ゲームにおける棋譜の性質と強さの関係に基づいた学習

佐藤佳州

システム情報工学研究科

筑波大学

2014年3月

目 次

図 目 次

表 目 次

概要

チェス，将棋，囲碁といったゲームは，ルールが明確で評価がしやすく，また，技術の進歩が一般の人にとってもわかりやすいことから，人工知能の一分野として古くから研究されてきた．コンピュータは，オセロ，チェス等のゲームでは既に人間のトッププレイヤを上回り，近年では，将棋や囲碁などのより複雑なゲームが主な研究対象となっている．

近年，ゲームの分野において，プログラムの性能を大きく向上させた要因の 1 つとして，機械学習が広く取り入れられるようになった点が挙げられる．ゲームの分野において，最も一般的に行われている学習は，棋譜を教師とした学習である．この手法は，評価関数の学習，探索深さの調整，モンテカルロ木探索のシミュレーション（playout）中の指し手の選択など，幅広い課題に対して良い結果を得ている．特に，近年では，将棋の評価関数の学習において高い効果を得ており，機械学習の成功によって現在のコンピュータ将棋は，人間のトッププレイヤに迫る強さを得ている．

一方，従来の学習手法の課題の 1 つとして，学習棋譜の性質が考慮されていない点が挙げられる．棋譜を教師とした学習では，教師データの性質によって学習されるパラメータに大きな違いが生じる．人間の棋譜は教師データとして均一な性質とはいえず，様々な強さのプレイヤの棋譜が含まれる，悪手が存在する，戦術や勝敗に与える重要度に違いが存在する，といったように様々な性質の学習データが混在した状態となっている．また，もう 1 つの問題として，従来の棋譜を教師とした学習では，プログラムの指し手と棋譜中のエキスパートの指し手の一致率を向上させることを目的としており，学習によって得られるパラメータを用いたプログラムの強さとの関係が不明であるという点が存在する．

本論文では，これらの問題を改善する手法として，学習データの性質と強さの関係を取り入れた学習手法を提案する．具体的には，(1) 教師データの重要度の学習，(2) 目的関数の学習，の 2 つの手法を提案し，実験により効果を検証する．(1) 教師データの重要度の学習では，学習データに重要度を導入し，学習により生成されるプログラムの強さに基づいた重要度の学習と重要度に従ったパラメータ学習を組み合わせた手法を提案する．(2) 目的関数の学習では，学習局面の性質と強さの関係を基に，目的関数自体のパラメータを学習する手法を提案する．提案手法は，棋譜を教師とした学習において，どのような局面を，どのように学習すれば「強い」プレイヤを学習できるかを自動的に獲得することを目的とする．

本論文では，将棋を題材として，提案手法の効果を検証した．重要度の学習では，評価関数の学習，探索深さの調整，モンテカルロ木探索のシミュレーション（playout）の方策の学習，といった現在棋譜を教師とした機械学習が成功を収めている課題に対して提案手法を適用し，その効果を検証した．実験の結果，提案手法を用いたプログラムが従来手法を用いたプログラムを有意に上回り，その有効性を示した．目的関数の学習では，現在，特に研究が盛んな評価関数の学習に提案手法を適用し，従来手法を大きく上回る結果を得ることに成功した．また，提案手法によって学習された目的関数は，異なる評価関数や異なる学習棋譜を用いた学習に対しても有効であり,「強い」評価関数を学習するための汎用的な目的関数が学習できることを示した．

第1章　序論

ゲームプログラミングは人工知能の一分野として，探索，機械学習，並列処理など幅広い技術のテストベッドとして研究が行われている．ゲームの研究は，チェスを中心として，1950年頃から本格的な研究が行われてきた[1, 2]．現在では，コンピュータは，オセロ，チェスといったゲームでは既に人間を上回る強さを得ており，さらに，近年の情報技術の進歩によって，将棋や囲碁などの複雑なゲームにおいても，人間のトッププレイヤに迫る性能を得ている．(以降，本論文中における「性能」とはプログラムの「強さ」を意味する．また「プログラム」は特に断りがない限り,「ゲーム（将棋）プログラム」といった意味で用いる．)

現在，ゲームプログラミングの研究領域は幅広いものとなっており，以下に示すような，様々な種類の研究が行われている．

(1) 強いプレイヤをコンピュータで実現するための研究

(2) ゲームを対象とした認知科学に関する研究

(3) 人間を楽しませるための研究

(4) ゲームやパズルの創作

中でも，古くから研究され，中心的な課題となっているのが，(1)の「強いプレイヤをコンピュータで実現するための研究」である．コンピュータでゲームを実現する際には探索と評価関数による手法が古くから用いられており[1]，強いプログラムを実現するための研究は，大枠では「探索」と「評価関数」の研究に集約される．また，これに加えて近年では,「モンテカルロ木探索」[3]という新たな手法が研究されている．この手法は，コンピュータ囲碁では探索と評価関数による手法を大きく上回る成功を収めており，モンテカルロ木探索も探索と評価関数に加え，ゲームプログラミングにおける主要な研究課題の1つとなっている．

近年，ゲームプログラミングの分野において，プログラムの性能を大きく向上させた要因の1つとして，機械学習が広く取り入れられるようになった点が挙げられる．オセロ[4]やバックギャモン[5]などでは早い段階から機械学習を採用したプログラムが成果を挙げてきたが，将棋や囲碁といった複雑なゲームにおいて機械学習が成功したの

は比較的最近のことである．これは機械学習の進歩に加え，計算機の計算能力の向上によって，数百万局面の学習データを現実的な時間内に効率良く処理できるようになったことが大きな要因と言える．現在では，コンピュータによって思考ゲームを実現するための主要なアルゴリズムである,「探索」,「評価関数」,「モンテカルロ木探索」のいずれにおいても，パラメータの調整に機械学習が取り入れられている．

現在のゲームプログラミングにおける機械学習では，エキスパートの棋譜を教師とした教師あり学習が一般的となっており，多くの課題に対して良い結果を得ている．しかし，従来の学習手法は，学習棋譜中のエキスパートの指し手とプログラムの指し手の一致率を向上させることを目的としており，学習によって得られるパラメータを用いたプログラムの強さが考慮されていない点に改善の余地がある．本論文では，これらの課題に対して，学習棋譜の性質と強さの関係を取り入れた学習手法を提案する．

本章の構成を以下に示す．1.1節ではまず，ゲーム木探索に関するアルゴリズムとして，min-max探索，alpha-beta探索，モンテカルロ木探索等の基礎的なアルゴリズムの概要を説明し，1.2節で，現在ゲームの分野において行われている機械学習の概要を述べる．その後，1.3節で現在のゲームにおける機械学習の課題と本論文の目的を述べ，1.4節で本論文の構成を示す.

1.1　ゲーム木探索

本節ではゲーム木探索の基本的なアルゴリズムについて説明する．本論文では，ゲームの中でも，特に，二人ゼロ和有限確定完全情報ゲームを対象とする．二人ゼロ和有限確定完全情報ゲームとは，以下の性質を満たすゲームを指す.

- 2人で行うゲームであること
- 両者の損得の合計が常にゼロであること
- 有限の手数で終了することが保証されていること
- 運の要素が入らず，また，双方のプレイヤの行動をすべて知ることができること

本論文で主な対象とする将棋はこの分類に属し，その他，オセロ，チェス，囲碁等のゲームもこの分類に該当する．一方，ポーカー，麻雀，バックギャモン等はこの分類には含まれない．以降，本論文中で単に「ゲーム」といった場合には二人ゼロ和有限確定完全情報ゲームを指すものとする．

コンピュータでゲームを実現する際の手法としては，大きく分けて，(1)探索と評価関数を用いる手法，(2)モンテカルロ木探索，の2通りの手法が存在する．(1)の探索と評価関数による手法は，多くのゲームにおいて用いられてきた古典的な手法である．

探索は人間でいうところの「読み」，評価関数は「形勢判断」「大局観」に対応する．探索と評価関数による思考は多くのゲームで成功を収めており，オセロやチェスでは人間のトッププレイヤ以上，将棋においてもトッププレイヤに迫る強さを得ている．(2)のモンテカルロ木探索は，近年提案された新しいゲームのアルゴリズムであり，乱数を用いたゲームのシミュレーションによって局面を評価する．この手法は評価関数の設計が困難であった囲碁において大きな成功を収めている．その他のゲームへの研究も行われている[7, 8, 9, 10, 11]ものの，チェスライクなゲームにおける研究結果は多くない．以降，それぞれの手法の基本的なアルゴリズムの概要について述べる．

1.1.1　Min-max探索

Min-max探索はコンピュータによりゲームを実現する際の，最も基本的なアルゴリズムである．図 1.1にmin-max探索の例を示す．この手法では，自分の手番（ノード）では評価値が最大となるノードを選択し，相手の手番では評価値が最小となる（つまり相手にとって評価値が最大となる）ノードを選択することで，最善手と評価値を求めることを可能とする．

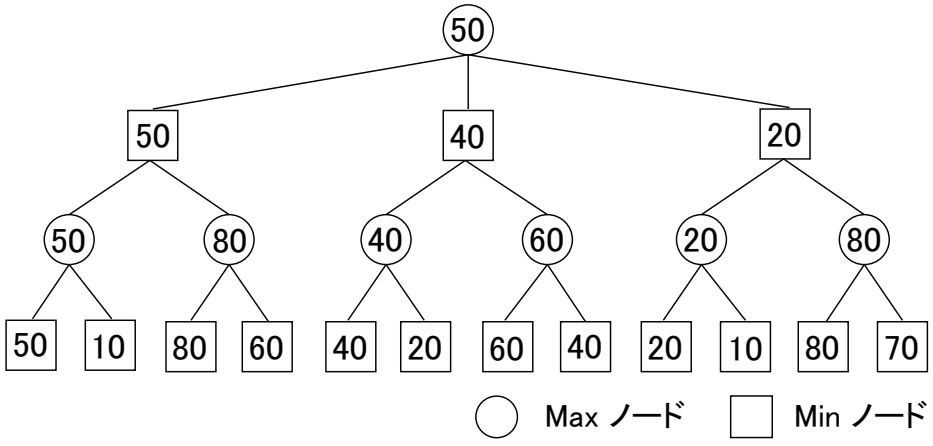

図 1.1: Min-max探索の例

図 1.1のような探索木はmin-max木と呼ばれ，ゲーム木探索の最も基本的な考え方となる．Min-max木は自分の手番に相当するmaxノードと，相手の手番に相当するminノードから構成される．Min-max探索では，すべてのノードを探索するため，効率が良い探索手法とはいえず，実際のゲームプログラムでは，以降で説明するalpha-beta探索などが用いられる．

なお，探索の末端ノードでは，評価関数を用いて局面の評価値を算出する．評価関数とは，局面の優劣を数値化する関数であり，例えば将棋の場合には駒の価値や駒の位置関係の評価などが用いられる．表 1.1に，評価関数の例として，将棋プログラムYSS[6]の評価関数の一部（駒の価値）を示す.

表 1.1: YSSの駒の価値

	歩	香	桂	銀	金	角	飛
基本価値	100	430	450	640	690	890	1040
駒が成る価値	320	200	190	30	0	260	260
持駒の付加価値	15	50	60	80	90	220	230

1.1.2　Alpha-beta探索

Alpha-beta探索[12]は，min-max探索と同じ解を保証しつつ，枝刈りを行うことにより，効率の良い探索を行う手法である．図 1.2にalpha-beta探索の例を示す.

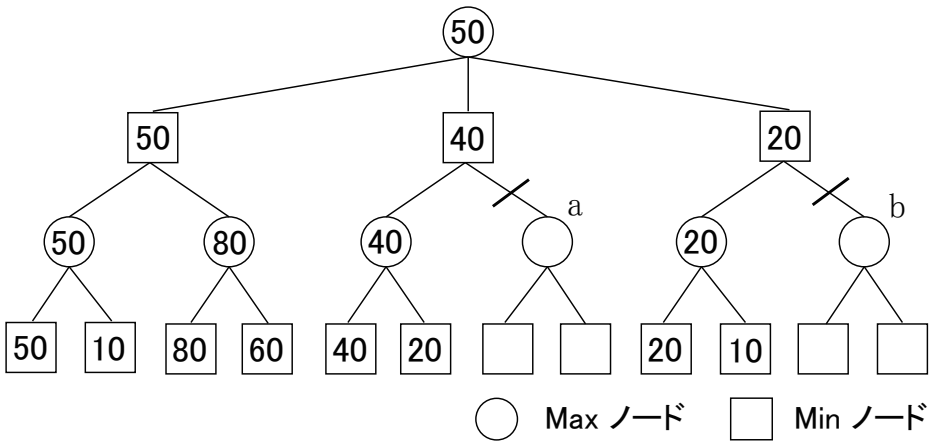

図 1.2: Alpha-beta探索の例

図 1.2では，ノードa，bはどのような値であったとしても，最善手の選択には影響を及ぼさない．例えば，ノードaの場合には，評価値40以上では，minノードである親ノードに採用されず，ルートで評価値50を上回ることはできないことがわかる．

なお，alpha-beta法では，指し手をランダムな順番で探索するよりも，評価値の高い指し手から順に探索したほうが枝刈りの効率が良くなるという特徴がある．そのため，

ヒューリスティックを用いた指し手の並べ替え（move ordering）を行うのが一般的である．

多くのゲームプログラムでは，alpha-beta探索をベースとした探索手法であるPVS（Principal Variation Search）[13]や，aspiration search[14]を用い，さらに深い探索を行うため，min-max探索と同じ解を保証しない枝刈りも組み合わせるのが一般的である．

現在，比較的良く使われている，枝仮り手法の例を以下に示す．これらの枝刈り手法は，ゲーム木探索の性質を利用したものであり，オセロ，チェス，将棋など幅広いゲームを対象として使用することができる．

- ProbCut[15]
- Null move pruning[16, 17]
- Futility pruning[18, 19]
- Late Move Reductions（LMR）[20]
- History pruning[21]

その他，ゲーム固有の知識に基づいた枝刈り（指し手の選択）が行われることもある．以前は，将棋などのゲームでは，探索空間が広いことから，ゲーム固有の知識に基づいた指し手の選択を行う，選択探索が中心であった．しかし，上記の枝刈り手法の確立や，計算機の処理能力の向上によって，現在の主要なプログラムでは，ゲーム固有の知識に基づいた強力な枝刈りが行われることは少なくなっている．

1.1.3　静止探索

静止探索（quiescence search）[22]は探索の末端で評価値を安定させるために行う手法である．

例えば，探索の末端で図 1.3のような局面が現れた場合，単純に評価関数を呼び出すと先手の角得という評価になる．しかし，実際には角はすぐに取り返されるため，この局面は互角と評価するのが正しいと言える．このような問題点を解決するため，探索の末端ではそのまま評価関数を呼び出すのではなく，静止探索を呼び出すのが一般的である．静止探索では，探索する指し手を，駒を取る手や王手などの戦術的な指し手に限定し，評価値が安定するまで探索を行う．具体的には，戦術的な手を深く探索したときに評価値が良くなる指し手が存在すれば探索を継続し，良くならない場合には，現在の局面を安定した局面とみなし，評価値を返す．なお，関連する技術として，Singular Extention[23]などの探索延長手法も，静止探索に近い効果を持つ．

図 1.3: 静止探索が必要な局面の例

1.1.4　AND/OR木の探索

Min-max木の特殊な形として，評価値を勝敗（1/0）に限定したAND/OR木が存在する．詰将棋や詰碁，またゲームの完全解析を行う場合などがAND/OR木の状態に相当する．AND/OR木の例を図 1.4に示す.

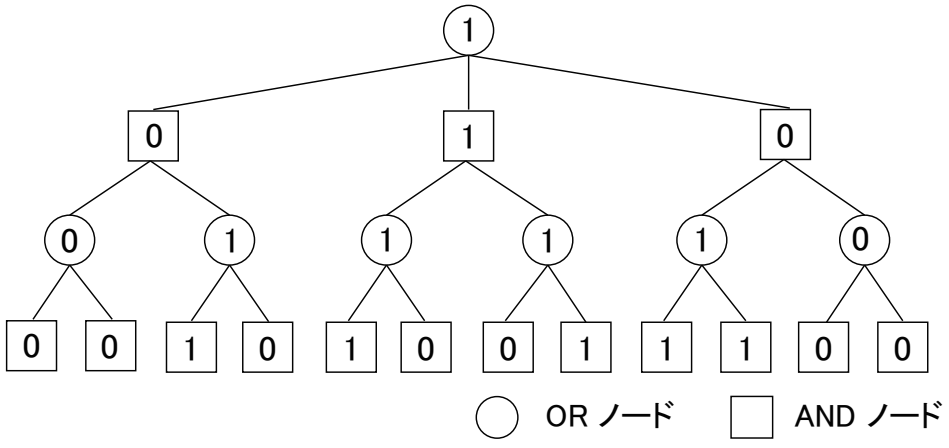

図 1.4: AND/OR木の例

AND/OR木は，子ノードの1つが1であれば1となるORノードと，子ノードのすべてが1であれば1となるANDノードによって構成される．詰将棋の例では，詰ませる側の

手番がORノード，逃げる側の手番がANDノードに相当する．

AND/OR木の探索では，効率の良い探索手法がいくつも提案されている．具体的には，proof-number search[24]，PN*[25]，PDS（Proof-number and Disproof-number Search）[26]，df-pn（Depth-First Proof-Number search）[27]，df-pn+[28]といった手法が存在する．これらの探索手法では，評価値を確定するために要するコストを示す，証明数，反証数という概念を利用することで，最良優先の探索を実現している．これらの探索技術によって，詰将棋ではミクロコスモス（1525手詰）など超長手数と呼ばれる問題も，コンピュータでは短時間で解答可能となっており，人間の能力を大きく上回っている．また，これらの技術の進歩もあって，チェッカー[29]など一部のゲームは，初手からの完全解析が終了している．

1.1.5　モンテカルロ木探索

モンテカルロ木探索[3]は，近年提案されたゲームの新たなアルゴリズムである．モンテカルロ木探索の例を図 1.5に示す.

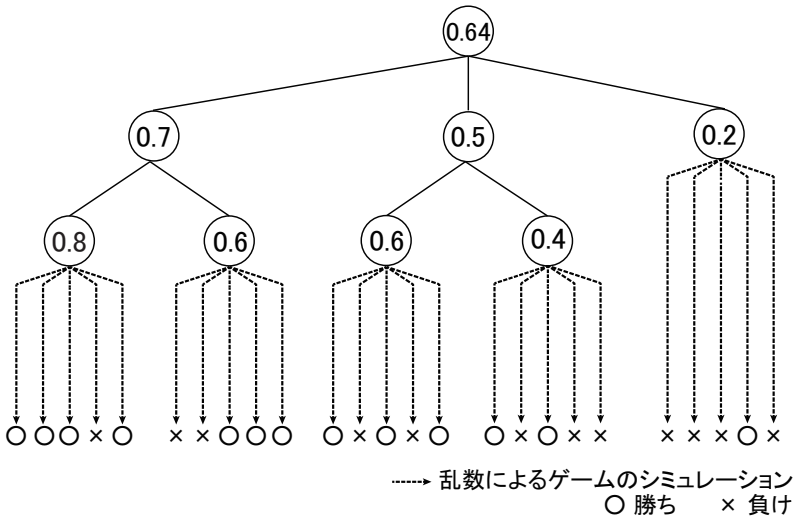

図 1.5: モンテカルロ木探索の例

この手法の特徴は，乱数を用いた指し手の選択を行うゲームのシミュレーション（playout）を終局まで何度も行い，その勝率によって局面を評価するというものである．モンテカルロ木探索は，このモンテカルロ法の考え方と木探索を組み合わせた手法となっている．この手法の大きな利点は，シミュレーションの勝率によって局面を評価するため，評価関数が不要となる点である．そのため，評価関数の設計が困難であった囲碁では，特に大きな成功を収めている．

1.2 ゲームにおける機械学習

ゲームプログラミングにおいて，機械学習は幅広いゲームにおいて研究されてきた．以下に，現在，ゲームの分野において，機械学習が成功を収めている課題の例を示す．

- 評価関数の学習
  評価関数の学習は，人間の知識を数値化することの難しさ，プログラムの強さに与える影響の大きさから，多くのゲームにおいて研究されてきた[5, 4, 30, 31]．特に，近年では，将棋においてプロの棋譜を教師とした機械学習が大きな成功を収めており，人間のトッププレイヤに迫る強さを得ることに成功している．

- 探索深さの調整
  コンピュータ将棋では，評価関数の学習以前から，探索深さの調整に機械学習が取り入れられている．これは，将棋ではチェスと比較して，合法手の数が多く，探索空間が広いため，より有力な手に絞った探索を行う必要があったことに起因する．代表的な手法としては，実現確率探索[32]が挙げられる．この手法では，指し手の選択確率を機械学習によって求め，その確率に従って探索深さを制御する．実現確率探索は将棋では高い効果が得られており，現在でもトップレベルのプログラムにおいて採用されている探索手法となっている．

- モンテカルロ木探索のシミュレーション中の指し手の選択
  モンテカルロ木探索では，シミュレーション中の指し手の選択が，プログラムの性能に大きな影響を及ぼすことが知られている．シミュレーション中の指し手の選択は完全にランダムでは良い性能を得ることは難しく，乱数を利用しつつも，何らかの形でヒューリスティックを導入し，対象とするゲーム（囲碁，将棋）らしい指し手の選択を行う必要がある．一方で，どのような指し手の選択が強いプログラムを実現するかは明らかになっておらず，人間が手作業で調整することが難しい課題の1つとなっている．コンピュータ囲碁では，このシミュレーション中の指し手の選択において，棋譜を教師とした学習[33]やSimulation Balancing[34]といった学習が成功を収めている．

上記の例から，一部強化学習などが成功を収めた例もあるものの[5]，いずれの課題に対しても棋譜を教師とした学習が広く成功を収めていることがわかる．

この背景としては，機械学習の進歩に加え，近年の計算機の性能向上から，複雑なゲームにおいても膨大な教師（棋譜）データを利用した機械学習が可能となったことが挙げられる．

1.3 本論文の目的

前節で述べたように，近年，ゲームの分野において，棋譜を教師とした機械学習は多くの課題に対して良い結果を得ている．一方で，ゲームで用いられている従来の棋譜を教師とした学習には以下に示す，2つの大きな課題が存在する．

(1) 棋譜の手を模倣することを目的としており,「強さ」を目的とした学習になっていない

(2) 学習棋譜の性質が考慮されていない

(1)について，現在の学習手法は，プログラムの指し手と学習棋譜中のエキスパートの指し手の一致率を向上させることを目的としており，学習によって得られるプログラムの強さは考慮されていない．実際，チェスにおいては，人間の強いプレイヤとの指し手の一致率を向上させることが必ずしも強さを向上させるわけではないことが以前から示されている[35, 30].

(2)について，ゲームにおける機械学習では，一般的に，人間のプレイヤの棋譜が用いられる．人間の棋譜は，教師データとして均一な性質とはいえず，戦術や学習局面が勝敗に与える重要度などについて違いがある．現在の機械学習では，単純に指し手の一致率を全体的に向上させることを目的としているが，局面の特徴を考慮して，勝敗に直結する局面を重視したほうが強いプログラムを学習できる可能性がある．また，人間の棋譜は信頼性という観点でも均一ではなく，プレイヤの強さに違いが存在する，棋譜中に悪手が含まれる，といった特徴がある．これまで，囲碁や将棋などの複雑なゲームでは，コンピュータは人間にはるかに及ばなかったため，この点は問題とならなかった．しかし，現在では，これらのゲームにおいてもコンピュータは人間のトッププレイヤに迫る強さとなっており，単純に人間の棋譜を模倣することが強さの向上につながるとは限らないと考えられる．

本論文では，これらの問題点を解決する手法として，学習棋譜の性質と強さの関係を取り入れた学習手法を提案する．具体的には，(1)教師データの重要度の学習，(2)目的関数の学習，の2つの手法を提案する．(1)では，学習データに重要度を導入し，学習されたプログラムの強さに基づいた重要度の学習と重要度に従ったパラメータ学習を組み合わせた手法を提案する．(2)では，(1)の拡張として，評価関数の学習を対象に，学習局面の特徴と強さの関係を基に，目的関数自体を学習する手法を提案する．学習局面の特徴に基づいて目的関数自体を学習することで,「どのような特徴の学習局面を，どのように学習すれば強い評価関数が得られるか」を自動的に獲得することを目的とする．

本論文では，実験対象のゲームとして将棋を用いる．将棋は，1997年にIBM Deep Blueがチェスで世界王者を破って以来，囲碁とともに次の大きなターゲットとしてゲーム分野における研究対象とされてきた．将棋には取った駒を再利用できる等の性質があり，

チェスライクなゲームでは複雑な部類に入る．コンピュータ将棋は10年ほど前までは，アマチュア高段者程度の棋力であったが，近年の機械学習の成功により，現在では人間のトッププレイヤに迫る強さとなっており，棋譜を教師とした機械学習の恩恵を最も強く受けたゲームの1つと言える．本論文では，将棋を題材に，評価関数の学習，探索深さの調整，モンテカルロ木探索のシミュレーションの学習，といった現在ゲームの分野で成功している各種学習アルゴリズムに対して提案手法を適用し，その効果を検証する．

1.4　本論文の構成

本論文の構成を以下に示す.

第2章では，新たなゲームのアルゴリズムである，モンテカルロ木探索を将棋へ適用し，その可能性を検証する．モンテカルロ木探索は，囲碁では大きな成功を収めたものの，その他のゲームにおける効果の検証は十分ではない．第2章では，まず，モンテカルロ木探索が将棋に対して有効かという根本的な課題についての検討を行う．

第3章では，学習棋譜の性質が強さに与える影響を考慮した学習を提案する．具体的には教師データに重要度を付与し，対局結果から算出した強さを基に重要度の学習を行う．提案手法では，重要度を用いることで，強さに寄与する教師データを重視した学習を実現する．実験では，提案手法を現在のゲームの主要なアルゴリズム（探索，評価関数，および第2章で検討したモンテカルロ木探索）へ適用し，提案手法の有効性を検証する．

第4章では，第3章の手法の拡張として,「強い」プログラムを学習するための目的関数の学習を提案する．学習局面の特徴に基づいて目的関数自体を学習することで，現在の学習手法の課題である，目的関数と強さの関係が不明となっている点，学習局面の特徴に応じた学習がなされていない点，を改善することを目的とする．実験では，機械学習が特に成果を挙げている，評価関数の学習を対象に提案手法を適用し，その有効性を検証する．

最後に，第5章で本論文のまとめについて述べる．

第2章　モンテカルロ木探索によるコンピュータ将棋

本章では，ゲームの新しい思考アルゴリズムであるモンテカルロ木探索を将棋へ適用し，その可能性を検証する．従来，思考ゲームをコンピュータで実現する際には，探索と評価関数を用いた手法が一般的であった．一方，近年，コンピュータ囲碁において，モンテカルロ木探索（Monte-Carlo Tree Search）という新たな思考アルゴリズムが提案され，大きな注目を集めている．この手法の特徴は，乱数を用いたゲームのシミュレーションの勝率によって局面を評価するというものであり，評価関数が不要という利点がある．モンテカルロ木探索は，囲碁以外のゲームへの適用も研究されているものの，チェスライクなゲームにおける有効性を検証した報告はこれまで多くない．本章では，モンテカルロ木探索による将棋を実装し，囲碁で成功した手法を基にいくつかの将棋向けの改良を行い，モンテカルロ木探索の将棋への適用可能性について検証する．

2.1　はじめに

将棋，チェス，オセロといった思考ゲームをコンピュータで実現する際には，探索と評価関数を用いるのが一般的である．一方，現在注目を集めている手法として，モンテカルロ木探索（Monte-Carlo Tree Search）[3, 36] がある．この手法の最大の特徴は，評価関数が不要となることであり，その設計が困難とされている囲碁では従来の手法を上回る成功を収めた．近年では，囲碁以外のゲームへのモンテカルロ木探索のゲームへの適用についても盛んに研究が行われている．モンテカルロ木探索の適用が研究されているゲームの例としては，チャイニーズチェッカー[7]，Amazons[8]，ヘックス[9]，Lines of Action[10]，Kriegspiel[11]などのゲームが挙げられる．

コンピュータ将棋は，囲碁に比べ評価関数が設計しやすいことや，近年の機械学習の成功から，従来の手法により人間のトッププレイヤに迫る性能を得ている．しかし，将棋においても万能な評価関数の設計は難しく，複雑な終盤や長期的な見通しが必要となる序盤の指し手選択には課題が存在する．モンテカルロ木探索は，これらの問題点を解決できる可能性があり，将棋の分野でも注目を集めている[37, 38, 39]ものの，従来良い結果を得ることには成功していなかった．モンテカルロ木探索によりコン

ピュータ将棋を実現する際の特有の問題としては，ランダムな指し手の選択によるシミュレーションでは終局条件（詰み）を満たすことが難しい，ゲームの性質として囲碁よりも読みに重点が置かれる，といった点が挙げられる．

本章では，モンテカルロ木探索のアルゴリズムとしてUCT（Upper Confidence bounds applied to Trees）[40] を採用したコンピュータ将棋を実現する．さらに，最近のモンテカルロ囲碁で成功した手法を元に，いくつかの将棋向けの改良を行うことで前述した問題点の改善を試みる．また，定跡選択部や探索と評価関数による手法では良い結果を得ることができなかった局面におけるモンテカルロ木探索の有効性について検証を行う．

以降の本章の構成を以下に示す．2.2節では，囲碁・将棋を中心にモンテカルロ木探索に関する関連研究を述べる．2.3節では，モンテカルロ木探索によるコンピュータ将棋の実現と将棋向けの改良について述べ，2.4節の実験でその有効性を検証する．実験では，問題集や対局実験による評価の他，現在のプログラムが不得意とする局面におけるモンテカルロ木探索の有効性についても検証する．2.5節で今後の展望について述べ，最後に2.6節で本章のまとめを述べる．

2.2　関連研究

思考ゲームを実現する際の最も一般的な手法は探索と評価関数である．探索は人間でいうところの「読み」，評価関数は「形勢判断」「大局観」といったものに対応する．

コンピュータ将棋の場合にも，多くのプログラムは探索と評価関数に基づいており，現在ではこの手法に基づくプログラムが，人間のトッププレイヤに迫る強さを得ている[41, 42, 43, 44, 45, 46].

しかし，将棋の場合にも，人間の知識をコンピュータに正しく反映させることは容易ではなく，複雑な終盤や長期的な見通しが必要となる局面では最善手が選択できない場合が存在する．また，コンピュータの場合，人間のように指し手を絞った深い読みはできない．また，現在のトップレベルのプログラムは，見かけ上は20手以上の深さまで探索しているものの，重要な指し手が途中で枝刈りされてしまっている場合も多い．このようなことから，現在のコンピュータ将棋は1秒間に数百万局面を探索するにもかかわらず，局面によっては人間のトッププレイヤに読み負けるケースが存在する．

一方で近年，探索と評価関数とは全く異なる考え方に基づく，モンテカルロ木探索という手法が注目を集めている．この手法の基となる，モンテカルロ法によるゲームの実現は，コンピュータ囲碁では1993年から研究されていた[47]．モンテカルロ法による思考では乱数を利用したゲームのシミュレーション（以下playout）を何度も行い，そのスコア（石の数の差，勝率など）によって局面を評価する．モンテカルロ法によるゲームの最大の特徴は，人間の知識に基づいた評価関数を用いる必要がないことであ

る．そのため，評価関数の設計が特に困難とされていた囲碁において大きな注目を集めた．しかし，相手のミスを期待した手を選択してしまう，ある程度以上playoutの回数を増やしても棋力が頭打ちになってしまう，といった問題があり従来の手法を上回るには至らなかった[48].

モンテカルロ法に木探索を組み合わせることによりこれらの問題点を大幅に改善した手法がモンテカルロ木探索である．モンテカルロ木探索の出現により，アマチュア級位者レベルであったコンピュータ囲碁は，9路盤ではプロに勝利するまでの強さになった．モンテカルロ木探索の最も一般的なアルゴリズムとしては，UCT（Upper Confidence bounds applied to Trees）[40]が挙げられる．UCT は，その時点までに行ったplayoutの回数と勝率に基づき，より有望な指し手（ノード）に多くのplayout を割り当て，探索木を成長させていく．Playout中の指し手の選択は，完全なランダムでは良い性能を得ることは難しく，多くの囲碁プログラムでは，ゲームの知識を利用して確率的に行っている[33, 49, 34, 50].

モンテカルロ法によるコンピュータ将棋の研究としては文献[37]が挙げられる．この研究では，囲碁プログラムCRAZY STONE[3]のアルゴリズムを将棋へ適用し，モンテカルロ法によるコンピュータ将棋の可能性を考察している．Playoutに長手数を要するという問題に対しては，最大手数を10程度に制限し，末端で評価関数を利用することにより解決を試みている．しかし，問題集の正答率は3%程度にとどまっており，有望とは言いがたい結果となっている．

本章では，探索手法としてUCTを採用したコンピュータ将棋を実現する．さらに，キラームーブ[51]やヒストリーヒューリスティック[52]の利用，囲碁において成功しているprogressive widening[33]を将棋向けに適用，並列化などを行うことにより性能の改善を試みる．また，playout 部分では最大手数を極端に制限することや評価関数を利用するといったことは行わず，ヒューリスティックの導入により自然な終局の実現を目指す.

2.3　モンテカルロ木探索によるコンピュータ将棋

本章では，まずモンテカルロ木探索とUCTについての説明を行う．その後，モンテカルロ木探索によるコンピュータ将棋を実現する際に行った，playout部分と探索部分の改良についてそれぞれ説明する．また，定跡選択におけるモンテカルロ木探索の利用，モンテカルロ木探索と静止探索の併用といった応用的な手法についても説明する.

2.3.1　UCT（Upper Confidence bounds applied to Trees）

モンテカルロ木探索の基本的な考え方は次の通りである.

1. 初期状態ではランダムに候補手を選びplayoutを繰り返し行う.
2. 徐々に（勝率が高い手など）有望な手で多くのplayoutを行っていく.
3. Playoutの回数が閾値を超えたノードでは探索木を成長させ，playoutを開始するノードを1段深くする.

この操作を繰り返すことで，有望なノード（指し手）ほど探索木が成長し，深く探索されることになる．ルートでは，勝率の最も良い手，あるいはplayoutが最も多く行われた手を最善手として選択する.

モンテカルロ木探索の最も一般的なアルゴリズムとしては，UCTが挙げられる．UCTでは，playoutを割り当てる子ノードの決定に以下の式，UCB1[53]を用いる．合法手がK手存在するとき，playoutを行う子ノード$I \in \{1, ..., K\}$は以下の式により決定される.

$$I = \underset{i \in \{1,...,K\}}{\operatorname{argmax}} \left\{ X_i + c\sqrt{\frac{2 \log n}{n_i}} \right\} \tag{2.1}$$

X_iはノード（指し手）iを選択した場合の勝率，n，n_iはそれぞれ親ノード，ノードiの訪問回数を示す．cの値は通常は1.0を用いる[36]．cの値を1.0より大きくした場合には勝率の低い手に，小さくした場合には勝率の高い手により多くのシミュレーションが割り当てられることになる.

このような式を用いることにより，UCTでは有望な手で多くのplayoutを行い深く探索する，といった手法を実現している.

2.3.2　Playoutの改良

モンテカルロ木探索では，playout 中の指し手の選択の方策が，プログラムの性能に大きな影響を及ぼすことが知られている．具体的には，playout 中の指し手の選択は完全にランダムでは良い性能を得ることは難しく，乱数を利用しつつも，何らかの形でヒューリスティックを導入し，対象とするゲーム（囲碁，将棋）らしい指し手の選択を行う必要がある．さらに，モンテカルロ木探索によりコンピュータ将棋を実現する際の最大の問題として，ランダムな指し手の選択によるシミュレーションでは終局条件（詰み）を満たすことが難しい点が挙げられる．モンテカルロ木探索におけるコンピュータ将棋を実現する際には，将棋らしいシミュレーションを実現しつつ，終局条件の課題をクリアする必要がある.

本章では指し手の選択に，Eloレーティングを利用した指し手の予測[33, 54]を用いた．Eloレーティング[55]は，本来，ゲームにおいて，勝敗からプレイヤの強さを推定する手法として用いられる．この予測手法では，指し手をいくつかの特徴の集合とみ

なし，プロの棋譜などを元にその選択されやすさをEloレーティングとして数値化し，その値に基づいて指し手の選択確率を決定するというものである．

この手法は，本質的には，ロジスティック回帰による指し手の予測[32]と同様のモデルによって，指し手の選択確率を算出しているが，モンテカルロ木探索において，初めて本格的な知識導入を行った手法であることから，現在でも良く用いられる手法となっている．

2.3.2.1　Bradley-Terryモデルと Eloレーティング

Bradley-Terryモデル[56]はある試合に関する勝敗を予測するモデルである．あるプレイヤiがγ_iという正の値（レーティング）を持つとすると，1からn人までのプレイヤの中で$i(1 \leq i \leq n)$が勝利する確率は，

$$P(i\text{が勝つ}) = \frac{\gamma_i}{\sum_{j=1}^{n} \gamma_j} \tag{2.2}$$

と表わされる．複数プレイヤによるチームでの試合は，チームを組むメンバの強さγ_iの積をチームの強さとすることで扱うことができる．プレイヤ$1, 2, 3, 4, 5, 6$が存在し，1-2-3, 4-5, 6というチームを組むとき，チーム4-5が試合に勝つ確率は，

$$P(\text{チーム4-5が勝つ}) = \frac{\gamma_4\gamma_5}{\gamma_1\gamma_2\gamma_3 + \gamma_4\gamma_5 + \gamma_6} \tag{2.3}$$

とモデル化される．一般的には，γ_iを$400\log_{10}\gamma_i$と変換したものをEloレーティングと呼ぶが，ここでは変換前の値γ_iをレーティングの値として扱う．将棋の指し手には,「駒の損得」,「王手」や「逃げる手」など様々な特徴がある．これらの特徴を個々のプレイヤとみなし，各特徴の選択されやすさをプレイヤの強さ（レーティング）と考えることで，上記のモデルを適用できる．この場合，指し手は複数の特徴からなるチームと考えることができる．このモデルを利用してplayout中の指し手の選択を行うことを考えた場合，指し手の選択確率はチームが勝つ確率と対応する．

各特徴のレーティングは，プロの棋譜を学習データとして算出する．棋譜中の各局面を1つの試合とみなし，実際に指された手を勝ったチーム，指されなかった手を負けたチームと考えることで，実際に指された手のレーティングが高くなるように，各特徴のレーティングを算出する．

指し手のEloレーティングの学習には，Minorization-Maximizationアルゴリズム[33]を用いる．Minorization-Maximizationアルゴリズムでは，指し手の特徴のEloレーティングγ_iを以下の式の反復計算によって求める．

$$\gamma_i \leftarrow \frac{W_i}{\sum_{j=1}^{N} \frac{C_{ij}}{E_j}} \tag{2.4}$$

ここで，Nを学習局面の数，W_iを特徴iを持つ手が選択された回数，C_{ij}を局面jにおける特徴iが属する指し手の他の特徴のレーティング，E_jを全指し手のレーティングを意味する．

2.3.2.2　用いた特徴

本研究において用いた主な特徴を以下に示す．それぞれの特徴について，さらに詳細な分類をしており，実際には200程度の特徴（項目）を用いている．

- 駒の損得（SEE）
- 駒を取る手
- 成る手
- 王手
- 相手の利きのある駒の移動（逃げる手）
- 位置テーブルの値の増減（駒を打つ場合には，駒を打つ位置のテーブルの値）
- 玉の位置，周囲の利き
- 直前および二手前に移動した駒との関係

駒の損得はSEE（Static Exchange Evaluation）[57]により求めた．駒の損得は表 2.1に示す値を用いて計算したSEEの値が，100点区切りでどの区分に属するかを特徴とした．駒の損得（SEE）の計算は，他の特徴に比べコストがかかるが，将棋において特に重要な特徴と考えられる．

表 2.1: 駒の交換値

歩	香	桂	銀	金	角	飛
100	280	300	420	530	620	700
と	成香	成桂	成銀	-	馬	龍
270	320	250	430	-	710	850

−62	−108	−64	−70	−88	−98	−88	−124	−98
−116	−112	−4	4	−22	−28	−6	−74	−94
−110	−46	−50	−26	−22	−2	−28	−44	−76
−64	−40	0	−16	10	−18	−8	−4	−50
−98	−52	−26	−18	−22	−14	−20	−26	−60
−78	−38	−20	−26	−20	−18	−14	−44	−66
−104	−20	2	18	−48	−22	−52	−64	−100
−120	X	46	−22	−2	−34	−52	−80	−162
−116	6	−58	8	−82	−30	−86	−74	−160

図 2.1: 玉が 8 八にいる場合の金の位置による点数

位置テーブルの値は文献[31]の手法により求めたものを用いた．図 2.1は玉が 8 八にいる場合の味方の金の位置による点数である．矢倉囲いや美濃囲いの金の位置の点数が高くなっており，玉から遠くなるほど点数が低くなっていることがわかる．このような表を自玉，相手玉がそれぞれ 1 一から 9 九にある場合について作成している．

前述した，位置テーブルの値の増減は，(移動先のテーブルの値 − 移動元のテーブルの値) が，−101以下，−100 〜 −11，−10 〜 −1，0，1 〜 10，11 〜 100，101以上のどの区分に当てはまるかを特徴としている．なお，駒を打つ場合には，単に打つ位置のテーブルの値がどの区分に当てはまるかを特徴としている．

2.3.3　UCTの改良

UCTの問題点としては，訪問回数の少ないノードにおいて効率の良いplayoutの割り当てができず，ランダムに近い子ノードの選択を行ってしまう点が挙げられる．本章では，キラームーブやヒストリーヒューリスティックの利用，progressive wideningなどによりこの問題を改善した．

2.3.3.1　キラームーブ，ヒストリーヒューリスティックの利用

本章では，UCTの子ノードの選択にキラームーブ[51]及びヒストリーヒューリスティック[52]を利用した．ともにコンピュータチェスやコンピュータ将棋において良く用いられている手法で，キラームーブは兄弟ノードの最善手，ヒストリーヒューリスティックはある指し手が最善手であった割合（回数）を示す.

囲碁では，キラームーブやヒストリーヒューリスティックに近い考え方を用いた，RAVE[58]と呼ばれる手法が成功を収めている．

将棋の場合には，ゲームの性質上多くの兄弟ノードでは最善手が同一であることが多く，キラームーブは特に重要な概念であると言える．実際，多くの将棋プログラムで

は，指し手の並べ替えなどにおいてキラームーブを利用しており[59]，これらの知識を利用することは，囲碁と比較した場合においても，高い効果を得ることが期待できる．

本研究では，キラームーブやヒストリーヒューリスティックの値が大きいノードでは，式 (2.1)におけるcを1.0よりも大きな値に設定する．具体的には，cの値を，キラームーブとなっている指し手では2.0，その他の指し手では以下の式により決定する．

$$c = 1.0 + \max(0, H_i - 0.6) \tag{2.5}$$

ここで，H_iはヒストリーヒューリスティックの値を意味し，その指し手が探索中で最善手であった割合を示す．

以上のような変更を加えることで，訪問回数が少ない間はキラームーブやヒストリーヒューリスティックの値が大きいノードが優先的に選択されることになる．

2.3.3.2 Progressive widening

Progressive widening[33]とは，新しいノードを作成する際，一度にすべての合法手を探索対象とするのではなく，ヒューリスティックを利用してよさそうな指し手から探索対象を広げていくという，枝刈り手法の一種である．

本章では，未訪問のノードでは式 (2.1)における勝率の値X_iの部分に，指し手のレーティングの値R_iを利用した以下の式を用いることでprogressive wideningを実現した．

$$U_i = \begin{cases} dR_i + c'\sqrt{\frac{2\log n}{e}} & (n_i = 0) \\ X_i + c\sqrt{\frac{2\log n}{n_i}} & (n_i \neq 0) \end{cases} \tag{2.6}$$

$$I = \underset{i \in \{1,\ldots,K\}}{\operatorname{argmax}} \{U_i\} \tag{2.7}$$

R_iの値は以下の式により指し手iのレーティングを0から1.0の値に正規化したものである．定数c', d, eの値は実験的に0.5, 4.0, 10.0とした．

$$R_i = \frac{\text{指し手}i\text{のレーティング}}{\text{合法手中の最大のレーティング}} \tag{2.8}$$

このような式を用いることで，レーティングの高い指し手から順に探索対象を広げていく．単純に駒得する手などレーティングが突出して高い手がある場合には，その手が特に重視されることになる．また，勝率の値X_iが高い指し手があるうちは，探索対象は広がりにくく，強い枝刈りとなる．逆に現在の探索対象の指し手の勝率X_iが低い場合には，速やかに探索範囲を広げることになる．

本章で用いたprogressive wideningは，囲碁の例[33]と比べると，比較的弱い枝刈りとなっている．このような枝刈りを用いた理由は，将棋の場合，一見駒損をするような手でも，読みを入れることにより実は好手であることがわかるといった指し手が多く，また，囲碁ほどは探索空間が広くないためである．

2.3.3.3　探索内部での詰み情報の利用

その他のUCTの大きな問題点としては，厳密な勝敗の証明に非常に多くのコストがかかるという点が挙げられる．将棋では，詰みは一手で勝敗を決める決定的な要因となるため，この問題は囲碁と比較して深刻である．

本章では，この問題への対策として，探索木の内部で見つけた詰み情報を利用している．探索中で詰みを見つけた場合には，シミュレーションを行うことなくスコア（勝敗）を求めることができる．また，詰みの情報を親ノードに伝えることで，勝敗が決定したノードでは，以降無駄なシミュレーションを行わないようにしている．モンテカルロ木探索の問題点の1つとして，厳密な勝敗の証明に大きなコストがかかるという点が挙げられるが，探索木の内部で見つけた詰み情報を利用することで，この問題を大幅に改善できると考えられる．

2.3.4　並列化

モンテカルロ木探索では，playoutの回数は棋力に大きな影響を及ぼす．playoutの部分は独立性が高いことから，並列化は有効である．

本章では，探索木の部分を共有した状態で，スレッドを用いplayoutを独立に実行する方法を採用した．具体的には，並列にplayoutを実行し，UCTの探索木の更新中のみ，更新中のノードをロックする．この方法では，探索木を更新するタイミングの関係上，単一スレッドで実行した場合とは異なる振る舞いをすることになるが，4CPU程度の並列化においては十分な性能向上が得られることが知られている[49]．

2.3.5　モンテカルロ木探索を用いた定跡選択

コンピュータ将棋において，定跡選択は，その後の優劣を決定する上で重要な部分であるが，この点に関する研究は少ない．現在多くの将棋プログラムは，定跡選択の部分に力を注いるとは言いがたく，定跡データベース（ここでは実践譜の集合とする）との一致のみで選択しているプログラムも多い．比較的良く用いられる手法としては，データベースとの一致数を見て，確率的に指し手を選択する手法が挙げられる．この

手法では，定跡データベース中のある候補手mが選択される確率を以下の式により決定する（以下確率による定跡選択と呼ぶ）．

$$P(m\text{が選択される}) = \frac{m\text{が指された回数}}{\text{現局面がデータベースに存在する数}} \tag{2.9}$$

この手法は，多くの将棋プログラムで用いられている一般的な手法と言えるが，データベースの指し手を絶対的に信頼し，コンピュータ自体は全く思考していないため，悪手を選択してしまうことも多い．また，プログラムがあまり得意でない展開も選択してしまうといった問題点も存在する．

上記の問題を改善するために，事前に膨大な対局実験を行い，勝率の低い定跡をデータベースから除外するといったことも行われるが，問題が完全に解決できているとはいえない状況となっている．

本章では，モンテカルロ木探索を用いた定跡選択を実装し，その有効性を検証した．モンテカルロ木探索を用いた定跡選択部の手順を以下に示す．

1. 定跡データベースに含まれる各棋譜を1回のplayoutとしたUCTの探索を行う．ただし，子ノードを生成する際にはすべての指し手を生成するのではなく，定跡データベース中に存在する指し手のみを生成する．

2. 1.で構成した探索木の上で，通常のUCTによる探索を行う．

定跡選択部分では，長期的な見通しが必要となること，また，ある程度有効な指し手が絞られているため，無駄な手にplayoutを割り当てる回数も抑えることができることから，モンテカルロ木探索の利点を活かすことができると期待できる．

2.3.6　静止探索との組み合わせ

将棋は，囲碁と比べて読みが重視されるゲームである．モンテカルロ木探索は，終局までのplayoutを何度も行いながら，徐々に探索木を成長させていくという性質上，駒の取り合いなどの直線的な読みは苦手と考えられる．このような問題から，モンテカルロ木探索では，特に思考時間が短い場合，駒の損得を正しく評価できない可能性がある．

本研究では，この問題の改善策として，静止探索[22]を用いた駒の損得評価を行うことを検討する．静止探索との組み合わせでは，駒の価値は表 2.1を用い，モンテカルロ木探索の評価が同等の指し手では，駒損しない手を優先して選択する．

2.4 実験

本章の実験では，提案手法の有効性を検証するため，(1)Eloレーティングを利用した際のplayoutの性質，(2)問題集による評価，(3)対局実験による評価，(4)プロの棋譜との一致率，(5)モンテカルロ木探索を用いた定跡選択の評価，(6)現在のプログラムが正解できない局面におけるモンテカルロ木探索の有効性の評価，を行った．

実験には，特に断りがない場合，表 2.2に示す環境を用いた．

表 2.2: 実験環境

CPU	Quad-Core Xeon X5355 2.66GHz × 2（８コア使用）
メモリ	2GB

2.4.1 Eloレーティングを利用した際のplayoutの性質

3.1節で述べた指し手の特徴のレーティングを表 2.3に示す．学習用の棋譜としては，名人戦の棋譜約300局を用い，各特徴のレーティングを算出した．

レーティングの値が大きいほどより選択されやすい特徴となる．表 2.3から，駒の損得や直前の駒の取り返し，王手などは特に強い特徴となっており，妥当なレーティングが算出できていると言える．位置テーブルの値の増減では，飛角より金銀などの小駒の方が，玉との位置関係が重視されていることがわかる．

表 2.4に指し手の選択の方策によるplayoutの性質の違いを示す．予測率は評価用の棋譜200局において実際に指された指し手に割り当てた確率の平均，終局率は平手の初期局面においてplayoutを10,000回行ったとき，256手以内に終局した割合を示す．速度は平手の初期局面において1秒間に実行できるplayout回数を計測したものである．実験環境はXeon X5355（1コアのみ使用）とした．

表 2.4から，playout中の指し手の選択にEloレーティングを導入することにより，速度は4分の1程度となっているものの，予測率が大きく向上していることがわかる．また，終局率の向上から，終局条件を満たすことが難しいという将棋特有の問題点を改善できていることがわかる．

なお，以降の実験において，終局しなかったplayoutは0.5勝0.5敗として扱っている．評価関数を用い，ヒューリスティックにより勝敗を決定する[37]といった手法も考えられるが，評価関数が不要というモンテカルロ木探索の利点を生かすため，本研究ではそのような手法はとっていない．

表 2.3: 指し手の特徴のEloレーティング

特徴	詳細	γ_i
駒の損得(SEE)	飛程度の得	20.04
	角程度の得	14.37
	金程度の得	9.54
	銀程度の得	6.08
	桂香程度の得	3.89
	歩程度の得	2.55
	損得なし	1.11
	歩程度の損	0.47
	桂香程度の損	0.32
	銀程度の損	0.10
	金程度の損	0.06
	角程度の損	0.05
	飛程度の損	0.02
駒を取る手	取り返し	12.75
	その他	1.88
王手		7.55
成る手		1.47
相手の利きのある駒の移動	歩	1.13 – 1.27
	香	1.90 – 3.19
	桂	1.70 – 2.59
	銀	2.20 – 2.60
	金	4.38 – 4.78
	角	1.77 – 5.12
	飛	6.85 – 23.68
	馬	3.26 – 9.98
	龍	8.53 – 14.62
	その他の成駒	0.82 – 1.65
位置テーブルの値の増減	歩	1.04 – 2.19
	香	0.16 – 0.90
	桂	0.57 – 2.25
	銀	0.60 – 3.00
	金	0.33 – 2.22
	角	0.66 – 1.45
	飛	0.38 – 1.00
	馬	0.78 – 1.49
	龍	0.66 – 0.99
	その他の成駒	1.05 – 2.21
位置テーブルの値（打つ場合）	歩	0.50 – 2.21
	香	0.24 – 0.97
	桂	0.17 – 1.66
	銀	0.19 – 1.35
	金	0.28 – 1.45
	角	0.16 – 1.17
	飛	0.22 – 1.89
玉移動（位置，周囲の利き）	通常時	0.30 – 1.01
	王手時	0.57 – 4.21
直前に移動した駒の移動		0.90 – 1.55
直前の駒との位置関係		0.89 – 1.97
二手前の駒との位置関係		0.94 – 1.86
直前の位置に戻る（序盤のみ）		0.10 – 0.87
...		...

表 2.4: 指し手の選択の方策によるplayoutの性質

指し手の選択	予測率	終局率	速度
ratingなし（完全乱数）	0.037	0.193	約3500回/秒
ratingあり	0.172	0.905	約 900回/秒

2.4.2 問題集の正答数による評価

モンテカルロ木探索によるコンピュータ将棋において，各改良がどの程度有効かを検証するために，問題集による評価を行った．問題としては，文献[60, 61]の98題を用い，解答時間は1問30秒とした．結果を表 2.5に示す.

表 2.5において，MC/UCTは単純なモンテカルロ木探索（カッコ内はスレッド数），checkはUCT内部での詰み情報を利用した場合，ratingはplayout中の指し手の選択にEloレーティングを用いた場合，PWはprogressive wideningを導入した場合，historyはキラームーブ，ヒストリーヒューリスティックを利用した場合を示す.

参考として，探索と評価関数に基づくアマチュア初段程度のプログラム「遠見」，およびアマチュア三段程度のプログラム「棋理」の正答数も示している.「遠見」「棋理」は詰探索ルーチンを持っているため，詰将棋を除いた問題86題の正答数による比較も行った.

表 2.5: 問題集の正答数

用いた手法(カッコ内はスレッド数)	正答数	除詰将棋
MC/UCT(1)	4 / 98	4 / 86
MC/UCT(8)	9 / 98	8 / 86
MC/UCT(8) + check	12 / 98	11 / 86
MC/UCT(8) + check + rating	36 / 98	32 / 86
MC/UCT(8) + check + rating + PW	41 / 98	38 / 86
MC/UCT(8) + check + rating + PW + history	49 / 98	45 / 86
遠見	60 / 98	48 / 86
棋理	71 / 98	59 / 86

表 2.5から，各手法を適用することによって，問題集の正答数が向上していることがわかり，モンテカルロ木探索によるコンピュータ将棋において有効な改良となっていることがわかる．中でも，特にEloレーティングを利用した指し手の選択の方策の改良は特に効果が大きく，囲碁同様，将棋においてもplayout中の指し手の選択の方策はモンテカルロ木探索の性能を決定する上で非常に重要な要素となっていると言える.

モンテカルロ木探索は，詰将棋を除いた問題では探索と評価関数によるアマチュア初段程度のプログラムに迫る正答数を得た．詰将棋などの問題では，正確な読みが必要となるためモンテカルロ木探索には適していないと考えられる．

なお，以降の実験において,「モンテカルロ木探索」と表記した場合には，表 2.5中の「MC/UCT(8) + check + rating + PW + history」を用いたことを示す.

2.4.3 「遠見」との対局による評価および静止探索の効果

探索と評価関数によるプログラム「遠見」との対局により，モンテカルロ木探索によるプログラムの評価を行った．結果を表 2.6に示す（太字は有意水準5%の二項検定で有意な結果）．思考時間はともに1手10秒，対局数は200局とした．なお，本実験では，お互いにルート局面で詰み探索を行い，即詰みが見つかった場合には投了するものとした．

表 2.6: 「遠見」との対局結果

用いた手法	勝率
モンテカルロ木探索	**0.04**
モンテカルロ木探索+静止探索	**0.32**

表 2.6の対局結果から，モンテカルロ木探索は，問題集の正答数では「遠見」とほぼ同等であったにもかかわらず，対局の結果では大きく負け越していることがわかる．

その理由として，モンテカルロ木探索の問題点として小さな駒損が多いことが挙げられる．特に序中盤の歩損などは，playoutの勝敗には結び付きにくく，意味のない歩の突き捨てなどを指してしまうことが多い．また，不利になるほど，相手のミスによる逆転に期待した悪手を指しやすくなるといった傾向もある．

一方，問題集のような戦術的な局面では，多少の駒損などは問題にならないことが多い．このようなことから，問題集の正答数による評価と対局結果による評価に大きな差が出たと考えられる．

「モンテカルロ木探索+静止探索」は，モンテカルロ木探索に静止探索を組み合わせ，評価値が互角の場合には駒損しない手を選択するようにしたものである．表 2.6の結果より単純なモンテカルロ木探索よりも勝率を改善できていることがわかる．

しかし，依然アマチュア初段程度のプログラムに達することはできておらず，モンテカルロ木探索により単純に従来の手法を上回ることは難しいと考えられる．

2.4.4　プロの棋譜との一致率

モンテカルロ木探索を用いた場合のプロの棋譜との一致率を局面の進行度別に計測した．k手で終局する棋譜中のn手目の局面の進行度は以下の式で示す値としている．

$$\text{進行度}(k,n) = \frac{n}{k} \tag{2.10}$$

図 2.2及び図 2.3に進行度別のプロの棋譜との一致率を示す．棋譜中の指し手が，モンテカルロ木探索の最善手と一致した割合，上位3手に含まれていた割合，参考として「棋理」の最善手との一致率も示している．

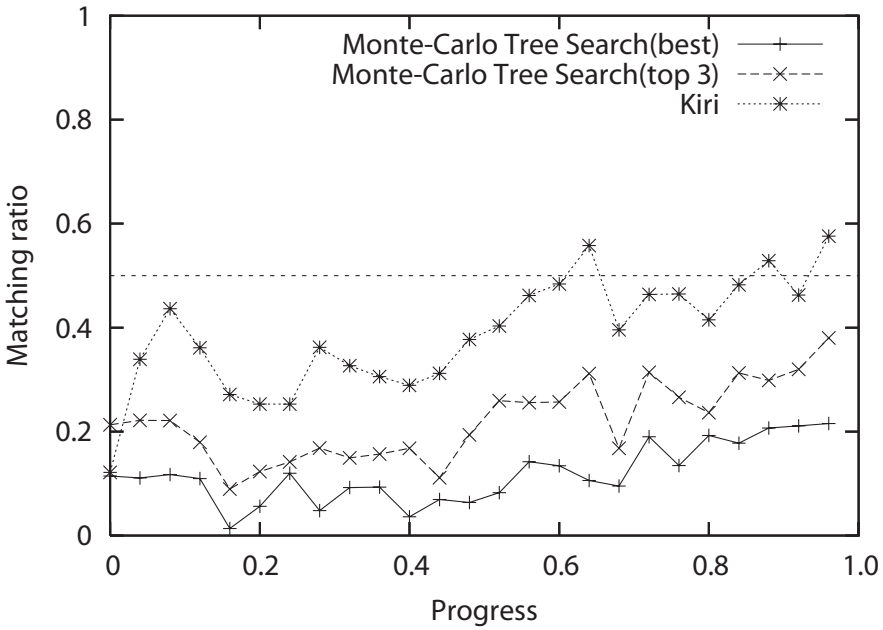

図 2.2: プロの棋譜との一致率（1手1秒の場合）

図 2.2は思考時間を1手1秒とした場合の一致率を示したものである．モンテカルロ木探索の一致率が非常に低くなっていることがわかる．これは1手1秒程度では十分なplayoutを行うことができていないためであると考えられる．一方，探索と評価関数によるプログラムでは思考時間が短い場合にも，ある程度の一致率を得ることができていると言える．

図 2.3は思考時間を1手10秒とした場合の一致率を示したものである．1手1秒の場合に比べ，モンテカルロ木探索の一致率が大きく向上していることがわかる．一定時間内により多くのplayoutを行うことができれば，性能はさらに向上すると考えられる．

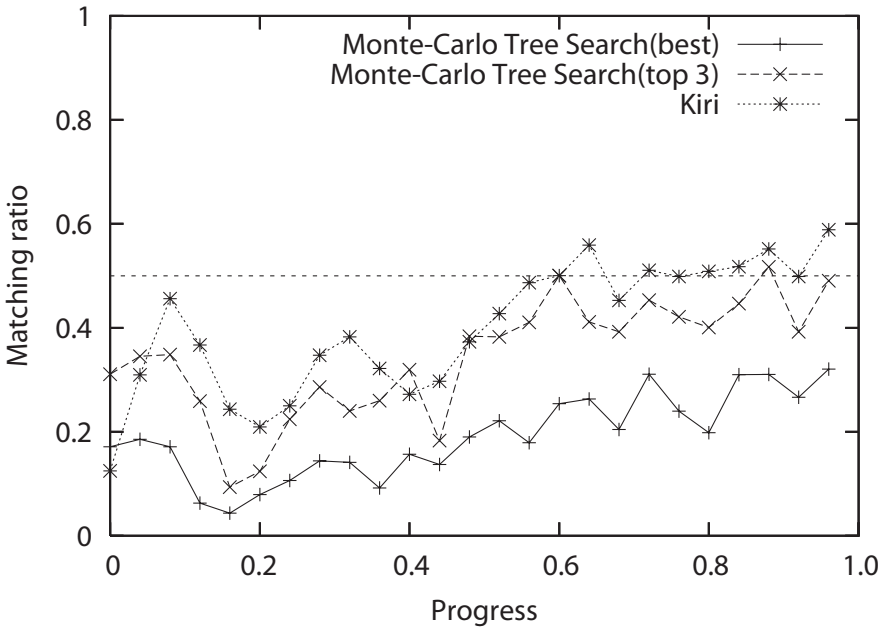

図 2.3: プロの棋譜との一致率（1手10秒の場合）

序盤から終盤にかけて徐々に一致率が向上している理由としては以下のような点が考えられる．

- 序盤の場合には，そもそも最善手を1つに決定することが難しい
- 終局（詰み）が近いほど時間あたりに行えるplayoutの回数が増加する
- 終局が近いほど指し手の善悪がplayoutの結果（勝敗）に反映されやすくなる

また，モンテカルロ木探索がプロの棋譜と一致した局面において,「棋理」がその手を選択した割合は約0.58となった（思考時間は1手10秒）．一致率の差から考えると低い値となっており，モンテカルロ木探索と探索と評価関数による手法は得意とする局面に違いがあると考えることができる．

2.4.5　モンテカルロ木探索を用いた定跡選択

モンテカルロ木探索を用いた定跡選択の評価として，定跡選択に他の手法を用いた場合との自己対局を行った．対局数は200局とし，定跡打ち切り後の思考部には「棋理」を用いた．定跡選択部，通常の思考部ともに思考時間は1手2秒とし，定跡データベースには，プロやアマチュア高段者の棋譜約3万局のうち手番側が勝ちの棋譜を用い

た．なお，定跡選択において，完全に同一の手順が選択された場合には，その対局は無効とし，200局の中にはカウントしていない．

結果を表 2.7に示す（太字は有意水準5%の二項検定で有意な結果）．比較手法のProbabilityは確率による定跡選択，Maximum Frequencyは定跡テータベース中で最も良く現れた指し手を選択する手法，Searchは通常の探索を行い定跡データベース中に存在する手の中での最善手を選択する手法である．

表 2.7: 定跡選択にモンテカルロ木探索を用いたプログラムの対局結果

比較手法	勝率	評価値
Probability	**0.61**	−2.01
Maximum Frequency	**0.57**	+8.10
Search	**0.58**	−68.24

実験の結果，いずれの手法と比較した場合にもモンテカルロ木探索を用いた定跡選択が勝ち越した（すべて有意水準5%の二項検定で有意）．通常の探索部分や評価関数との相性の関係もあるため，モンテカルロ木探索を用いた定跡選択がすべてのプログラムに対して優れていると言い切ることはできないが，有力な手法の1つと考えられる．

評価値は，定跡が打ち切られた時点での局面における「棋理」の評価関数の値の平均である．定跡が打ち切られた時点での評価値は，歩の価値が100点であることを考慮すると，ほぼ互角と言える．勝率と比較すると，序盤の優劣を評価関数で正しく判断することの難しさが示されていると言える．

また，定跡選択のアルゴリズムに求められる要件として，ある程度選択される指し手にばらつきがあることが望ましいと言える．実験として，モンテカルロ木探索を用いた定跡選択を初期局面から乱数の初期値を変えて100回行ったところ，32パターンの異なる手順を選択した．この結果から，モンテカルロ木探索による定跡選択では，ある程度ばらつきのある指し手の選択を実現することもできていると考えることができる．

2.4.6　現在のコンピュータ将棋の問題点とモンテカルロ木探索の利点

前節までの実験結果では，定跡選択では一定の成果をあげたものの，通常探索部分での性能では探索と評価関数によるプログラムには及ばないという結果になった．しかし，探索と評価関数による手法にも，いくつか問題点があることがわかっており，そのような局面ではモンテカルロ木探索が有効である可能性がある．

本節では，2007年3月に行われた渡辺竜王対Bonanzaの対局[62]を例に，現在のコンピュータ将棋が抱える問題点とモンテカルロ木探索の利点について述べる．この対局

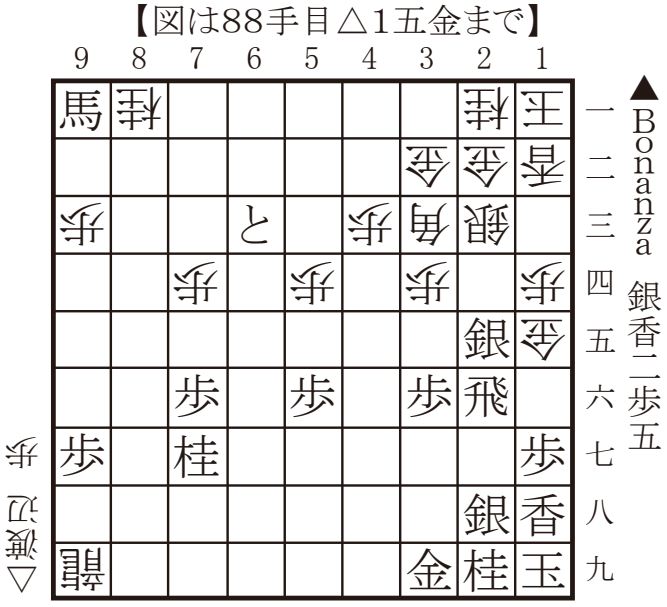

図 2.4: 渡辺竜王 対 Bonanza（局面１）

表 2.8: モンテカルロ木探索による図2.4の局面の評価

オーダー	指し手	評価値	読み筋
1（最善手）	▲２七香	0.471	△２六金▲同香△７九飛▲３八金打
2	▲３七馬	0.412	△２四歩▲２七香△２六金▲同香
3	▲３七銀打	0.398	△２六金▲２七香△３九竜▲同銀
...	...	...	
21	▲２四歩	0.301	△２六金▲２三歩成△同金右▲８一馬
...	...	...	

は，中盤までは互角の形勢に持ち込んでいたものの，終盤に典型的にコンピュータが苦手とする局面に入り，Bonanzaが敗れた．

図 2.4において，Bonanzaは▲２四歩を選択した．しかし，この局面は攻め合いでは先手に勝機はなく，実際この手が敗着となった．この局面の最善手は▲２七香でこれならまだ難しかったとされている[63]．相穴熊のような局面では，正しい局面の評価をするためには，特定の手順について深く読む必要がある．探索と評価関数による手法では，基本的に一定の深さで探索を打ち切るため，このような局面は苦手と言える．

表 2.8に，図 2.4に今回実装したモンテカルロ木探索によるプログラムに解答させた

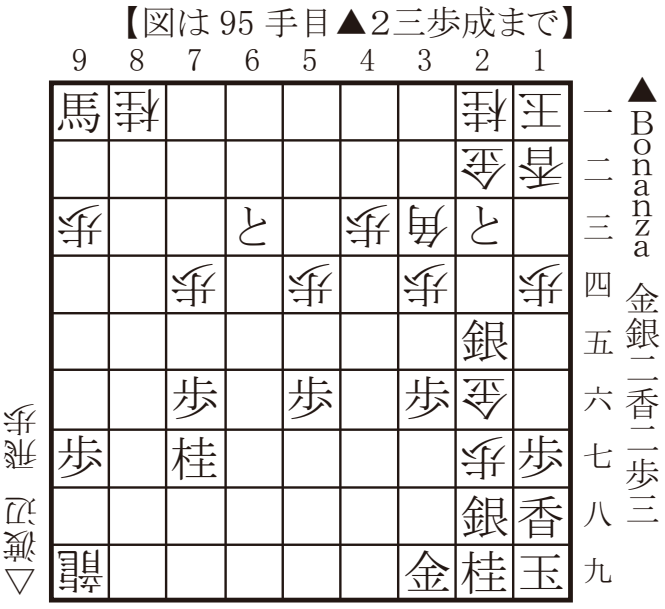

図 2.5: 渡辺竜王 対 Bonanza（局面２）

表 2.9: モンテカルロ木探索による図2.5の局面の評価

オーダー	指し手	評価値	読み筋
1（最善手）	△７九飛	0.622	▲２二と△同角▲３八金打△２八歩成
2	△５九飛	0.607	▲２二と△同玉▲４九歩△２八歩成
3	△３九龍	0.599	▲２二と△同角▲３九銀△２八金
...	...	...	

ときの結果を示す（思考時間は1手60秒）．評価値はモンテカルロ木探索の勝率を示しており，読み筋は訪問回数が最大の手順を示したものである．

表 2.8の結果から，本章の手法を適用したモンテカルロ木探索によるプログラムでは，渡辺竜王が最善手として示した▲２七香を選択できていることがわかる．また，Bonanzaが最善手として選択した▲２四歩の評価値は0.301となっており，この手順の攻め合いでは勝ちにくいことが正しく評価できていると言える．

図 2.5における，渡辺竜王の指し手は△３九龍である．しかし，Bonanzaをはじめとした多くのプログラムでは，この手を正しく評価することはできていない．この局面も，従来の手法を用いて正しく評価するためには相当深く読む必要がある局面である．

表 2.9は，図 2.5の局面を今回実装したモンテカルロ木探索によるプログラムに解答

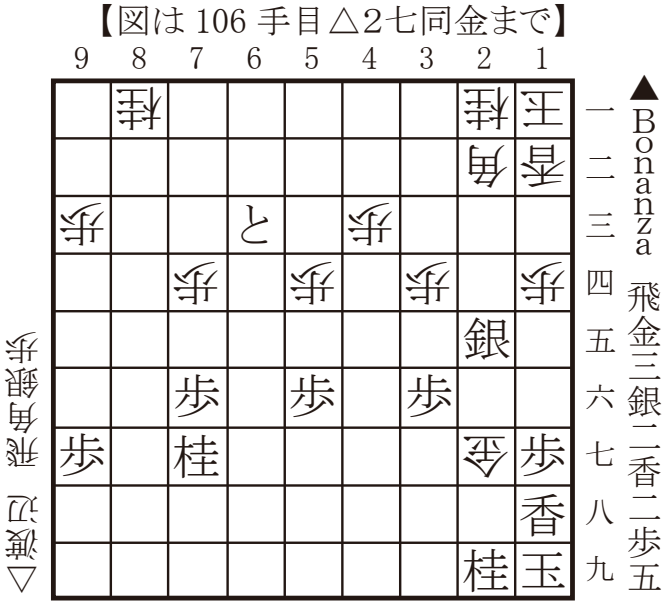

図 2.6: 渡辺竜王 対 Bonanza（局面３）

表 2.10: モンテカルロ木探索による図2.6の局面の評価

オーダー	指し手	評価値	読み筋
1（最善手）	▲２八金	0.334	△４八飛▲３九金△４七飛成▲２七金
2	▲３七銀	0.326	△２六銀▲４八銀打△８二角▲３四銀
3	▲３九銀	0.315	△２六飛▲２八銀打△同金▲同銀
...	...	...	

させた結果である．△３九龍を選択することはできていないものの，僅差で上位3手に含まれており，有力な手として評価できていることがわかる．モンテカルロ木探索では，playoutが直線的な読みの働きをするため，このような局面では探索と評価関数による手法よりも良い結果が得られることが多いと考えられる．

図 2.6は，すでに先手が敗勢の局面であるが，現在の将棋プログラムではほぼ互角の評価をしてしまう．この局面では，先手の持ち駒の多さ，後手玉に王手がかからないことを評価することの難しさなどから，評価関数により正しい形勢判断を行うことが難しい局面と言える．

表 2.10は，図 2.6の局面をモンテカルロ木探索によるプログラムに解答させた結果である．評価値（勝率）が3割程度となっており，かなりの劣勢と評価していることがわかる．

評価関数を設計する際の大きな問題点として，ある程度複雑なゲームでは，局面によって評価すべき項目が大きく異なるということが挙げられる．たとえば，図 2.6のような局面では，駒の損得よりも手番や相手玉に王手がかかるかといったことを重視しなければならない．多くの将棋プログラムでは，局面の進行度により評価関数のパラメータの値を変化させるなどの手法をとっているものの，あらゆる局面に対応できる評価関数の設計は困難である．

モンテカルロ木探索では，駒の損得や働き，手番や玉の危険度など多くの項目を個別に評価することはなく，シミュレーションの勝率というゲームの知識に依存しない評価指標を用いるため，より汎用性のある局面の評価が可能である．

2.5　今後の展望

本章では，コンピュータ将棋ではモンテカルロ木探索により従来の手法を単純に上回ることは難しいものの，部分的には十分に有効であることを示した．今後コンピュータ将棋にモンテカルロ木探索を利用する場合，従来の手法と併用し，その利点を生かすことが必要となると考えられる．本章では，モンテカルロ木探索の定跡選択部分における利用，通常探索との併用について今後の展望を述べる．

2.5.1　モンテカルロ木探索を用いた定跡選択

実験の結果から，定跡選択部分ではモンテカルロ木探索の一定の有効性が示されたと言える．問題点としては，通常の思考部（探索と評価関数）との相性が考慮されていない点が挙げられる．将棋の序盤は確実な最善手を求めることは困難といえ，現実的にはプレイヤ（プログラム）がより勝ちやすい手順を選択することが目標となる．そのため通常の思考部との相性を考えることは重要であり，大きな課題と言える．

改善策の1つとしては，playout 中の指し手の選択に，プログラムの評価関数の値を用いることが考えられる．この場合，実行時間が問題となるが，定跡選択部のように指し手が絞られている局面では数千〜数万程度の少ないplayout数でもある程度良い指し手の選択が可能であると考えられる．

また，本章では定跡データベース中に存在しない手は全く生成していないが，この方法では，一致するデータベースが少ない局面では，良い手を見逃してしまう可能性がある．このような問題を解決するため，データベースに存在する指し手の他に，ヒューリスティックにより有力と考えられる指し手を生成することなども有効と考えられる．

2.5.2 従来からの手法による思考との併用

モンテカルロ木探索は単純に探索と評価関数による手法を上回ることは難しいと考えられるものの，2.4節で示したように，局面によっては非常に有効と言える．モンテカルロ木探索を単独で用いるのではなく，従来からの手法である探索と評価関数による手法と併用することは棋力の向上に有効である可能性がある．以下に具体的な例を示す．

- 通常の探索において評価値が安定しないとき，モンテカルロ木探索を利用する
- 通常の探索とモンテカルロ木探索を同時に利用し，評価が大きく違う場合には探索の深さ，思考時間を延長する
- 合議アルゴリズム[64, 65]の1つの思考とする

通常の探索において，時間をかけるほど評価値が下がるなど結果が安定しないことがある．このような状況は，深さを基準に探索を打ち切るという通常の手法では，正しい評価を行うために必要な深さまで読むことができていないために起きていると考えられる．しかし，モンテカルロ木探索では，playoutが直線的な読みの働きをすることが期待できるため，探索の深さを十分にとることが難しい状況において従来の手法より有効に働く可能性がある．

また，現在のコンピュータは，プロセッサのマルチコア化が進んでおり，従来の手法とモンテカルロ木探索の両手法を同時に行うことも可能である．全く異なる手法による思考を同時に行い，参考にすることは棋力向上に有効である可能性が高いと考えられる．

近年では，合議アルゴリズムと呼ばれる，異なるプログラムの思考結果の多数決で単体のプログラムより，性能を容易に改善できることも明らかになっている．モンテカルロ木探索による思考は，従来のプログラムとは全く異なる特徴を持つため，このような合議アルゴリズムの思考の1つとすることも有効である可能性があると考えられる．

2.6 まとめ

本章では，ゲームの新たな実現法であるモンテカルロ木探索を将棋へ適用し，その可能性を検証した．囲碁において成功した手法を将棋向けに適用することに加え，チェスや将棋で古くから用いられているキラームーブやヒストリーヒューリスティックといった考え方を利用することにより，その性能を改善できることを示した．問題集による性能評価では，単純にモンテカルロ木探索を利用した場合の正答数は非常に低いものとなっているのに対し，本章の手法を適用した場合には，アマチュア初段程度のプログラムに迫る正答数を得ることに成功した．

有用性という点では，現在のコンピュータ将棋は人間のトッププレイヤに匹敵する強さにまで達しており，モンテカルロ木探索によるプログラムが従来の手法を単純に上回ることは難しいと考えられる．しかし，序盤の定跡選択や一部の終盤では，従来の手法を上回る有望な結果を得ることに成功し，コンピュータ将棋においてモンテカルロ木探索の部分的な利用が棋力の向上に有効となり得ることを示した．近年では，評価関数を利用した手法[66, 67]によって，モンテカルロ木探索によるコンピュータ将棋でアマチュア高段者の棋力が得られるといった報告もされており，モンテカルロ木探索によるコンピュータ将棋自体にもさらに改善の余地はあると考えられる．

本章の主要な貢献は以下の3点である．

1. 従来研究が少なかった，チェスライクなゲームにおけるモンテカルロ木探索の有効性を検証し，一定の可能性を示したこと．

2. キラームーブ，ヒストリーヒューリスティック等の従来alpha-beta法で用いられていた知識をモンテカルロ木探索に応用することで，性能を改善できることを示したこと．

3. 序盤の定跡選択，一部の終盤でモンテカルロ木探索を部分的に利用することにより，現在のコンピュータ将棋を上回る結果を得られたこと．

現在のコンピュータ将棋は，ほぼすべてのプログラムが同様のロジックに基づいており，抱えている問題点もある程度共通していると言える．モンテカルロ木探索は，これらの問題点を改善できる可能性のあるコンピュータ将棋の新たな実現法として，今後が期待される手法と言える．

第3章　対局に基づいた教師データの重要度の学習

本章では，ゲームで用いられている主要な機械学習を対象に，対局に基づいた教師データの重要度の学習を提案する．コンピュータにより思考ゲームを実現する主な手法としては，探索と評価関数を用いた手法，および，前章で述べたモンテカルロ木探索の2種類が存在する．近年，ゲームプログラミングの分野では，機械学習が成功を収めており，探索，評価関数，モンテカルロ木探索のそれぞれにおいて，機械学習によるパラメータ調整が取り入れられている．具体的には，探索深さの調整，評価関数の学習，モンテカルロ木探索におけるplayout中の指し手の選択確率の学習（以降，playoutの方策の学習）等が行われている．現在のゲームプログラミングにおける機械学習では，人間のエキスパートの棋譜を教師とした教師あり学習が最も一般的となっている．この学習手法は，多くの課題に対して良い結果を得ているものの，どのようなパラメータが得られるかは学習に使用する局面に依存する．学習に使用する局面は，学習データとしては均一な性質ではなく，プレイヤの強さや戦術，局面が勝敗に与える重要度などについて違いがある．例えば，将棋などのゲームでは，コンピュータは既に人間のトッププレイヤに迫る強さとなっており，人間の指し手をすべて真似することが必ずしも「強い」プレイヤの学習に結びつくとは限らない．本章では，このような課題を解決するため，教師データに重要度を導入した学習手法を提案する．ここでの重要度は，各教師データがどの程度「強い」パラメータの学習に寄与するかを意味する．本章では，強さを適応度とした進化的計算による重要度の学習と，重要度に従ったパラメータ学習を組み合わせた学習を提案し，現在ゲームプログラミングで用いられている主要な学習における有効性を検証する．

3.1　はじめに

近年，ゲームプログラミングの分野では，機械学習が大きな注目を集めている．現在のゲームプログラミングで用いられている学習手法の多くは，人間のエキスパートの指し手を教師とし，その指し手に近づけるように各特徴のパラメータを調整している．この方法は，評価関数，探索深さの決定など，多くの場面で成功し，プログラムの性能向上に大きく貢献してきた．

一方で，このような人間の棋譜を教師とした学習によって獲得されるパラメータは，教師データの性質に大きく依存する．人間の棋譜は教師として完全に理想的なものとはいえず，悪手が含まれる場合が存在する，人間の手を真似ることが必ずしもコンピュータにとって最善とは限らない，といった問題点が存在する．従来は，将棋，囲碁などの複雑なゲームでは，コンピュータの強さは人間のプレイヤに大きく劣っていたため，このような問題を含む棋譜も理想的な教師データとして近似することができた．しかし，現在では，将棋などの複雑なゲームにおいても，コンピュータが人間のトッププレイヤに迫る強さとなっており，単純に人間の指し手を教師とする手法では，性能に限界が生じると考えられる．

このような問題は，本来「強い」パラメータを学習したいのに対して，それを人間の指し手との一致率で近似しているために生じていると言える．人間の棋譜を教師としない学習も研究されているものの，現在までのところ将棋などの複雑なゲームでは人間の棋譜を教師とした学習を明らかに上回る手法は存在していない．

本章ではこのような現在の機械学習の問題点を解決するため，教師データに重要度（その学習局面が強いプレイヤの学習にどの程度寄与するか）の概念を導入し，対局による重要度の学習と重要度に従った重み付き学習を組み合わせた学習手法を提案する．また，コンピュータ将棋を題材に，評価関数の学習，実現確率の学習，モンテカルロ木探索におけるplayoutの方策の学習，に提案手法を適用し，その有効性を検証する．

以降の本章の構成は以下の通りである．3.2節では，現在成功を収めている，棋譜を教師とした機械学習を中心に，ゲームプログラミングで用いられる機械学習の手法について述べる．3.3節では，現在のゲームプログラミングにおける機械学習の問題点を述べ，その問題を解決する手法として，重要度を導入した学習を提案する．3.4節では，現在，ゲームプログラミングにおいて広く用いられている学習として，(1)評価関数，(2)実現確率，(3)playoutの方策の学習に対して提案手法を適用し，その効果を検証する．その後，3.5節で今後の課題について述べ，3.6節で本章のまとめを述べる．

3.2　関連研究

現在，ゲームの分野では様々なパラメータ調整において，人間の棋譜を教師とした教師あり学習が成功を収めている．特に将棋では，評価関数の学習や探索深さの調整において，大きな成果を挙げている．本章では，現在，ゲームで成功を収めている棋譜を教師とした学習として，評価関数の学習，探索深さの調整（実現確率探索），モンテカルロ木探索におけるplayoutの方策の学習，について概要を説明する．また，ゲームにおける提案手法と関連するその他の学習手法についても述べる．

3.2.1　評価関数の学習

ゲームの分野において，評価関数の機械学習は古くから研究されていた課題である．近年では，将棋において，機械学習による評価関数の学習が成功を収め，人間のトッププレイヤの強さを獲得することに成功している[31]．文献[31]の学習手法はゲームではComparison Trainingと呼ばれる学習手法の一種であり，特にコンピュータ将棋ではBonanzaメソッドとも呼ばれる．Comparison Trainingの概要を図 3.1に示す．

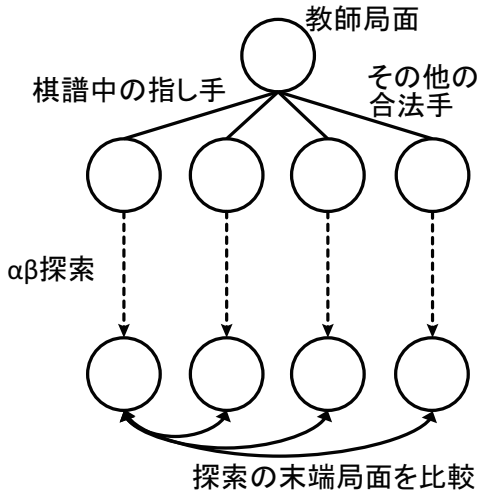

図 3.1: Comparison Trainingの概要

この手法では，棋譜中で実際の指し手と，それ以外の合法手との比較によって学習を行う．具体的には，棋譜中で指された手を正解手とし，それ以外の合法手の評価値を上回るように評価関数のパラメータを学習する．

パラメータ学習時には，PV（Principal Variation，最善手順）の生成と評価関数のパラメータの学習を繰り返し行うことを特徴としている．同様の学習手法はチェスでも行われており，Deep Blueの一部のパラメータは棋譜を教師とした学習で求められている[30]が，計算環境などの制約からチェスでは，将棋ほど良い結果は得られなかった．文献[31]の学習手法では，具体的には式(3.1)の目的関数の最適化を行っている．

$$J_1(P, \boldsymbol{v}) = \sum_{p \in P} \sum_{m=2}^{M_p} T\big(\xi(p_m, \boldsymbol{v}) - \xi(p_1, \boldsymbol{v})\big) \tag{3.1}$$

ここで，Pは学習対象の局面集合，M_pは局面pにおける合法手数，p_1は棋譜中で実際に指された手，p_mはそれ以外の手，$\xi(p_m, \boldsymbol{v})$はp_mにおいて探索を行った場合の末端局面の特徴$\boldsymbol{v}$によって算出された評価値を示す．また$T(x)$はシグモイド関数を表す．実際の

学習時には，パラメータの値を安定させるために，駒の価値の総和等で拘束条件を課していることが多い．また，評価関数の学習では，数十万以上の非常に多くのパラメータを学習するため，L1正則化を行うのが一般的であるが，式 (3.1)ではこれらの項は省略して表記している．

評価関数の学習は，コンピュータ将棋の強さを大きく向上させることに成功し，現在ではトップレベルのプログラムの多くが，Bonanzaの学習をベースとした学習を用いている[43, 44, 45].

3.2.2　実現確率探索

実現確率による探索打ち切りアルゴリズム[32]（以下，実現確率探索）は，知識に基づいた代表的な探索手法である．この探索手法は将棋を中心に盛んに研究が行われており[68, 69, 70, 71]，いくつかのトップレベルのプログラムでも採用されている．一般的な探索手法では，深さを探索の打ち切り条件とするのに対して，実現確率探索では深さの代わりに局面の実現確率を利用する．実現確率探索の探索例を図 3.2に示す.

局面の実現確率は以下の式によって再帰的に定義される値である．

$$（局面の実現確率）=（親の局面の実現確率）\times（遷移確率）\tag{3.2}$$

ルート局面の実現確率は1.0 とし，遷移確率には指し手が選択される確率を用いる．指し手の選択確率は，指し手を特徴ごとに分類し，その特徴に対応する指し手がプロの棋譜中でどれくらいの割合で指されたかを算出することにより求める．このように局面の実現確率を探索の閾値とすることで，実際に起こりやすい展開を重点的に深く探索できる点が実現確率探索の大きな特徴である．

実現確率探索では，図 3.2のように，確率の高いノードを優先的に深く探索する．実現確率探索では，確率の低い手が再探索を起こした場合，その手の確率を高くして再探索する．これは確率が低いことによる水平線効果（horizon effect）を防ぐためである．なお，式 (3.2)の確率は対数をとることで深さと同等のものとして扱うことができる．

遷移確率の学習は，従来は単純に実際の棋譜における選択確率を用いていたが，近年はロジスティック回帰による予測が良い結果を得ており[72]，本章でもこの手法を用いる．なお，以降論文中で，実現確率探索における遷移確率の学習を指す場合，単に「実現確率の学習」と表現する．

ロジスティック回帰による実現確率の学習の手順を以下に示す．この手法では，n個の特徴が存在し，$i\,(1 \leq i \leq n)$番目の特徴の値をx_iとすると，特徴の値$(x_1, x_2, ..., x_n)$を持つ指し手の遷移確率pは式(3.3)で表される．

$$p(x_1, x_2, ..., x_n) = \frac{1}{1 + \exp(-\sum_{i=1}^{n} w_i x_i)}\tag{3.3}$$

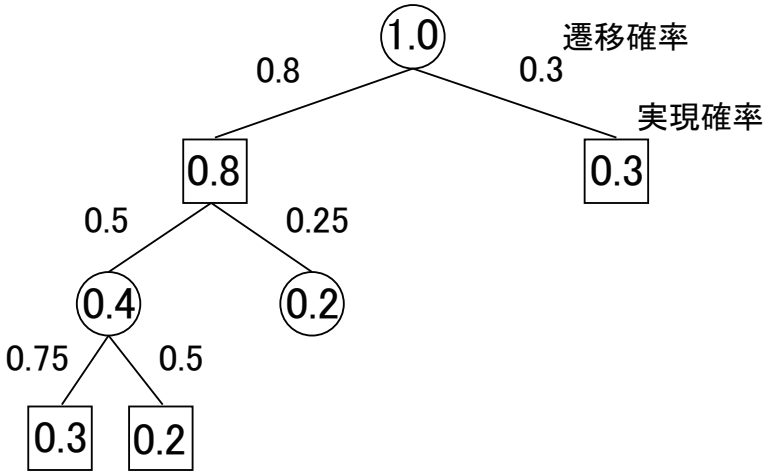

図 3.2: 実現確率探索の概要

ロジスティック回帰による実現確率探索では，プロの棋譜から各特徴が選択される割合を求める代わりに，式(3.3)における各特徴の重みw_iを学習により求める．学習の際には，プロの棋譜において実際に指された手を正例，指されなかった手を負例とすることで，各特徴の重みw_iを推定する．

この重みw_iの算出には，L1正則化ロジスティック回帰が用いられることが多い．具体的には式(3.4)の最適化を行うことによってw_iを算出する．

$$\min_{\boldsymbol{w}} \quad |\boldsymbol{w}| + C \sum \log(1 + e^{-y_j \boldsymbol{w}^T \boldsymbol{x}_j}) \tag{3.4}$$

ここで，y_jは正例，負例のラベル，Cは正則化の重みを制御する定数である．

実現確率探索も将棋において成功を収めている探索手法であり，評価関数の学習と同様，多くのプログラムが採用している手法となっている．

3.2.3　モンテカルロ木探索中のplayoutの方策の学習

モンテカルロ木探索中のplayoutの指し手の選択にも棋譜を教師とした学習が取り入れられている．第2章で述べた，指し手のEloレーティングの学習には，Minorization-Maximizationアルゴリズムと呼ばれる手法が用いられる[33]．また，モンテカルロ木探索中のplayoutの方策には前節で述べた実現確率を用いることも可能であり[73]，本質的には同等の意味合いを持つ．

3.2.4　その他の学習手法

ゲームプログラミングの分野では，前述の手法に限らず，従来から様々な学習手法が提案されている．例えば，文献[74]では，強化学習の一種であるTD法を用いた駒の価値の学習が行われており，文献[75]では進化的計算を用いた評価関数の学習が行われている．

また，これらの学習手法を組み合わせた学習手法も提案されている．文献[76]では，局所探索を得意とするTD学習と大域的探索を得意とするGAを組み合わせたハイブリッドGAにより評価関数のパラメータ学習を行っている．この方法により，オセロの実験では従来手法（単一の学習手法を用いた場合）を上回る結果を得ている．

しかし，これらの強化学習や進化的計算によるパラメータ学習は，現在までのところ将棋のような複雑なゲームでは，人間の棋譜を教師とした学習を明確に上回る成果は得られていない．

3.2.5　学習棋譜がプログラムの強さに与える影響

学習棋譜が学習された評価関数の強さに与える影響についても，これまでにいくつかの研究が行われている[77, 78]．文献[77]では，プロ，コンピュータ，アマチュアの棋譜を学習棋譜した時の評価関数の強さの比較を行っており，アマチュアよりもプロ，コンピュータの棋譜を教師としたほうがやや強い評価関数を学習できるといった結果が報告されている．

また，一部のトップレベルのプログラムでは，学習棋譜の取捨選択，強いプレイヤの棋譜の重みを大きくするといったことも行われている[79]．ただし，これらの調整は，開発者が手作業で行ったものであり，学習棋譜の性質を十分に考慮した学習が行われているとはいえず，どの程度の効果が得られているかも明らかになっていない．

3.3　提案手法

3.3.1　人間の棋譜を教師とした学習の問題点

本節では，現在のゲームプログラミングにおいて主流となっている人間の棋譜を教師とした学習の問題点について述べる．人間の棋譜を教師とした学習は，現在のところ多くの課題に対して良い結果を得ているものの，以下に示すような問題点が存在する．

第一に，人間の棋譜を絶対的に信頼しているため，教師とするのにふさわしくない局面も学習対象とすることがある．人間の棋譜を教師とした場合，様々な強さの対局者の棋譜が混ざったものを用いることになる．一般的に，学習棋譜としては，プロやアマチュア高段者などある程度の強さの対局者のものを利用するが，対局者の強さに

ばらつきが存在するという点は本質的には解決できていない．また，人間のプレイヤの場合，ミスがあるため，棋譜中には悪手が含まれることがある．特に対局時間の短い棋譜では，プロの棋譜でも悪手が数多く含まれる場合がある．これまで，囲碁や将棋等といった複雑なゲームでは，コンピュータは人間にはるかに及ばなかったためこの点は問題とならなかった．しかし，現在では，これらのゲームにおいてもコンピュータは人間のトッププレイヤに迫る強さとなっており単純に人間の強いプレイヤの棋譜を信頼した場合，性能向上に悪影響を及ぼす可能性があると考えられる．

第二に，従来手法では，すべての棋譜，局面を均等に扱っており，全体としてプロの指し手との一致率が上がるようにパラメータの調整を行っている．しかし，これは必ずしも強いプレイヤの実現に結びつくとは限らない．棋譜中に含まれる局面には，最善手が指せないと即負けにつながるような重要な局面もあれば，有力な手が多数存在するような局面も存在する．棋譜を教師として，一致率を向上させるようにパラメータを学習する手法では，序盤の駒の位置関係が重視されがちになるが，中盤や終盤の，より戦術的な，勝敗に直結する局面において良い手が指せるかを重視したほうが強いプレイヤとなる可能性があると考えられる．

第三に，本来は学習手法によって有効な教師データには違いがあると考えられる．例えば複雑な局面における好手，妙手といった教師例は，駒の価値などの単純な特徴しか用いていない学習ではその意味を表現することは難しく，意味のない教師データとなる可能性が高い．一方で，十分に表現力のある特徴を用いた場合には，そのような戦術的な局面は非常に有用な学習対象となると考えられる．

これらの問題は本質的には「強い」パラメータを得ることが目的であるのに対して，それを「人間の指し手との一致率」といった基準で近似していることに起因していると言える．強化学習など人間の棋譜によらない手法が成功しているゲームもあるものの[80]，現在までのところ，将棋などの複雑なゲームでは人間の棋譜を教師とした学習を明らかに上回る成果は得られていない．

本章では，人間の棋譜を教師とした学習において教師データに重要度を導入し，対局による重要度の学習と重要度に基づく重み付き学習を組み合わせた学習手法を提案する．

3.3.2　重要度を導入した学習

前節で述べた課題の通り，教師あり学習によって学習されるパラメータは教師データの性質に大きく依存する．しかし，従来の棋譜を教師とした機械学習では，教師データとの指し手の一致率を向上させることのみを目的としており，どのような教師データを用いることによって,「強い」プログラムが学習できるかといった点については考慮されていない．提案手法では「強い」パラメータを得るため，教師データに重要度を導入し，強さに寄与する局面を重視した学習を行う．ここでの重要度は，ある

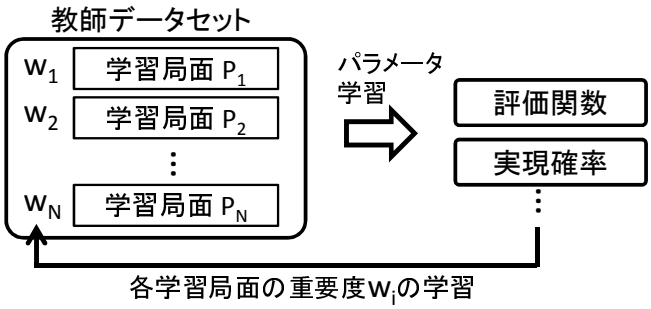

図 3.3: 提案手法の概要

教師局面が「強い」パラメータの学習にどの程度寄与するかを意味する．提案手法では，この重要度を学習されたプログラムの強さが最大化されるように，機械学習によって求める．重要度の値は，評価関数のパラメータ等と異なり，勾配法を使った学習が困難であるため，進化的計算を用いて学習を行う．具体的には，重要度は，多数の対局から算出したEloレーティングを適応度とした進化的計算によって学習し，重要度の学習と，重要度に基づいたパラメータ学習を組み合わせることにより,「強い」プログラムを実現するパラメータを獲得する．なお，ここでのEloレーティングは，前章で述べた指し手の予測とは関係なく，プログラム間の対局結果を基に算出された，プログラムの相対的な強さを表す値を意味するものである．

提案手法の概要を図 3.3に示す．提案手法では図 3.3のように，教師あり学習による個体の学習と，学習された個体同士の対局に基づいた進化的計算による教師データの重要度の学習を繰り返し行うことで,「強い」個体を学習する．

重要度は進化的計算の一種であるPBILc（Continuous Population Based Incremental Learning）[81] を用いて学習する．学習の際には，大規模な計算環境を用いた対局を行い，勝率の高い個体を学習するように重要度を調整する．強化学習等の学習手法と進化的計算を組み合わせた手法は過去にも提案されている[76]が，前述のとおり，将棋等の複雑なゲームでは強化学習や進化的計算により人間の棋譜を教師とした学習を上回ることは現状では困難と考えられる．本手法のポイントは，進化的計算により評価関数や実現確率のパタメータを直接求めるのではなく，教師あり学習において「強い」パラメータを学習するための教師データの重要度を学習する点である．

図 3.4に提案手法の具体的な処理フローを示す．ステップ(1)では，重要度を正規乱数$\mathcal{N}(1.0, \sigma^2)$に従って生成し，その重要度により教師データの各局面に重み付けしたパラメータ学習を行う．この操作を繰り返し，N個の初期個体を生成する．本章では実験的に，$\sigma^2 = 0.1$，$N = 50$とした．重要度は局面単位に持たせても良いし，棋譜単位など意味を持ったまとまり毎に持たせても良い．局面単位に重要度を持たせる場合，個別の学

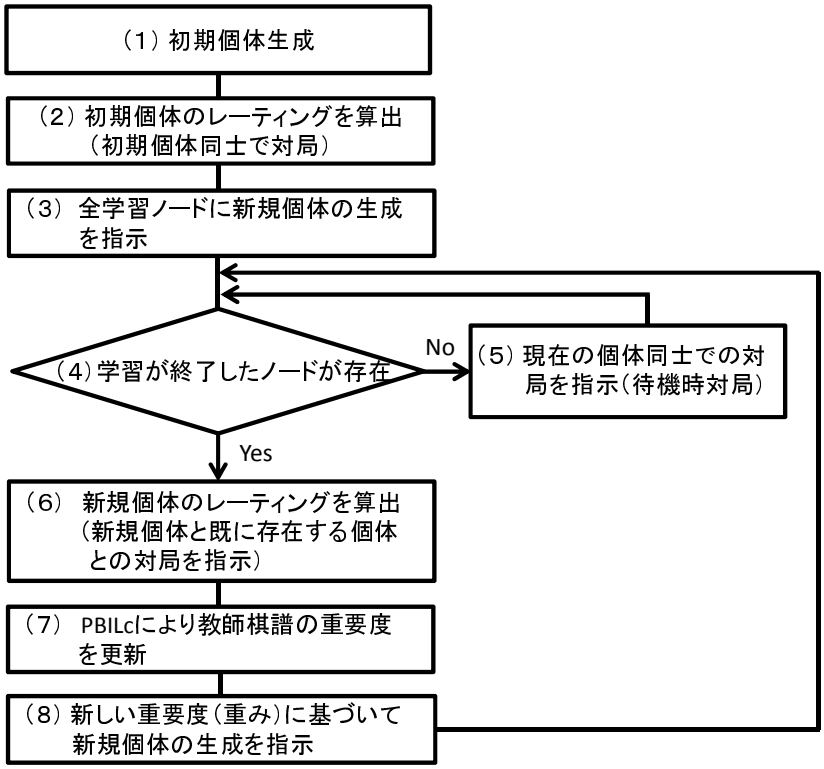

図 3.4: 提案手法のメイン処理フロー

習局面それぞれの重要度を算出できるため，悪手を1手だけ省くといったような詳細な重要度の割り当てが可能となる反面，求めるパラメータの数が膨大となり，非常に多くの計算資源，計算時間を要するという欠点が存在する．そのため，本章では，重要度を進行度に応じて割り当てる手法，及び棋譜毎に割り当てる手法の2種類を検討する．

ステップ(2)では，ステップ(1)で生成された各初期個体同士で対局を行い，その対局結果から，個体の相対的な強さをEloレーティングとして算出する．

ステップ(3)〜(8)では，対局結果に基づいた重要度の更新と，その重要度に従った新規個体の生成を繰り返す．対局結果からEloレーティングを算出する処理には非常に多くの時間を要する．本章における実験では，これらの処理に，クラスタ環境を利用する．対局からEloレーティングを算出し重要度を学習する処理と，新規個体の生成（重要度に従った評価関数，実現確率などの学習）は，全く別の処理として並列に処理することができる．重要度を学習するマシンでは，新規個体が生成されていた場合には，新規個体と既に存在する個体と対局を行うことによって新規個体のレーティングを算出し，その結果に基づいて重要度を更新する．その後，新規個体生成側のマシンに，新たな重要度に従った新規個体の生成を指示する．新規個体生成側のマシンは，受け取った重要度に従ったパラメータ学習を行い，新たな個体を生成する．

重要度の学習では，正確な順位を算出するために非常に多くの対局を行う必要がある．ステップ(2)では各個体4,000局ずつ，ステップ(6)では2,000局の対局を行う．ただし，ステップ(6)では，対局数が1,000局以上で，レーティング算出対象の新規個体の順位が最下位の場合，学習時間を短縮するため，その時点で対局を打ち切っている．

本手法では，重要度の更新には，進化的計算の一種であるPBILcを用いる．PBILcは進化的計算の中でも，分布推定アルゴリズム（EDA: Estimation of Distribution Algorithm）と呼ばれる手法である．通常の遺伝的アルゴリズム（GA: Genetic Algorithm）が直接個体を進化させるのに対して，EDAでは個体の生成確率を進化させる点が特徴である．

PBILcでは，次世代の個体を$\mathcal{N}(X,\sigma^2)$の正規乱数に従って生成する．X，σは以下の式により更新する．

$$X^{t+1} = (1-\alpha)X^t + \alpha(X^t_{best,1} + X^t_{best,2} - X^t_{worst}) \tag{3.5}$$

$$\sigma^{t+1} = (1-\alpha)\sigma^t + \alpha\sqrt{\frac{\sum_{j=1}^{K}(X^j - \overline{X})^2}{K}} \tag{3.6}$$

ここで$(t+1)$世代目のX^{t+1}は，前世代の最良個体$X^t_{best,1}$と2番目に良い個体$X^t_{best,2}$，及び最も悪い個体X^t_{worst}から求められる．また，σの値は，上位K個の最良個体の標準偏差を基に更新する．このような式を用いることで，よさそうなパタメータの値の付近かつ不確定な（よさそうな個体群のなかでばらつきが大きい）部分を重点的に探索することができる手法となっている．本章における学習では，PBILcのパラメータは$K = 10$，$\alpha = 0.01$とした．

3.4　実験

提案手法を，現在ゲームの分野で行われている主要な棋譜を教師とした学習に対して適用し，その効果を検証した．具体的には，以下の3つの学習における提案手法の有効性を検証した．

1. 評価関数の学習
2. 実現確率の学習
3. モンテカルロ木探索におけるplayoutの方策の学習

評価関数のパラメータ学習は文献[31]の手法を用い，学習時の探索は静止探索のみとした．実現確率のパラメータ学習には，LIBLINEAR[82]を各学習局面に重み付けして学習できるよう改変したものを用いた（オプションはL1正則化ロジスティック回帰「-s

6」とした）．また，モンテカルロ木探索の実験では，第2章の手法をすべて適用したプログラムを用い，playoutの方策には前章と同様，文献[33]の手法を用いた．

3.4.1 学習条件

実験環境を表 3.1に示す．重要度を学習するための個体同士の対局は15台のクラスタ環境で行い，新規個体の生成には18台の学習用マシンを用いた．学習は，提案手法における図 4.1のステップ(4)〜(8)の処理を400回行った．学習に要した時間は，評価関数の学習では約12日，実現確率の学習では約3日となっている．

表 3.1: 実験環境

用途	CPU	メモリ	台数
対局（重要度学習）	Core2 Quad Q9650	8GB	15
パラメータ学習	Core i7-3930K	16GB	7
	Core i7-990X	12GB	1
	Xeon E5506×2	24GB	6
	Opteron 6134×2	16GB	4

なお，重要度の学習，及び対局実験における探索ノード数は1手10万ノードとした．学習では，以下に示す将棋における一般的な特徴を用いた．

(1) 駒割り

(2) 自玉，敵玉との位置関係

(3) 利きの関係に基づく駒の位置関係のパターン[83]

(4) 王手

(5) 駒を取る手，リキャプチャ

(6) 成る手

(7) あたりをかける手

(8) 逃げる手

(9) 持ち駒を打つ手

(10) 玉の移動

このうち，(4)〜(10)は指し手の特徴となるため，評価関数の学習では(1)〜(3)の特徴のみを用いている．

(3)の利きの関係に基づく駒の位置関係のパターンは，利きの重なりがある2駒以上の駒の位置関係のパターンを意味する．具体的には，あるマスに着目した時，そのマスに利きのあるすべての駒の位置関係を特徴として利用する．利きの関係に基づく駒の位置関係の例を図 3.5で示す.

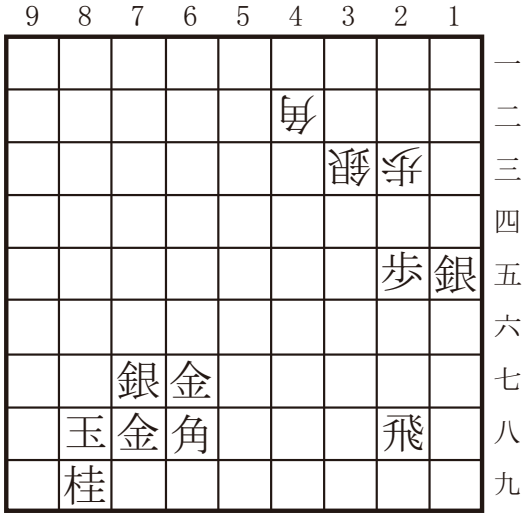

図 3.5: 利きの関係に基づく位置関係のパターンの例

図 3.5において，例えば２四の地点に着目すれば（▲２五歩，▲１五銀，▲６八角，▲２八飛，△２三歩，△３三銀，△４二角）というパターンが，７七の地点に着目すれば（▲８九桂，▲７七銀，▲６七金，▲７八金，▲６八角，▲８八玉）というパターンが抽出される．将棋の学習に用いられる特徴は，一般的に2駒間，3駒間の位置関係が用いられるが，この特徴では，多くの数の駒の位置関係を表現できる．また，利きの情報に基づいているため，意味のあるパターンを生成しやすく，利き情報を持つデータ構造を使用しているプログラムでは，高速にパターンを生成することも可能である．

なお，学習にはプロやアマチュア高段者の棋譜を用いた．学習棋譜数は，評価関数は10,000局，実現確率は5,000局とした．

3.4.2　対局のマッチメイク方法

提案手法では，複数の個体同士の対局結果を基に算出したEloレーティングを強さの指標として用いる．なるべく少ない対局数で正確な個体の強さを算出するために，マッチメイクの方法は重要である．本実験では，囲碁や将棋のオンライン対局サーバ

[84, 85]で良く用いられている手法を参考に，どのようなマッチメイク方法が適しているかを検討した．

図 3.6，図 3.7は，強さが既知の50プレイヤで繰り返し対局を行った時の，理想的な順位との誤差の平均を示したものである．本実験では，1,000回のマッチメイクを行う実験を10回行い，各プレイヤの理想的な順位との誤差の平均値を求めている．なお，本実験では，1回のマッチメイクにつき，同一局面から先後を入れ替えた2局の対局をセットで行っている．

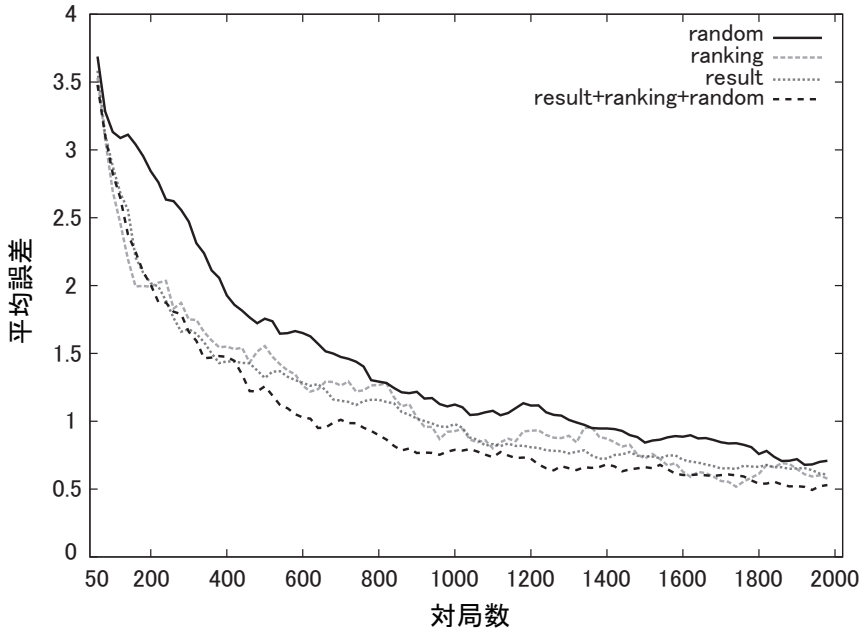

図 3.6: マッチメイク手法による理想的な順位との誤差（同一プログラムで探索ノード数を変更したプログラム間の対局）

図 3.6は，同一プログラムでノード数を変更したプレイヤ間での実験，図 3.7は，評価関数学習時におけるすべての初期個体同士で対局を行った実験の結果である．

図 3.6の実験では，プレイヤ$n(0 \leq n \leq 49)$の探索ノード数を$1,000 + 200 \times n$とした．この条件では，同一の思考を持つプログラム同士の対局において，探索ノード数が多いほど強いことは自明であるため，相性が存在しない理想的な順位を持つプログラム間で対局を行う環境での実験となっている．図 3.7の実験では，すべての個体同士で事前に1,000局ずつ総当りの対局を行い，その結果を理想的な順位としている．総当りの対局は非常に時間がかかるため，本実験では探索ノード数は10,000とした．図 3.7は相性や同程度の強さのプログラムが複数存在する，より実際の状況に近い環境でのマッチメイクの妥当性を検証することを目的とした実験となっている．

実験では，次の4通りのマッチメイクの方法を実験した．

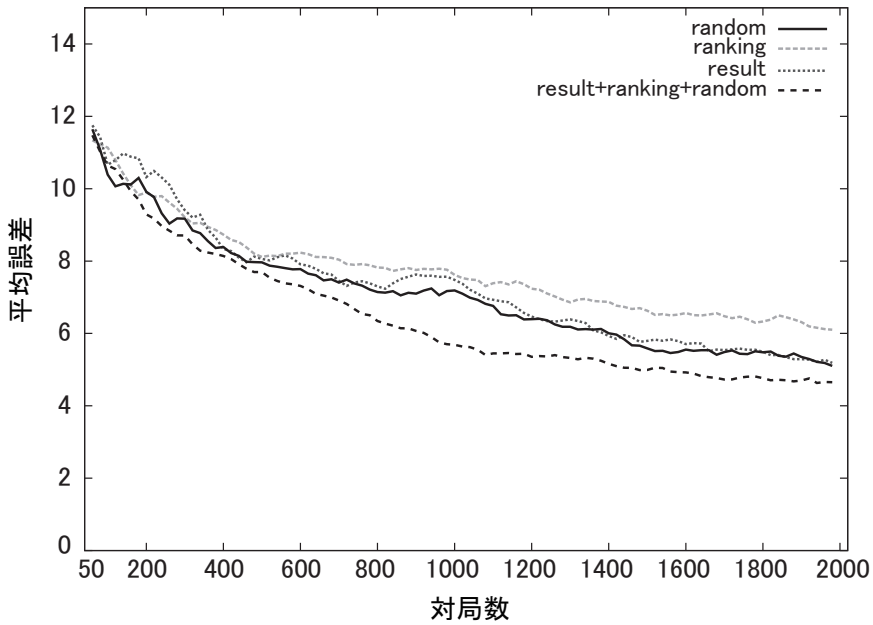

図 3.7: マッチメイク手法による理想的な順位との誤差（異なるパラメータの評価関数を持つプログラム間の対局）

- 完全にランダムなマッチメイク（random）
- 正規乱数により順位の近いプレイや同士を優先的に対局させる方策（ranking）
- 勝ったプレイヤ同士，負けたプレイヤ同士をランダムに対局させる方策（result）
- プレイヤ同士，負けたプレイヤ同士で順位の近いものを優先して対局させ，さらに10回に1回完全にランダムなマッチメイクを混ぜた方策(result+ranking+random)

4つ目の方策で，ランダムなマッチメイクを混ぜている意図は，順位の近いプレイヤで勝ち同士，負け同士で対局を組むと，同じプレイヤでの対局のループが生じることがあるため，これを防ぐためである．

実験の結果，図 3.6の理想的なプレイヤ同士の対局では，ranking，result，result+ranking+randomのいずれの方策もランダムなマッチメイクと比較して良い結果を得た．一方で，図 3.7の実験では，resultはrandomとほぼ同等，rankingはrandomに劣る結果となった．これは，マッチメイクの工夫がプレイヤ間の相性に影響されやすいことを示していると言える．特にrankingによるマッチメイクでは，順位の近いプレイヤとの対局回数が多くなるため，局所的には正確な順位を求めることができるものの，相性が存在する場合には全体の中での順位に不適当な偏りが生じることが多くなったと考えられる．

result+ranking+randomによるマッチメイクは，図 3.6および図 3.7の実験において，

ともに安定して最も良い結果を得た．これは，rankingによって順位の近いプレイヤとの優劣を正確に求めつつ，result，randomを組み合わせることによって，局所解に陥らないようなマッチメイクが実現できているためであると考えられる．以降の実験では，提案手法における個体のEloレーティングを算出するための対局のマッチメイクには，result+ranking+randomを用いた．

3.4.3　対局実験による提案手法の評価

提案手法をコンピュータ将棋における，(1)評価関数の学習，(2)実現確率の学習，(3)モンテカルロ木探索中のplayoutの方策，に適用した際の有効性を対局実験により評価した．比較対象のプレイヤ（従来手法）は，教師データに重要度による重み付けをせず学習したものである．評価関数の学習では，予備実験においてそれ以上反復回数を増やしても性能向上が認められなくなるまで十分な学習を行ったものを比較対象とした．対局は定跡で16手進めた局面から先後を入れ替えて1,000セット，2,000局を行った．

3.4.3.1　評価関数の学習における提案手法の評価

表 3.2，表 3.3に評価関数学習時における，提案手法を用いて学習したプログラムの，従来手法，および最良の初期個体との対局結果を示す（表中の太字の数値は有意水準5%の二項検定で有意な結果）．

表 3.2: 従来手法に対する勝率（評価関数）

重要度の学習単位	提案手法	初期個体（best）
進行度	**0.561**	**0.523**
棋譜	**0.581**	**0.552**

表 3.3: 最良の初期個体に対する勝率（評価関数）

重要度の学習単位	勝率（提案手法）
進行度	**0.544**
棋譜	**0.531**

実験結果から，提案手法が従来手法と比較し，有意に勝ち越していることがわかる．最良の初期個体の重要度を用いて学習を行った場合にも従来手法を上回っているが，提案手法で学習された個体は，より高い勝率を得ることができている．提案手法に

よって学習された評価関数を用いたプログラムは，最良の初期個体に対しても有意に勝ち越しており，進化的計算によって，より強い個体を学習するための重要度の学習が行われていると考えられる．

3.4.3.2　実現確率の学習における提案手法の評価

表 3.4，表 3.5に実現確率学習時における，提案手法を用いて学習したプログラムの，従来手法，および最良の初期個体との対局結果を示す（太字は有意水準5%の二項検定で有意な結果）．

表 3.4: 従来手法に対する勝率（実現確率）

重要度の学習単位	提案手法	初期個体（best）
進行度	**0.527**	0.506
棋譜	**0.536**	0.486

表 3.5: 最良の初期個体に対する勝率（実現確率）

重要度の学習単位	勝率（提案手法）
進行度	0.514
棋譜	**0.542**

実験結果から，実現確率探索においても提案手法が従来手法を有意に上回る結果を得た．ただし，評価関数の学習と比較すると，その効果はやや低い結果となった．これは，評価関数の場合には評価値にわずかでも差があれば選択される指し手に直接影響を及ぼすのに対して，実現確率探索の場合には，予測確率を探索深さに変換して利用することになるが，その際に予測確率の差が小さい指し手間では探索深さとしては差がつきにくいこと等が原因として考えられる．

3.4.3.3　モンテカルロ木探索のplayoutの方策の学習における提案手法の評価

表 3.6，表 3.7にモンテカルロ木探索のplayoutの方策の学習時における，提案手法を用いて学習したプログラムの，従来手法，および最良の初期個体との対局結果を示す（太字は有意水準5%の二項検定で有意な結果）．

実験結果から，モンテカルロ木探索のplayoutの方策では，実現確率と比較してやや高い効果を得た．これは，モンテカルロ木探索では，終局までの長手数のplayoutを行

表 3.6: 従来手法に対する勝率（モンテカルロ木探索のplayoutの方策）

重要度の学習単位	提案手法	初期個体（best）
進行度	**0.539**	0.505
棋譜	**0.548**	0.514

表 3.7: 最良の初期個体に対する勝率（モンテカルロ木探索のplayoutの方策）

重要度の学習単位	勝率（提案手法）
進行度	**0.540**
棋譜	**0.544**

うため，通常の探索よりも指し手の予測が結果に与える影響が大きくなったことが考えられる．なお，モンテカルロ木探索の場合，playoutの性質がプログラムの強さに与える影響が明らかになっていないという特徴がある．このような，どのようなパラメータが強いプログラムを実現するのか明らかでない課題に対しても，提案手法は効果を発揮する可能性が高いと考えられる．

なお，今回は棋譜単位，進行度単位で重要度を割り当てたが，指し手の予測確率により明確な差が付けられるよう，局面単位で重要度を割り当てることや，正例（棋譜中で指された手）と負例（それ以外の手）で異なる重要度を与えること等により，さらなる性能向上を実現する余地があると考えられる．

3.4.4　進行度単位の重要度の分析

図 3.8に提案手法を用いて学習した進行度別の重要度（評価関数学習時）を示す．進行度は，単純にある時点の手数を終局までの手数で割った値を意味する（1に近づくほど終盤であることを意味する）．図 3.8では，提案手法によって学習された重要度，および参考として初期個体のうち最もレーティングの高かった個体の重要度を示している．なお，重要度の平均は常に1.0になるように正規化している．

図 3.8から，提案手法を用いて学習した重要度は，序盤から中盤にかけてやや下がり，終盤で最も高くなっていることがわかる．終盤で重要度が最も高くなった理由としては，終盤では駒の位置関係が直接勝敗に影響しやすく，良い局面，悪い局面がはっきりと分かれるためであると考えられる．文献[44]では，進行度に応じて終盤ほど正解手とその他の合法手の評価値のマージンが大きくなるように調整しており，終盤の重要

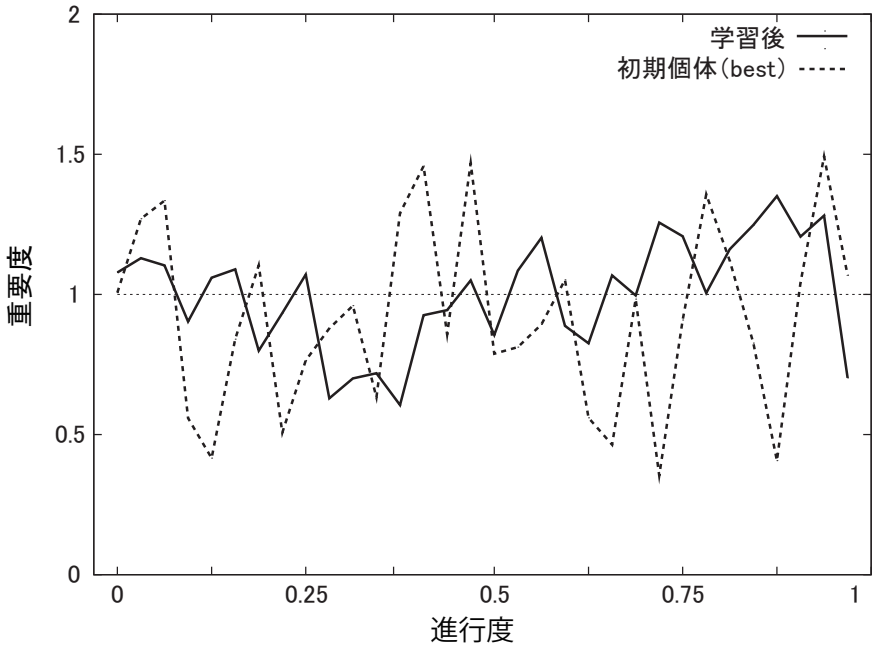

図 3.8: 進行度に応じた教師データの重要度の変化

度が高くなっている点では，今回の実験結果と傾向が一致していると言える．

また，今回の実験では，序盤よりも中盤の前半において重要度が低くなる結果となった．これは，序盤は駒組みが明確で評価関数で局面評価がしやすいのに対して，中盤は駒組み（囲い）が崩れることも多く，明確な目指すべき形が定義しにくいことが原因と考えられる．序盤，終盤はある程度有力な手が限られているのに対して，中盤は自由度が高く局面の評価が難しいと言える．

さらに，最終盤（終局直前の5〜10手程度に相当）では，重要度が低くなっていることもわかる．これは最終盤では既に勝敗が決しており教師として不適切な局面が存在することや，今回の実験で評価関数の学習に用いた探索が静止探索という非常に浅い探索であったことに起因すると考えられる．文献[31]の学習手法では，探索結果の末端局面を学習に用いるため，学習時に用いる探索深さと局面を正確に評価するために必要な探索深さが大きく異なる局面は本質的に学習対象としてあまり向いていないと言える．特に今回の評価関数の学習では，静止探索という非常に浅い探索を用いたため，直接詰みが絡むことが多い最終盤は適切な探索結果を得ることができず，重要度が低くなったと考えられる．これらの結果から，提案手法では学習の性質に応じた教師データの重要度を算出することができていると考えられる．

表 3.8: 戦術による重要度の違い

順位	戦型	戦術		先手側勝率（棋譜数）	重要度の平均
		先手戦術	後手戦術		
1	四間飛車	四間飛車穴熊	銀冠	0.566 (30)	1.121
2	四間飛車	居飛穴模様	藤井システム	0.500 (20)	1.100
3	矢倉	森下システム	△９五歩・８四歩型	0.500 (96)	1.088
4	矢倉	▲４七銀３七桂	金矢倉	0.500 (18)	1.079
5	相掛かり	▲２六飛	△５四歩型	0.542 (24)	1.061
…	…	…	…	…	…
96	三間/四間飛車	居飛車穴熊	本美濃	0.700 (30)	0.951
97	矢倉	引き角	△６二飛戦法	0.474 (19)	0.922
98	三間/四間飛車	左美濃	高美濃	0.611 (18)	0.914
99	中飛車	角交換型	ゴキゲン中飛車	0.519 (27)	0.901
100	角換わり	相腰掛け銀	△６五歩型	0.538 (26)	0.899

3.4.5　棋譜単位の重要度の分析

表 3.8に，棋譜毎の重要度を学習した時の，戦術による重要度の違いを示す．戦型，戦術の分析には，文献[86]のソフトを用いた．戦術は，出現数の多いもの上位100個について分析を行った．

表 3.8から，穴熊や銀冠，矢倉を始めとする，囲いを発展させる戦術の重要度が高くなっていることがわかる．一般的に，穴熊等はコンピュータ将棋でも勝ちやすい戦術と言われており妥当な結果と考えられる．一方で，今回の実験では美濃囲いはやや重要度が低くなる傾向があった．これは美濃囲いで戦うよりも，銀冠，穴熊等の囲いに発展させたほうがより高い勝率を得ることができる可能性が高くなるため，相対的に美濃囲いの重要度を低くする方向に学習が働いたためと考えられる．表 3.8から，特に対穴熊では，通常の美濃囲いを選択するよりも，銀冠に発展させる，藤井システムによって穴熊を阻止する等の戦術をとる重要度が高くなっていることがわかる．また，今回の実験結果では，戦術で分類した時に先手と後手の勝率が互角に近い棋譜の重要度が高い傾向があった．これは，本実験では棋譜単位で重要度を付与しており，先手，後手の戦術の重要度をそれぞれ表現することができないことが原因だと考えられる．棋譜単位の重要度では，一方が戦術的に有利である棋譜に対して，先後両方の戦術の重要度を同等として学習してしまうため，先後で互角の棋譜の価値が高くなったと考えられる．今回は棋譜単位で重要度を割り当てたが，同じ棋譜でも戦術や対局者は先手と後手で異なるため，先後で異なる重要度を用いることも有効と考えられる．

表 3.9，表 3.10に棋戦による重要度の違い，対局者による重要度の違いをそれぞれ示す．棋戦は，棋譜数が10局以上あるもの，対局者は，対局数の多いプレイヤ50人を分析対象とした．表 3.9から，棋戦の分析では，アマチュアや女流の棋戦の重要度が低く

表 3.9: 棋戦による重要度の違い

順位	棋戦	棋譜数	重要度の平均
1	名将戦	41	1.049
2	近将カップ	64	1.043
3	天王戦	85	1.025
4	十段戦	118	1.020
5	達人戦	27	1.020
…	…	…	…
31	全日プロ	400	0.989
32	朝日アマ名人戦	20	0.974
33	早指し戦	299	0.972
34	女流名人戦	62	0.946
35	全日アマ名人戦	10	0.917

表 3.10: 対局者による重要度の違い

順位	プレイヤ名	棋譜数	重要度の平均
1	中川大輔	159	1.040
2	北浜健介	125	1.033
3	屋敷伸之	279	1.028
4	三浦弘行	176	1.023
5	井上慶太	169	1.020
…	…	…	…
46	小林健二	174	0.980
47	神谷広志	107	0.979
48	中村修	219	0.977
49	島朗	297	0.973
50	日浦市郎	112	0.973

なる傾向があったが，プロ棋士の棋戦の間ではほぼ差は見られなかった．また，表 3.10 の結果から，一定以上の対局を行っているプレイヤ間では，棋譜の重要度に明確な傾向は得られなかった．

これらの結果から，今回の実験で用いた評価関数の特徴，および学習方法では，女流やアマチュア等明らかに実力の差がある棋譜は学習に悪影響をおよぼす可能性があるものの，一定以上の強さが保証されている場合には，棋戦やプレイヤの差はそれほど重要ではないと考えられる．

表 3.11: 実際の対局における戦術選択の違い

戦術	各戦術の選択回数	
	従来手法	提案手法
本美濃	131	77
高美濃	45	54
銀冠	61	175
流れ矢倉	28	15
金矢倉	33	45
居飛車穴熊	210	215
四間飛車穴熊	38	40
三間飛車穴熊	65	39

その他，以下に示す特徴での分析も行ったが，今回の実験ではいずれの項目も重要度は0.98–1.02の範囲に収まっており，明確な傾向は見られなかった．

- 対局日時（5年区切りの年代に分類）
- 持ち時間
- 戦型（戦術で分類しない）
- 終局までの手数（20手ごとに分類）

以上の結果から，今回の実験において学習された重要度は，(1)コンピュータが勝ちやすいと言われている戦術が重視されている，(2)教師としての質（プレイヤの強さ）に大きく差があるものは除外する方向に調整されている，といった傾向が読み取れる結果となった．

3.4.6　実際の対局時における戦術選択の違い

提案手法では，棋譜単位の重要度を用いた場合，より囲いを発展させる戦術の重要度が高くなる傾向があった．本実験では，この重要度の違いが，実際の対局においてどのように反映されているかを検証した．

表 3.11に，棋譜単位の重要度を用いた場合の2,000局の対局実験において，提案手法，従来手法のプログラムが選択した戦術の違いを示す．表 3.11から，今回の実験結果では，実際の対局時に選択された戦術として，美濃囲い（本美濃）と銀冠において，提案手法と従来手法に特に大きな差が表れていることがわかる．提案手法では，美濃囲い

のまま戦うよりも，高美濃，銀冠といった囲いに発展させている傾向が顕著に表れている．その他にも提案手法では，流れ矢倉よりも，より固さを重視した金矢倉が選択される割合が高くなるといった傾向が表れており，学習時の重要度の違いが，実際の対局時の戦術にも影響を与えていることが確認できた．

なお，穴熊については，実際の対局時には，提案手法で選択した割合は従来手法と比較して大きな差は見られなかった．これは，従来手法においても，穴熊は十分に高い価値を獲得しており，提案手法と差が生じにくかったことなどが理由として考えられる．

3.5　今後の課題

本章における実験では，教師データに重要度を導入し，対局による重要度の学習と重要度に基づく重み付き学習を繰り返すことによって，性能向上が実現できることを示した．

今後の課題としては，以下の2点が挙げられる．

(1) 重要度の学習手法，重要度を割り当てる単位の検討

(2) 他のゲームへの適用

(1)について，本提案手法では，重要度の学習には分布推定アルゴリズムの一種であるPBILcを用いた．分布推定アルゴリズムの手法としては，その他にも様々な手法が存在し[87, 88, 89]，それらの学習手法を用いることによってより精度の高い重要度が獲得できる可能性がある．また，本章では，実験の都合上，初期個体50，学習ステップ400という条件で学習を行ったが，これらの条件を増やすことによる性能向上も期待できる．今回の実験では，進行度単位，棋譜単位に重要度を割り当てた．局面単位，戦術単位，先手と後手で異なる重要度を用いるなど，重要度を割り当てる単位にも改善の余地があると考えられる．

(2)について，提案手法は「強さ」に基づいた学習であるため，将棋以外の多くのゲーム対して有効であることが期待できる．具体的には，近年研究が盛んな囲碁や，さらに対象とするゲームを限定しないGeneral Game Playing[90, 91]への適用については，検討する価値が高いと考えている．

3.6　まとめ

本章では，教師データに重要度を導入し，対局による重要度の学習と重要度に基づいたパラメータの重み付き学習を組み合わせた学習手法を提案した．重要度は，個体の強さを適応度とした進化的計算により学習した．強さを表す指標としては，大規模

な計算環境を用いて，個体間で対局を行い，その対局結果から算出したEloレーティングを用いた．

提案手法をコンピュータ将棋を対象として，(1)評価関数の学習，(2)実現確率の学習，(3)モンテカルロ木探索のplayoutの方策の学習，に適用した結果，従来手法を有意に上回ることに成功し，その有効性を示した．また，学習された重要度を分析した結果，終盤の重要度が高くなる，穴熊など一般的にコンピュータが勝ちやすいと言われている戦術の重要度が高い傾向となる，といった傾向となっており，提案手法を用いることによってゲームの性質にあった教師データの重要度が算出されていることが確認できた．

本章における貢献は，以下の2点である．

1. 学習データに重要度を導入し，その値を強さを基に学習することによって，従来手法を上回る強さのプログラムを学習することに成功したこと．

2. 提案手法が，評価関数，実現確率，playoutの方策といった現在のゲームにおける主要な学習において，汎用的に使用できることを実験によって示したこと．

現在，ゲームプログラミングの分野では，評価関数の学習，探索深さの調整，モンテカルロ木探索のplayoutの方策等，多く課題において棋譜を教師とした機械学習が取り入れられており，その重要性は，今後も増していくと考えられる．一方で，コンピュータの強さは多くのゲームにおいて確実に人間の強さに迫るものとなっており，今後は単純に人間の指し手を教師とした学習では不十分となる場合も考えられる．本章の提案手法は，教師データに対して重要度という形で意味付けを行うことで，単純な棋譜を教師とした学習を上回ることを示したものであり，今後このような学習手法はさらに重要になると考えられる．

第4章　評価関数の学習における目的関数の学習

本章では，前章の重要度の学習手法を拡張して，評価関数の学習を対象とした，目的関数の学習を提案する．評価関数の学習は，プログラムの強さへの影響が大きいことや，人間の知識を手作業で数値化することの困難さから，ゲームプログラミングにおいて中心的な研究課題の1つとされてきた．評価関数の学習手法として，現在，成功を収めている手法としては，エキスパートの棋譜を教師とした教師あり学習が挙げられる．特に将棋では，プロの棋譜を教師とした機械学習の成功により，現在のコンピュータ将棋は人間のトッププレイヤに迫る強さを得ている．しかし，現在の学習手法は，学習棋譜中のエキスパートの指し手とプログラムの指し手の一致率を向上させることを目的としており，学習によって得られるプログラムの強さを考慮したものではないという点に改善の余地がある．実際，チェスにおいては，人間の強いプレイヤとの指し手の一致率を向上させることが必ずしも強さを向上させるわけではないことが以前から示されている[35, 30]．評価関数の学習には，一般的に，コンピュータの指し手を，棋譜中のエキスパートの指し手に近づけるような目的関数が用いられるが，この目的関数については様々なものが検討されているものの，どれが優れているか明確な結論は得られていない．本章で提案する手法では，このような問題点を解決するために，従来用いられている目的関数に強さとの関係を表現できるパラメータを導入し，そのパラメータを，学習によって得られる評価関数を用いたプログラムの強さが最大化されるように調整する．

4.1　はじめに

人工知能の分野において，エキスパートの行動からの学習は広く研究されてきた[93, 94]．ゲームプログラミングでは，エキスパートの棋譜を教師とした学習が従来から研究されており，オセロ[4]，チェス[30]，将棋[31, 32]，囲碁[33]など，多くのゲームにおいて成功を収めている．

棋譜を教師とした機械学習が取り入れられている具体例としては，評価関数の学習，探索深さの調整，モンテカルロ木探索のplayoutの方策の学習，などが挙げられる．中

でも，特に，評価関数の学習については，プログラムの強さを決める重要な要因となること，また，人間の判断基準を数値化することの難しさから，ゲームの分野における中心的な課題の1つとされてきた．

エキスパートの指し手を教師とした評価関数の学習では，プログラムの指し手と棋譜中のエキスパートの指し手の一致率を向上させる目的関数を用いることにより，パラメータを学習する．チェスでは，1997年に初めて人間のチャンピョンを破ったIBM Deep Blueが，この手法を用いて玉の安全度に関する評価関数のパラメータを調整している[30, 95]．また，近年では，将棋においてプロの棋譜を用いた評価関数の学習が成功しており，盛んに研究が行われている[31, 92, 96, 97, 98, 99]．最新の将棋プログラムでは，ほぼすべてのプログラムが学習による評価関数を用いており，棋譜を教師とした学習が最も成功を収めたゲームの1つと言える．

しかしながら，これらの学習手法には，目的関数と学習されるプログラムの強さとの関係が不明であるという問題点が存在する．従来の目的関数は，学習棋譜中のエキスパートの指し手とプログラムの指し手の一致率を向上させるように設計されているため，その目的関数によって学習された評価関数を用いたプログラムが必ずしも強いプログラムとなっている保証はない．実際，エキスパートの指し手とプログラムの指し手の一致率を向上させることは必ずしも強いプレイヤを生み出さないということがチェスの研究において示されている[35, 30]．本章では，これらの問題点を解決するために，学習されるプログラムの強さを最大化するように，目的関数自体を学習する手法を提案する．

また，第3章の課題と共通して，従来手法による目的関数では，学習局面の性質が考慮されていない．学習棋譜に含まれる各局面の性質は均一ではないため，強い評価関数を学習によって得るためには，学習棋譜の性質に応じたパラメータの学習が必要になると考えられる．第3章で提案した学習棋譜の重要度の学習は，この問題に対する1つの改善策となっているが，本章では，この学習をさらに拡張し，目的関数のパラメータを学習局面の特徴に応じて制御する．

学習の手法としては，第3章と同様の枠組みを用いる．すなわち，多数の対局から算出したEloレーティングを適応度とした進化的計算によって目的関数のパラメータを調整する．従来手法による目的関数では，単に棋譜との一致率を向上させることを目的としていたのに対して，提案手法により学習された目的関数では,「強い」プログラムを学習するように，棋譜を教師とした評価関数の学習を行うことが可能となる．

以下に，本章の構成を示す．5.2節では，棋譜を教師とした評価関数の学習に関する関連研究について述べる．5.3節では，目的関数と強さの関係が不明であるという従来手法の問題点を解決する手法として，強さに基づいた目的関数の学習を提案する．5.4節では，提案手法の効果を示すため，対局実験，および学習された目的関数のパラメー

タの分析を行う．最後に，5.5節で今後の課題を述べ，5.6節で本章のまとめを述べる．

4.2 関連研究

エキスパートの行動を基にした学習は，ロボット，自然言語，画像認識，ゲーム等，人工知能の幅広い分野において研究されてきた[93, 100, 101, 102].

ゲームの分野では，オセロ[4]，チェス[30]，将棋[31, 32]，囲碁[33]など，多くのゲームにおいて人間などのエキスパートの棋譜を用いたパラメータの学習が行われてきた．これらの学習手法では，棋譜の指し手，あるいは棋譜の勝敗を教師として用いている．また，ゲームにおける機械学習の特徴的な点としては，学習の過程にゲーム木探索が組み込まれている点が挙げられる[103, 30, 104, 34, 31].

本節では，提案手法のベースとなる，棋譜を教師とした2つの評価関数の学習手法について述べる．1つは棋譜中の指し手を教師とした学習，もう1つは棋譜の勝敗を教師とした学習である．

4.2.1 棋譜の指し手を教師とした学習

棋譜の指し手を教師とした学習は，ゲームの分野において，最も良く用いられている学習手法と言える．本章では，評価関数の学習手法として，前章でも述べたComparison Training[30]をベースとする．Comparison Training は，棋譜中のエキスパートの指し手とそれ以外の合法手の比較によって，棋譜中の指し手の評価値が高くなるように評価関数の学習を行う手法である．以下に第3章で述べた，文献[31]で用いられている目的関数を再掲する．

$$J_1(P, \boldsymbol{v}) = \sum_{p \in P} \sum_{m=2}^{M_p} T\big(\xi(p_m, \boldsymbol{v}) - \xi(p_1, \boldsymbol{v})\big) \tag{3.1}$$

Pは学習棋譜に含まれるすべての局面の集合，M_pは局面pにおける合法手の数，p_mは局面pにおいて合法手mを指した時の局面を示す．ここで，$m = 1$の場合は実際の棋譜において選択された指し手を示す．また，$\xi(p_m, \boldsymbol{v})$はp_mにおいて探索を行った場合の末端局面の特徴$\boldsymbol{v}$によって算出された評価値を示す．また，$T(x)$はシグモイド関数$T(x) = \frac{1}{1+\exp(-a \cdot x)}$を表す．定数 a $(a > 0)$ はシグモイド関数の傾きを制御するパラメータである．

式 (3.1)の目的関数では，正解手（棋譜中の指し手）を選択した時の探索結果の評価値が，その他の手を選択した時の評価値を十分に上回るように，評価関数のパラメータを調整することを目的としている．この手法を用いた評価関数の学習によって，コ

ンピュータ将棋では，人間のトッププレイヤに迫る強さを得ることに成功している．また，現在では，ほぼすべての将棋プログラムがこの手法によって学習された評価関数を用いている．

4.2.2　棋譜の勝敗を教師とした評価関数の学習

文献[96, 104]では，棋譜の勝敗を教師とした評価関数の学習が提案されている．棋譜の勝敗と教師とした学習では，評価値から勝敗を精度良く求めることができるように，評価関数のパラメータを調整する．この手法では，ロジスティック回帰モデルによって勝敗の予測を行う．具体的には，ある局面の評価値がxの場合の勝率$g(x)$を以下の式によって予測する．

$$g(x) = \frac{1}{1 + \exp(x)} \tag{4.1}$$

ここで，式 (4.1)と局面pにおける勝敗との尤度は以下の式で表される．

$$likelihood(p, y_p) = g\big(\xi(p_1, \boldsymbol{v})\big)^{y_p} \big(1 - g(\xi(p_1, \boldsymbol{v}))\big)^{(1-y_p)} \tag{4.2}$$

y_pは局面pでの勝敗（先手勝ちの場合1，後手勝ちの場合0）を表す.

ここで，以下の式に示す，すべての局面についての負の対数尤度を最小化することにより，局面の評価値から勝敗を精度良く予測できるように評価関数のパラメータを調整することができる．

$$J_2(P, \boldsymbol{v}) = -\sum_{p \in P} \Big\{ y_p \cdot \log\big(g(\xi(p_1, \boldsymbol{v}))\big) + (1 - y_p) \cdot \log\big(1 - g(\xi(p_1, \boldsymbol{v}))\big) \Big\} \tag{4.3}$$

なお，この学習手法は，主にComparison Trainingなどで一度学習された評価関数の調整に用いられることが多く，勝敗を教師とした学習のみでは，十分な効果を得ることは難しいと考えられている．この理由としては以下の原因が考えられる．

- Comparison Trainingが指し手を教師とするのに対して，勝敗を教師とした学習では局面を教師とするため，教師データの数が少なくなる．
- 棋譜毎に与えられた勝敗の値を，棋譜に含まれるすべての学習局面のラベルとするため，局面単位で見た時の実際の優劣はわからず，個々の学習局面に付与されたラベルの信頼性が高くない．

この様な欠点から，棋譜の勝敗を教師とした学習は，オセロなどの比較的単純なゲームでは成功を収めたものの[4]，チェスや将棋では，単体では十分に強い評価関数の学習には成功していない．

一方で，この手法は，一度学習された評価関数の調整には優れており，いくつかのトップレベルの将棋プログラムではこの手法が取り入れられている．

4.3 提案手法

前節で述べたように，エキスパートの棋譜を教師とした評価関数の学習はいくつものゲームにおいて成功を収めている．

しかしながら，従来手法では，学習棋譜中のエキスパートの指し手を真似ることを目的としており，学習された評価関数を用いたプログラムがどのような強さになるかは考慮されていない．将棋における研究では，学習に用いる目的関数はいくつかの種類のものが検討されている[105]が，どのような目的関数が優れているか，明らかな結論は得られていない．式 (3.1)の目的関数を用いた場合に限定しても，シグモイド関数$T(x)$の形状によって，学習の結果として得られる評価関数の値には違いが生じるが，これらの目的関数に関する重要なパラメータは，今なお手作業により調整されている．

さらに，学習棋譜に含まれる各局面は，それぞれ異なる特徴を持っている．学習局面の特徴としては，例えば，進行度，合法手の数や正解手を求める難しさ，正解手を指すことによって勝敗に与える影響の大きさなどが考えられる．これらの特徴の異なる学習局面を従来手法では均等に扱っていたが，学習局面の特徴を捉えることで，より強いプログラムの学習が可能となると考えられる．

本章では，これらの従来手法の問題点を解決する手法として，目的関数に局面の特徴に応じたパラメータを導入し，そのパラメータをプログラムの強さを基に最適化する手法を提案する．以降，本節では，学習局面の特徴に応じたパラメータを導入した新たな目的関数の設計と，そのパラメータの学習手法について述べる．

4.3.1 パラメータ化された目的関数の設計

提案手法では，まず，現在将棋の評価関数の学習において一般的に用いられている式 (3.1)と式 (4.3)を組み合わせた新たな目的関数を設計し，目的関数と学習されるプログラムの強さを表現できるよう，局面の特徴に応じたパラメータを導入する．

まず，提案手法のベースとなる，目的関数の式 (3.1)にパラメータを導入する．式 (3.1)は現在，将棋では最も一般的に用いられている目的関数である．この目的関数では，シグモイド関数$T(x)$の形状が，学習される評価関数の値に強い影響を及ぼす．本章の提案手法では，式 (3.1)中のシグモイド関数に目的関数と学習されたプログラムの強さを表現できるよう2つのパラメータを導入する．

$$T(x) = \frac{1}{1 + \exp(-W_{1,p} \cdot (x + W_{2,p}))} \tag{4.4}$$

ここで，$W_{1,p}$，$W_{2,p}$は局面pの特徴によって決定される値であり，シグモイド関数の形状を制御するパラメータである．$W_{1,p}$はシグモイド関数の傾き，$W_{2,p}$は正解手とそ

れ以外の合法手の評価値の差を制御するパラメータである．

次に，2つの目的関数の式 (3.1)と式 (4.3)を組み合わせた新たな目的関数を設計する．式 (3.1)は学習棋譜中のエキスパートの指し手とプログラムの指し手の一致率を向上させる目的関数であり，式 (4.3)は勝敗を精度良く予測できるよう評価関数のパラメータを調整する目的関数である．2つの目的関数を組み合わせることによって，異なる性質を持つ学習を取り入れることが可能になると期待できる．

目的関数を組み合わせる際，各目的関数に与える重みの大きさを決定する必要がある．それぞれの目的関数は異なる性質を持つため，局面の特徴によってどちらの目的関数を重視するべきかについても違いが生じると考えられる．提案手法では，以下の式に示すように，各目的関数に局面の特徴に応じたパラメータ$W_{3,p}$，$W_{4,p}$を付与し，局面の特徴に応じて，目的関数の重みを制御できるように変更した．

$$J_1'(P,\boldsymbol{v}) = \sum_{p\in P} W_{3,p} \sum_{m=2}^{M_p} T\big(\xi(p_m,\boldsymbol{v}) - \xi(p_1,\boldsymbol{v})\big) \tag{4.5}$$

$$J_2'(P,\boldsymbol{v}) = -\sum_{p\in P} W_{4,p}\Big\{ y_p \cdot \log\big(g(\xi(p_1,\boldsymbol{v}))\big) + (1-y_p)\cdot \log\big(1-g(\xi(p_1,\boldsymbol{v}))\big)\Big\} \tag{4.6}$$

最終的には，式 (4.5)と式 (4.6)を組み合わせた以下の目的関数を用いる．

$$J(P,\boldsymbol{v}) = J_1'(P,\boldsymbol{v}) + J_2'(P,\boldsymbol{v}) + w_0\cdot|\boldsymbol{v}| \tag{4.7}$$

ここで，$|\boldsymbol{v}|$はL1正則化，w_0は正則化項の重みを表す．

4.3.2　局面の特徴の導入

提案手法では，式 (4.7)の目的関数に導入されたパラメータ$W_{1,p}$，$W_{2,p}$，$W_{3,p}$，$W_{4,p}$，および正則化の重みw_0を評価関数の強さを最大化するように制御する．

ここで，$W_{i,p}$　$(i=1,2,3,4)$は局面の特徴に応じた値$w_{i,j}$の積によって計算する．

$$W_{i,p} = c_i \prod_{j\in C_p} w_{i,j} \tag{4.8}$$

C_pは学習局面pに含まれる局面の特徴を表し，$w_{i,j}$は特徴jに対するパラメータの値を表す．このような式を用いることで，学習局面の特徴に応じて目的関数を変形させることが可能となる．定数c_iの値は，Bonanzaで用いられている値[43]，および予備実験によって決定した$(c_1=0.0237,\, c_2=64,\, c_3=1,\, c_4=16)$．

本章では，以下に示す3種類の局面の特徴を用いた．

- 進行度

  進行度は，ゲームの分野では，局面の性質を表す指標として以前から広く用いられている[4]．将棋では進行度の計算方法には様々なものが存在する[106, 6, 107]が，本章では，最も単純な指標としてある時点の手数を終局までの手数で割った値を基に，8段階に分類したものとした．

- 最善手が変わった回数（Best-move Change）

  最善手が変わった回数（以降，Best-move Change）は，対象の局面において探索深さを徐々に深くしていったとき，最善手が変化した回数を示す．この指標は，チェスの文献[108]でも用いられており，Best-move Changeの値が大きい局面は，コンピュータにとって複雑な局面と考えることができる．本章の実験では，各学習局面に対して深さ3までの探索を行い，最善手が変化した回数（0回〜2回）をBest Changeの値とした．

- 正解手を求めるために要した探索深さ（Correct Depth）

  本章では，各学習局面に対して探索を行った結果，正解手（棋譜の手）を選択できるのに要した探索深さ（以降，Correct Depth）も特徴の1つとして使用している．プログラムが正解手を選択できない原因としては，評価関数が適切でない場合と，探索深さが十分ではない場合の2種類が存在する．評価関数の学習では，一般的に浅い探索を用いた学習が行われるため後者が原因で正解手を選択できない局面も多いが，本質的に学習したい局面としては，前者がより重要であると考えられる．本章では，Best-move Change同様，各学習局面に対して深さ3までの探索を行い，正解手を求めるのに要した探索深さ（1，2，3，不正解）をCorrect Depthの値とした．

これらの特徴は，将棋特有の知識に依存したものではないため，チェスなど他のゲームにおいても使用できる特徴となっている．

4.3.3　目的関数の学習

目的関数のパラメータは，前章の提案手法と同様の枠組み（強さを適応度とした進化的計算）によって学習する．図 4.1に目的関数のパラメータを学習する手順を示す．

前章と同様，強さの指標としてはEloレーティングを用い，目的関数のパラメータの学習にはEloレーティングを適応度としたPBILcを用いる．本章では，重要度の代わりに，目的関数に導入された局面の特徴に応じたパラメータ$w_{i,j}$を学習する．

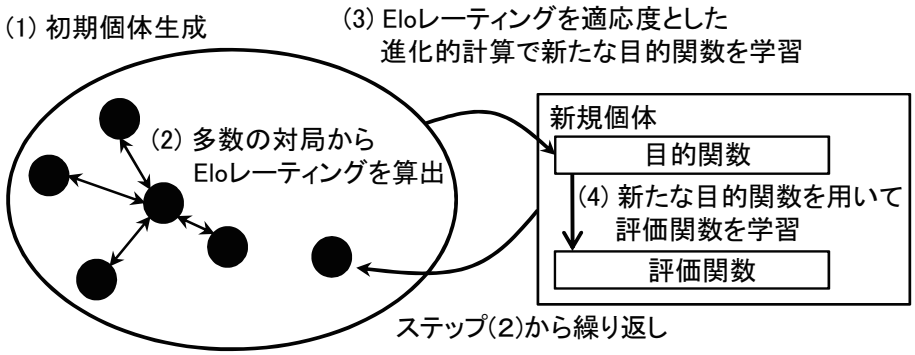

図 4.1: 目的関数のパラメータを学習する手順

4.4　実験

4.4.1　実験環境

学習に使用した環境を表 4.1に示す．本実験では，99台のクラスタ環境を用いて目的関数の学習を行った．目的関数の学習には，約10日を要した．学習には，プロやアマチュア高段者の棋譜10,000局を用いた．

表 4.1: 実験環境

CPU	メモリ	ノード数
Xeon E5-2680 × 2	132GB	64
Core i7-3930K	16GB	7
Xeon X3440	8GB	8
Core2 Quad Q9650	8GB	10

評価関数の特徴としては，前章の実験と同様，以下の項目を用いた．

1. 駒の価値
2. 玉との1つの駒の位置関係
3. 利きが関連する駒の位置関係のパターン

使用した特徴の総数は約40万となっている．なお，学習の際，駒の価値は予め定められた値を初期値とした．これは，学習の際に生じるローカルミニマムの問題[97]を軽

減すること，また，学習時間の短縮を目的としたものである．また，実験を簡単にするため，評価値$\xi(p_m, \boldsymbol{v})$の算出には，静止探索[22]を使用した．

PBILcのパラメータは，個体数$N = 50$，学習率$\alpha = 0.0025$，$K = 10$，反復回数は4,000回とした．本手法では，個体を1つずつ入れ替える処理をとっているため，初期収束を防ぐため学習率αは小さな値を採用している．また，PBILcではKの値は，学習結果に大きな影響を及ぼさないことが知られている[81]ため，10で固定とした．

4.4.2　対局実験による評価

提案手法の有効性を示すため，対局実験による評価を行った．実験では，提案手法を用いて最適化された目的関数を用いて学習された評価関数を用いたプログラムと，従来手法の目的関数で学習された評価関数を用いたプログラムで対局を行った．従来手法を用いたプログラムとしては，以下に示す2種類のプログラムを用いた．

- Comparison: 式 (3.1)により評価関数を学習
- Combination: 式 (4.7)により評価関数を学習

Comparisonは，Bonanzaを始めとして，将棋の評価関数の学習に良く用いられている目的関数の式 (3.1)を用いて評価関数を学習したプログラム，Combinationは，関連研究で示した2つの目的関数を単純に足し合わせた目的関数の式 (4.7)（パラメータの学習を行っていないもの）を用いて評価関数を学習したプログラムを用いた．Combinationのプログラムでは，W_1, W_2，W_3の値はBonanzaと同一とし，W_4の値は予備実験によって決定した($W_1 = 0.0237$，$W_2 = 0$，$W_3 = 1$，$W_4 = 16$，$w_0 = 0.00625$)．なお，すべての学習において，L1正則化を行っている．

表 4.2に提案手法により学習された評価関数を用いたプログラムと，従来手法のプログラムの対局結果を示す．思考時間は，各プログラムともに1手10万ノードとした（太字は有意水準5%の二項検定で有意）．

表 4.2の実験結果から，提案手法が従来手法の2つのプログラムを大きく上回っており，提案手法によって最適化された目的関数によって「強い」評価関数が学習できていることがわかる．特に，現在標準的に用いられているComparisonに対しては，6割以上の勝率を得た．

表 4.3は各手法によって評価関数を学習したプログラムをBonanza (version 6.0）と対局させた結果である．この実験の目的は，他のプログラムと対局を行うことで，提案手法によって学習された評価関数の強さが，自己対局に最適化されたものではないことを確認することである．本実験では，Bonanzaの思考時間は，1手12,000ノードとし

表 4.2: 従来手法のプログラムに対する勝率

	提案手法の勝率 (勝ち-負け-引き分け)
vs Comparison	**0.619** (1169-719-112)
vs Combination	**0.576** (1063-782-155)

表 4.3: Bonanza (12,000 nodes) に対する勝率

	Bonanzaに対する勝率 (勝ち-負け-引き分け)
Comparison	0.497 (975-987-38)
Combination	**0.531** (1033-913-54)
提案手法	**0.615** (1205-754-41)

た．これは，Comparisonとほぼ同等の強さにすることで，提案手法，Comparison, Combination間の比較を行いやすくするためである．

表 4.3から，他のプログラムとの対局結果においても，表 4.2の結果と同様，提案手法が従来手法のプログラムを大きく上回っており，提案手法によって学習された評価関数の「強さ」が，自己対局に特化したものではなく，他のプログラムに対しても有効であることがわかる．

4.4.3　学習された目的関数のパラメータ

本節では，提案手法によって学習された目的関数のパラメータを，局面の特徴（進行度，Best-move Change，Correct Depth）ごとに分析する．なお，L1正則化の重みを表す値w_0については，ほとんど変化がなかったため，ここでは議論しない(学習前0.00625，学習後0.00607)．

4.4.3.1　進行度

図 4.2に学習された進行度に応じた目的関数のパラメータを示す．進行度は1に近づくほど終盤に近いことを意味する．

$w_{1,i}$は目的関数中のシグモイド関数の傾きを制御するパラメータである．図 4.2から，$w_{1,i}$の値は，進行度に従って小さくなっており，シグモイド関数の傾きが緩やかとなるような値が学習されていることがわかる．これは，序盤と終盤で正解手とそれ以外の

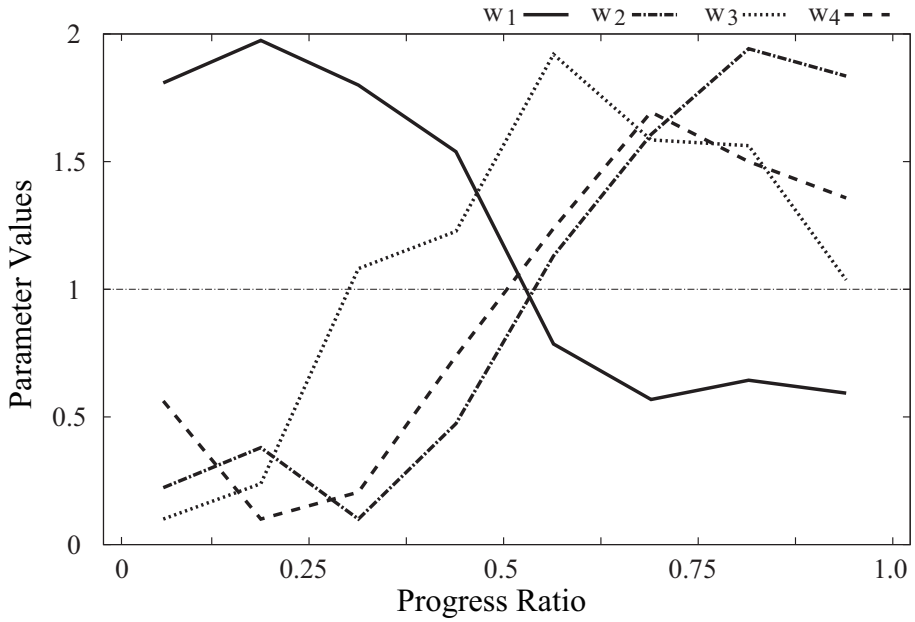

図 4.2: 進行度に応じたパラメータの値

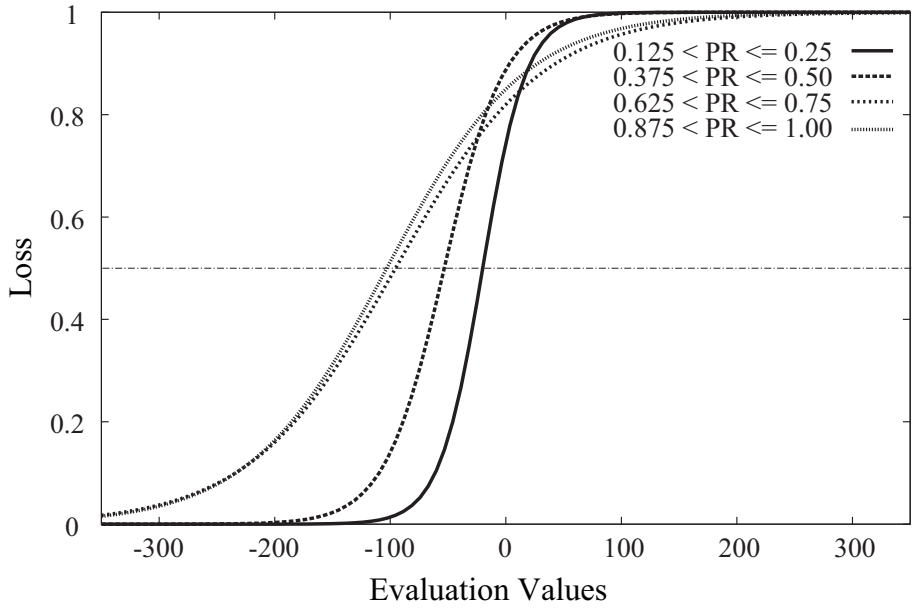

図 4.3: シグモイド関数の形状 (進行度: PR)

評価の差が大きくなるためであると考えられる．シグモイド関数による目的関数は，傾きを持つ領域に含まれるデータのみが学習対象として有効となる性質を持ち，正解手と評価値が大きく離れている学習データ（指し手）を学習対象から除外する効果を持つ．終盤では，序盤に比べ，評価に差がつきやすい状況となるため，広い範囲で傾きを持つようなパラメータの値が算出されたと考えられる．

この，序盤と終盤の評価値の差の違いは，正解手とそれ以外の指し手の評価値の差を調整するパラメータ$w_{2,i}$にも影響している．図 4.2から，$w_{2,i}$の値は，進行度に従って大きな値となっていることがわかる．これは，終盤ほど正解手以外の指し手は決定的な悪手となることが多いため，評価値の差を大きくするようなパラメータが学習されたと考えられる．一方で，将棋の性質として，序盤では有力な指し手が複数存在し，どの手を選択しても優劣に大きな差がない局面が多く存在するため，評価値の差を小さくするような値が学習されたと考えられる．

図 4.3に，進行度に関するパラメータ$w_{1,i}$と$w_{2,i}$を反映させたときのシグモイド関数の形状を示す．図 4.3から，から，進行度に従って広い領域の傾きを持ち，正解手との差を大きくするような関数の形状になっていることがわかる．

パラメータ$w_{3,i}$および$w_{4,i}$は基本となる目的関数の式 (4.5)，式 (4.6)の進行度に応じた重みを表している．図 4.2から，$w_{3,i}$，$w_{4,i}$ともに，進行度に応じて大きな値となっていることがわかる．これはゲームの性質として，序盤に比べ，終盤のほうがより勝敗に直結しやすい性質があるため，終盤で正解手を選択できることを優先しているためであると考えられる．この結果は，第3章の進行度に基づく重要度の学習と同様の傾向となっていると言える．また，序盤から中盤では$w_{3,i}$が，$w_{4,i}$の値と比較して，相対的に大きな値となっていることがわかる．これは，序盤，中盤では勝率が0.5 付近に偏りやすいため，勝敗の予測により適切な評価関数の学習が難しいことが原因として考えられる．一方，そのような局面においても指し手の一致による学習は有効であると考えられる．これらの結果から，異なる性質を持つ目的関数の組み合わせにおいて，学習データの性質に応じた重み付けが実現できていると考えられる．

4.4.3.2　Best-move Changeに応じたパラメータの値

図 4.4に，Best-move Changeの値に応じた，目的関数のパラメータの値を示す．

図 4.4から，$w_{3,i}$，$w_{4,i}$の値がBest-move Changeの値に従って，小さくなっていることがわかる．Best-move Changeの値は，駒を取る手や王手など，評価値に影響を与える指し手が複数存在する局面において大きな値となりやすい．これらの局面では，探索深さを増やす毎に評価値や最善手が変化しやすく，局面が静止していない状態になっていると言える．

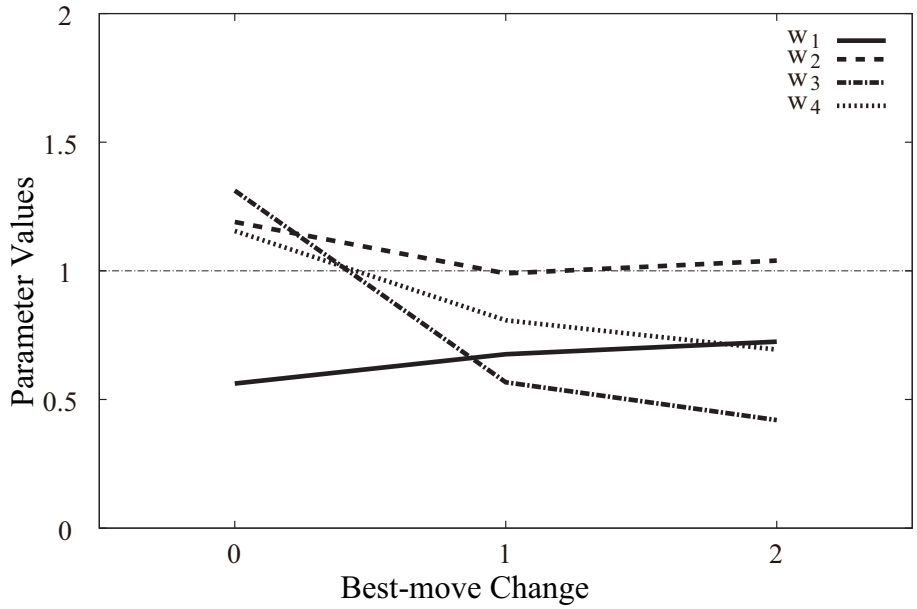

図 4.4: 最善手が変化した回数（Best-move Change）に応じたパラメータの値

図 4.4から，Best-move Changeの値が大きくなるに従って$w_{3,i}$，$w_{4,i}$の値が減少していることから，評価関数の学習では，学習に用いられる探索の末端局面が静止している局面を重視していると言える．頻繁に探索結果が変わる局面では，末端の学習局面が安定しないため，評価関数の学習において，静止している局面を重視することは理にかなっていると考えられる．

シグモイド関数の形状を制御するパラメータ$w_{1,i}$，$w_{2,i}$については，Best-move Changeによる値の変化は小さい結果となった．$w_{1,i}$はBest-move Changeの値に従ってやや大きな値となっており，最善手が変わりやすい，複雑な局面ではシグモイド関数が傾きを持つ領域を広く取るように調整されていることがわかる．

4.4.3.3　Correct Depthに応じたパラメータの値

図 4.5に，Correct Depthに応じた目的関数のパラメータの値を示す．図 4.5において，Correct Depthの値「>3」は，深さ3までの探索において，正解手を選択できなかったことを表す.

図 4.5では，$w_{3,i}$について，2つの特徴的な傾向が見られる．1点目として，$w_{3,i}$の値はCorrect Depthの値が小さいときに大きな値となっている．これは， Correct Depthの値が小さい局面は，静止した局面となっており，評価に適した局面となっているためであると考えられる．通常のalpha-beta探索でも，局面を静止させるための静止探索が用い

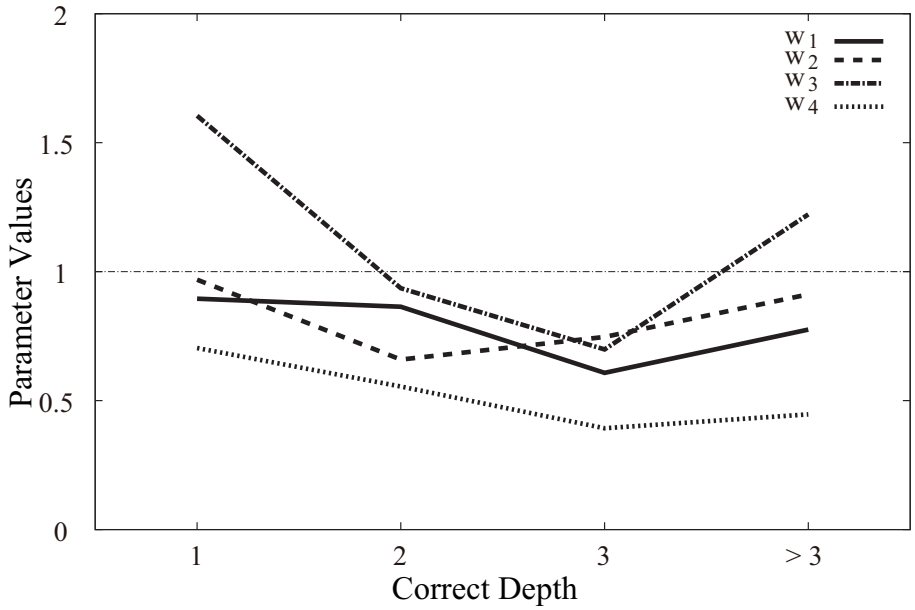

図 4.5: 正解手を求めるために要した探索深さ（Correct Depth）に応じたパラメータの値

られるように，静止していない局面（探索によって評価値が変化する，有力な指し手が存在する局面）は評価関数により正しい評価を行うことが難しいと言える．これは，学習時にも同様と考えられ，その結果，Correct Depthの値が小さい，静止した局面を重視して学習するように目的関数のパラメータが学習されたものと考えられる．

一方で，深さ3までの探索において正解手を選択できなかった局面（>3）についても，$w_{3,i}$の値が大きくなっていることがわかる．プログラムが正解手を選択できない理由としては，探索の深さが足りない場合と評価関数が不適当な場合が存在する．Correct Depthを求める際に正解手を選択できなかった局面に大きな重みを与えることで，後者の評価関数が不適当であった局面を重視した学習を行う目的関数となっていると考えられる．

なお，その他のパラメータの傾向として，$w_{4,i}$については，変化の度合いは小さいもの$w_{3,i}$と同様の傾向の結果となった．シグモイド関数の形状を制御するパラメータ$w_{1,i}$，$w_{2,i}$については，Correct Depthの変化によって明らかな傾向は見られなかった．

Correct Depthに応じたパラメータの値の傾向として，プログラムがなぜ正解手を指せていないかを目的関数に反映させるパラメータの学習が行われていると言える．

表 4.4: 4.4.2節とは異なる評価関数の特徴と学習棋譜を用いた実験の対局結果

	提案手法の勝率 (勝ち-負け-引き分け)
vs Comparison	**0.592** (1107-762-131)
vs Combination	**0.566** (1051-805-144)

表 4.5: Bonanza (15,000 nodes) に対する勝率 （4.4.2節とは異なる評価関数の特徴と学習棋譜を用いた場合）

	Bonanzaに対する勝率 (勝ち-負け-引き分け)
Comparison	0.506 (989-965-46)
Combination	**0.544** (1070-898-32)
Proposed method	**0.587** (1151-810-39)

4.4.4 学習された目的関数の再利用性

提案手法によって学習された目的関数は，異なる特徴を持つ評価関数の学習や，異なる学習棋譜を用いた場合の評価関数の学習においても有効であると期待できる（以降，この性質を目的関数の再利用性と呼ぶ）.

本節では，提案手法により学習された目的関数の再利用性を検証するため，異なる評価関数と学習棋譜を用いた場合の有効性を検証した．学習棋譜としては，目的関数の学習時とは別の学習棋譜10,000局を用いた．また，学習する評価関数の特徴は以下のように変更した.

- 駒の価値
- 2つの玉とその他1つの駒の位置関係
- 1つの玉とその他2つの駒の位置関係

これらの評価関数の特徴は，Bonanzaで使用されているものと同一である.

実験では，4.4.2節の対局実験と同様，提案手法で最適化された目的関数を用いて学習した評価関数を用いたプログラムと，従来の目的関数で学習された評価関数を用いたプログラム（Comparison，Combination）で対局実験を行った．実験条件は4.4.2節と同様，思考時間は1手10万ノードとし，対局数は2,000局とした.

表 4.4に提案手法と従来手法のプログラム（Comparison，Combination）の対局結果，表 4.5に各プログラムとBonanzaとの対局結果を示す（太字は有意水準5%の二項検定で有意）．

表 4.4および表 4.5から，いずれの実験においても，提案手法を用いて最適化された目的関数で学習された評価関数を用いたプログラムが，従来手法によるプログラムを上回る結果となった．実験結果から，提案手法で学習された目的関数は，異なる特徴の評価関数や異なる学習棋譜を用いた場合にも有効であり，再利用性を持つことが確認できた．この性質を利用することで，プログラムを変更し，評価関数を学習し直す際などにおいて，目的関数を学習し直さずに評価関数の学習を行うことができる．提案手法による目的関数の最適化はコストの大きい処理になるため，何度も目的関数から学習し直さなくても良い点は，実用的な観点から見たときに有益であると考えられる．

4.5　今後の課題

本章では，前章の教師データの重要度の学習手法を拡張し，棋譜を教師とした将棋の評価関数の学習を対象に,「強い」プログラムを学習するよう，目的関数自体のパラメータを学習する手法を提案した．

今後の重要な課題として，再利用性のより詳細な検討が挙げられる．実験では，異なる評価関数と異なる学習棋譜を用いた場合にも提案手法で学習された目的関数が従来の目的関数を上回ることが示された．一方で，評価関数の特徴にどの程度の違いがあった場合に再利用性を確保できるか，どのような場合に目的関数の再学習が必要となるかについては，今後より詳細な実験が必要であると考えられる．関連して，異なるゲーム間の目的関数の再利用性についても検討を行う価値があると考えられる．例えば，チェスと将棋は，似たような性質を持つゲームといえ，このようなゲーム間では，将棋で学習された目的関数がチェスの学習でも使用可能である可能性がある．このような複数のゲームに共通する枠組みは，現在ゲームの分野で研究が盛んなGeneral Game Playing等へ応用できる可能性もあり，今後検討する価値は高いと考えられる．

また，前章の課題と同様，提案手法の他のゲームへの適用も今後の課題として挙げられる．提案手法は，強さをベースとした学習手法であるため，ゲームにおける多くの課題に対する機械学習に適用可能である．例えば，モンテカルロ木探索におけるplayoutの方策では，文献[3]に示すような棋譜を教師とした機械学習が用いられることが多いものの，どのようなplayoutの方策が強いプログラムに結びつくのか，現在までに明らかにはなっていない．前章の重要度の学習が一定の効果を得たように，提案手法はこのような問題に対して効果を発揮できると期待できる．

4.6 まとめ

本章では，将棋の評価関数の学習を対象に，強さとの関係を表現するためのパラメータを導入し，そのパラメータを対局から得たEloレーティングを適応度とした進化的計算によって学習する手法を提案した．

現在，ゲームプログラミングの分野では，多くのパラメータの調整において，棋譜を教師とした機械学習が行われている．中でも，将棋の評価関数の機械学習は，特に大きな成功を収めており，多くのプログラムで取り入れられている．一方で，従来手法による棋譜を教師とした評価関数の学習では，棋譜の指し手との全体的な一致率を向上させることを目的としており，目的関数と学習によって得られるプログラムの強さの関係が不明であるという問題点が存在した．また，前章と同様，学習局面の性質を考慮していない点にも改善の余地が存在した．

これらの課題を解決するために，本章では，前章の教師データの重要度の学習における提案手法を拡張し，目的関数のパラメータを，学習された評価関数を用いたプログラムの強さを用いて学習する手法を提案した．提案手法では，目的関数に，学習された評価関数の強さとの関係を表現することができるよう，局面の特徴に応じたパラメータを導入した．目的関数のパラメータは，前章と同様の手法，個体間の対局から算出したEloレーティングを適応度とした進化的計算（PBILc）によって,「強い」評価関数を学習するように学習した．

本章では，提案手法を将棋の評価関数の学習へ適用し，その効果を検証した．実験の結果，提案手法によって学習された目的関数を用いて学習された評価関数を用いたプログラムが従来手法を用いて学習されたプログラムを有意に上回り，提案手法の有効性を示した．学習された目的関数のパラメータを分析した結果，局面の特徴と目的関数の関係を捉えたパラメータが学習されていることを確認した．また，提案手法によって学習された目的関数を用いて学習したプログラムは，再利用性を有し，異なる特徴を持つ評価関数や異なる学習棋譜を用いた場合にも有効であることを示した．

本章の主要な貢献は以下の2点である．

1. 目的関数に，局面の特徴に応じたパラメータを導入し，そのパラメータをEloレーティングを適応度とした進化的計算で学習することで,「強い」評価関数を学習可能となる目的関数を得ることができることを示したこと．

2. 学習された目的関数は，異なる評価関数の学習や異なる学習棋譜を用いた場合にも有効であり，再利用性を持つことを示したこと．

ゲームプログラミングの分野において，評価関数の機械学習は幅広いゲームで研究され，中心的な課題の1つとされてきた．一方で，評価関数のパラメータを学習する際に用いる目的関数自体を調整する研究は非常に少なく，ほとんどの場合では，単純に

エキスパートの指し手を真似ることを目的とした学習が行われている．本章で提案した目的関数の学習手法は,「どのような特徴の学習局面を，どのように学習すれば強い評価関数が得られるか」を学習可能な手法となっており，単純な模倣による学習を上回る可能性のある手法として，多くのゲームに対して現在の性能を上回る結果が得られる可能性があると考えられる．

第5章　結論

5.1　本論文のまとめ

本論文では，ゲームプログラミングにおいて用いられている，棋譜を教師とした機械学習を対象に，学習データの性質と強さの関係を取り入れた学習を提案した．現在のゲームプログラミングでは，棋譜を教師とした学習が成功しているが，従来手法には，以下に示す2つの課題が存在した．

(1) 棋譜の手を模倣することを目的としており,「強さ」を目的とした学習になっていない

(2) 学習棋譜の性質が考慮されていない

本論文では，これらの問題点を解決する手法として，学習棋譜の性質と強さの関係を取り入れた学習手法として，(1)教師データの重要度の学習，(2)目的関数の学習，の2つの手法を提案した．実験では，対象のゲームとして将棋を用い，ゲームにおける主要な学習（評価関数の学習，探索深さの調整，モンテカルロ木探索のplayoutの方策の学習）に提案手法を適用し，その有効性を示した．以下に本論文で得られた知見を各章毎にまとめる．

第2章では，思考ゲームの新たな実現法であるモンテカルロ木探索を将棋へ適用し，その可能性を検証した．モンテカルロ木探索は，囲碁では大きな成功を収めたものの，将棋などチェスライクなゲームへの適用を検討した研究は少なかった．特に将棋では，詰みという終局条件をランダムなシミュレーションで満たすことの難しさから，従来研究では，単純にモンテカルロ木探索を将棋へ適用した場合の性能は非常に低いものとなっていた．本論文では，モンテカルロ木探索によるコンピュータ将棋を実現するため，以下の手法を用いた．

- 囲碁において成功している手法である，UCTによる探索やEloレーティングによるplayoutの方策を将棋向けに適用
- チェスや将棋で古くから用いられている，キラームーブやヒストリーヒューリスティックをモンテカルロ木探索に応用

問題集による評価では，単純なモンテカルロ木探索を用いたプログラムの正答数は，従来研究と同様非常に低いものであった．一方，本論文で示した改良を適用したモンテカルロ木探索によるプログラムでは，アマチュア初段程度のプログラムに迫る正答数を得ることに成功し，本論文で用いた各手法により大幅な性能改善を実現できること示した．特にplayoutの方策に機械学習による指し手の選択を導入した効果は大きく，将棋を対象としたモンテカルロ木探索においても，playoutの方策における機械学習は重要な位置付けであることが示された．また，機械学習を取り入れることでランダムな指し手の選択では終局条件を満たすことが難しいという将棋特有の課題を改善できていることが確認できた．

将棋におけるモンテカルロ木探索の有用性という観点では，現在のコンピュータ将棋は人間のトッププレイヤに匹敵する強さに達しており，単純に従来の手法を上回ることは難しいと考えられる．一方で，序盤の定跡選択や一部の終盤では，従来の手法を上回る有望な結果を得た．現在のコンピュータ将棋の課題として，長期的な見通しが必要になるような局面や有力な指し手に絞った直線的な深い読みが必要となる局面への対応が難しいといった点が挙げられる．序盤の定跡選択や，本論文中で例として挙げた終盤の局面はこのような局面に該当し，従来手法で良い結果を得ることは現在のところ困難である．一方，モンテカルロ木探索は終局までのゲームのシミュレーションが直線的な読みの働きをするため，探索と評価関数による思考が不得意とする局面において，有効性を発揮できたと考えられる．以上の結果から，コンピュータ将棋において，モンテカルロ木探索単体で従来手法を上回ることは難しいものの，その部分的な利用は棋力の向上に有効となり得ることを示した．

第3章では，ゲームプログラミングにおける主要な機械学習に対して，教師データに重要度を導入し，対局結果に基づいた重要度の学習と，重要度に従ったパラメータの学習を組み合わせた手法を提案した．

現在のゲームプログラミングにおける機械学習では，人間などのエキスパートの棋譜を教師とした学習が成功しており，評価関数の学習，探索深さの調整，モンテカルロ木探索のplayoutの方策の学習など，幅広い課題において用いられている．

棋譜を教師とした学習の特徴として，学習によって得られるパラメータが学習データの性質に大きく依存するという点が挙げられる．しかし，学習棋譜に含まれる局面は，その性質や信頼性において均一ではないという課題が存在した．具体的には，学習棋譜には，様々な強さのプレイヤ棋譜が含まれる，人間のプレイヤの棋譜には悪手も多数含まれる，といった課題が存在する．また，特徴の違いという観点では，学習局面によって，戦術や正解手を選択することの重要性に違いが存在する．

第3章では，これらの課題を改善する手法として，教師データに重要度を導入し，対局による重要度の学習と重要度に基づく重み付き学習を組み合わせた学習手法を提案した．重要度は，個体の強さを適応度とした進化的計算により学習した．強さを表す指標としては，大規模な計算環境を用いて個体間で対局を行い，その対局結果から算出したEloレーティングを用いた．また，学習手法としては，分布推定アルゴリズムの一種であるPBILcを用いた．

実験では，提案手法をコンピュータ将棋を対象として，(1)評価関数の学習，(2)実現確率の学習，(3)モンテカルロ木探索のplayoutの方策の学習，に適用した．ここで，モンテカルロ木探索の実験には第2章の手法をすべて適用したプログラムを用いた．実験の結果，(1)〜(3)の各学習において，提案手法を適用することによって従来手法を有意に上回る結果を得ることに成功し，提案手法の有効性を示した．実験では，評価関数の学習では提案手法を適用することによって，特に高い効果を得た．また，学習された重要度を分析した結果，終盤の重要度が高くなる，穴熊など一般的にコンピュータが勝ちやすいと言われている戦術の重要度が高い値となる，といった傾向となっており，提案手法を用いることによって，ゲームの性質にあった教師データの重要度が算出されていることを確認した．

提案手法は，コンピュータの強さが人間のトッププレイヤに匹敵しようとしている現在においては，有効な学習手法として今後重要となると考えられる．現在，コンピュータの強さは，教師となる人間のプレイヤのうちの大部分を上回りつつある．しかし，このような状況においても，すべての局面において人間がコンピュータを上回っているわけではなく，多くの局面の評価能力については，依然として人間が上回っていると言える．提案手法を用いることによって，コンピュータが人間を上回った場合にも，参考となる部分のみを学習として取り入れるといったことも可能となると考えられる．

第4章では，第3章の手法の拡張として，評価関数の学習を対象として，強いプログラムを学習するための目的関数の学習手法を提案した．評価関数の学習では，棋譜を教師とした学習が広く行われており，特に将棋では，機械学習の成功によって人間のトッププレイヤに迫る強さを得ている．棋譜を教師とした学習では，一般的にプログラムの指し手と学習棋譜の指し手の一致率を向上させる目的関数を用いるが，従来手法では，目的関数と学習によって得られるプログラムの強さの関係が不明であるという問題点が存在した．

この課題を解決するために，第4章では，第3章の教師データの重要度の学習を拡張し，目的関数のパラメータを，学習された評価関数を用いたプログラムの強さを用いて学習する手法を提案した．提案手法では，目的関数に，学習された評価関数の強さとの関係を表現することができるよう，局面の特徴に応じたパラメータを導入し，そのパラメータを「強さ」が最大と成るように学習した．目的関数のパラメータの学習

には，第3章と同様，個体間の対局から算出したEloレーティングを適応度とした進化的計算（PBILc）を用い,「強い」評価関数を学習できる目的関数の学習を行った．

第4章では，提案手法を将棋の評価関数の学習へ適用し，その効果を検証した．実験の結果，提案手法によって学習された目的関数を用いて学習された評価関数を用いたプログラムが従来手法を用いて学習されたプログラムを大きく上回り，提案手法の有効性を示した．学習された目的関数のパラメータを分析した結果，

- 終盤ほど正解手と不正解手の評価値の差を大きくする
- 探索深さを増やしても正解できない局面を重視した学習を行う

といったように進行度や局面の複雑さに関する特徴と目的関数の関係を捉えたパラメータが学習されていることを確認した．また，提案手法によって学習された目的関数を用いて学習したプログラムは再利用性を有し，異なる特徴を持つ評価関数や異なる学習棋譜を用いた場合にも有効であることを示した．

5.2　今後の課題

本論文の提案手法は,「どのような特徴の教師データを，どのように学習するか」といったこと自体を学習するものであり，従来から多数存在する棋譜の指し手を模倣する学習を拡張する位置付けとなっている．論文中の実験から，提案手法は，従来の学習を上回る成果を得ており，その有効性が示されたと言える．

本研究の，今後の課題としては以下の2点が挙げられる．

(1) 他のゲームにおける有効性の評価

(2) ゲーム以外の分野への応用の検討

(1)について，本論文の提案手法である，重要度の学習，目的関数の学習の2つの手法は,「強さ」をベースとした学習となっている．本手法では,「強さ」の指標を個体同士の対局から算出する手法となっているため，特定のゲームへ依存することなく，幅広いゲームに対して適用可能である．現在も研究が盛んなコンピュータ囲碁など，多くのゲームの性能を，本手法により改善できる可能性があると考えられる．また，対象とするゲームを限定しないGeneral Game Playingへの提案手法も検討の価値があると考えられる．

(2)について，本論文の提案手法は，一般的な教師あり学習の枠組みに従うものであるため，ゲームに限定されず他分野への学習の応用も考えられる．

人工知能の分野では，ゲームに限らず，人間の行動を教師とした学習が広く行われている．これらの中には，(1)人間の行動を学習すること自体が目的であるもの，(2)本来得たいものの近似として人間の行動を模倣しているもの，の2種類が存在している．後者に該当するような対象では，単純な人間の模倣が必ずしも良い結果をもたらすとは限らず，本論文で提案した手法の枠組みで，より本来得たいものに適した学習が実現できる可能性があると考えられる．

謝辞

本論文の執筆にあたり，ご指導いただきました筑波大学システム情報系 高橋大介教授に深く感謝の意を表します.

また，お忙しい中，本論文の審査をお引受けいただきました筑波大学システム情報系 朴泰祐教授，筑波大学システム情報系 狩野均教授，筑波大学システム情報系 佐久間淳准教授，東京大学大学院工学系研究科電気系工学専攻 鶴岡慶雅准教授に深く感謝いたします.

また，鶴岡慶雅准教授，東京工科大学 Reijer Grimbergen教授，マンチェスター大学 三輪誠氏，JST ERATO湊離散構造プロジェクト 竹内聖悟氏からは，本研究にあたり多くの有益な助言を頂きました.

最後に，筑波大学HPCS研究室の先生方，学生の皆様には，社会人博士としてつくばから離れた地で研究を進めるにあたり，本当に多くの御助力を頂きました．ここに，心より感謝の意を表します.

参考文献

[1] Shannon, C. E.: XXII. Programming a computer for playing chess, *Philosophical Magazine (Series 7)*, Vol. 41, No. 314, pp. 256–275 (1950).

[2] Turing, A. M.: Digital computers applied to games, *Faster than Thought*, pp. 286–310 (1953).

[3] Coulom, R.: Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search, *In Proceedings of the 5th International Conference on Computer and Games*, Lecture Notes in Computer Science, Vol. 4630, pp. 72–83 (2006).

[4] Buro, M.: Improving heuristic mini-max search by supervised learning, *Artificial Intelligence*, pp. 85–99 (2002).

[5] Tesauro, G.: Temporal Difference Learning and TD-Gammon, *Communications of the ACM*, Vol. 38, No. 3, pp. 58–68 (1995).

[6] 山下宏：YSS—「コンピュータ将棋の進歩2」以降の改良点，アマトップクラスに迫る—コンピュータ将棋の進歩5，共立出版，pp. 1–32 (2005).

[7] Sturtevant, N.: An Analysis of UCT in Multi-Player Games, *ICGA Journal*, Vol. 31, No. 4, pp. 195–208 (2008).

[8] Lorentz, R.: Amazons Discover Monte-Carlo, *In Proceedings of the 6th International Conference on Computer and Games*, Lecture Notes in Computer Science, Vol. 5131, pp. 13–24 (2008).

[9] Cazenave, T. and Saffidine, A.: Utilisation de la recherche arborescente Monte-Carlo au Hex, *Revue d'Intelligence Artificielle*, Vol. 23, No. 2–3, pp. 183–202 (2009). (in French).

[10] Winands, M. and Björnsson: Evaluation Function Based Monte-Carlo LOA, *In Proceedings of the 12th Advances in Computer Games*, Lecture Notes in Computer Science, Vol. 6048, pp. 33–44 (2009).

[11] Ciancarini, P. and Favini, G.: Monte Carlo Tree Search Techniques in the Game of Kriegspiel, *In Proceedings of the 21st International Joint Conference on Artifical intelligence*, pp. 474–479 (2009).

[12] Knuth, D. E. and Moore, R. W.: An analysis of alpha-beta pruning, *Artificial Intelligence*, Vol. 6, No. 4, pp. 293–326 (1975).

[13] Marsland, T. A. and Campbell, M.: Parallel Search of Strongly Ordered Game Trees, *ACM Computing Surveys*, Vol. 14, No. 4, pp. 533–551 (1982).

[14] Hermann, K., Reza, S. and Helmut, H.: Minimax Search Algorithms With and Without Aspiration Windows, *IEEE Transaction on Pattern Analysis and Machine Intelligence.*, Vol. 13, No. 12, pp. 1225–1235 (1991).

[15] Buro, M.: ProbCut: An Effective Selective Extension of the alphabeta Algorithm, *ICCA Journal*, Vol. 18, pp. 71–76 (1995).

[16] Beal, D.: Experiments with the Null Move, *Advances in Computer Chess 5*, pp. 65–79 (1989).

[17] Heinz, E.: Adaptive null-move pruning, *ICCA Journal*, Vol. 22, No. 3, pp. 123–132 (1999).

[18] Heinz, E.: Extended Futility Pruning, *ICCA Journal*, Vol. 21, No. 2, pp. 75–83 (1998).

[19] 保木邦仁：コンピュータ将棋における全幅探索とfutility pruningの応用，情報処理，Vol. 47, No. 8, pp. 884–889 (2006).

[20] Romstad, T.: An Introduction to Late Move Reductions. http://www.glaurungchess.com/lmr.html

[21] Schroder, E.: History Reductions. http://members.home.nl/matador/hr.htm

[22] Beal, D.: A generalised quiescence search algorithm, *Artificial Intelligence*, Vol. 43, No. 1, pp. 85–98 (1990).

[23] Anantharaman, T. S., Campbell, M. and hsiung Hsu, F.: Singular Extensions: Adding Selectivity to Brute-Force Searching., *Artificial Intelligence*, Vol. 43, No. 1, pp. 99–109 (1990).

[24] Allis, L. V., van der Meulen, M. and van den Herik, H. J.: Proof-number Search, *Artificial Intelligence*, Vol. 66, No. 1, pp. 91–124 (1994).

[25] Seo, M., Iida, H. and Uiterwijk, J. W.: The PN*-search algorithm: Application to tsume-shogi, *Artificial Intelligence*, Vol. 129, No. 1-2, pp. 253–277 (2001).

[26] Nagai, A.: A new and/or tree search algorithm using proof number and disproof number, *In Proceedings of the Complex Games Lab Workshop*, pp. 40–45 (1998).

[27] 長井歩，今井浩：df-pnアルゴリズムの詰将棋を解くプログラムへの応用，情報処理学会論文誌，Vol. 43, No. 6, pp. 1769–1777 (2002).

[28] Nagai, A. and Imai, H.: Application of df-pn+ to Othello endgames, *In Proceedings of the 5th Game Programming Workshop*, pp. 16–23 (1999).

[29] Schaeffer, J., Al, E., Schaeffer, J., Burch, N., Björnsson, Y., Kishimoto, A., Müller, M. and Lake, R.: Checkers is solved, *Science* (2007).

[30] Tesauro, G.: Comparison training of chess evaluation functions, *Machines that learn to play games*, Nova Science Publishers, Inc., pp. 117–130 (2001).

[31] 保木邦仁：局面評価の学習を目指した探索結果の最適制御，第11回ゲームプログラミングワークショップ，pp. 78–83 (2006).

[32] Tsuruoka, Y., Yokoyama, D. and Chikayama, T.: Game-Tree Search Algorithm Based On Realization Probability, *ICGA Journal*, Vol. 25, No. 3, pp. 145–152 (2002).

[33] Coulom, R.: Computing Elo Ratings of Move Patterns in the Game of Go, *ICGA Journal*, Vol. 30, No. 4, pp. 198–208 (2007).

[34] Silver, D. and Tesauro, G.: Monte-Carlo Simulation Balancing, *In Proceedings of the 26th International Conference on Machine Learning*, pp. 945–952 (2009).

[35] Hsu, F.-h., Anantharaman, T. S., Campbell, M. S. and Nowatzyk, A. G.: Deep Thought, *Computers, Chess and Cognition*, pp. 55–78 (1990).

[36] 美添一樹：モンテカルロ木探索—コンピュータ囲碁に革命を起こした新手法，情報処理，Vol. 49, No. 6, pp. 686–693 (2008).

[37] 橋本隼一，橋本剛，長嶋淳：コンピュータ将棋におけるモンテカルロ法の可能性，第11回ゲームプログラミングワークショップ，pp. 195–198 (2006).

[38] 伊藤毅志，新沢剛：モンテカルロ法を用いた5五将棋システム，情報処理学会研究報告. GI, [ゲーム情報学]， Vol. 2007, No. 62, pp. 1–6 (2007).

[39] 佐藤佳州，高橋大介：モンテカルロ法によるコンピュータ将棋の実現，情報処理学会第70回全国大会(2007).

[40] Kocsis, L. and Szepesvari, C.: Bandit based Monte-Carlo Planning, *In Proceedings of 15th European Conference on Machine Learning*, Lecture Notes in Computer Science, Vol. 4212, pp. 282–293 (2006).

[41] 棚瀬寧：棚瀬将棋の技術背景，情報処理， Vol. 49, No. 8, pp. 987–992 (2008).

[42] 保木邦仁，金子知適，横山大作，小幡拓弥，山下宏：あから2010勝利への道：2．あから2010のシステム設計と操作概要，情報処理， Vol. 52, No. 2, pp. 162–169 (2011).

[43] 保木邦仁：「Bonanza 4.1.3」ソースコード，コンピュータ将棋の進歩6，共立出版，pp. 1–22 (2012).

[44] 鶴岡慶雅：「激指」の最近の改良について–コンピュータ将棋と機械学習–，コンピュータ将棋の進歩6，共立出版，pp. 71–83 (2012).

[45] 金子知適：「GPS将棋」の評価関数とコンピュータ将棋による棋譜の検討，コンピュータ将棋の進歩6，共立出版，pp. 25–44 (2012).

[46] 大槻知史：コンピュータ将棋プログラム「大槻将棋」，コンピュータ将棋の進歩6，共立出版，pp. 47–69 (2012).

[47] Brügmann, B.: Monte Carlo Go, http://ideanest.com/vegos/MonteCarloGo.pdf.

[48] Yoshimoto, H., Yoshizoe, T., Kaneko, T., A., K. and K., T.: Monte Carlo Go Has a Way to Go, *In Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, pp. 1070–1075 (2006).

[49] Gelly, S., Wang, Y., Munos, R. and Teytaud, O.: Modification of UCT with Patterns in Monte-Carlo Go, Technical Report 6062, INRIA, France (2006).

[50] 美添一樹，山下宏：コンピュータ囲碁: モンテカルロ法の理論と実践，共立出版(2012).

[51] Akl, S. and Newborn, M.: The Principal Continuation and the Killer Heuristic, *In Proceedings of the 1977 ACM Annual Conference*, pp. 466–473 (1977).

[52] Schaeffer, J.: The History Heuristic, *ICCA Journal*, Vol. 6, No. 3, pp. 16–19 (1983).

[53] Auer, P., Cesa-Bianchi, N., and Fischer, P.: Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, Vol. 47 No.2–3, pp. 235–256 (2002).

[54] 金子知適：兄弟節点の比較に基づく評価関数の調整，第12回ゲームプログラミングワークショップ，pp. 9–16 (2007).

[55] Elo, A.: The Rating of Chessplayers, Past and Present (1978).

[56] Bradley, R. A. and Terry, M. E.: Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons, *Biometrika*, Vol. 39, pp. 324–345 (1952).

[57] Reul, F.: Static Exchange Evaluation with $\alpha\,\beta$-Approach，*ICGA Journal*, Vol. 33, No. 1, pp. 3–17 (2010).

[58] Gelly, S. and Silver, D.: Combining Online and Offline Knowledge in UCT, *In Proceedings of the 24th International Conference on Machine Learning*, pp. 273–280 (2007).

[59] 鶴岡慶雅：将棋プログラムの現状と未来，情報処理， Vol. 46, No. 7, pp. 817–822 (2005).

[60] 松原仁：コンピュータ将棋の進歩2，共立出版 (1998).

[61] 将棋タウン棋力判定問題集，http://www.shogitown.com/school/judge/judgetop.html

[62] 大和証券杯ネット将棋公式ホームページ，http://www.daiwashogi.net/

[63] 渡辺明：渡辺明ブログ，http://blog.goo.ne.jp/kishi-akira/

[64] 杉山卓弥，小幡拓弥，斎藤博昭，保木邦仁，伊藤毅志：将棋における合議アルゴリズム—局面評価値に基づいた指し手の選択，情報処理学会論文誌，Vol. 51, No. 11, pp. 2048–2054 (2010).

[65] 伊藤毅志，小幡拓弥，杉山卓弥，保木邦仁：将棋における合議アルゴリズム–多数決による手の選択，情報処理学会論文誌，Vol. 52, No. 11, pp. 3030–3037 (2011).

[66] 竹内聖悟，金子知適，山口和紀：将棋における,評価関数を用いたモンテカルロ木探索，第15回ゲームプログラミングワークショップ，pp. 86–89 (2010).

[67] 竹内聖悟：将棋における, 評価値に基づくモンテカルロ木探索へのDynamic Komiの応用，第18回ゲームプログラミングワークショップ，pp. 50–57 (2013).

[68] 竹歳正史，橋本剛，梶原羊一郎，長嶋淳，飯田弘之：コンピュータ将棋における実現確率探索の研究，第7回ゲームプログラミングワークショップ，pp. 87–92 (2002).

[69] 橋本剛，長嶋淳，作田誠，Uiterwijk, J.，飯田弘之：実現確率探索のゲーム全般への応用 – Lines of Action を題材にして，第7回ゲームプログラミングワークショップ，pp. 81–86 (2002).

[70] 三輪誠，横山大作，近山隆：指し手の履歴の抽出に基づくカテゴリの拡張，第11回ゲームプログラミングワークショップ，pp. 64–69 (2006).

[71] 佐藤佳州，高橋大介：探索結果を利用した実現確率探索，情報処理学会論文誌，Vol. 51, No. 11, pp. 2021–2030 (2010).

[72] 鶴岡慶雅：最近のコンピュータ将棋の技術背景と激指，情報処理， Vol. 49, No. 8, pp. 982–986 (2008).

[73] 関栄二，三輪誠，鶴岡慶雅，近山隆：シミュレーション・バランシングを用いたモンテカルロ将棋の方策学習，情報処理学会論文誌，Vol. 53, No. 11, pp. 2533–2543 (2012).

[74] Beal, D. F. and Smith, M. C.: First Results from Using Temporal Difference Learning in Shogi, *In Proceedings of the First International Conference on Computers and Games*, Lecture Notes in Computer Science, Vol.1558, pp. 113–125 (1999).

[75] 鈴木彰，柴原一友，但馬康宏，小谷善行：条件付き確率PIPEによる将棋の評価関数の生成，第10回ゲームプログラミングワークショップ，pp. 56–62 (2005).

[76] 矢野友貴，柴田剛志，横山大作，田浦健次朗，近山隆：GAとTD(λ)学習の組み合わせによるゲーム局面評価パラメータの調整，情報処理学会研究報告. GI, [ゲーム情報学]， Vol. 2009, No. 27, pp. 63–70 (2009).

[77] 金子知適：コンピュータ将棋の評価関数と棋譜を教師とした機械学習(＜レクチャーシリーズ＞コンピュータ将棋の技術〔第5回〕)，人工知能学会誌，Vol. 27, No. 1, pp. 75–82 (2012).

[78] 川上裕生，浦晃，三輪誠，鶴岡慶雅，近山隆：将棋の評価関数の学習に有用な局面の自動選択，第18回ゲームプログラミングワークショップ，pp. 66–72 (2013).

[79] 竹内章：コンピュータ将棋「習甦」開発記，コンピュータ将棋協会誌， Vol. 22, pp. 21–26 (2010).

[80] Tesauro, G.: Programming backgammon using self-teaching neural nets, *Artificial Intelligence*, Vol. 134, pp. 181–199 (2002).

[81] Sebag, M. and Ducoulombier, A.: Extending Population-Based Incremental Learning to Continuous Search Spaces, *In Proceedings of the 5th International Conference*

on Parallel Problem Solving from Nature, Lecture Notes in Computer Science, Vol. 1498, pp. 418–427 (1998).

[82] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R. and Lin, C.-J.: LIBLINEAR: A Library for Large Linear Classification, *Journal of Machine Learning Research*, Vol. 9, pp. 1871–1874 (2008).

[83] 佐藤佳州，高橋大介：特徴の生成を組み合わせた機械学習，第16回ゲームプログラミングワークショップ，pp. 135–142 (2011).

[84] CGOS. http://cgos.boardspace.net/

[85] floodgate. http://wdoor.c.u-tokyo.ac.jp/shogi/floodgate.html

[86] http://www.geocities.jp/saltedeggplant/

[87] 佐久間淳，小林重信：確率分布推定に基づく実数値GAの新展開(＜特集＞遺伝的アルゴリズムの発展)，人工知能学会誌，Vol. 18, No. 5, pp. 479–486 (2003).

[88] Larranaga, P., Etxeberria, R., Lozano, J. A. and Pena, J.: Optimization by learning and simulation of Bayesian and Gaussian networks, Technical report, University of the Basque Country (1999). Technical Report EHUKZAA-IK-4/99.

[89] Bosman, P. A. and Thierens, D.: Expanding From Discrete to Continuous Estimation of Distribution Algorithms: The IDEA, *In Parallel Problem Solving From Nature - PPSN VI*, Lecture Notes in Computer Science, Vol. 1917, pp. 767–776 (2000).

[90] Pitrat, J.: Realization of a general game-playing program, *In proceedings of the IFIP Congress (2)*, pp. 1570–1574 (1968).

[91] Genesereth, M. and Love, N.: General game playing: Overview of the AAAI competition, *AI Magazine*, Vol. 26, pp. 62–72 (2005).

[92] 金子知適，田中哲朗，山口和紀，川合慧：駒の関係を利用した将棋の評価関数の学習(評価関数,＜特集＞ゲームプログラミング)，情報処理学会論文誌，Vol. 48, No. 11, pp. 3438–3445 (2007).

[93] Abbeel, P. and Ng, A. Y.: Apprenticeship Learning via Inverse Reinforcement Learning, *In Proceedings of the Twenty-first International Conference on Machine Learning* (2004).

[94] Argall, B., Chernova, S., Veloso, M. and Browning, B.: A Survey of Robot Learning from Demonstration, *Robotics and Autonomous Systems*, Vol. 67, pp. 469–483 (2009).

[95] Campbell, M., Hoane Jr, A. J. and Hsu, F.-h.: Deep Blue, *Artificial Intelligence*, Vol. 134, No. 1-2, pp. 57–83 (2002).

[96] 竹内聖悟，林芳樹，金子知適，山口和紀，川合慧：勝率に基づく評価関数の評価と最適化(評価関数,＜特集＞ゲームプログラミング)，情報処理学会論文誌，Vol. 48, No. 11, pp. 3446–3454 (2007).

[97] Hoki, K. and Kaneko, T.: The Global Landscape of Objective Functions for the Optimization of Shogi Piece Values with a Game-Tree Search., *In Proceedings of the 13th Advances in Computer Games*, Lecture Notes in Computer Science, Vol. 7168, pp. 184–195 (2011).

[98] Kaneko, T. and Hoki, K.: Analysis of Evaluation-Function Learning by Comparison of Sibling Nodes, *In Proceedings of the 13th Advances in Computer Games*, Lecture Notes in Computer Science, Vol. 7168, pp. 158–169 (2011).

[99] Ura, A., Yokoyama, D. and Chikayama, T.: Comparison Training of Shogi Evaluation Functions with Self-Generated Training Positions and Moves, *In Proceedings of the 8th International Conference on Computers and Games (CG 2013)*, pp. 66–72 (2013).

[100] Neu, G. and Szepesvári, C.: Training parsers by inverse reinforcement learning, *Machine Learning*, Vol. 77, pp. 303–337 (2009).

[101] He, H., Daumé III, H. and Eisner, J.: Imitation Learning by Coaching, *In Proceedings of the Advances in Neural Information Processing Systems 25* (2012).

[102] Ross, S. and Bagnell, D.: Efficient Reductions for Imitation Learning, *Journal of Machine Learning Research - Proceedings Track*, Vol. 9, pp. 661–668 (2010).

[103] Baxter, J., Tridgell, A. and Weaver, L.: Knightcap: A chess program that learns by combining td(λ) with game-tree search, *In Proceedings of the 15th International Conference on Machine Learning*, pp. 28–36 (1998).

[104] Takeuchi, S., Kaneko, T., Kazunori, Y. and Kawai, S.: Visualization and Adjustment of Evaluation Functions Based on Evaluation Values and Win Probability, *In*

Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07), pp. 858–863 (2007).

[105] 金子知適，山口和紀：将棋の棋譜を利用した，大規模な評価関数の調整，第13回ゲームプログラミングワークショップ，pp. 152–159 (2008).

[106] 鶴岡慶雅：将棋プログラム「激指」，アマ4段を超える—コンピュータ将棋の進歩4，共立出版，pp. 1–17 (2003).

[107] 竹内聖悟，林芳樹，金子知適，川合慧：勝率と評価値の歪みに基づく評価関数調整法–将棋における進行度差の評価–，第11回ゲームプログラミングワークショップ，pp. 56–63 (2006).

[108] Heinz, E. A.: DarkThought Goes Deep, *ICCA Journal*, Vol. 21, No. 4, pp. 228–229 (1998).