修 士 論 文 の 和 文 要 旨

研究科・専攻	大学院　情報理工　学研究科　　情報・通信工学　　専攻　博士前期課程		
氏　　名	荒井　光	学籍番号	1331005
論 文 題 目	モンテカルロ木探索における予測読みに関する研究		

要　旨

本研究は，モンテカルロ木探索アルゴリズムの探索木の成長に大きく関わるパラメタを変更することにより，コンピュータ囲碁における予測読みの強化を試みた．予測読みとは，相手が探索しているときにこちらも探索を行い，その探索結果を次の自身の着手を決定する探索に使用する技術のことである．予測読みの情報を次の手番時に活かすためには実際の相手の着手を深く探索することが一番良い．しかし，これには 2 つの相反する要素を持つ．つまり，相手の着手を当てるためには手を広く読む必要があるのに対し，一方で深く探索する方が良いという 2 つの要素である．よって，予測読みは良い着手を探索することよりも，実際の相手の着手を深く探索することが要求される．このことから，手番時と同じ探索を行うのではなく，他の探索手法を行うことを検討する余地があると考えられる．

モンテカルロ木探索には，木の成長に大きく関わるパラメタが 2 つある．1 つめは UCB1 値の勝率とバイアスのバランスを決定するパラメタである．このパラメタ値が大きいとバイアスの値が大きくなり，選択回数が少ない着手が多く読まれやすくなる．よって，より広く手を探索するようになる．2 つめは Progressive Widening の閾値を決定するパラメタである．このパラメタ値を大きくすると，候補手に設定される各閾値が大きくなり，結果として探索の候補手を減らす．この 2 つのパラメタ値を変更することにより，手番時と異なった探索を実現する．

本研究では，まず村松研究室の学生により開発されている囲碁プログラム storm に基本的な予測読みを実装しどの程度棋力が向上するかオープンソース囲碁プログラム GNU Go との対局実験により検証した．そして，探索時にパラメタ値を変更することにより，生成された探索木の変化を示した．また，予測読み時にパラメタ値の変更を行う手法を storm に実装し，13 路盤では適切なパラメタ値を設定することで，GNU Go との対戦において棋力が向上すること示した．そして，コンピュータ囲碁専用の対局サーバ CGOS に提案手法を実装した storm を接続し，複数の異なるプログラムと対戦させた場合に，適切なパラメタ値を設定することで，予測読みを行わない場合と比較して，Elo レーティングが 100 以上向上することも示した．

平成 26 年度電気通信大学情報・通信工学専攻 修士論文

モンテカルロ木探索における予測読みに関する研究

主指導教員　村松 正和 教授
指導教員　岡本 吉央 准教授

平成 27 年 1 月 30 日

電気通信大学情報理工学研究科情報・通信工学専攻
情報数理工学コース

学籍番号　1331005
氏名　荒井 光

目次

図目次

表目次

第 1 章　序論

1.1　背景

近年，囲碁プログラムの棋力はアマチュア高段者レベルまで到達している．その棋力まで引き上げた技術のひとつが，Kocsis と Szepesvári により提案されたモンテカルロ木探索である [6]．この探索手法はランダムシミュレーションの結果を用いて木探索を行う特徴があり，シミュレーション中の状態を評価する必要がない．この特徴は途中の局面を評価することが困難な囲碁において有効であり，局面を評価する従来の手法に比べて良く機能することから，今日開発されている囲碁プログラムの多くはこの探索手法を採用している [3, 5].

囲碁プログラムをより強くするための研究はさまざま行われている．そのひとつに予測読みに関する研究が挙げられる．予測読みとは，相手が探索しているときにこちらも探索を行い，その探索結果を次の自身の手番時に使用する技術である．囲碁プログラム Erica は相手の思考時に自身も探索を行い，相手が選択した着手に対応する部分木を再利用することにより，GNU Go [10] を相手に 54.8% から 67.4% と勝率が大幅に向上したことが報告されている [7]．予測読みは棋力の向上に大きく貢献し得ることから，どのように予測読みが有効かを調べる研究は重要であると考えられる．

1.2　目的

本研究の目的は，村松研究室で開発されている storm に予測読みを実装し，予測読みをどのように行うことが効果的であるか調査することである．

予測読みの情報を次の手番時に活かすためには実際の相手の着手を深く探索することが一番良い．しかし，これは 2 つの相反する要素を持つ．つまり，相手の着手を「当てる」ためには広く手を読まねばならないのに，一方で深く探索した方が良い，という 2 つの要素である．したがって，予測読みでは良い着手を探索することより，実際の相手の着手を多く探索することが要求される．そこで，手番時の探索と同じ探索ではなく，他の探索手法を行うことを検討する余地があると考えられる．

本研究では探索に大きく関わる 2 つのパラメタを変更することにより，手番時の探索とは異なる探索を実現し，それが棋力にどのように関係するか調査した．変更したパラメタの 1 つは探索の指標となる UCB1 値の期待値とバイアスのバランスを決定するパラメタ

である．このパラメタの値を大きくすることにより，探索回数が少ない着手を多く読まれやすくすることができる．つまり，探索を 1 つの手に集中させるのではなく，より広く手を探索することになる．2 つめは探索の候補手の数に関わる Progressive Widening のパラメタである．このパラメタの値を大きくすることにより，探索の候補手が少なくなり，1 つの候補手に対する探索回数を増加させることができる．本研究では storm に予測読みアルゴリズムを実装し，2 つのパラメタの値を変えて対局実験を行った．そして，対局での探索情報を分析した．その結果から，パラメタの値を変えると，手番時の探索とは異なる探索が実現でき，棋力が大きく変化することがわかった．また，13 路盤では適切なパラメタ値を設定することで，手番時と同じ探索を行う標準的な予測読みを行うよりも強くなることが確認された．

1.3 本論文の構成

本論文の構成は次のとおりである．

第 2 章は，本論文で取り扱う囲碁のルールを簡単に述べる．第 3 章はコンピュータ囲碁における関連研究を記す．第 4 章では，予測読みを storm に実装することによる棋力の変化を検証した実験結果を記す．第 5 章では，パラメタの変更方法と，変更することにより生成される探索木がどのように変化するか記す．そして，予測読み時にパラメタを変更することによる棋力の変化等を調査した実験結果を記し，また考察を述べる．最後に第 6 章で，本論文のまとめと今後の課題を述べる．

第 2 章　囲碁

本章では囲碁のゲーム進行手順と着手に関するルール，勝敗の決定方法について説明する．

2.1　ゲームの進行手順

囲碁は碁盤の上に黒石と白石を交互に置いて行くことにより，ゲームが進行する．ゲーム開始時の碁盤の状態を図 1 に示す．一般的に使われる碁盤は，19 路盤と呼ばれるもので，縦に 19 本，横に 19 本の線が引かれており，361 個の交点から構成されている．黒石と白石は交点上に置かれる．また，入門者用に縦横の線を少なくした 13 路盤や 9 路盤が考案されている．13 路盤は縦横の線を 13 本，9 路盤は縦横の線を 9 本にしたものである．石が置ける場所を少なくすることにより，ゲームに要する時間の短縮などの効果がある．本稿では，9 路盤も用いてルールを説明する．ゲームの具体的な進行は図 2 に示すとおりである．また，囲碁の用語の説明は付録 A にまとめて記しているため，適宜参照願いたい．

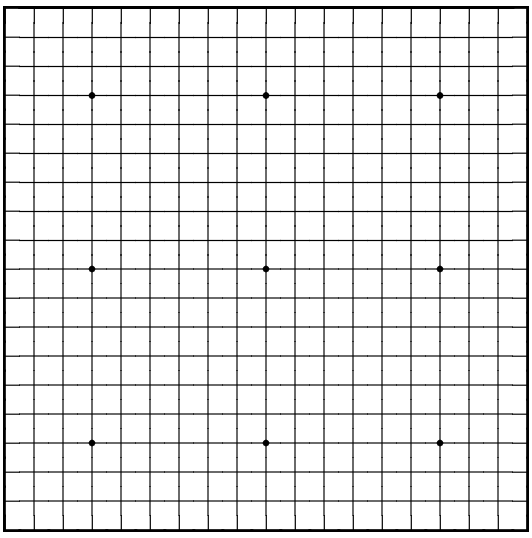

図 1 ゲーム開始時の盤面

Step 1. 手番決定 先手番と後手番を決定する．先手番になったプレイヤは黒石を持ち，後手番になったプレイヤは白石を持つ．本稿では先手番は黒番，後手番は白番と表記する．

Step 2. 着手 先手番から交互に石を交点の上に置いていく．両者が連続してパスした場合は 3 へ進む．

Step 3. 勝敗の決定 局面から勝敗を決定する．

図 2 囲碁におけるゲームの進行

2.2 着手に関するルール

2.2.1 石の打ち上げ

相手の石に隣接した交点に着手したときに，相手の石を囲った場合，その石は盤上から打ち上げられる．図 3 において，黒石を置くと白石が打ち上げられる交点には X 印が記されている．例えば黒石を左上隅に置いた場合，四角形が記されている隣接する 2 つの白石を盤上から打ち上げる．一方で Y 印が記された交点に黒石を置いても黒石が Z 印が記された交点に置かれてなく囲っていないため，隣接する白石は打ち上げられない．

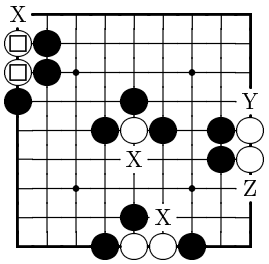

図 3 白石を打ち上げられる交点 X と打ち上げられない交点 Y の例

2.2.2 同型反復となる着手を禁止

図 4 において X 印が記された交点に黒石を置いた場合，図 5 の局面になる．次に図 5 において Y 印が記されている交点に白石を置くと図 4 の局面に戻ってしまう．この着手を許すとゲームが終わらない場合が考えられるため，この例の場合にはすぐに白石を Y

印が記された交点に置くことが禁止されている．また，次の着手時には白石を Y 印が記された交点に置くことが許されている．

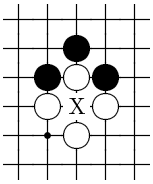

図 4 コウが発生する前の局面

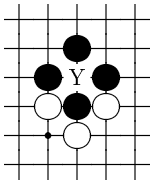

図 5 コウが発生した局面

2.2.3 自殺手となる着手の禁止

同型反復となる着手となる着手の他に，自殺手となる着手も禁止されている．ここで自殺手とは着手した直後に打ち上げれてしまうような着手のことである．例を図 6 に示す．図において X 印が記されている交点に白石を置くことは自殺手となるため禁止されている．一方で，Y 印が記されている交点には隣接する四角形が記さている黒石を打ち上げられることから自殺手とならないため，白石を置くことが許されている．

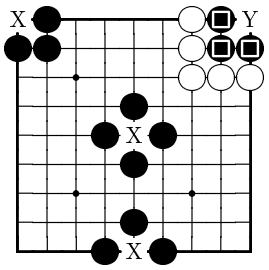

図 6 白番にとって自殺手となる交点 X と自殺手とならない交点 Y の例

2.3 勝敗の決定方法

勝敗の決定方法は大別して日本ルールによるものと中国ルールによるものが知られている．本稿では中国ルールを扱っているため，中国ルールによる勝敗の決定方法を説明する．

中国ルールは黒番の活き石の数と地の数の合計数と白番の活き石の数と地の数，コミの合計数の大きさを比較して勝敗を決定する．ここで，白番は後手番であるため不利であることから，コミと呼ばれる数値が足されている．

例として図 7 の局面を用いて活き石と地を説明する．原則として，中国ルールではゲームが終了する前に，打ち上げられる石は打ち上げる．図の局面において，活き石とはゲームが終了したときに盤面に残っている石のことである．例では黒石の活き石の数は 22，白石の活き石の数は 22 である．また，地とは一方の石で囲まれた石が置かれていない交点のことである．図では黒番の地を B，白番の地を W で記されている．例では黒番の地の数は 20，白番の地の数は 17 である．したがって，黒番の活き石の数と地の数の合計は 22+20=42，白番の活き石の数と地の数の合計は 22+17=39 である．また，コミを 7.5 とすれば，白番の合計数は 39+7.5=46.5 になる．よって黒番の合計数は 42 であり，白番の合計数は 46.5 である．よって白番の数の方が多いことから，このゲームは白番の勝ちと決定される．

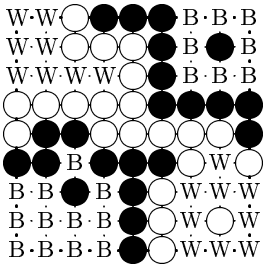

図 7 終局時の盤面

第 3 章　基礎知識と関連研究

3.1　ゲーム木

囲碁のゲームの進行はゲーム木と呼ばれる木構造で表すことができる．図 8 にゲーム木の例を示す．図において，根は現在の局面を表している．また，根節点以外の節点は親節点から遷移可能な次局面のひとつを表している．ここでアルファベットによって印付けられている節点は親節点がもつ盤面からその交点に着手した状態を表しているものとし，白色の節点は白番を黒色の節点は黒番を表しているとする．以上のように節点に情報を持たせれば，例えば現在の局面から黒番が交点 A に着手し，次に白番が交点 D に着手したゲームの進行は根節点から節点 A，節点 D と降りていくことで表現することができる．

現在の局面から生じ得る局面をすべて書き下したゲーム木よりゲームのすべての進行を把握できることから，そのゲーム木を探索することで最善の着手を決定することができる．しかし，状態数に相当するそのゲーム木の節点の個数は囲碁の場合には 10^{360} [11] とも考えられており，実時間で探索し終えることは困難である．このことから，探索できる範囲からより良い着手を選択する方法が研究されてきた．

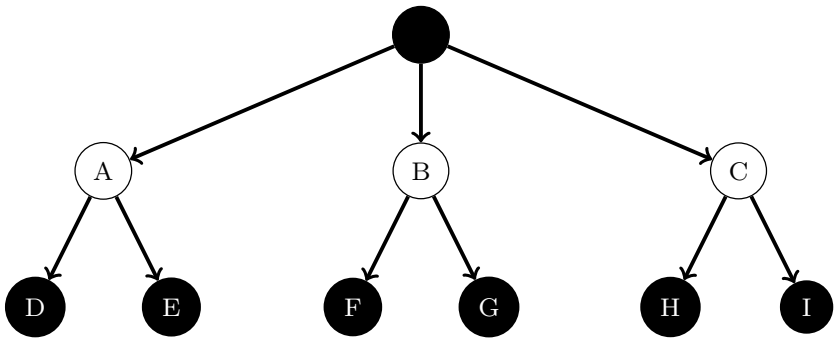

図 8 ゲーム木の例

3.2　原始モンテカルロ囲碁

乱数を用いて着手選択を行う囲碁プログラムの研究は 1993 年の Brügmann から始められていた [2]．この研究で開発されたプログラムは，現在の局面から乱数を用いてゲーム終了まで打ち切り，その結果により局面を評価し，一番良かったものを次の着手として

選択したものであった．このプログラムが作るゲーム木は図 9 で表される．図 9 において，点線はシミュレーションを表している．図からわかるように 1 手しか先を読まないため，より先を読む必要があるときに探索を行っても最適性が保証されない問題がある．そのため，このプログラムの棋力はあまり強くなく，現在のモンテカルロ囲碁と区別するために，原始モンテカルロ囲碁と呼ばれている．

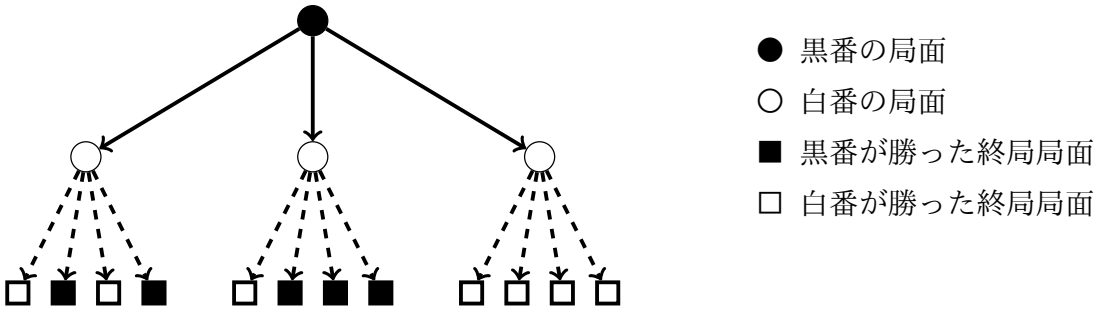

図 9 原始モンテカルロ囲碁が探索するゲーム木

3.3 Multi-Armed Bandit 問題

Multi-Armed Bandit 問題は機械学習で取り扱われてきた問題のひとつである．それぞれ異なる報酬の確率分布を持つスロットマシンが複数存在し，それぞれの確率分布が未知であるとき，得られる報酬を最大とする手順を求めることが目的である．

一定回数ずつ各スロットマシンにコインを投入し，その結果から一番良かったものに残りのコインを全て投入する戦略が考えられるが，はじめの一定回数が少ないと一番良いスロットマシンを間違う可能性が大きくなり，一方で多くした場合にはすべてのスロットマシンにそれぞれコインを大量に投入するため，無駄が生じてしまう．このように，現在までに探索して得た情報を利用し利益を最大化することにするか，または他にも探索をしてより情報を得るかのジレンマが生じる代表的な問題である．囲碁において，着手の結果の盤面がある確率で勝ちという報酬を与えるものとみなせば，どのスロットマシンを選択するかという問題を囲碁の着手選択の問題に置き換えることができる．

3.4 UCB1 戦略

Multi-Armed Bandit 問題に対して，現実的な計算量と消費メモリで動作するアルゴリズムとして，Auer ら [1] により提案された UCB1 戦略がある．この戦略は，次式により

計算される UCB1 値と呼ばれる値が最大であるスロットマシンを選択するものである．

$$\mathrm{UCB}(n) := \frac{W(n)}{N(n)} + C\sqrt{\frac{2\log_e N}{N(n)}}, \qquad (N(n) > 0). \tag{1}$$

ここで，$W(n)$ はそのスロットマシン n を選択して得た報酬であり，$N(n)$ はそのスロットマシンを選択した回数である．また，C は定数である．N はこれまでにスロットマシンを選択した総回数である．また，一度もスロットが選択されていない場合，UCB1 値は無限大と定義されている．

この式の右辺は [期待値] + [バイアス] のように言い換えることができる．[期待値] に相当する部分は投入したコインに対して利益が大きければ値が大きくなり，[バイアス] に相当する部分は全体に対して投入したコインが少ないと値が大きくなる．したがって，この値に従ってコインを投入するスロットマシンを決定すると，現在までの探索により良かったスロットマシンを選択しつつ，あまり選択していないスロットマシンを選択するような動作が期待できる．また，C は [バイアス] の重み付けをする定数とみることができる．これから，Multi-Armed Bandit 問題において生じるジレンマに上手く対応できると考えられる．また，選択回数が大きくなれば，$(\log n)/n \to 0$ $(n \to \infty)$ となることから，最善でないスロットに投入されるコインの割合は 0 に収束することが証明されている [6]．選択回数を十分に多くすれば，ほとんどのコインが最善のスロットマシンに投入されることになる．

3.5 UCT

原始モンテカルロ囲碁はゲームが終了するまでの手数が 1 より大きいと最善手を返す保証がなかった．Kocsis と Szepesvári により提案された UCT (Upper Confidence bound applied for Tree) は UCB1 戦略を用いて効率的にこの問題を解決する木探索アルゴリズムであり，モンテカルロ木探索の代表的なものである．以下にコンピュータ囲碁における UCT の動作を説明する．

UCT は手の選択，探索木の拡張，シミュレーション，探索木の更新の 4 つの段階を繰り返し行う．各段階で行われる処理は図 10 に示すとおりである．図 10 の Step 1 から Step 4 の段階を 1 回行うことを 1 プレイアウトと呼び，プレイアウトを一定回数行い，最後に根節点の子節点の中で一番多く選択された小節点に対応する着手を次の一手として出力する．

UCT はプレイアウトを多くすればするほど，探索木が大きくなり，プレイアウトを十

Step 1. 手の選択 可能な着手を探索木の根節点から UCB1 値に従って選択し，木を降りていく．

Step 2. 探索木の拡張 葉節点から新たに子節点を作成する．

Step 3. シミュレーション 終局まで乱数を用いてシミュレーションを行う．

Step 4. 探索木の更新 終局した盤面から結果を取得し，降りてきた節点の情報を更新する．

図 10 UCT の 1 プレイアウトで行われる処理

分多くすれば，すべての状態を保有するゲーム木を作成することができる．このことから，原始モンテカルロ囲碁のアルゴリズムにはなかった最適性を UCT は持つことが知られている [6]．また，手の選択の段階で UCB1 戦略を行うことから，無駄な手の選択をあまり行わず探索することができ，実時間でも十分良い選択を行うことができるように設計されている．

3.6 RAVE

UCT では探索木の更新をする際に降りてきた節点に対して情報の更新を行う．図 10 からわかるように，囲碁のように巨大なゲーム木に対応する場合にはプレイアウトを大量に行う必要がある．この問題を解決する手法として考案されたのが RAVE (Rapid Action Value Estimation) である [4]．この手法は，囲碁において，手順ではなくある交点を占有することが重要となる場合が多いことを利用したものである．そのような局面の例を図 11 に示す．この局面において次の手番が白番であるときに，交点 X に着手すれば有利にゲームを進めることができる．一方で黒番に着手されると不利になると考えられる．UCT により探索を行った場合に，手の選択の段階においてはじめに交点 X に着手しなかった場合でも，次以降の手番時で選択していれば，シミュレーションにより得る結果は勝っている場合が多いと考えられる．そこで，RAVE では手の選択の段階からシミュレーションの段階までどこに着手したか記録しておき，探索木の更新の際に途中で打たれた着手でもはじめに着手したとして節点の情報を更新する．この処理により図 11 における交点 X への着手が良いことが通常よりも早く特定でき，交点 X への着手からはじまるプレイアウトが行われやすくなる．これより，最終的に選択される着手も X が選ばれやすくなる．

実際には RAVE 値と呼ばれる値を導入して，説明した処理を実現している．着手 n の RAVE 値は次のように定義される値である．

$$\mathrm{RAVE}(n) := \frac{W_{RAVE}}{n_{RAVE}} + C_{RAVE}\sqrt{\frac{\log N_{RAVE}}{n_{RAVE}}}. \tag{2}$$

ここで，W_{RAVE} はこの節点に対応する着手で勝った RAVE 回数，n_{RAVE} はこの節点に対応する着手を選択した RAVE 回数，C_{RAVE} は割合を決める定数，N_{RAVE} は子局面の RAVE の回数の総和である．この RAVE 値を UCB1 値に導入し，UCB1 値に代わる指標として着手 n の UCB-RAVE 値を以下のように定義し，これを用いて着手選択を行う．

$$\beta(n) := \sqrt{\frac{k}{3N(n)+k}}, \tag{3}$$

$$\text{UCB-RAVE(n)} := \beta(n)\mathrm{RAVE}(n) + (1-\beta(n))\mathrm{UCB}(n). \tag{4}$$

ここで，k は UCB1 値と RAVE 値を $N(n)$ に対してどの程度重みを置くかを設定する定数である．$N(n)$ と k が等しいときに $\beta(n)$ は 0.5 を取るようになる．この指標を使うことで，初めは $\beta(n)$ が大きいことから，RAVE 値に重きを置いた探索を行い，$N(n)$ が k より多くなったときには UCB1 値に重きを置いた探索を行うことが期待される．これにより，はじめに説明した動作を実現することができる．

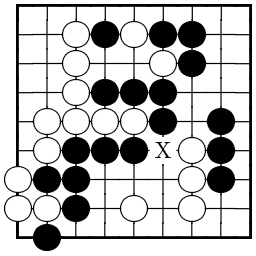

図 11 交点 X への着手が重要である局面

3.7 シミュレーションの改良

シミュレーションの質を向上させることで，棋力の向上を試みる研究も行われている．ここで言う質を向上させるとは，棋力が強い人が着手したようにシミュレーションを実現させることを指す．プロ棋士に近いレベルまで向上できた場合には，各候補手に対して数回のプレイアウトを行うだけで最善手に近い着手を選択し得ることから，重要な研究であ

ると考えられる．これに関する研究で一番良く知られているのが Bradley-Terry モデルを用いたものである [8]．ここではそれに関して簡単に説明する．

3.7.1 Bradley-Terry モデル

Bradley-Terry モデルは 1 人または複数人からなるチームの強さを推定することができるモデルである．数理モデルに基いて，実際には対戦がないチーム同士でも勝率を予測することができる．プレイヤ i の強さを γ_i とし，プレイヤ j の強さを γ_j としたときに，このモデルにおいてプレイヤ i がプレイヤ j と対戦して勝つ確率は次式で与えられる：

$$[i \text{ が } j \text{ と対戦して勝つ確率}] = \frac{\gamma_i}{\gamma_i + \gamma_j}. \tag{5}$$

また，i に対して j の他に強さが γ_k のプレイヤ k が対戦に加わった場合にも，次のように勝率を表す：

$$[i \text{ が } j \text{ と } k \text{ に対戦して勝つ確率}] = \frac{\gamma_i}{\gamma_i + \gamma_j + \gamma_k}. \tag{6}$$

4 人以上からなる対戦も同様に表すことができる．また，このモデルでは対戦者が複数人からなるチームであっても適用することができる．例えば i と j からなるチームが j と k からなるチームと i と k からなるチームと対戦したときに勝つ確率は次のように表す：

$$[i \text{ と } j \text{ からなるチームが勝つ確率}] = \frac{\gamma_i\gamma_j}{\gamma_i\gamma_j + \gamma_j\gamma_k + \gamma_i\gamma_k}. \tag{7}$$

式からわかるように，チームの強さはチームを構成するプレイヤの強さの積により表される．

コンピュータ囲碁でこのモデルを扱う場合，プレイヤを「着手を特徴付ける要素」に置き換える．具体的には，着手した交点の周辺の石の配置や直前に打たれた着手からの距離や相手の石を打ち上げられるかなどが特徴付ける要素として使用されている．このようにすると，その着手の強さはその着手にある各特徴の強さの積により表すことができ，候補手において勝つ確率を計算することができる．本稿では，ある着手に対してその着手を特徴付ける要素からなるチームの勝つ確率をレート値と記し，レート値を計算することをレーティングと記す．

3.7.2 Minorization-Maximization 法による γ 値の算出

Bradley-Terry モデルにおけるプレイヤの強さ γ は対戦記録があれば，Minorization-Maximization 法により求めることができる．具体的な手順を以下で説明する．

ここで，n 回の対戦 $R_1, R_2, \ldots, R_n$ があったとする．このとき，あるチームが対戦 R_j で勝つ確率は次のように要素に分解して表現することができる：

$$[\text{あるチームが勝つ確率}] = \frac{A_{ij}\gamma_i + B_{ij}}{C_{ij}\gamma_i + D_{ij}}. \tag{8}$$

ここで，A_{ij}や B_{ij}, C_{ij}, D_{ij} は γ_i を含まない要素である．例えば，$\gamma_j\gamma_k$ は γ_i が現れてないので，γ_i を含まない要素といえる．具体的な例を当てはめると次のようになる．i と j, k からなるチームが m と n からなるチームと i と n からなるチームと対戦して勝つ確率は次のように表せる：

$$[i \text{ と } j, k \text{ からなるチームが勝つ確率}] = \frac{\gamma_j\gamma_k\gamma_i + 0}{(\gamma_j\gamma_k + \gamma_n)\gamma_i + \gamma_m\gamma_n}. \tag{9}$$

この式から，$A_{ij} = \gamma_j\gamma_k$，$B_{ij} = 0$，$C_{ij} = \gamma_j\gamma_k + \gamma_n$，$D_{ij} = \gamma_m\gamma_n$ と対応していることがわかる．ここで $E_j = C_{ij}\gamma_i + D_{ij}$ と置き，γ_i が対戦で勝った回数を W_i としたとき，Minorization-Maximization 法により γ_i の更新値は次式により与えられる：

$$\gamma_i = \frac{W_i}{\sum_{j=1}^{n} \frac{C_{ij}}{E_j}}. \tag{10}$$

この式を用いて繰り返し更新することで各プレイヤの強さを求めることができる．また，各プレイヤの強さの初期値として普通は 1 が用いられる．

3.8 Progressive Widening

囲碁は合法手が多く，すべての着手に対して探索を行っていると重要でない着手に対しても一定の計算資源を割り当てることになり非効率である．そこで，重要であると考えられるものから順番に探索するようにする仕組みのひとつとして考案されたのが Progressive Widening [8] である．

まず，候補手に対してレーティングを行い，1 番目にレート値が最大の着手が来るように降順に並び替えを行う．そして，はじめはレート値が最大の着手のみを候補手として探索し，選択回数が表 1 に表される値になった場合には，対応するレート順位であった着手を候補手に取り入れる．例えば，レート順位が 2 番目の着手は選択回数が 41 回目から候補手に取り入れられる．この閾値の値は次式から導かれる値で，本研究で使用する囲碁プログラム storm は K=1.4 を使用している．

$$\left[\text{上位 1 位の閾値}\right] := 0, \tag{11}$$

$$\left[\text{上位 } n+1 \text{ 位の閾値}\right] := [\text{上位 } n \text{ 位の閾値}] + \lfloor 40K^n \rfloor, \quad (n > 0). \tag{12}$$

上の式を用いることで，候補手の数は探索回数に対して対数関数的に増加する．また，候補手が増える数は多くても 1 である．このことから，この手法を用いることにより，重要そうな着手から徐々に候補手を増やし探索することができ，かつ候補手が増えたとしても，その候補手がそれまで探索された候補手よりも良いか評価する探索を十分に行うことができる．

表 1 候補手に取り入れるときの選択回数の閾値とレート順位の関係

レート順位	2	3	4	5	6	...
閾値	40	96	174	283	436	...

3.9 同心円配置アルゴリズム

本研究では探索木の様子を確認するために，同心円配置アルゴリズム [9] と呼ばれる方法で探索木の節点を配置し，描画している．ここでは，同心円配置アルゴリズムを簡単に説明する．

同心円配置アルゴリズムは多くの節点からなる木構造を見やすく描画するための手法である．例えば，図 12 に示している木構造に対して，このアルゴリズムを用いると，図 13 のように節点が配置される．なお，図 13 には同心円配置アルゴリズムの説明のための補助線が記されている．木構造を図 12 のように描画する場合，節点数が多くなると横幅を大きくとる必要がある．また，どの節点が子節点を多く持つかなどがすぐにはわかりづらい欠点がある．図 13 を見てもわかるように，同心円配置アルゴリズムを使用すると，そのような欠点がなく，本研究で描画の対象となる探索木を描画するのに適していると考えられる．

次に図 13 を例に描画方法を説明する．同心円配置アルゴリズムは各節点に領域角と呼ばれる角度を割り当てることにより節点の配置を行っている．はじめに根節点の領域角を 360 度と割り当て，円の中心に配置する．次に各根節点の子節点が持つ子節点の数を数え上げる．ここで，子節点が葉節点であった場合には子節点の数を自分自身の 1 と数える．図 13 の例では，節点 C の子節点の数は 3 であり，節点 B と D，E は 1 として数えられる．したがって合計した数は 6 であることがわかる．そして，数え上げた数に応じて領域角の割り当てを行う．より具体的には，親節点が持つ領域角を合計した数で割り，その数に自身が持つ子節点の数をかけた値を自身の領域角として割り当てる．図 13 の例では，親節点である根節点の領域角は 360 度であり，合計した子節点数は 6 であることか

ら，割った値は 60 度である．よって，節点 C は子節点を 3 つ持つことから，節点 C の領域角は 3 倍した 180 度が割り当てられる．同様に節点 B と D，E は 60 度が割り当てられる．次に節点の配置を行う．節点は割り当てられた領域角の半分の傾きの線を根節点から引き，深さに応じた同心円の円周上と交わる点に配置される．図 13 では，節点 C は領域角の半分である 90 度の位置に配置されていることがわかる．以上の処理を 1 ステップとし，すべての節点に領域角を割り当てて配置し終えるまで行う．

以上のアルゴリズムの動作説明から，子節点を多く持つ節点の領域角は大きいことが予想される．したがって，図を見たときに領域角が大きい節点は子節点を多く持つと予想することができる．また，根節点からの距離は木の深さに相当しており，根節点から等距離にある節点は同じ深さを持つ．これから，節点の最大の深さはすぐに判断できる．本論文では，多くの節点からなる探索木を描画する場合にこの手法を用いている．

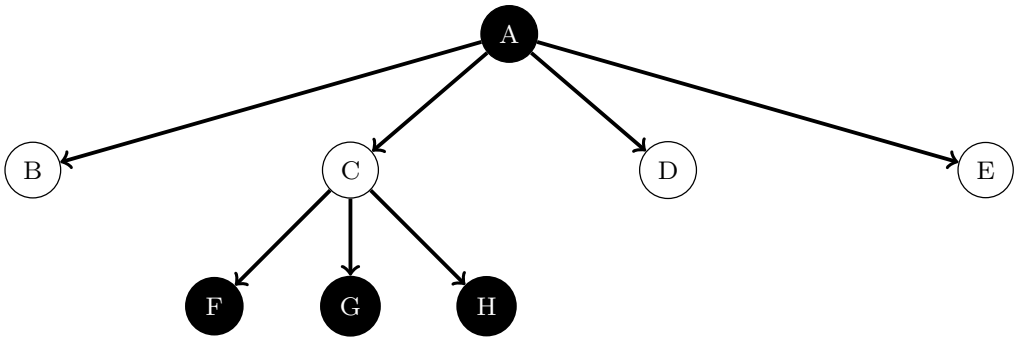

図 12 木構造の例

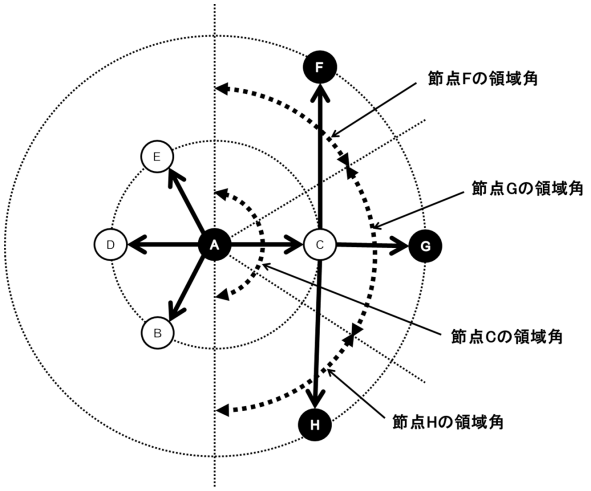

図 13 図 12 の木構造を同心円配置アルゴリズムにより配置した例

第 4 章　予測読みに関する検証実験

本章では，村松研究室で開発されている囲碁プログラム storm に予測読みを実装し，どの程度強くなるかを検証した対局実験の結果を示す．

4.1　予測読みアルゴリズムの実装

storm は手番がくるたびに UCT を一から行う処理が実装されていた．そこで，図 14 に示すアルゴリズムに変更することにより，予測読み処理を実現した．図 14 のアルゴリズムを図を用いて説明する．

まず，アルゴリズムの処理内容で使用されている用語について説明する．予測読みの説明をわかりやすくするために，2 つの用語を導入している．1 つめは，予測探索木である．これは，予測読み時に生成される探索木のことである．2 つめは，手番探索木である．これは，手番時に生成される探索木のことである．次に，各段階の処理内容を説明する．

Step 1 での処理の例を図 15 と図 16 に示す．図 15 は，予測探索木である．ここで，相手が交点 C への着手を選択したとする．この場合には，その着手に対応する節点 C があるので，図 16 のように節点 C を根節点として探索を行う．また，相手が交点 B や C，D 以外の着手を行った場合には，探索木に対応する節点がないため，新たに根節点を作成する．

Step 2 での処理の例を図 17 に示す．図 17 は，図 16 を用いて探索が行われた後の手番探索木を示している．ここで，交点 H への着手が最も多く探索されたとすれば，交点 H を次の一手として選択する．

Step 3 での処理の例を図 18 に示す．図 18 は，図 17 で交点 H を選択した場合において，予測探索木の初期状態を示している．Step 1 から Step 4 の処理を繰り返し行うことで，相手番のときに探索を行う予測読みができ，相手が選択した節点が予測読みの段階で探索されていた場合には探索して得た情報を次の探索時に繰り越して使用することができる．

4.2 実験

4.2.1 実験 1：自己対局

予測読みがどの程度有効か調査するため，標準的な予測読みアルゴリズムを実装した storm (Version 1.14.2) と実装していない従来の storm (Version 1.14.1) の対戦を行った．また，これ以降では標準的な予測読みアルゴリズムを実装した storm を storm-ponder と表記する．

4.2.2 実験環境と条件

9 路盤と 13 路盤，19 路盤での実験環境は下記のとおりである．

Algorithm 1 : 標準的な予測読みアルゴリズム

Step 1 予測探索木において，深さ 1 の節点に相手が実際に選択した節点がある場合には，その節点を新たに根節点として手番探索木を作成し，探索を行う．選択した節点がない場合には，従来どおりに新たに根節点のみからなる手番探索木を作成し，探索を行う．

Step 2 探索し終えたら根節点に対する子節点のなかで一番多く選択された子節点に対応する着手を次の一手として選択する．

Step 3 選択した節点を新たに根節点として予測探索木を作成し，相手番から予測読みに相当する探索を行う．

Step 4 相手が着手した後に予測読みに相当する探索を終了し，Step 1 へ進む．

図 14 標準的な予測読みアルゴリズム

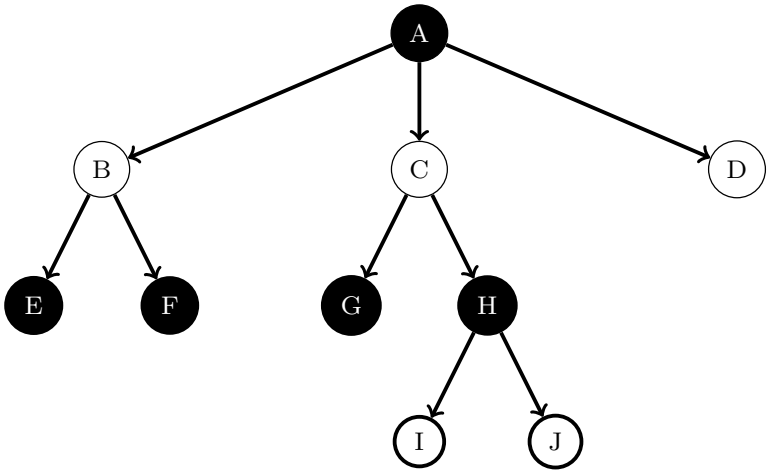

図 15 予測読みで生成された予測探索木

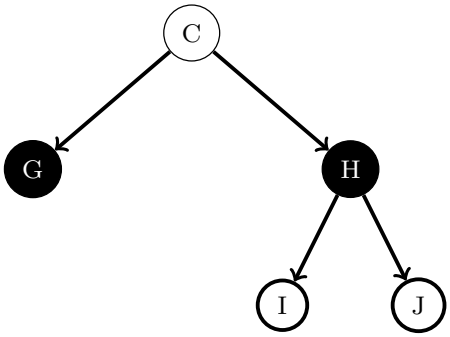

図 16 図 15 において相手が C に着手した場合の手番探索木の初期状態

・**OS** Linux Mint 16 Cinnamon 64-bit.
・**CPU** Intel© Core^{TM} i7-2630QM CPU @ 2.00GHz x 4.
・**メモリ** 8GB.

9 路盤と 13 路盤での実験条件は下記のとおりである.

・ 中国ルール，コミは 7 目半で 500 局（黒番 250 局，白番 250 局）.
・ storm と storm-ponder の手番時のプレイアウト数は，3,000，6,000 と固定．使用するスレッド数は 4，式 (1) における C は 1.0，式 (12) における K は 1.4 と設定.

19 路盤での実験条件は下記のとおりである.

・ 中国ルール，コミは 7 目半で 500 局（黒番 250 局，白番 250 局）.

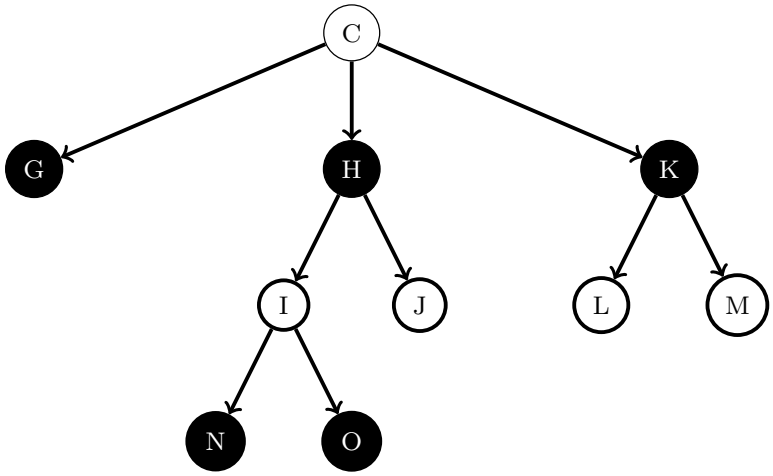

図 17 図 16 を用いて探索が行われて生成された手番探索木

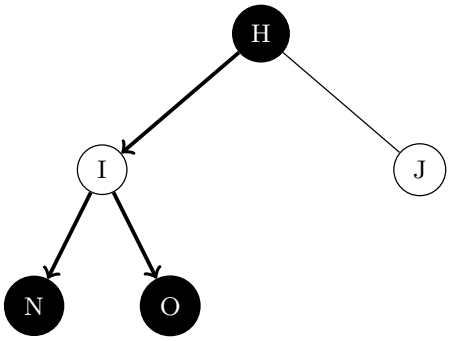

図 18 図 17 において H を着手選択した後の予測読みで使用する予測探索木の初期状態

・ storm と storm-ponder の手番時のプレイアウト数は，3,000 と固定．使用するスレッド数は 4，式 (1) における C は 1.0，式 (12) における K は 1.4 と設定．

4.2.3 結果と考察

9 路盤と 13 路盤，19 路盤での実験結果を表 2 に示す．いずれの実験条件においても，予測読みを実装したプログラムが勝ち越していることがわかる．また，プレイアウト数が増加しても勝率の変化はあまりないこともわかる．また，盤面の大きさが異なっても，同程度勝ち越していることもわかる．

表 3 に，各手番時に生成した手番探索木の節点数の平均と根節点の子節点数の平均をそれぞれ示す．表 3 において，探索終了時の根節点の子節点の平均は探索終了時の候補手

の平均数を表している．この表から，9 路盤と 13 路盤においては，予測読みをした場合はしなかった場合と比べて生成した探索木の節点数はいずれも 1.5 倍程度多いことがわかる．また，候補手の平均数の差は 1 以上ある．このことから，予測読みをした場合はより多くの候補手に対して，多くのプレイアウトによる探索を実現でき，勝ち越したと考えられる．19 路盤においては，候補手の平均数は変化しなかったが，生成した節点数が増加していることがわかる．これが予測読みをしたプログラムが勝ち越した要因であると考えられる．

どの程度予測読みした探索結果を利用できていたかの情報を表 4 に示す．表において，繰り越したプレイアウト数の割合とは，予測読み時に行ったプレイアウト数に対して，実際の相手の着手から始まるプレイアウトが行われた回数の割合である．表より，13 路盤に比べて 9 路盤の方が繰り越したプレイアウト数の割合は大きいことがわかる．これは，9 路盤の方が候補手を探索により絞りやすく，相手の実際の着手に対してより探索できたからだと考えられる．19 路盤に比べて 13 路盤の方が繰り越したプレイアウト数の割合が大きいことも同様の理由だと考えられる．

表 2 storm との対戦結果

盤面の大きさ	手番時のプレイアウト数	storm-ponder の勝率
9 路盤	3,000	66.0%
9 路盤	6,000	65.4%
13 路盤	3,000	65.6%
13 路盤	6,000	65.6%
19 路盤	3,000	64.0%

4.2.4 実験 2：GNU Go との対局

予測読みにより別のプログラムに対してどの程度有効か調査するため，storm-ponder と GNU Go 3.8 [10]，storm と GNU Go 3.8 の対戦を行った．GNU Go はオープンソースの囲碁プログラムである．

4.2.5 実験環境と条件

9 路盤での実験環境は下記のとおりである．

・**OS** Ubuntu 12.04 32bit.

表 3 storm との対戦時に探索された探索木の節点数の平均と根節点の子節点数の平均

盤面の大きさ	手番時のプレイアウト数	予測読みの有無	生成した節点数の平均	探索終了時の根節点の子節点の平均
9 路盤	3,000	なし	1,696	11
9 路盤	3,000	あり	2,786	13
9 路盤	6,000	なし	3,391	13
9 路盤	6,000	あり	6,010	14
13 路盤	3,000	なし	1,570	11
13 路盤	3,000	あり	2,143	12
13 路盤	6,000	なし	3,057	13
13 路盤	6,000	あり	4,450	14
19 路盤	3,000	なし	1,431	11
19 路盤	3,000	あり	1,713	11

表 4 各条件において予測読みにより繰り越したプレイアウト数の割合

盤面の大きさ	手番時のプレイアウト数	繰り越したプレイアウト数の割合
9 路盤	3,000	39.2%
9 路盤	6,000	41.9%
13 路盤	3,000	28.3%
13 路盤	6,000	31.3%
19 路盤	3,000	20.3%

・**CPU** Intel® Core™ i7-7620M CPU @ 2.70GHz x 4.
・**メモリ** 4GB.

9 路盤での実験条件は下記のとおりである.

- 中国ルール，コミは 7 目半で 500 局（黒番 250 局，白番 250 局）.
- storm と storm-ponder の手番時のプレイアウト数は 500 と固定．使用するスレッド数は 3，式 (1) における C は 1.0，式 (12) における K は 1.4 と設定.
- GNU Go のレベルはデフォルトの 10 と設定.

13 路盤での実験環境は下記のとおりである．

・**OS** Ubuntu 12.04 64bit.
・**CPU** Intel® Xeon(R) CPU E5430 @ 2.66GHz x 8.
・メモリ 8GB.

13 路盤での実験条件は下記のとおりである．

・ 中国ルール，コミは 7 目半で 500 局（黒番 250 局，白番 250 局）．
・ storm と storm-ponder の手番時のプレイアウト数は 3000 と固定．使用するスレッド数は 6，式 (1) における C は 1.0，式 (12) における K は 1.4 と設定．
・ GNU Go のレベルは 10 と設定．

19 路盤での実験環境は下記のとおりである．

・**OS** Ubuntu 14.04 64bit.
・**CPU** Intel® Xeon(R) CPU E5430 @ 2.66GHz x 8.
・メモリ 8GB.

19 路盤での実験条件は下記のとおりである．9 路盤や 13 路盤と比較して 19 路盤ではプレイアウトを行う計算時間が増加する．十分な数のプレイアウトを storm-ponder に行わさせるために，GNU Go の着手は 2 秒経過するまで着手を返さないように設定している．

・ 中国ルール，コミは 7 目半で 500 局（黒番 250 局，白番 250 局）．
・ storm と storm-ponder の手番時のプレイアウト数は 7,000 と固定．使用するスレッド数は 6，式 (1) における C は 1.0，式 (12) における K は 1.4 と設定．
・ GNU Go のレベルは 0 と設定．
・ storm-ponder の対戦時には GNU Go の着手は 2 秒経過するまで着手を返さないように設定．

4.2.6 結果と考察

実験結果を表 5 に示す．表からわかるように，いずれの盤の大きさにおいても storm より storm-ponder の勝率が高いことがわかる．また，9 路盤と 13 路盤，19 路盤でのそれぞれの storm との勝率の差の検定による求まる p 値は 5.2×10^{-6} と 0.019，0.013 であった．このことから予測読みを実装することにより有意水準 5% で有意に強くなったことがわかる．

実際に相手が選択した節点が予測読みの段階で平均的にどの程度プレイアウトが行われていたかを表 6 に示す．表において，平均プレイアウト回数とは，予測読み時に相手の着手から始まるプレイアウトを行った回数のことである．表からわかるように，相手が選択した手を予測読み時に平均して 1,000 回程度プレイアウトを行っていたことがわかる．これが storm-ponder の勝率が storm よりも高かった要因であると考えられる．

表 5 GNU Go との対戦結果

盤面の大きさ	対戦プログラム	対戦プログラムの勝率
9 路盤	storm-ponder	56.2%
9 路盤	storm	41.8%
13 路盤	storm-ponder	55.6%
13 路盤	storm	48.2%
19 路盤	storm-ponder	49.6%
19 路盤	storm	41.8%

表 6 予測読み時に相手の着手から始まるプレイアウトを行った回数

盤の大きさ	平均プレイアウト回数
9 路盤	1,327
13 路盤	1,063
19 路盤	957

4.3 まとめ

実験 1 の自己対局では，9 路盤や 13 路盤，19 路盤において，予測読みを行った storm-ponder の勝率が 60% 以上勝ち越していることが確認された．この結果から，予測読みにより storm の棋力は向上したと考えられる．また，繰り越したプレイアウト数は盤が大きくなるほど小さくなることも確認された．これは版が大きくなるほど予測読みにより相手の手を読むことが困難になることを意味していると考えられる．

実験 2 の GNU Go との対局では，9 路盤や 13 路盤，19 路盤において，storm よりも storm-ponder の方が勝率が 7% 以上高くなることが確認された．この結果から，自己対

戦以外でも強くなることがわかった．また，予測読み時の相手の着手から始まるプレイアウトを行った回数を計測したところ，どの盤の大きさにおいても十分にプレイアウトを行えていたことが確認できた．このことが，勝率が向上した要因であると考えられる．

第 5 章　パラメタ調整による予測読みの強化

5.1　概要

第 4 章では，手番時と同じ探索を行う予測読みを実装したことで，棋力が向上することを確認した．本章では，新たに予測読みの強化方法を提案し，予測読みによりさらなる棋力の向上を試みる．

予測読みで生成した探索木において得られた情報を次の手番に使用できるのは，実際に相手が選択した着手に対応する節点以下の部分木のみである．したがって，相手が実際に選択した着手以外から行われた探索の情報は使用されない．このことから，予測読みでの探索は手番時のようにより良い着手を探索することよりも，実際の相手の着手をより多く読むことが要求される．そこで，モンテカルロ木探索の木の生成に大きく関わるパラメタを変更することにより，予測読み時により良い探索を実現する手法を提案する．提案する手法において変更するパラメタ数は 2 つである．

1 つめは，式 (1) で表される UCB1 値にある C である．このパラメタは勝率の期待値に対してバイアスを重み付けるものである．パラメタ値が大きくなるとバイアスの値が大きくなり，結果として，広く探索するようになる．

2 つめは，式 (12) で表される Progressive Widening における閾値に関係する K である．Progressive Widening は徐々に候補手を増やしていく手法である．K の値を大きくすると候補手を増やす閾値が大きくなる．結果として，候補手の数が少なくなり，重点的な探索が行われやすくなる．

上記の 2 つのパラメタ値を変えることにより，生成される探索木の傾向を変えることができる．そこで，提案手法では予測読み時は手番時と同じパラメタ値を使用するのではなくより適切な値を設定することにより，予測読みで得た情報において使用できる量を増やし，棋力の向上を試みる．

5.2　パラメタ調整を行う予測読みアルゴリズム

提案したパラメタ調整を行う予測読みのアルゴリズムを図 19 に示す．このアルゴリズムは図 14 に示す標準的な予測アルゴリズムに Step 4 のパラメタ値を変更する処理を加えたものである．この処理を追加することで，予測読みを行うときは手番時の探索とは異なったパラメタ値を用いた探索を実現することができる．また，変更したパラメタ値は

Algorithm 2 : パラメタ調整を行う予測読みアルゴリズム

Step 1 手番時で使用するパラメタ C と K とは別に，予測読み時に使用するパラメタ C_{ponder} と K_{ponder} の値を設定する．

Step 2 予測探索木において，深さ 1 の節点に相手が実際に選択した節点がある場合には，その節点を新たに根節点として手番探索木を作成し，探索を行う．選択した節点がない場合には，従来どおりに新たに根節点のみからなる手番探索木を作成し，探索を行う．

Step 3 探索し終えたら根節点に対する子節点のなかで一番多く選択された子節点に対応する着手を次の一手として選択する．

Step 4 パラメタ C に C_{ponder}，K に K_{ponder} のそれぞれの値を設定する．

Step 5 選択した節点を新たに根節点として予測探索木を作成し，相手番から予測読みに相当する探索を行う．

Step 6 相手が着手した後に予測読みに相当する探索を終了する．そして，パラメタ C と K の値を元に戻し，Step 2 へ進む．

図 19 提案手法の予測読みアルゴリズム

Step 6 で元に戻している．

5.3 実際の探索木の変化

式 (1) のパラメタ C と式 (12) のパラメタ K を変更することにより，探索により生成される探索木がどのように変化するか実例を用いて説明する．図 20 にある局面でデフォルト値である $C = 1$，$K = 1.4$ としたときに storm が 10,000 回のプレイアウトを行うことにより生成する探索木は例として図 21 のようなものである．この図は探索木を同心円配置アルゴリズム [9] により配置した図である．このアルゴリズムの利点は多くの節点を含む探索木の全体を把握しやすくなることである．また，UCT によって生成された探索木の節点を同心円配置アルゴリズムにより配置した場合，領域角が大きい節点では多くのプレイアウトが行われていると考えられる．

図 21 において，A，B と印付けられた節点（節点 A と節点 B）は図 20 の印付れれた交点への着手に対応している．図より，交点 A から始まるプレイアウトが多く行われていることがわかる．また，同一局面においてパラメタ値を変えた場合に探索により生成さ

れた探索木の例を図 22 から図 25 に示す．

図 22 は $C = 0.5$，$K = 1.4$ と C の値をデフォルト値より小さくした場合の例である．C の値を小さくすると UCB1 値における期待値に重きを置くことなる．よって図にあるように，期待値が大きい節点 A に対してより重点的に探索が行われやすくなる．

図 23 は $C = 2.0$，$K = 1.4$ と C の値をデフォルト値より大きくした場合の例である．C の値を大きくすると，先程とは逆に UCB1 値におけるバイアスに重きを置くことになる．このことから，図にあるように候補手に対応する根節点の子節点はいずれも平均的に探索が行われやすくなる．図 21 と比較すると，節点 A から行われるプレイアウト数が減っていることが確認できる．また節点 B は子節点を持たないことから，節点 B から始まるプレイアウト数も減っていることも確認できる．これはバイアスに重きを置いたことで，他の候補手に対してプレイアウトが行われたことが要因として考えられる．

図 24 は $C = 1.0$，$K = 1.1$ と K の値をデフォルト値より小さくした場合の例である．K の値を小さくすると，候補手を追加する閾値が小さくなり候補手をより多く取ることになる．よって図にあるように，根節点の子節点の数が多くなりやすくなる．図 21 と比較すると，根節点の子節点の数が増えていることが確認できる．また，節点 A 以外の節点に対してもプレイアウトを平均的に行っていることが確認できる．これは，候補手の数が増えたことにより，広く読むようになったことが要因として考えられる．

図 24 は $C = 1.0$，$K = 2.8$ と K の値をデフォルト値より大きくした場合の例である．K の値を大きくすると，先ほどとは反対に候補手を追加する閾値が大きくなり候補手の数が少なくなる．よって図にあるように，根節点の子節点の数が少なくなりやすくなる．

以上の例からも，同一局面においても，パラメタ C と K の値を変えることで生成される探索木の傾向が異なることがわかる．提案手法ではこの性質を利用する．

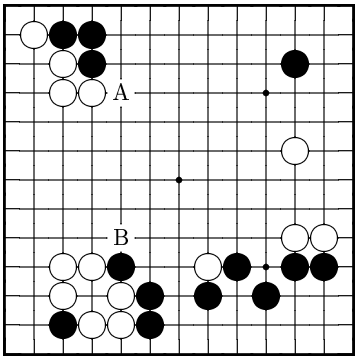

図 20 探索対象の局面（黒の手番）

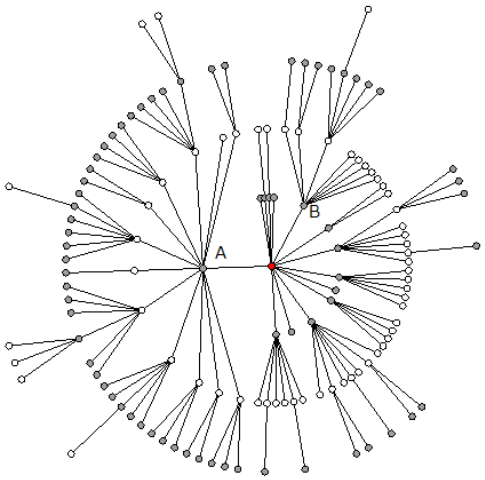

図 21 $C = 1.0$, $K = 1.4$ (デフォルト値) であるときに生成された探索木の例

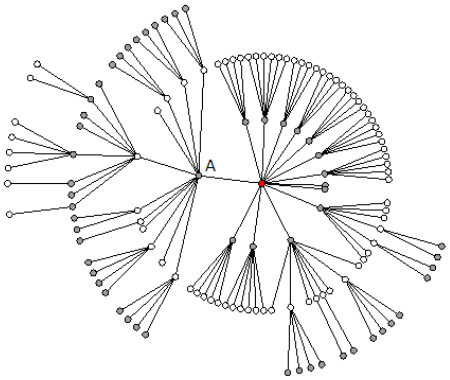

図 22 $C = 0.5$, $K = 1.4$ であるときに生成された探索木の例

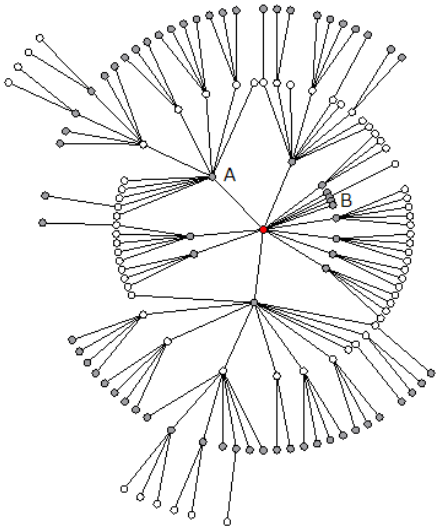

図 23 $C = 2.0$, $K = 1.4$ であるときに生成された探索木の例

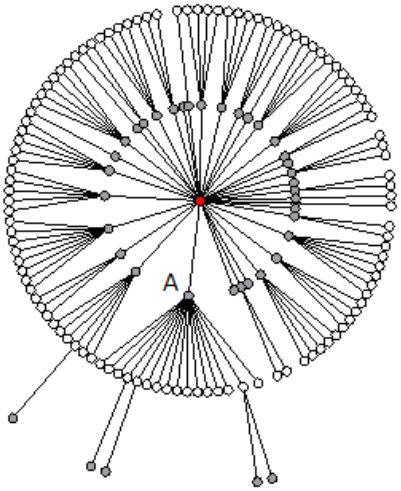

図 24 C= 1.0, $K = 1.1$ であるときに生成された探索木の例

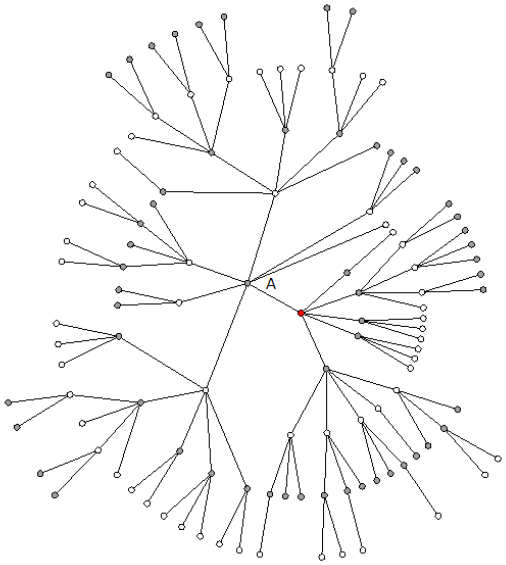

図 25 $C = 1.0$, $K = 2.8$ であるときに生成された探索木の例

5.4　実験 3 の概要

提案した予測読みを storm-ponder に実装した場合に，適切な C_{ponder}，K_{ponder} を探すため，提案した予測読みアルゴリズムを実装した storm (Version 1.14.3) と GNU Go 3.8 の対戦を行った．C_{ponder} は 0 から 1.5 の範囲，K_{ponder} は.1.1 から 2.0 の範囲からそれぞれ設定した．計測はデフォルト値 C_{ponder}=1.0，K_{ponder}=1.4 の周辺から行った．また，これ以降では提案した予測読みアルゴリズムを実装した storm を storm-ponder-param と表記する．

5.4.1　9 路盤，13 路盤での実験環境

9 路盤での実験環境は下記のとおりである．

・**OS** Ubuntu 12.04 32bit
・**CPU** Intel® Core™ i7-7620M CPU @ 2.70GHz x 4
・メモリ 4GB

9 路盤での実験条件は下記のとおりである．

・ 中国ルール，コミは 7 目半で 500 局（黒番 250 局，白番 250 局）
・ storm-ponder-param の手番時のプレイアウト数は 500 と固定，使用するスレッド数は 3
・ GNU Go のレベルはデフォルトの 10 と設定

13 路盤での実験環境は下記のとおりである．

・**OS** Ubuntu 12.04 64bit
・**CPU** Intel® Xeon(R) CPU E5430 @ 2.66GHz x 8
・メモリ 8GB

13 路盤での実験条件は下記のとおりである．

・ 中国ルール，コミは 7 目半で 500 局（黒番 250 局，白番 250 局）
・ storm-ponder-param の手番時のプレイアウト数は 3,000 と固定，使用するスレッド数は 6
・ GNU Go のレベルはデフォルトの 10 と設定

5.4.2 実験 3 の結果と考察

9 路盤での実験結果を表 7 に示す．この表の各欄には 3 つの数値が記されており，上段は storm-ponder-param の勝率，中段には繰り越した平均プレイアウト数，下段には繰り越した確率が記されている．ここで，繰り越した確率とは予測読みを行った回数に対して，予測読み時に相手の着手を 1 回以上探索していた割合のことである．

表 7 より，C_{ponder} の値が 1.1 以下では，その値を小さくすると繰り越した平均プレイアウト数が増加し，勝率は減少している傾向が見られる．また，K_{ponder} の値を大きくすると，繰り越した平均プレイアウト数が増加する一方で，繰り越した確率は減少する傾向が見られる．

表 7 から，$C_{ponder} = 1.1$ で $K_{ponder} = 1.4$ のとき勝率が最大となっていることがわかる．なお，デフォルト値 ($C_{ponder} = 1.0$ で $K_{ponder} = 1.4$) との勝率差の検定により求まる p 値は 0.158 であり，有意水準 5% で有意差は認められなかった．しかし，デフォルト値での勝率の 56.2% と比較すると，$C_{ponder} = 1.1$ で $K_{ponder} = 1.4$ のときの勝率が 60.6% を取っていることから，このパラメタ値の周辺をより調査することにより，棋力が向上し得ると考えられる．また，このパラメタ値の組み合わせは K の値は手番時と同じであるが，C の値をデフォルト値の 1.0 から若干大きくしたものであり，広く候補手を探索することが有効であることを示唆していると考えられる．

13 路盤での実験結果を表 8 に示す．各欄は表 7 と同様な項目で構成されている．表 8 より，C_{ponder} の値が 1.3 以下では，その値を小さくすると繰り越した平均プレイアウト数が増加し，勝率は減少している傾向が見られる．また，K_{ponder} の値を大きくすると，繰り越した平均プレイアウト数が増加する一方で，繰り越した確率は減少する傾向が見られる．

表 8 から，$C_{ponder} = 1.3$ で $K_{ponder} = 1.4$ のとき勝率が最大となっていることがわかる．デフォルト値 ($C_{ponder} = 1.0$ で $K_{ponder} = 1.4$) との勝率差の検定により求まる p 値は 0.0398 であり，有意水準 5% で有意に強くなっていることがわかる．このパラメタ値は，K の値は手番時と同じであるが，C の値をデフォルト値の 1.0 から 1.3 と大きくしたものである．これより，予測読みでは広く候補手を探索することが有効であったと考えられる．また，$C_{ponder} = 1.3$ で $K_{ponder} = 1.4$ の場合とデフォルト値の場合との繰り越した平均プレイアウト数と繰り越した確率を比較してもあまり変化がないことがわかる．このことから，繰り越した平均プレイアウト数と繰り越した確率からは，勝率が向上した要因を説明することができないことがわかった．勝率が向上した要因を説明するためには，局面毎に予測読みを行わなかった場合と行った場合とで選択した着手がどのように変化す

るかなどを詳細に解析する必要があると考えられる．

9 路盤と 13 路盤両方において，$C_{ponder} = 0$ で $K_{ponder} = 1.4$ のときに繰り越した平均プレイアウト数が最大となっている．しかし，勝率は最大となっていない．これはひとつの着手に候補手を絞ることにより，実際の相手の着手を読んでいた場合には繰り越されるプレイアウト数が多くなる．一方で，手を広く探索しないため，繰り越した確率が低く，プレイアウトが繰り越される場合が他のと比べて少ないことが原因だと考えられる．

また，$C_{ponder} = 1.1$ で $K_{ponder} = 1.4$ のとき 13 路盤では勝率が最大であり，デフォルト値の場合と差があったが 9 路盤では勝率はデフォルト値の場合とあまり差がないことがわかる．このことから，盤の大きさにより，パラメタの値も変える必要があると考えられる．

表 7 9 路盤における対戦結果．各欄の上段は勝率，中段は繰り越した平均プレイアウト数，下段は繰り越した確率を記している．

C_{ponder} \ K_{ponder}	1.1	1.2	1.3	1.4	1.5	1.6	1.7	2.0
0.0				53.6% 1,814 81.2%				
0.5				55.6% 1,380 80.5%				
1.0		57.8% 1,206 85.3%	56.0% 1,257 82.8%	56.2% 1,327 80.4%	58.0% 1,308 77.2%	59.2% 1,368 76.2%	59.0% 1,375 74.1%	53.4% 1,529 71.6%
1.1		55.2% 1,169 86.0%	59.2% 1,235 82.9%	60.6% 1,286 80.5%	58.6% 1,308 77.6%			
1.2				52.8% 1,380 80.1%				
1.5	52.0% 891 89.5%							

表 8 13 路盤における対戦結果．各欄の上段は勝率，中段は繰り越した平均プレイアウト数，下段は繰り越した確率を記している．

C_{ponder} \ K_{ponder}	1.2	1.3	1.4	1.5
0.0			53.8% 1,368 71.4%	
1.0	56.2% 946 77.8%	53.2% 1,046 74.5%	55.6% 1,063 71.3%	
1.1		58.6% 994 73.4%	56.6% 1,035 71.4%	57.4% 1,186 69.5%
1.2		56.4% 911 73.9%	58.2% 1,033 71.7%	57.6% 1,141 69.9%
1.3		53.8% 932 74.0%	62.0% 996 71.5%	58.8% 1045 69.1%
1.4			52.0% 990 71.6%	

5.5 実験 4 の概要

実験 3 での 13 路盤において最も勝率が高かったパラメタ $C_{ponder} = 1.1$ で $K_{ponder} = 1.4$ を 19 路盤で使用した場合でも有効か調べるために対局実験を行った．

5.5.1 19 路盤での実験環境

19 路盤での実験環境は下記のとおりである．

・**OS** Ubuntu 14.04 64bit
・**CPU** Intel® Xeon(R) CPU E5430 @ 2.66GHz x 8
・メモリ 8GB

19 路盤での実験条件は下記のとおりである．

- 中国ルール，コミは 7 目半で 200 局（黒番 100 局，白番 100 局）
- storm-ponder-param の手番時のプレイアウト数は 7,000 と固定．使用するスレッド数は 6，式 (1) における C は 1.0，式 (12) における K は 1.4 と設定し，$C_{ponder} = 1.3$ とし $K_{ponder} = 1.4$ と設定
- GNU Go のレベルは 0 と設定
- GNU Go の着手は 2 秒経過するまで着手を返さないように設定

5.5.2 実験 4 の結果と考察

実験結果を表 9 に示す．実験 2 の結果と比較すると，対局数が異なるが storm-ponder と同じ程度の勝率が得られていることがわかる．その原因として，表 9 に記している繰り越したプレイアウト数の割合が低いことが挙げられる．ことのことから，19 路盤では繰り越したプレイアウト数の割合を高めるように K_{ponder} の値を大きめに設定する必要があると考えられる．

表 9 実験 4 の実験結果

storm-ponder-param の勝率	繰り越したプレイアウト数の割合	繰り越した確率
48.5%	11.7%	65.4%

5.6 実験 5 の概要

囲碁プログラムを変更したときに，自己対戦の勝率を計測し，その勝率により変更を適用することで棋力が向上したか評価することが考えられる．しかし，自己対戦の勝率のみで変更点の良し悪しを評価することはあまり行われていない [12]．その変更が自己対戦のみに有効な変更である場合が考えられるからである．つまり，変更により自己対戦の勝率が良い場合であっても，他のプログラムとの勝率の変動が見られないことや，変更前より悪くなってしまう場合もある．このことから，勝率により変更の良し悪しを評価するには，対戦するプログラムの数を増やした方が良いと考えられる．しかし，対戦の組み合わせなどの管理をする必要があることなどから，プログラムの数を増やし実験対局することは容易ではない．

CGOS (Computer Go Server)[1]は上記の問題を解決してくれるサービスである．具体

的には，このサービスはコンピュータ囲碁プログラム同士のみ対局を行うサーバを提供している．囲碁プログラムの開発者は CGOS の Web ページで配布されているクライアントプログラムを使用して囲碁プログラムをサーバに接続するだけで，自動的にサーバに接続しているプログラムとの対局を行える．対局が行われるごとに接続したプログラムの Elo レーティングが変動し，その値により修正前より良くなっているか判断することができる．本実験では，パラメタ値 $C_{ponder} = 1.1$ と $K_{ponder} = 1.4$ を用いた storm-ponder-param と storm がどの程度のレーティング差を持つか，CGOS を使用して 13 路盤の対局を行った．

5.6.1 プログラムの動作環境

プログラムの動作環境は下記のとおりである．

・**OS** Ubuntu 12.04 64bit
・**CPU** Intel® Xeon(R) CPU E5430 @ 2.66GHz x 8
・メモリ 8GB

プログラムの設定は下記のとおりである．

・ storm と storm-ponder-param の手番時のプレイアウト数は 3,000 と固定．使用するスレッド数は 3，式 (1) における C は 1.0，式 (12) における K は 1.4 と設定し，$C_{ponder} = 1.3$ とし $K_{ponder} = 1.4$ と設定

CGOS の 13 路盤での対局は中国ルール，コミは 7 目半で行われる．

5.6.2 実験 5 の結果と考察

実験結果を表 10 に示す．表より，storm-ponder-param の Elo レーティングは storm より 100 以上大きいことがわかる．これより，適当なパラメタを設定した予測読みを行うことで，棋力が向上していると考えらる．また，2 つのプログラムの Elo レーティング差が 100 あると，Elo レーティングが大きい方が小さい方に対しての勝率がおよそ 64% であること意味している．これは実験 1 の自己対戦の結果ともおよそ一致する結果が得られていると考えられる．

Elo レーティングが上位の対戦プログラムの消費時間と勝敗の関係を表 11 に示す．表において，1 手あたりの平均消費時間は storm-ponder-param との対戦時のものを記して

[1] http://cgos.boardspace.net/

いる．表から 1 手あたりの平均消費時間が 1 秒付近である mogoRel3_10k に対してのそれぞれの勝率はあまりかわらないことがわかる．これは相手の消費時間が少なかったために，予測読みを十分に行えなかったことが要因として考えられる．一方で，2 秒付近であるプログラムに対しての storm-ponder-param の勝率は storm に比べて向上している場合が多いことがわかる．例えば，pachi10_Pat_10k に対して，storm の勝率は 0.00% であるが，storm-ponder-param は 41.2% まで向上している．これは，相手の平均消費時間が多く十分に予測読みが行えたことが要因として考えられる．

表 10 実験 5 の実験結果

プログラム名	Elo レーティング	対局数
storm-ponder-param	1708	472
storm	1602	463

表 11 実験 5 における Elo レーティング上位の対戦プログラムの消費時間と勝敗の関係

対戦プログラム名	1 手あたりの 平均消費時間 [s]	storm-ponder-param の 勝ち数 / 試合数，勝率	storm の 勝ち数 / 試合数，勝率
Valkyria3.5.19_4t	3.53	2/ 40，5.00%	0 / 25，0.00%
fuego1604_20k	1.93	0 / 20，0.00%	0 / 11，0.00%
Aya781c_1k	1.63	1 / 53，1.89%	3 / 47，6.38%
Valkyria3.5.19_4k	2.79	5/ 30，16.7%	3 / 23，13.0%
pachi10_Pat_10k	2.19	7 / 17，41.2%	0 / 19，0.00%
mogoRel3_10k	1.01	2 / 19，10.5%	2/ 14，14.29%
Gnugo-3.7.10-a1	1.93	36 / 66，54.6%	23 / 48，47.9%

5.7 まとめ

はじめにパラメタ調整による予測読みの強化方法を提案した．実験 3 では，提案した強化方法を storm に実装し，GNU Go との対局を行った．パラメタ値の組み合わせを複数試し，GNU Go に対する勝率や繰り越した平均プレイアウト数，繰り越した確率を計測

した．その結果，13 路盤では勝率が向上することがわかったが，その要因を繰り越した平均プレイアウト数と繰り越した確率からは説明することができないこともわかった．

実験 4 では，実験 3 において勝率が最大となったパラメタ値を用いて，19 路盤で GNU Go との対局を行った．その結果，得られた勝率はパラメタ値がデフォルトの値であった場合と同程度の勝率が得られることが確認された．繰り越したプレイアウト数の割合が 13 路盤の場合に対して低く，パラメタ値を変更しても予測読みに　より得られる効果は変わらなかったことが要因として考えられる．

実験 5 では，CGOS に storm と storm-ponder-param をそれぞれ接続し，GNU Go 以外との対戦も行った．storm-ponder-param のパラメタ値は実験 3 において勝率が最大となったパラメタ値を用いた．400 以上の対局を行った結果，storm-ponder-param の Elo レーティングは storm と比較して 100 以上向上していることが確認された．これから，GNU Go 以外の囲碁プログラムに対しても棋力が向上していることが確認できた．

第 6 章　結論

本稿では，まず村松研究室で開発されている囲碁プログラム storm に一般的な予測読みを実装し，棋力が向上することを確認した．次に，探索パラメタを変えて行う予測読みの強化方法を提案し，パラメタの値によって棋力がどのように変化するかを示した．

予測読み時のパラメタの設定により，繰り越した平均プレイアウト数や繰り越したプレイアウト数の割合が大きく変動することが確認され，棋力も変動することが新たにわかった．13 路盤ではパラメタを手を広く読むように適切に設定することで，標準的な予測読みを行う場合よりも強くなることを示した．また，盤の大きさにより適切なパラメタを設定する必要があることを示した．CGOS を用いた実験では，適当に設定したパラメタを用いた予測読みを実装することで，予測読みを行わないプログラムより Elo レーティングが 100 以上向上することを確認した．

本稿で試みたパラメタによる予測読みの強化方法はどの局面でも設定した同一のパラメタを用いるものであった．しかし，重点的にある着手を探索する必要がある局面や広く候補手を読む必要がある局面がそれぞれ存在すると考えられる．そこで，今後の課題としては局面毎に適切なパラメタ値を探索情報を用いて事前に設定することが挙げられる．

参考文献

[1] P. Auer, N. Cesa-Bianchi and P. Fischer, Finite-time Analysis of the Multi-armed Bandit problem, *Machine Learning*, Vol. 47, pp. 235-256, 2002.

[2] B. Brügmann, Monte Carlo Go, *Technical report*, Physics Department, Syracuse University, 1993.

[3] M. Enzenberger and M. Müller, Fuego — an open-source framework for board games and Go engine based on Monte-Carlo tree search, *TR 09-08*, University of Alberta, 2009.

[4] S. Gelly and D. Silver, Combining Online and Offline Knowledge in UCT, *Proceeding of the 24th International Conference on Machine Learning*, pp. 273-280, ACM Press, New York, 2007.

[5] S. Gelly, Y. Wang, R. Munoud and O. Teytaud, Modification of UCT with Patterns in Monte-Carlo Go, *Technical Report RR-6062*, INRIA, 2006.

[6] L. Kocsis and Cs. Szepesvári, Bandit based Monte-Carlo planning, In *proceedings of the 17th European Conference on Machine Learning*, Vol. 4212 of lecture Notes in Computer Science, pp. 282-293, Springer, 2006.

[7] S.-C. Huang, New Heuristics for Monte Carlo Tree Search Applied to the Game of Go, *PhD thesis*, Nat. Taiwan Normal Univ., Taipei, 2011.

[8] R. Coulom, Computing Elo ratings of move patterns in the game of Go, *ICGA Journal*, Vol. 30, No. 4, pp. 198-208, 2007.

[9] G. J. Wills, NicheWorks — Interactive Visualization of Very Large Graphs, *In Graph Drawing '97 Conference Proceedings*, pp. 403-414, 1997.

[10] GNU Go GNU Project, http://www.gnu.org/software/gnugo/, 2009.

[11] 小谷義行 編著, ゲーム計算メカニズム, 株式会社コロナ社, p. 31, 2010.

[12] 松原仁　編者，コンピュータ囲碁 — モンテカルロ法の理論と実践，共立出版株式会社，pp. 203-204, 2012.

付録 A　囲碁用語

以下に箇条書きで，本稿で扱う囲碁における用語を説明する．

黒石 囲碁に用いられる黒色の石のこと．

黒番 盤面に黒石を着手するプレイヤのこと．

局面 盤面の状態のこと．

コウ 同型反復を禁止するために直ぐに取り返すことができない石の状況のこと．

合法手 着手選択を行う局面において，打つことができる着手のこと．

コミ ゲーム開始前に与えるハンデキャップのこと．勝敗判定時に加算される．一般の対局では黒番が有利であることから，黒番は白番に対してコミを出す．

地の数 同一色の石で囲まれた交点の数のこと．

白石 囲碁に用いられる白色の石のこと．

白番 盤面に白石を着手するプレイヤのこと．

終局 黒番と白番のプレイヤがどちらもパスしたときのゲーム状態のこと．

対局 囲碁の対戦のこと．

着手 盤面に描かれている線の交点上に碁石を置くこと．

手 着手のこと．

手番 着手する側の黒番または白番のこと．

盤面 囲碁に用いられる盤において，石が置かれる面のこと．

謝辞

本研究を進めるにあたって，多くの方にお世話になりました．

村松正和教授には適切な助言やご指摘を多くいただきました．また，村松研究室の学生の皆様にも多々助けていただきました．皆様には深く感謝いたします．

平成 27 年 1 月 30 日