3$ $C_{nobi,liberty} = 3$ $C_{nobi,increase} = 2$ $update$ 不使用
VI. アテ	$C_{ate,stone} = 3$ $update$ 使用	$C_{ate,stone} = 3$ $update$ 不使用
VII. 救出トリ	$C_{nuki,stone} = 3$	$C_{nuki,stone} = 3$
VIII. 自殺手	不使用	$C_{suicide,stone} = 3$
IX. パターン	$C_{pattern,distance} = 7$	$C_{pattern,distance} = 7$

3.3 評価関数係数の学習

Nomitan では勾配法を用いた学習を行っている [6]．ここでは Nomitan で行っている学習について簡単に説明する．棋譜サンプル中のある局面 p_h で実際に打たれた手 $p_{h,0}$ を教師信号とし，教師信号以外の全ての手 $p_{h,j}$ を訓練集合としたとき誤差関数 $J_{\vec{x}}(H)$ を以下のように定義する．

$$J_{\vec{x}}(H) \quad := \quad \sum_{h=1}^{H} \sum_{j=1}^{M_h} T[g_{\vec{x}}(p_{h,j}) - g_{\vec{x}}(p_{h,0})] \tag{3.8}$$

ここで H は局面のサンプル数，M_h は局面 h での合法手数である．$g_{\vec{x}}$ は評価関数（式 (3.3)）であり $g_{\vec{x}}(p_{s,j})$ は局面 p_s で手 $p_{s,j}$ を打ったときの評価値となる．$T[]$ は誤差汎化のために以下のシグモイド関数を用いる．これによって悪手や難解な手の影響を軽減し学習を安定化させている．

$$T[x] \quad = \quad \frac{1}{1 - e^{1000 \cdot x}} \tag{3.9}$$

パラメータベクトル $\vec{x}$ による誤差関数 $J_{\vec{x}}(S)$ の x_i に対する偏微分は以下のようになる．

$$\frac{\partial J_{\vec{x}}(H)}{\partial x_i} \quad = \quad \sum_{s=1}^{H} \sum_{j=1}^{M_h} \left[\frac{\partial J_{\vec{x}}(H)}{\partial \vec{x}} \right] \tag{3.10}$$

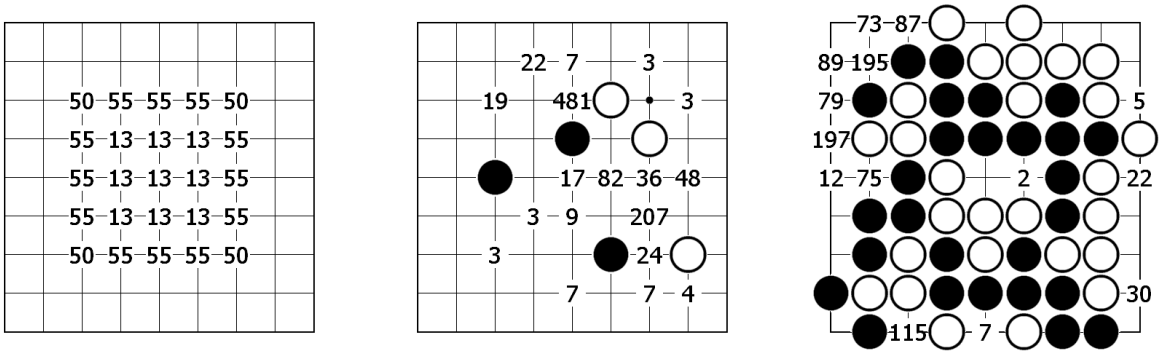
$J_{\vec{x}}(H)$ を最小化するためパラメータベクトル x_i はこの勾配が負になる方向に反復的に更新して行く．このとき更新幅を徐々に小さくすることで値を収束させている．この勾配法による学習は特徴量の数によらないため，本論文では全てこの方法を用いた係数の学習を行っている．

3.4 モンテカルロシミュレーション

モンテカルロシミュレーション（プレイアウト）では次式（3.4 を再掲）に基づいて確率的に手を選択して行く．

$$f(m_i) = \frac{g_{\vec{x}}(m_i)}{\sum_{j=1}^{M} g_{\vec{x}}(m_j)}$$

初期局面の確率分布の例を図 3.9 に示す．数値の単位は ‰（パーミル）で 1000 で 100% となる．また 1 ‰ 以下の点の数値は省略している．また現在はある一定確率以下の手は選択しないようにしている．これによって明らかに良くなさそうな手を除外している．図 3.9 でいえば数字の無い点である．普通であればこのような個所に打つことは無いのでこれによってプレイアウトの質の向上を図っている．この確率 P_{cutoff} は 9 路では 0.01，13 路では 0.005，19 路では 0.002 としている．図 3.9(b) を見ると人間であれば打ちたいと思う (5, 3)，(7, 6) の点が高くなっている．また図 3.9(c) を見ると (3, 9) のトリや (1, 4)，(2, 2) のアタリにする手の確率が高くなっている．



(a) 初期局面　(b) 6 手進んだ局面（黒番）　(c) 56 手進んだ局面（黒番）

図 3.9 確率分布の例

第4章　コラージュパターンの提案と評価

ここでは従来手法の問題点を解決する方法としてコラージュパターンを提案する．さらにパターンマッチに関する多様な評価法の必要性を整理した上で提案手法とその評価を行う．

4.1　従来のパターンマッチングの問題点

4.1.1　従来研究とその問題点

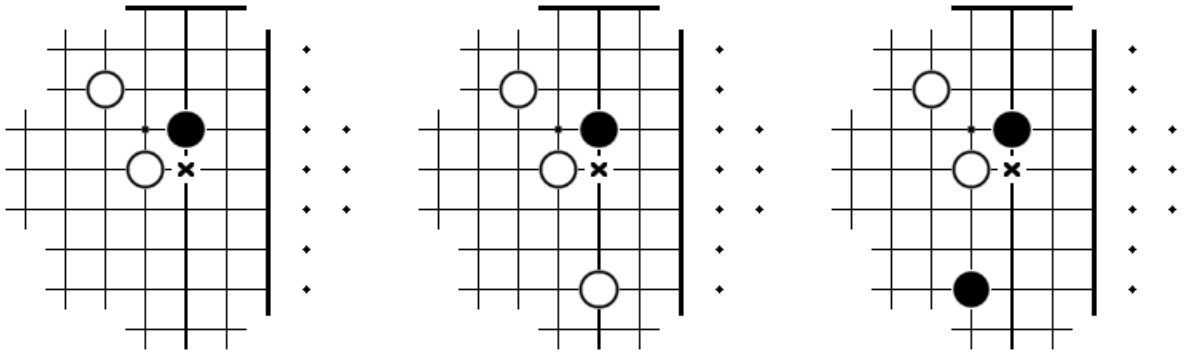
従来のパターンマッチングでは通常着手点を中心とするパターンを用いるが，広い範囲でマッチングを行おうとすると多くの点の石の状態を見る必要があった．その分パターン数は膨大な量となってしまい，メモリに保存できる数の制限にかかると同時に，完全に一致する割合が著しく減少することで汎化性能にも問題が生じる．表 4.1 に 19 路の場合の Remi 距離（2, 3, . . . , 9）の範囲をマッチングする際に見る点の数とその大体のパターン数を示す．表の通りパターンが大きいほどパターン数は膨大となる．メモリに全パターンをのせることができるのは Remi 距離 5 ぐらいが限界であるが，この大きさでは着手の善悪を精度良く評価するには不十分である．そのため 3.2 節のようにパターンをハッシュ化して用いている．

表 4.1 Remi 距離によるパターンのパターン数の変化

Remi 距離	点の数	パターン数
2	4	10^2
3	8	10^4
4	12	10^6
5	20	10^{10}
6	28	10^{14}
7	36	10^{17}
8	48	10^{23}
9	60	10^{28}

4.1.2　従来パターンで問題となる実例

例えば図 4.1 の 3 つの局面を考える．これは着手点（×印）から Remi 距離で 9 の範囲のパターンを見ており，右方の点はパターンの範囲が盤外にあることを示す．これらは左上の石の配置は同一で下方の石の状態のみが異なっている．石の配置が異なるため異なるパターンとなるが，どれも候補手としてはありそうな（着手が評価されるべき）手である．ここで重要となるのは左上の石であり下方の石の存在は別に捉えても良いのではないかという考え方が以下の提案手法である．



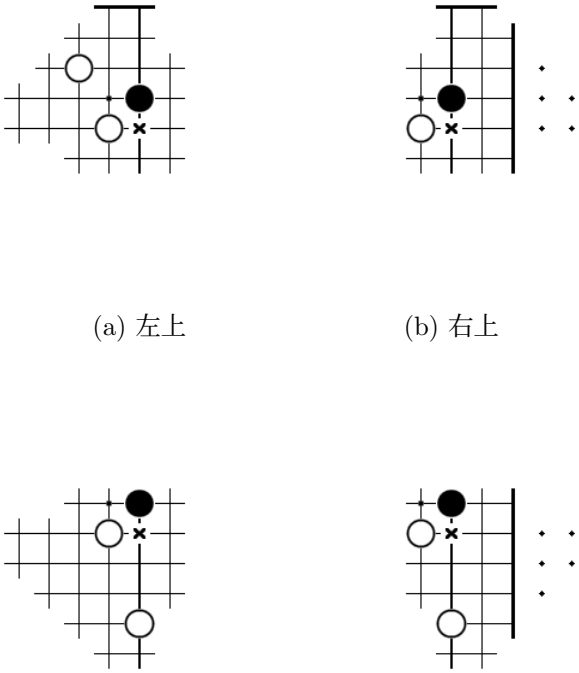
(a) 局面 A　　(b) 局面 B　　(c) 局面 C

図 4.1 従来手法において問題となる例

4.2　コラージュパターン

本論文では上記の問題点を改善するために，従来の着手点を中心とする円のような形のパターンを 4 分割し，これを貼り絵のように組み合わせて大きな範囲を表することを提案する．貼り絵のように重ね合わせることからコラージュパターン（以下 CP）と名付けた．例えば図 4.1(b) の局面を図 4.2 の様に左上，右上，右下，左下の 4 つに分割する．これによって図 4.1 の 3 つの局面について左上は同一として処理できる．全体としては図 4.3 のように図 4.1(b) とほぼ同じ範囲を見ることになるが，4 点だけ少なくなっている．

CP は図 4.4 のように段階を踏んで拡大することとした．重複する点があるのは最低限で周囲 4 点，次に周囲 8 点を見るようにするようにしたいということと，一間トビ，二間トビやケイマのような囲碁において重要なパターンを取り入れるためである．使用する際はマッチした最大のパターンを用いることとした．CP の各サイズの大体のパターン数を表 4.2 に示す．点の数が変わっていないのに表 4.1 よりパターン数が増えているのは 4 つに分割しているためである．最大パターンは Remi 距離で 9 とほぼ同等の範囲（4 点少ない）を見ながらも見ている点の数は距離 6 より少ない．その分大きい範囲でのマッチングがし易くなると考えた．



(a) 左上　(b) 右上

(c) 左下　(d) 右下

図 4.2 CP の例

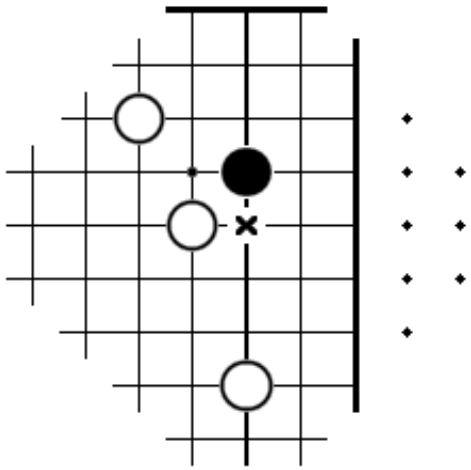


図 4.3 CP の全体の大きさ

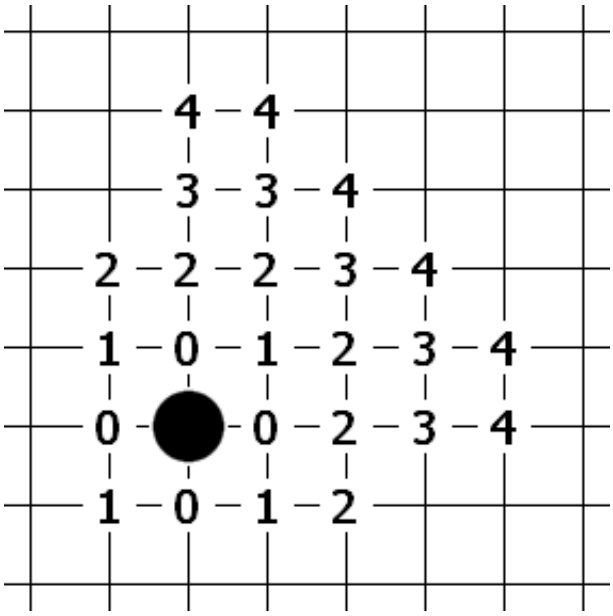


図 4.4 CP のサイズと段階

表 4.2 CP のパターン数の変化

段階	点の数	パターン数	従来パターンとの被覆の比較
0	4	10^3	= Remi 距離 2
1	8	10^5	= Remi 距離 3
2	14	10^8	= Remi 距離 5
3	19	10^{10}	= Remi 距離 7
4	25	10^{13}	≒ Remi 距離 9

CP の疑似コードを Algorithm 1 に示す．$calculateCPScore$ は盤の状態 $board$ のときに盤上のある点 $position$ に手番 $color$ が置いたときのスコアを計算するアルゴリズムである．ある方向について各 $size$ でのハッシュ値 $hashValue_{direction,size}$ を計算し，次に最大サイズからマッチングするか最小サイズになるまで $size$ を減少させ，そのサイズのスコア $Score_{direction}$ を計算する．そして 4 つの方向のスコアの積を返す．

Algorithm 1 コラージュパターンの疑似コード

Algorithm $calculateCPScore(board, position, color)$
for $direction = 1$ **to** 4 **do**
  for $size = 0$ **to** 4 **do**
    $hashValue_{direction,size,color}$ を $board$, $position$ から計算
  end for
  $size \leftarrow maxSize$
  while マッチングする **or** $size = 0$ まで **do**
    $size = size - 1$
  end while
  $hashValue_{direction,size}$ より $Score_{direction}$ を計算する
end for
return $\prod_{direction=1}^{4} Score_{direction}$

4.3 パターンマッチングの階層的評価

パターンマッチングの評価方法には，調べたい特徴やパターンの用とによって様々なものが考えられるが．パターンの抽出時に決まるもの，学習によって決まるもの，フィルタ値によるもの，実際にプレイアウトに用いて分かるもの，そして実際のプログラムの強さなどがある．そこでこれらの評価項目をまとめることにする．これは様々な項目で多くの

視点から評価することでどのようなプレイアウトが良いか，確率はどうあるべきか，何が強さにとって重要なのかを調べるためである．

現在我々のプロジェクトで用いている評価項目は表 4.3 の通りである．決定時期はその項目が決定する時期，フィルタ値はフィルタ値に依存するか，番号は各項目の番号でパターン抽出時に決定するものは P，学習時に決定するものは L，使用によって決定するものは U が付き，またフィルタ依存のものは f が付く．また似た項目の場合は a, b が付く．基本的にこの表は階層を示しており，上の項目の評価は下の項目の評価に影響を与えると考えることができる．例えば自己対戦はプレイアウト速度，占有率（プレイアウトの質），平均選択確率などの影響を受ける．次に各項目について説明する．なお各項目はテストに用いた全局面について評価関数によって合法手の各手に評価値を与え評価値の大きい順に並べているとする．この中にはその善悪をスカラで表現できないものも含む．

表 4.3 Nomitan で用いている評価項目

決定時期	フィルタ値	番号	項目名	略称
パターンの抽出時	非依存	$P-1$	パターン数に関する情報	$Pattern$
学習時	非依存	$L-1$	誤差関数値	$Error$
		$L-2.a$	平均順位	$Rank$
		$L-2.b$	平均逆数順位	$Rank_{Inv}$
		$L-2$	誤差関数値	$Error$
		$L-3.a$	累積一致率分布	$Match_n$
		$L-3.b$	累積一致率係数	Co_{Match}
	依存	$Lf-4.a$	平均選択確率	$Select$
		$Lf-4.b$	平均ルート選択確率	$Select_{Root}$
		$Lf-5$	選択確率 α 到達割合	$Over_\alpha$
		$Lf-6$	α 非着手率	Not_α
		$Lf-7$	平均対数尤度	MLE
使用時	非依存	$U-1$	パターンの最大マッチングサイズ	$Size_{Max}$
	依存	$Uf-2$	占有率	$Ownership$
		$Uf-3$	形勢判断問題	$Judge$
		$Uf-4$	次の一手問題	$Nextmove$
		$Uf-5$	プレイアウト速度	$Speed$
		$Uf-6$	自己対戦	$Selfplay$
		$Uf-7$	他のプログラムとの対戦	$Battle$
		$Uf-8$	CGOS でのレーティング計測	$Rating$

$P-1$ パターン数に関する情報（$Pattern$）

これはパターンの抽出によって得られたパターンについての情報である．主な情報はパターン数であるが，他にも登場回数の多いパターンがどのようなパターンかを調べるという使い方もできる．

$L-1$ 誤差関数値（$Error$）

最適化対象である誤差関数（式 3.8 ）の値を**誤差関数値** $Error$ と呼ぶ．小さいほど最適化されているといえる．式 3.8 はどちらかというと順位を優先する指標であり，確率分布については考慮しないためフィルタ値を導入している．これは UCT 用評価関数の評価に用いることができる．またこの値は誤差関数が変われば何を最適化するかも変わるため，それによって用途も変わり得る．

$L-2.a,b$ 平均順位と平均逆数順位（$Rank, Rank_{Inv}$）

教師信号が平均的に何位であったかを**平均順位** $Rank$ と呼び，低い方が良い．また**平均逆数順位** $Rank_{Inv}$ は順位の逆数の和の逆数をとったもので，低い方が良い．これらは r_h を局面 h での教師信号の順位としたとき，次のように表される．

$$Rank = \frac{\sum_{h=1}^{H} r_h}{H} \tag{4.1}$$

$$Rank_{Inv} = \frac{\sum_{h=1}^{H} \frac{1}{r_h}}{H} \tag{4.2}$$

順位が重要となる UCT 用評価関数において重要である．$Rank_{Inv}$ は $Rank$ に比べると上位の順位を重視する傾向がある．

$L-3.a,b$ 累積一致率分布と累積一致率係数（$Match_n, Co_{Match}$）

教師信号が 1 位（評価値最大）となった割合を**1 位一致率**と呼ぶ．同様に n 位となった割合を**n 位一致率**と呼ぶ．これを 1 位から合法手数まで累積したグラフを**累積一致率分布**と呼ぶ．ここで n 位までの一致率の和を**n 位累積一致率**と呼び，$Match_n$ と表記する．累積一致率分布の例を図 4.5 に示す．A，B 2 つのグラフがあるがどちらも 1 位一致率は 30% である．しかしその後の分布が異なる．B の方が上位に集中しているため，B の方が良いといえる．この累積一致率の良さを示す指標として**累積一致率係数** Co_{Match} と

いうものを考える．これは式 4.3 で表される．m は合法手数で $Match_i$ は i 位累積一致率である．

$$Co_{Match} = \frac{\sum_{i=1}^{m} Match_i}{m} \tag{4.3}$$

もし全ての手が 100% で教師信号と一致するとこの値は 1 となる．この値が大きいほど上位に手が集中していると言える．これは図 4.6 のように累積一致率を棒グラフで表したときの面積を求め，それを合法手数で割っている．UCT 用評価関数では手の制限に用いるため教師信号がある程度上位（例えば 20 位以内）にあるほうが好ましいのでこれらの指標が重要となる．

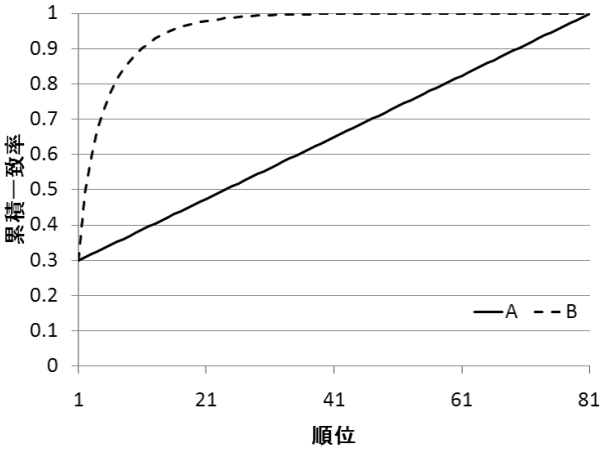


図 4.5 累積一致率の例

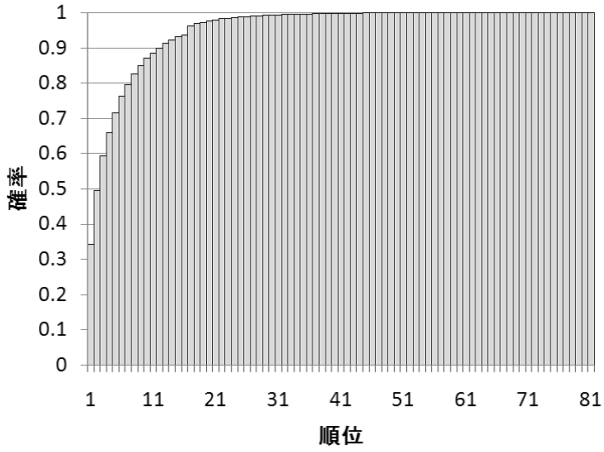


図 4.6 棒グラフで表した累積一致率

$Lf-4.a,b$ 平均選択確率と平均ルート選択確率（$Select, Select_{Root}$）

教師信号の選択確率の平均を**平均選択確率** $Select$ とする．高い方が良く，特に MC 用評価関数において重要となる．また中程度の確率の場合を相対的に強調するために教師信号の確率のルートを取り平均したものの 2 乗を**平均ルート選択確率** $Select_{Root}$ とする．

これらは m_h を局面 h での教師信号，$f(m_h)$ をその確率としたとき次のように表される．

$$Select = \frac{\sum_{h=1}^{H} f(m_h)}{H} \tag{4.4}$$

$$Select_{Root} = \left(\frac{\sum_{h=1}^{H} \sqrt{f(m_h)}}{H} \right)^2 \tag{4.5}$$

$Lf-5$ 選択確率 α 到達割合（$Over_n$）

教師信号の確率が α 以上であった割合を**選択確率 α 到達割合** $Over_\alpha$ と呼ぶ．たとえば選択確率 20 到達割合 $Over_{20}$ が 0.3 だと全局面の 3 割で教師信号の確率が 20% 以上だったということになる．これは高い方が良いが UCT ではあまり関係がない．これらは $\mathscr{H}$ を局面集合，P_h を局面 h での教師信号の確率としたとき次のように表される．

$$Over_\alpha = \frac{|\{h \in \mathscr{H} | P_h > \alpha\}|}{H} \tag{4.6}$$

$Lf-6$ α 非着手率（Not_n）

ある手の確率が α 以上であるのに教師信号でなかった割合を α **非着手率** Not_α と呼ぶ．たとえば 30 非着手率 Not_{30} が 0.5 だとある手の確率が 30% 以上であるのに棋譜では打たれなかった割合が 5 割であったことになる．特に MC 用評価関数では非常に高い確率で選択される手が実は良くない手であるとするとプレイアウトに与える悪影響は非常に大きい．これは次のように表される．式 4.7 の右辺第 2 項は α 着手率で，これを 1 から引いたものが α 非着手率となる．

$$Not_\alpha = 1 - \frac{|\{h \in \mathscr{H} | P_h > \alpha\}|}{|\{h \in \mathscr{H}, j \in M_h | P_{h,j} > \alpha\}|} \tag{4.7}$$

$Lf-7.a,b$ 平均対数尤度とその平均（$MLE, MLE_{Average}$）

教師信号の確率の log を取り全棋譜の n 手目についてその平均を取ったものを**平均対数尤度**（Mean Log Evidence）と呼び，他論文 [11][9] でも用いられている．n 手目の平均対数尤度 MLE_n は次のように表される．k は全棋譜で n 手目が表れた回数，$f_{n,i}$ は n 手目の i 番目の確率である．

$$MLE_n = \frac{\sum_{i=1}^{k} \log(f_{n,i})}{k} \tag{4.8}$$

もし教師信号の確率が 100% であれば 0 となるので，0 に近いほど良いといえる．log を取るため $Select_{Root}$ に近いが，低い確率をより重視している．また全手数での平均対数尤度を平均した $MLE_{Average}$ も計算できる．ただしこの値は他の項目で代用できるため Nomitan では用いていない．

$U-1$ パターンの最大マッチングサイズ（$Size_{Max}$）

プレイアウトにおいてどれだけ大きいサイズにマッチングしているかを示す．初期局面（石がまったく置かれていない局面）からのプレイアウト中の各手数で全合法手についてどのサイズのパターンにマッチングしたかをグラフにしたもの．できるだけ大きなサイズでマッチングしていた方が局所化性能が高いと言える．

$Uf-2$ 占有率（$Ownership$）

複数回プレイアウトを行った時盤上の各点が最終的に黒白どちらの地であったかの割合のことを**占有率**と呼ぶ（図 4.10 がその例である）．普通に打てば黒地になる点について黒の占有率が高ければ良いプレイアウトと言える．これはその値の大小を比較することで善し悪しを評価する．定量的な判断は全ての交点に最終的にどちらの地になるかを指定しなくてはならないため難しいが，人間が見て石の強弱や勢力の大小を正しく反映しているかを見るには適している．

$Uf-3$ 形勢判断問題（$Judge$）

a の方が勝率が高くなるような 2 つの似た局面 a，b を与え，それぞれに対して複数回プレイアウトを行いその勝率を比較するというものである．a の勝率が b より大きいほどプレイアウトの質が良いと言える．

$Uf-4$ 次の一手問題（$Nextmove$）

ある局面について人間が各交点にスコアを与えておき，その局面に対して探索を行い選ばれた最善手の交点のスコアがいくつかを求める．探索は複数回行い平均でどれだけのスコアを取れたかを調べる．プレイアウトの質が良ければ良い手を早く，重点的に調べることができるのでプレイアウトの評価に用いることができる他，探索の評価にも用いることができる．囲碁の問題集の多くはこの形式に近いが，正解手を一手に限らずまた失敗手の問にも序列を設けることでより頑健な評価を可能にしている．

$Uf-5$ **プレイアウト速度（*Speed*）**

1 秒当たりのプレイアウトの回数をプレイアウト速度と呼ぶ．広い範囲を見るパターンマッチングでは速度が落ちやすい傾向がある．モンテカルロ囲碁ではプレイアウト速度は速い方が良いが，プレイアウトの質が良ければ遅くても強いということはありうる．

$Uf-6$ **自己対戦（*Selfplay*）**

Nomitan 同士で対戦をさせ，その勝率を見るもの．本論文ではこれまで用いていた着手点を中心とするパターンと CP で対戦を行い勝率を求める．勝率は 95% 信頼区間で区間推定する．

$Uf-7$ **他のプログラムとの対戦（*Battle*）**

フリーである GNU Go や Fuego と Nomitan を対戦をさせその勝率を見るもので，自己対戦より客観的な評価を行える．

$Uf-8$ **CGOS でのレーティング計測（*Rating*）**

CGOS（Computer Go Server）とはサーバ上でプログラム同士を対戦させ，そのレーティングを測ることのできるサイトである．客観的な評価ができるが，レーティングが安定するまで時間がかかるのと，投入時期によってレーティングのインフレ，デフレが生じやすいという問題点もある．

4.4 パターンマッチング性能に関する評価

提案手法 CP に大きく分けて 2 種類の評価を行う．1 つはパターンマッチングそのものの精度に関するもの，1 つはそれがプレイアウトの精度や囲碁プログラムの強さにどう貢献するかの評価である．本節では前者について評価を行う．

4.4.1 評価方法

係数の学習を行った際は学習性能と汎化性能を評価することができる．学習性能は学習に用いた棋譜に対する性能で，汎化性能は学習に用いていない棋譜に対する性能である．以下はすべて汎化性能について評価を行ったが，この 2 つの差を見ることで過学習 (Overfitting) の度合いを判断することもできる．

評価項目を順位項目と確率項目の 2 つに大別する．順位項目はフィルタ値によらず最適化や順位に関する項目で表 4.3 における，$L-X$ となっている項目である．確率項目

はフィルタ値によって変化する確率に関する項目で，表 4.3 では $Lf - X$ としている．順位項目は順位のみが重要である UCT 用評価関数の評価に，確率項目は選択確率の大小が重要となる MC 用評価関数の評価に用いることができる．確率項目と次節で示すプレイアウトの質，そして実際のプログラムの強さに相関があることが期待される．今回は順位項目として誤差関数値，平均順位，平均逆数順位，累積一致率分布，累積一致率係数を用い，確率項目として平均選択確率，平均ルート選択確率，選択確率 α 到達割合，α 非着手率を用いた．

4.4.2 結果

学習には 9 路には CGOS から十分強い（レーティング 2200 以上，およそアマ 3 から 5 段程度）プログラム同士の対戦棋譜を，19 路にはプロの棋譜を用いた．学習に用いた棋譜の枚数は表 4.4，また棋譜から表 3.1 の定数を用いて抽出した各パターン数は表 4.5 の通り．以下 Remi 距離の範囲を用いた従来のパターンを NX と表記する．X には距離が入る．表 4.5 を見ると 9 路において N7 よりも N9 の方がパターン数が少なくなっている．これはパターン抽出の際にハッシュ表がある程度埋まると登場回数が一定以下のものを削除する操作において削除されすぎてしまったと考えられ，表 3.1 の値を変更することで改善する可能性がある．

表 4.4 学習に用いた棋譜の枚数

	9 路	19 路
パターンの抽出	150889	42641
学習	148889	41641
テスト	2000	1000

表 4.5 抽出できたパターン数

	9 路	19 路
N5	270311	651775
N7	540323	997801
N9	508050	1189956
CP	990131	2534430

フィルタ値については事前実験より従来手法は 3，CP は 2 とした．CP の方が小さいのは CP ではパターンについて 4 回乗算されるため従来手法に比べてフィルタの影響が大きくなるためである．

また 19 路の CP については参考としてもう一つ実験を行った．これは目的関数を次のように変更したものである．

$$J_{\vec{x}}(H)' := \sum_{h=1}^{H}\sum_{j=1}^{M_h} f(p_{h,j})^2 \cdot T[g_{\vec{x}}(p_{h,j}) - g_{\vec{x}}(p_{h,0})] \tag{4.9}$$

元の目的関数に手 $p_{h,j}$ の選択確率 $f(p_{h,j})$ の 2 乗を掛けており，これによって MC 用評価関数において重要な部分を重視するようになると考えられる．これを用いた CP を CP_2

とし，一部の結果を併記する．ただし CP_2 はパターンマッチング性能に関する評価のみ行った．

以下の結果では表中の各項目で最も良いものは太文字，最も悪いものはイタリック体とした．

平均順位，平均逆数順位，誤差関数値を表 4.6 に示す．CP は全体的に良い結果であったが，$Rank_{Inv}$ のみ悪かった．CP_2 は上位の確率を重視するため平均順位や誤差関数値は悪くなった．なお表 4.6 の CP_2 の $Error$ は式 4.9 でなく式 3.8 での誤差関数値である．

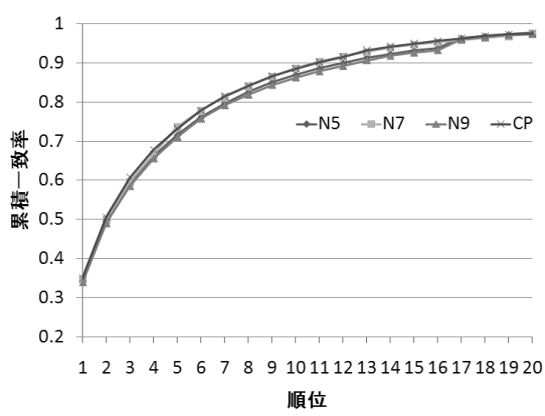
表 4.6 平均順位，平均逆数順位，誤差関数値

	9 路			19 路		
	$Rank$	$Rank_{Inv}$	$Error$	$Rank$	$Rank_{Inv}$	$Error$
N5	4.89158	1.96725	0.0743205	15.0814	2.30357	0.0583133
N7	4.70748	1.94233	0.0717104	13.953	2.24169	0.0530841
N9	*4.98541*	*1.98019*	*0.0767214*	13.6629	2.22354	0.0515967
CP	**4.59338**	**1.92491**	**0.0695717**	**12.9594**	*2.34857*	**0.0488924**
CP_2	-	-	-	*20.3145*	**2.13547**	*0.0844835*

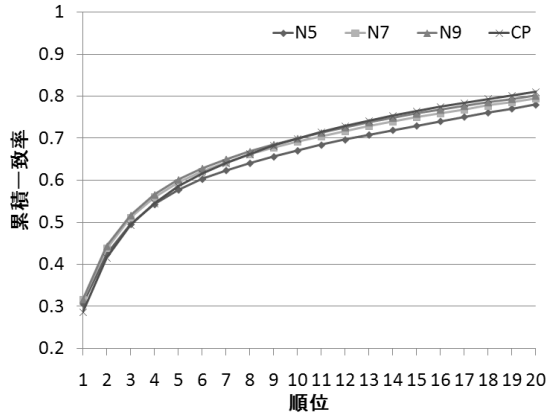
累積一致率分布の一部 を図 4.7 に示す．また表 4.7 に一部の累積一致率（$Match_n$）と累積一致率係数（Co_{Match}）を示す．表より 9 路では n 位以内率，累積一致率係数共に CP が最も良いという結果であった．19 路では 1 位以内率は悪いが他は良くなっていた．CP_2 は 1 位以内率は高くなったが 20 位以内率は下がった．これは CP_2 が教師信号が上位にくることを重視するためと考えられる．

表 4.7 累積一致率分布の値

		$Match_1$	$Match_{10}$	$Match_{20}$	Co_{Match}
9 路	N5	0.341878	0.869778	**0.976809**	0.951955
	N7	0.347401	**0.886436**	*0.973728*	0.954229
	N9	*0.340024*	*0.862754*	0.974042	*0.950797*
	CP	**0.351443**	0.885583	0.976318	**0.955637**
19 路	N5	0.306947	*0.67089*	0.780028	*0.960993*
	N7	0.315922	0.692149	0.794764	0.964119
	N9	0.317284	**0.699931**	0.801451	0.964923
	CP	*0.28638*	0.699315	**0.810263**	**0.966872**
	CP_2	**0.350134**	0.689693	*0.777006*	-



(a) 9 路



(b) 19 路

図 4.7 累積一致率分布

平均選択確率と平均ルート選択確率，5% 以上率，20% 以上率，30% 非着手率，70% 非着手率を表 4.8 に示す．表より CP は項目によっては良いものもあるが悪いものもあり，特に 9 路の 5% 以上率，19 路の 20% 以上率，両方の α 非着手率が悪かった．CP_2 は選択確率については高くなったがその分 α 非着手率などは低くなった．

表 4.8 平均選択確率と平均ルート選択確率，選択確率 α 到達割合，α 非着手率

		$Select$	$Select_{Root}$	$Over_5$	$Over_{20}$	Not_{30}	Not_{70}
9 路	N5	*0.213811*	*0.153671*	0.705398	*0.379795*	**0.607219**	0.440385
	N7	0.221583	0.159057	**0.705623**	0.388909	0.607313	0.426272
	N9	0.216527	0.15453	0.692899	0.380364	0.611088	**0.424728**
	CP	**0.258224**	**0.168948**	*0.649253*	**0.413267**	*0.664488*	*0.497237*
19 路	N5	*0.138083*	*0.0886114*	0.530874	0.25353	**0.508277**	0.395813
	N7	0.147069	0.0954932	0.547708	0.274271	0.526431	**0.379943**
	N9	0.148815	0.0973675	0.555261	0.279813	0.544173	0.406643
	CP	0.144231	0.0934535	**0.554825**	*0.25171*	0.57426	0.393792
	CP_2	**0.28566**	**0.147167**	*0.508846*	**0.376443**	*0.636694*	*0.505567*

CP のパターンマッチング性能について順位項目と確率項目に分けて評価を行ったところ，順位項目については良い結果が得られたが確率項目についてはあまり良い結果は得られなかった．順位項目が良いことから最適化についてはうまくいっていると考えられる．また CP_2 は 1 位以内率や確率が高くなったがその分誤差関数値や α 非着手率は低くなった．このことから CP も目的関数を変えることで良い結果を得ることができうるという

ことを示すことができた．また UCT 用・MC 用など，用途に応じて目的関数を変えることで同じパターンを使っても多様な特性を持たせることができることも示せた．

4.5 プレイアウト性能，強さに関する評価

ここでは CP のプレイアウト性能と実際の強さについて評価を行う．

4.5.1 評価方法

評価項目は表 4.3 の中からパターンの最大マッチングサイズ，占有率，プレイアウト速度，自己対戦を用いた．これらによりそれぞれ汎化局所化ができているか，プレイアウトは自然であるか，大きなパターンを使うことで速度がどの程度犠牲になるのか，総合的な強さはどう変化するかなどを調べる．

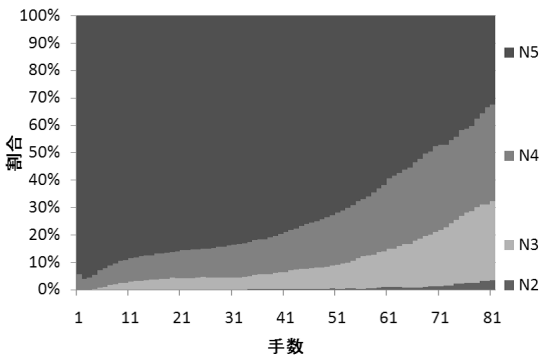
4.5.2 結果

実験に用いたマシンは Xeon 2.27GHz 8cores で対戦を行った．UCT 用評価関数のパターンは全て N7 を用いた．

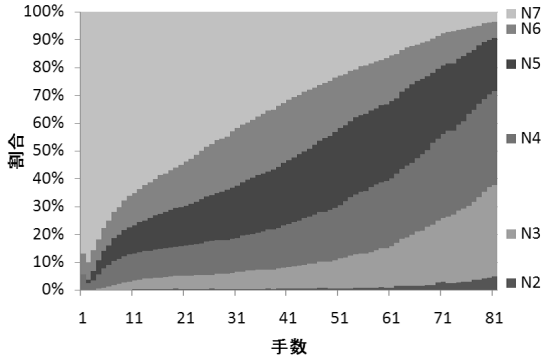
9 路と 19 路のパターンの最大マッチングサイズのグラフを図 4.8，4.9 に示す．これは 100 プレイアウトを行い手数毎に盤上の各点（空点のみ）が最大どのサイズでマッチングしたかを表すグラフである．手数は盤上がほぼ埋まる 81 手と 361 手までとした．9 路 19 路共に傾向は同じで手数が増えて行くと盤上の石も増えて行くため徐々に大きなサイズではマッチングしなくなっているのがわかる．N5 の図 4.8(a) において N5 の部分が N7 の図 4.8(b) では N5，N6，N7 に細分化されている．同様に N7 の図 4.8(b) の N7 の部分が N9 の図 4.8(c) では N7，N8，N9 に細分化されており，これは 19 路も同じである．また N9 の図より最大サイズの N9 では中盤当たりでマッチングしなくなっているのが分かる．しかし CP の図 4.8(d) では終盤になっても最大サイズが残っている．CP の最大サイズの見る点の数は表 4.2 のように 25 点でこれは Remi 距離で 5 と 6 の間程度（5 は 20 点，6 は 28 点）であるので CP の最大サイズでマッチングする割合は N5 と N7 の間程度となっている．すなわち CP では領域分割の効果により，終盤でも広い領域を見てマッチング，着手選択することができていると言える．

図 4.10 に 13 路での 4 つの局面の N9 と CP での占有率を示す．図の数値は黒にとっての占有率で，1000 回のプレイアウトを行ったときその点が黒の地になった割合（%）を示している．

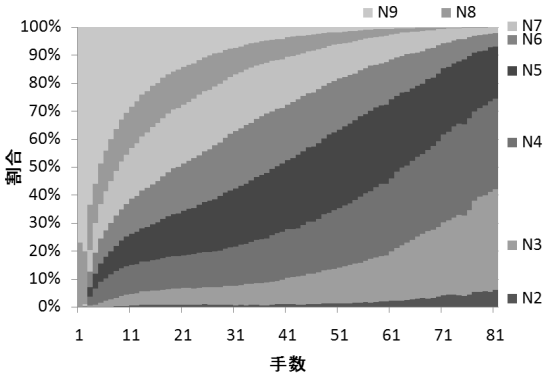
局面 1 の下辺を見るとだいたい 80% から 90% であるが，普通に打てば黒石が取られることはないがプレイアウト中では 10% から 20% は取られているということである．CP の値と比べると N9 の方が高いのでこの部分については N9 の方が精度は良いといえる．



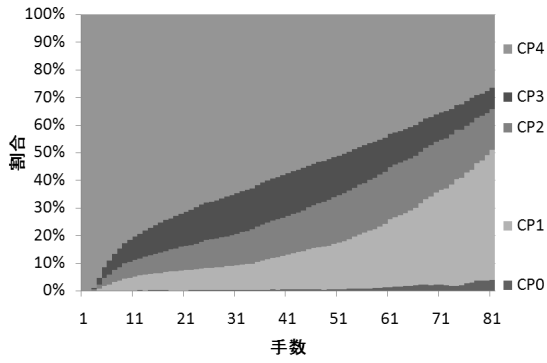
(a) N5



(b) N7

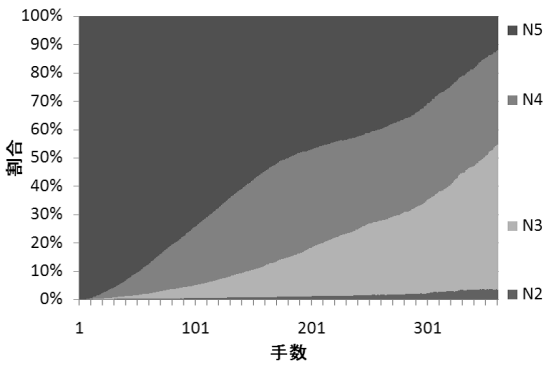


(c) N9

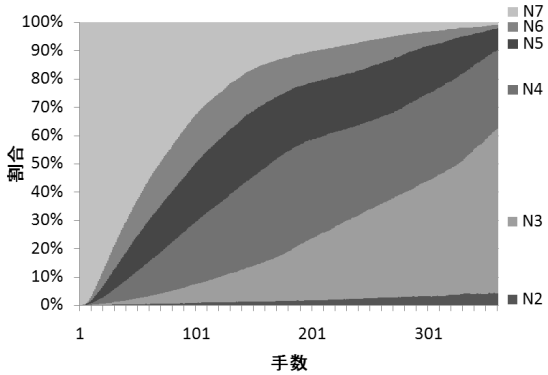


(d) CP

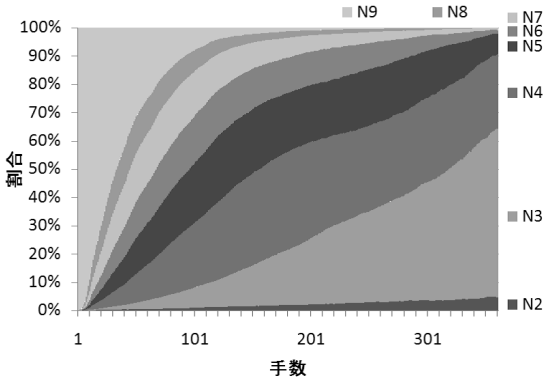
図 4.8 9 路の各パターンの最大マッチングサイズ



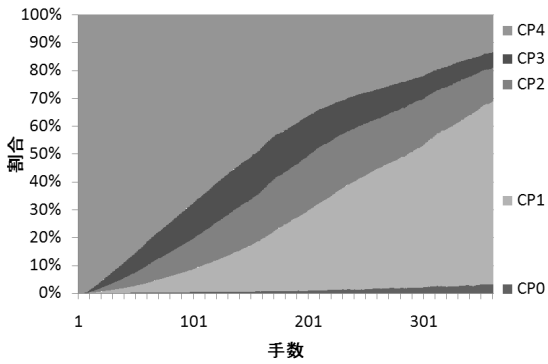
(a) N5



(b) N7



(c) N9



(d) CP

図 4.9 19 路の各パターンの最大マッチングサイズ

上辺を見ると (6, 1) の周辺の値が N9 の方が高くなっている．ここは普通に打つと黒の地にはならないので CP の方が良いといえる．

局面 2 を見ると左上や左下は黒の地になりそうであるが，N9 の方が若干値が高い．また右上の白は生きているが CP では 1% で取られてしまっているので N9 の方が良いといえる．

本実験の 2 例だけからでは両者の優劣は言えないが，より曖昧又は複雑な局面を与えることで問題点の発見ができると考えている．

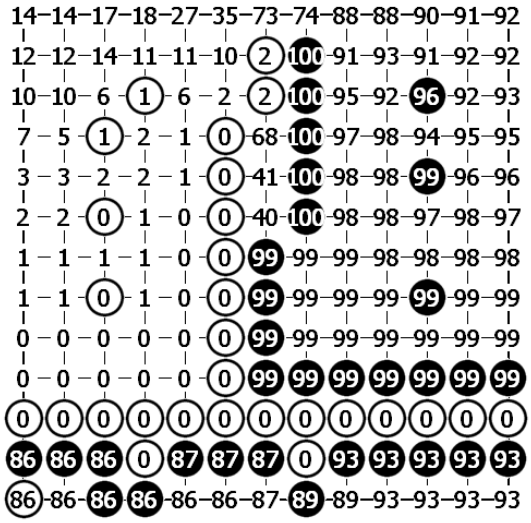
表 4.9 に 9 路でのプレイアウト速度を示す (コア数は 1)．これは m 手まで n 回のプレイアウトを 1 セットとし，これを 12 回行って最良と最悪の 2 回を除いた 10 回の平均速度から計算した 1 秒当たりのプレイアウト回数である．m と n の値は盤の大きさによって異なり，表 4.10 の通り．表 4.9 より CP は N5 の 4 割，N7 の 5 〜 6 割，N9 の 8 割程度の速度になっていた．これは CP では 4 つに分けて計算するため例えば 4 つのパターンが重なっている着手点の周囲 8 点に石が置かれると 4 つのパターン全てを計算する必要があるためからである．また現在プレイアウト速度に対するチューニングは不十分であり，改善の余地はあるためこれらを修正することで良くなる可能性がある．

表 4.9 9 路でのプレイアウト速度

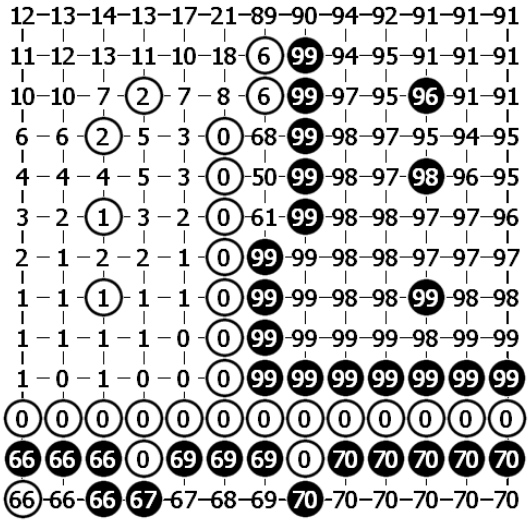
	プレイアウト速度 [po/sec]		
	9 路	13 路	19 路
N5	1297.9	545.5	238.0
N7	884.8	364.6	156.5
N9	625.6	246.5	103.6
CP	512.1	192.2	81.0

表 4.10 測定に用いた値

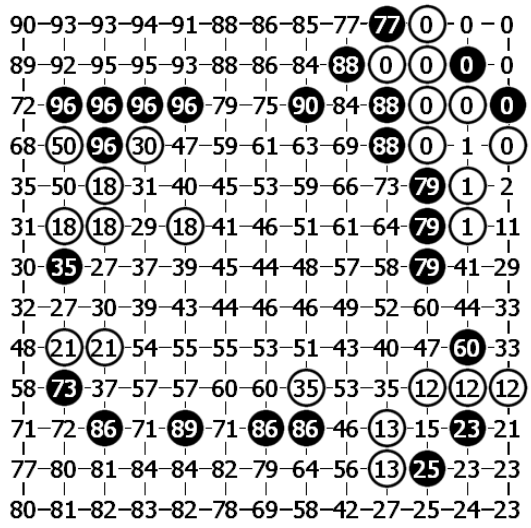
	m	n
9 路	80	10000
13 路	160	1000
19 路	300	1000



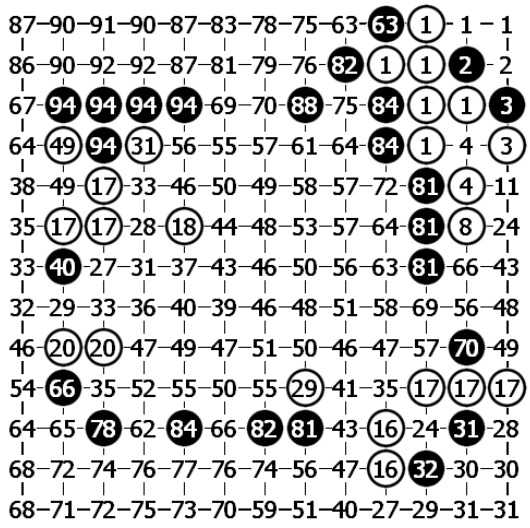
(a) 局面 1 N9



(b) 局面 1 CP



(c) 局面 2 N9



(d) 局面 2 CP

図 4.10 13 路での占有率の例

表 4.11 に自己対戦の結果を示す．表中の値は CP の勝率である．9 路は一手 5 秒で 500 対局，13 路は一手 20 秒で 200 対局行った．13 路盤用の係数を学習するための十分な棋譜の入手は困難なため，19 路盤用に学習した係数を流用するした．表より自己対戦では CP は良い結果を示さなかった．この原因としては表 4.9 に示すようにプレイアウト速度が大幅に遅くなっていたことが考えられる．また N9 に対しては他に比べて少し勝率が高くなっており特に 13 路では他に比べて有意に勝率が高い．19 路での N9 の順位項目，確率項目の結果は良いため N5，N7 と差があるとするとプレイアウト速度である．表 4.9 より N9 のプレイアウト速度は N5 の約 5 割，N7 の約 7 割となっている．このことからプレイアウト速度はプログラムの強さに大きく影響を与えると言える．

表 4.11 9 路と 13 路の自己対戦の結果

	9 路	13 路
N5	34.0 ± 4.2	22.5 ± 5.8
N7	34.8 ± 4.2	21.5 ± 5.7
N9	38.4 ± 4.3	36.5 ± 6.7

第5章　考察

結果より CP は順位項目については良い結果を得ることができており，マッチングの性能は向上しているといえる．この点では着手点を中心とする従来手法に比べて良くなっており一定の成果を得ることができた．しかし確率係数はあまり良くはなく，プレイアウト速度も遅くなってしまったことから自己対戦では従来手法より良い結果は得られなかった．占有率の結果を見ると CP は N9 よりプレイアウトの質が良くなっているとは言えず，実際に自己対戦の結果も悪かった．このことから占有率の値などを用いて定量的なプレイアウトの質の評価をすることができればプログラムの強さと相関を持たせ，よりよい評価の知見発見ができる可能性がある．またプレイアウト速度について自己対戦では N5, N7，N9 はそれぞれプレイアウト速度は異なるのに勝率はそれほど変化は無いためプレイアウトの質が良ければ多少プレイアウト速度が遅くても強くなることは有り得る．

またプレイアウト中でのパターンマッチングの最大サイズについては良かったことからプレイアウト中という普通は現れないであろう局面に対してもある程度のマッチングをしていたといえる．このため確率をうまく調整することができれば従来手法よりも自然な着手をすることができることが期待できる．

順位項目は良かったことから学習の目的関数を変えることで改善することもありうる．現在の目的関数は確率の差のシグモイドなので，事実上「教師信号の順位の平均値」に近く，この方法では確率の調整はできないため現在フィルタ値によって調整している．そのためフィルタ値によって挙動も大きく変わってしまう．そこでこの目的関数自体を変更することが考えられ，実際に新たな目的関数（式 4.9）を定義し CP_2 として評価したところ良い結果を得ることができた．

評価項目についてはそれぞれに関係性があるということは分かったので総合的に評価することができれば実際に対戦させなくても大体の強さが類推できるかもしれない．しかし評価項目はまだ十分とは言えず，評価項目の充実が必要である．

第6章　おわりに

本論文では着手点を中心とする従来のパターンにおける問題点を改善するためにパターンを 4 つに分割したコラージュパターンを提案し，その評価を行った．またそのために多くの階層的な評価項目を提案し，様々な視点から評価することでどのようなプレイアウトが良いか，確率はどうあるべきか，何が強さにとって重要かを調べた．最適化に関しては良い結果を得ることができたが，自己対戦では速度が大きく影響して良い結果は得られなかった．しかしパターンの最大マッチングサイズについては良い結果が得られたことから本質的な方向性や潜在能力は高いと考えており，実際に新たな目的関数を定義することで違った特性を持った係数を得ることができた．

今後評価項目を充実させることでプレイアウトの質をうまく評価することを可能にし，棋力との関係を調べることでプレイアウトの質とは何か，どのようなプレイアウトが良いのか，より良い目的関数はないかといった研究を進めより強い囲碁プログラムを目指す．

謝辞

研究を行なうに当たりご指導をしていただいた飯田弘之教授，池田心准教授に深謝の意を表する．またご討論ご助言をいただいた松江工業高等専門学校　情報工学科　橋本剛准教授，橋本隼一氏，野口陽来氏，松井利樹氏，他飯田・池田研究室の方々に感謝の意を表する．

参考文献

[1] C. Shannon. *Programming a Computer for Playing Chess*, Vol. 41 of 7. Bell Telephone Laboratories, Mar. 1950.

[2] L. Allis. *Searching for solutions in games and artificial intelligence*. Ph.d. thesis, Vrije Universitat, Amsterdam, 1994.

[3] H. Iida, M. Sakuta, and J. Rollason. Computer shogi. *Artificial Intelligence*, Vol. 134, pp. 121–144, 2002.

[4] C.J. Cox. Analysis and implementatioin of the game arimaa. Master 's thesis, Universiteit Masstricht, Mar. 2006.

[5] S Gelly, Y. Wang, R. Munos, and O. Teytaud. Modification of uct with patterns in monte-carlo go. In *Technical Report RR-6062*, 2006.

[6] 松井利樹. 囲碁における勾配法を用いた局面評価関数の強化学習. 修士論文, 北陸先端科学技術大学院大学, 2008.

[7] 野口陽来. モンテカルロ法を用いたコンピュータ囲碁の性能向上に関する研究. 修士論文, 北陸先端科学技術大学院大学, 2008.

[8] B. Brügmann. Monte carlo go, Oct. 1993. ftp://ftp.cgl.ucsf.edu/pub/pett/go/ladder/mcgo.ps.

[9] R. Coulom. Computing elo ratings of move patterns in the game of go. In *the Computer Games Workshop 2007*, Amsterdam, The Netherlands, 2007.

[10] 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, pp. 97–104, 1994.

[11] D. Stern, R. Herbrich, and T. Graepel. Bayesian pattern ranking for move prediction in the game of go. In *Proceedings of the International Conference of Machine Learning*, pp. 873–880, USA, Jan. 2006.

[12] A. Zobrist. A new hashing method with application for game playing. In *Tech. Rep. 88*, University of Wisconsin, Apr. 1970.

[13] 荒木伸夫, 吉田和弘, 鶴岡慶雅, 辻井潤一. 囲碁における正確な着手予測のためのファジーパターンマッチング. In *the 20th annual conference of the japanese society for artificial intelligence*, 2006.

[14] L. Kocsis and C. Szepesvari. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, Vol. 4212, pp. 282–293, Sep. 2006.

[15] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. In *Machine Learning*, Vol. 47, pp. 235–256, 2002.