修　　士　　論　　文

投入計算量の有限性に基づく UCT探索の枝刈り

Pruning in UCT search based on limitedness of computational resource

指導教員　　近山 隆 教授

東京大学大学院
新領域創成科学研究科
基盤情報学専攻

氏 名　　47-076308 北川 竜平

提 出 日　　平成 21 年 1 月 27 日

概 要

本研究では，計算機によって解を得るために用いることのできる時間が有限であることから，投入計算量の有限性を意識したアルゴリズムを提案し，ゲーム木探索に対して実装した．

ゲーム木探索において，UCT 探索は知識表現の難しいゲームや新しいゲームに関しても有効な探索手法である．探索しても無駄になりそうな枝を探索しない手法である枝刈りを行うことで，選ばれそうな合法手に対してより多くのプレイアウトを行うことができ，それらの合法手に対してより高い精度の評価値を得ることが可能である．本研究では探索に用いることのできる時間が有限であることから，残り時間内にプレイアウトを行うことができる回数を推定し，それぞれの合法手に関して到達することのできる勝率の区間を推定することで枝刈りを行った．この手法を囲碁の 9 路盤のコンピュータゲームプレイヤに組み込んだところ，元の手法に対して勝率 8 割という結果を得ることができた．

目 次

図 目 次

表 目 次

第1章　序論

1.1　背景と目的

近年，問題解決の手段として計算機上には様々なアルゴリズムが実装されている．しかし，その全てが必ずしも精度の高い解を得られるとは限らない．非常に難しい問題を解こうとする場合や，組み込みで用いる CPU など処理性能の低い計算機上で実行させる複雑なアルゴリズムなど，精度の高い解を求めるのに時間が掛かるという状況は存在する．本研究では，解を得るまでに用いることのできる時間が有限であることから，投入計算量の有限性を意識したアルゴリズムを提案する．そのためにコンピュータゲームプレイヤの最善手選択アルゴリズムに，投入計算量の有限性に基づいた最適化を行うことで，このアプローチの有効性を示す．

一般的にオセロ・チェス・将棋のコンピュータプレイヤでは，最善手を判断するのにゲーム木探索と評価関数を用いている．ゲーム木探索では現在の局面をルートノードとし，合法手によって推移可能な局面を子ノードとすることで将来の局面を展開する．評価関数とは局面の優劣を評価するのに用いられ，その精度が高いほどコンピュータプレイヤの性能は向上する．古くからゲームプレイヤではこの二つを組み合わせることで，将来の局面で自分が有利となるような合法手を選ぶという手法が用いられていた．しかし精度の高い評価関数を作成するには，そのゲームに関する深い知識持つことやその知識をプログラムとして表現することが必要となる．そのため知識の溜まっていない新しいゲームや囲碁のように知識表現の難しいゲームに関しては精度の高い評価関数の作成が困難となる．

このような問題点を改善した探索手法にモンテカルロ木探索 [4] がある．モンテカルロ木探索はその局面からゲームの終局までランダムに合法手を選ぶことで局面を展開するランダムシミュレーション (プレイアウト) を繰り返し，その勝敗の結果の平均 (勝率) によって局面評価を行う手法である．この手法ではゲームの勝敗の結果が局面評価に利用されるために，評価関数を必要としない．そのため新しいゲームや知識表現の難しいゲームにおいても有効である．特に囲碁では Crazy Stone[1] や MoGo[2] といったトップクラスのコンピュータプレイヤにこの探索手法が使われており，その有効性が示されている．近年では，モンテカルロ木探索による局面の評価を評価関数の強化学習 [16] に用いるなど，プレイヤとしての利用以外にも用いられている．

モンテカルロ木探索は評価関数を用いる必要が無いという利点はあるが，大数の法則に基づいているために局面評価の精度はプレイアウトの回数に依存する。一般的に一つの行動選択のためにゲーム木探索に用いることのできる時間は数秒から数分程度であり，全ての合法手をランダムに選んでいると高い精度で局面評価を行うことは困難である．ゲームでは選択される合法手は通常は一つなので，選択されそうな合法手の評価値の精度が正確であれば良い．よって合法手の選択を完全にランダムではなく，ある程度選択される可能性の高そうな合法手に対して重点的にプレイアウトを行

うことでプレイヤとしての性能は向上する．その問題に対する代表的なアルゴリズムとして UCT 探索 (Upper Confidence Bounds applied to Trees) [5] がある．UCT 探索では勝率と探索された回数に依存する UCB 値によって探索を行う合法手を選択し，結果的に選択される可能性の高い合法手に対して多くのプレイアウトが行われる．しかし UCT 探索であっても選択される可能性の無い合法手のプレイアウトの回数を 0 にすることは無い．よって，そのような合法手を見つけ出し探索対象から外すこと (枝刈り) で，選択されそうな合法手の評価値の精度を更に上げ，プレイヤとしての性能を向上させることが可能となる．

本研究では，探索時間の有限性から，プレイアウトを行うことのできる回数が有限であることと，それまでの探索結果がある程度信頼できることから，それぞれの合法手が探索終了時で到達しそうな勝率の区間を推定した．そして，その区間から選択される可能性が低いと思われる合法手に対して枝刈りを行った．この手法はゲームの知識や性質に依存しない点で様々なゲームに適応可能であると考えられる．

本研究では，知識表現の難しいゲームである囲碁にこの手法を実装することで性能評価を行った．

1.2 本論文の構成

以降，本論文の構成は次のようになっている．

第 2 章 関連研究
モンテカルロ木探索と先行研究について述べる．

第 3 章 提案手法
本研究の提案手法について述べる．

第 4 章 実験
本研究の実験結果と評価について述べる．

第 5 章 結論
本研究の結論と今後の課題について述べる。

第2章　関連研究

本章では関連研究として，2.1 でゲーム木探索について説明する．そして 2.2 でモンテカルロ木探索について説明し，2.3 で UCT 探索とその改善手法について説明する．

2.1　ゲーム木探索

ゲーム木探索は，コンピュータゲームプレイヤがある局面からの最善手を求めるための手法として古くから用いられている探索手法である．図 2.1 に○×ゲームのゲーム木の一部を示す．ゲーム木探索は，現在の局面をルートノード，局面からの合法手を枝，親ノードから合法手によって推移可能な局面を子ノードとして用いることでゲームの将来の局面を木構造によって展開したものである．

○×ゲームのように簡単なゲームであれば，全ての局面を展開した後に勝敗を評価すればよい．しかし，より複雑なゲームではゲーム木は非常に大きなものとなってしまう．例として表 2.1 に主なゲームの探索空間の大きさの概算を示す．仮にメモリ容量が無限大のコンピュータで 1 秒間に 1 兆局面に対して展開・評価ができたとしても，チェスでは 10^{109} 秒程，囲碁では 10^{349} 秒程の時間が必要となる．これは宇宙が生まれてから現在までの時間が 10^{17} 秒程であることを考えると実現不可能な時間である．そのため終局までの結果を全て展開し，勝敗を評価するという方法は実質的に不可能である．よって実際のゲーム木探索では，時間制限等によって探索を打ち切り，局面の優劣を判断する評価値を与える．評価値は現在の局面のプレイヤが有利なほど大きな値にするのが一般的である．古くから用いられていた手法では，評価値は 2.1.1 で説明する評価関数によって与える．

表 2.1: 探索空間の大きさ

チェッカー	10^{30}
オセロ	10^{60}
ブロックスデュオ	10^{80}
チェス	10^{120}
将棋	10^{220}
囲碁	10^{360}

2.1.1 評価関数

評価関数とは局面を入力することで，局面の優劣を付ける評価値を出力する関数である．一般的に評価関数は，局面からその基準となる評価要素 (特徴) を抜き出し，それぞれの評価要素に対応した重みを与え，その線形和で表すことが多い．

評価関数の作成には，そのゲームに関する深い知識による評価要素の生成と多大な労力による重みの調整が必要であった．近年では重みの調整は，ゲームの記録からの教師あり学習 [7][17][18] や，自己対戦による強化学習 [9][10] によって自動化されることが多い．また，ゲームの記録からの評価要素の自動生成 [8][14] についても研究されている．

しかし，ゲームの知識が溜まっていない新しいゲームや囲碁のように知識表現の難しいゲームでは，最適な重みの調整や精度の高い評価関数の作成が困難であるというのが現実である．

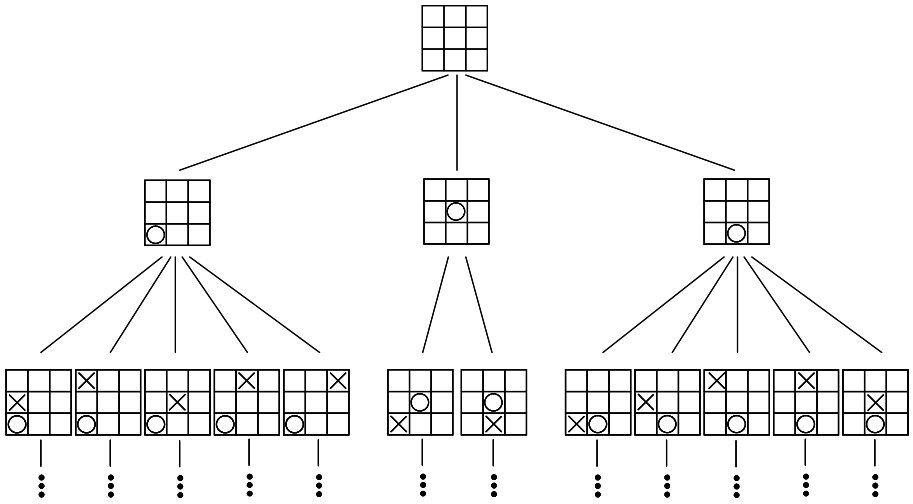

図 2.1: ゲーム木探索

2.2 モンテカルロ木探索

2.1.1 で示したように，精度の高い評価関数の作成が困難なゲームが存在する．基本的なモンテカルロ木探索では評価関数を必要としないために，そのようなゲームでも局面に評価値を与えることが可能である．

モンテカルロ木探索 [4] ではモンテカルロ法に基づいたゲームのシミュレーション (プレイアウト) によって現在局面の評価を行う．プレイアウトでは現在局面からゲームの終局までランダムに合法手を選ぶことで局面の展開を行う．そして現在局面からのそれぞれの合法手に対し，プレイアウトによる勝敗を与える．このプレイアウトを繰り返すことで，現在局面からのそれぞれの合法手を選択した場合の勝率を得ることができる．

合法手 i を選択してプレイアウトを行った回数を s_i，プレイアウトによって得られた報酬の和を X_i とする．報酬としてはプレイアウトの結果が勝ちであれば 1 を，負けであれば 0 を与える．このとき勝率 $\bar{X}_i$ は

$$\bar{X}_i = \frac{X_i}{s_i} \tag{2.1}$$

によって与えられる．モンテカルロ木探索では勝率 $\bar{X}_i$ を合法手 i の評価値として利用する．

この手法は評価関数を用いていないために，新しいゲームや知識表現の難しいゲームにおいて有効とされている．

モンテカルロ木探索では大数の法則に基づいているために，評価値の精度はプレイアウトの回数に依存している．一般的に一つの行動選択のためにゲーム木探索に用いることのできる時間は数秒から数分程度であり，その時間内のプレイアウトの回数では高い精度の評価値を得ることは困難である．よって少ない時間であっても多くのプレイアウト回数を行ったのと同様の結果を得る必要が出てくる．ゲームでは選択される合法手は通常は一つなので，選択されそうな合法手の評価値の精度が正確であれば良い．したがって合法手の選択を完全にランダムではなく，ある程度選択される可能性の高そうな合法手に対して重点的にプレイアウトを行うことで，プレイアウトの回数が増えたのと同等の結果を得ようとするのがモンテカルロ木探索の性能改善の基本的な方針である．

2.2.1 Progressive Pruning

B.Bouzy による Progressive Pruning[11][12] では，それぞれの合法手の勝率は正規分布に従うと仮定し，勝率と得られた報酬の標準偏差から勝率が取り得る値の区間を推定する．その結果，選ばれる見込みの無い合法手を求め，探索を行わないこと (枝刈り) でモンテカルロ木探索の効率を改善する．

合法手 i の勝率を $\bar{X}_i$，報酬の標準偏差を σ_i，プレイアウト回数を s_i とした時に，区間推定を用いて以下のようにそれぞれの合法手の勝率が取り得る値の区間を求める．

$$\bar{X}_{Li} = \bar{X}_i - r\frac{\sigma_i}{\sqrt{s_i}} \tag{2.2}$$

$$\bar{X}_{Ri} = \bar{X}_i + r\frac{\sigma_i}{\sqrt{s_i}} \tag{2.3}$$

ここで $\bar{X}_{Li}$ は合法手 i の勝率がどの程度まで下がる可能性があるかを推定した値であり，$\bar{X}_{Ri}$ は合法手 i の勝率がどの程度まで上がる可能性があるかを推定した値である．つまり合法手 i の勝率は $[\bar{X}_{Li}, \bar{X}_{Ri}]$ となる可能性が高い．ここで r は信頼係数である．図 2.3 に Progressive Pruning のイメージ図を示す．図の横軸は勝率を表し，それぞれの合法手に対し $\bar{X}$ は現在の勝率を，$\bar{X}_L, \bar{X}_R$ は勝率が取り得る値の区間を示す．合法手 i に関して，$\bar{X}_{Ri} < \bar{X}_{Lj}$ であるため，合法手 i の勝率が合法手 j の勝率より高くなる可能性は低い．よって合法手 i が選択される可能性は低いために，これ以上探索しても探索結果に影響を与える可能性は低く，枝刈りをすることができる．合法手 k に関しては $\bar{X}_{Lj} < \bar{X}_{Rk}$ であることから，合法手 k の勝率は合法手 j の勝率より高くなる可能性があるので探索を続けるべきである．

この手法では推定勝率区間 $[\bar{X}_{Li}, \bar{X}_{Ri}]$ を狭めるのに多くのプレイアウト回数を必要とするために，あまり多くないプレイアウト回数では効果を得ることができない．

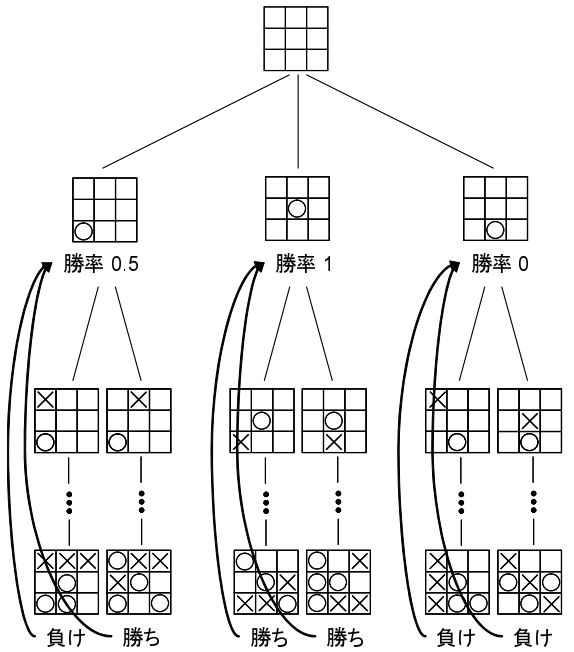

図 2.2: モンテカルロ木探索

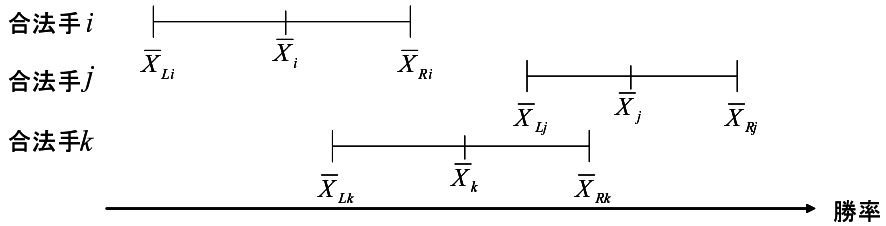

図 2.3: Progressive Pruning

2.3 UCT 探索

本節では，多腕バンディット問題のアルゴリズムを適用することでモンテカルロ木探索の改良を行った UCT 探索について説明する．また UCT 探索に対する各種関連研究についても述べる．

2.3.1 多腕バンディット問題

多腕バンディット問題とは，多くの腕を持つスロットマシンをプレイするギャンブラーに基づく機械学習の問題である．

スロットマシンをプレイすると，ギャンブラーはある確率分布に基づいた報酬を得ることができる．しかしそれぞれの腕毎に報酬の確率分布は異なっている．ギャンブラーは繰り返しスロットマシンをプレイし，その報酬の合計を最大化することを目的とする．ギャンブラーは最初はスロットマシンに関する知識を持たず，プレイを繰り返すことで報酬を高く得られる腕を見つけ出し，集中的にプレイすることができるようになるとされている．

多腕バンディット問題は，プレイによって得た知識によって現時点で最善の腕を選び報酬を最大化することと，他の腕をプレイし，その腕に関する知識を増やすこととのバランスを取る問題であり，強化学習では収穫と探検のジレンマとして知られている．

モンテカルロ木探索は，それぞれの合法手を多腕バンディット問題におけるそれぞれの腕とし，プレイアウト勝敗の結果を得られる報酬とすると，多腕バンディット問題と類似している点は多い．このことから近年では，モンテカルロ木探索に多腕バンディット問題のアルゴリズムを利用した手法が多く用いられている．

2.3.2 UCT 探索

L.Koscis らによる UCT 探索 [5] はプレイアウトを行う合法手に制御を与え，結果的に選択されそうな手に対して多くのプレイアウト回数を行うアルゴリズムである．この制御のために，多腕バンディット問題に対する解法として用いられている UCB1 アルゴリズム [13] を利用している．

UCT では未探索の局面に達した時はそこから先の探索ではモンテカルロ木探索を行う．またある局面において既に探索したことのある合法手と未探索の合法手が存在している場合は未探索の合法手を優先して探索する．ある局面において全ての合法手が 1 度以上探索されていた場合，プレイアウト中の各プレイヤは UCB 値に基づいて合法手の選択を行う．$\bar{X}_i$ を合法手 i の勝率，s_i を探索中に合法手 i が選択された回数，n を合法手 i の親局面を通った回数とすると，合法手 i の UCB 値は以下のように定義される．

$$\mathrm{UCB}(i) = \bar{X}_i + \sqrt{c\frac{\log n}{s_i}} \tag{2.4}$$

プレイアウト中では UCB 値が最も高い合法手を探索する．UCT では UCB 値の第一項により，勝率の高い合法手ほど選択されやすい．しかし探索回数の少ない合法手の勝率は信頼できないために，より多く探索する必要がある．そのため第二項により比較的他の合法手よりも探索回数が少ない合

法手は選択されやすくなる．式 2.4 の c は探索されやすさに探索回数による影響の大きさを与える定数であり，$c = 2$ が用いられる場合が多い．

2.3.3 UCB1-Tuned

UCB1-Tuned は多腕バンディット問題における UCB1 アルゴリズムを改善したものであり，式 2.4 の c を定数として与えるのではなく，以下の式によって与える．

$$V_i = \sigma_i^2 + \sqrt{\frac{2 \log n}{s_i}} \tag{2.5}$$

$$c = \min\left(\frac{1}{4}, V_i\right) \tag{2.6}$$

UCT 探索においても UCB 値をこの式によって与えることで，探索が進んだ時に極端に勝率の低い合法手が探索される回数が少なくなり，プレイヤがより強くなることが報告されている [2]．

2.3.4 Discounted UCB

Discounted UCB[6] は，2.3.3 の UCB1-Tuned と同様に多腕バンディット問題における UCB1 アルゴリズムを改善したものであり，UCT 探索に利用することで強いプレイヤの作成が可能であることが報告されている．

合法手 i で実際にプレイアウトが行われた回数を d_i とすると，勝率 $\bar{X}_i$ を以下のように計算する．

$$X_i = \sum_{\eta=0}^{d_i} X_i(\eta) \gamma^{d_i - \eta} \tag{2.7}$$

$$s_i = \sum_{\eta=0}^{d_i} \gamma^{d_i - \eta} \tag{2.8}$$

$$\bar{X}_i = \frac{s_i}{X_i} \tag{2.9}$$

ここで $X_i(\eta)$ は η 回目のプレイアウトによって得られた報酬である．また γ は過去の報酬の割引率であり，[0,1] の範囲で与える．

このように勝率を与え，式 2.4 の第一項と評価値として用いる．このアルゴリズムは過去の報酬より最近の報酬を重視することになる．特に UCT 探索では，木構造となっている関係上，得られる報酬の質に差があるために，この手法による性能向上が大きく期待できる．

2.3.5 ゲームの知識を用いたランダムシミュレーションの改善

ランダムシミュレーションの際に合法手を完全にランダムで選ぶと，プレイアウトの際に意味の無い手順でゲームが進むことが多くなる．ここでランダムシミュレーション部で何らかの制限を与え，意味のある手順に対する勝率を用いた方が，その局面の真の評価値に近い値を得られるということは明白である．よってゲームの知識を用いることで，意味の無い合法手を判断し，それらを除いた中からランダムに合法手を選択することで，プレイアウトの質を上げることができる．

囲碁プレイヤである MoGo[2] では，囲碁の知識や 3×3 のパターンを用いることで，選択してもいい合法手と選択すべきでない合法手を判断している．

この手法は，ある程度のゲームの知識が必要となるが，完全なランダムシミュレーションと比べ，高い性能を出している．

表 2.2: ランダムシミュレーションの違いによる対戦結果 (S. Gelly et al. 2006.)

ランダムモード	黒の勝率	白の勝率	計
完全ランダム	46%	36%	41.2%±4.4%
改善手法	77%	82%	80%±2.8%

2.3.6 UCT-RAVE

UCT-RAVE (Rapid Action Value Estimate) [3] は，UCT のプレイアウト中に選択された行動を，全て現在局面からの一手目に選択されたものとすることで，見た目のプレイアウトの回数を増やす手法である．

通常の UCT では，プレイアウトによって得られた報酬は，そのプレイアウト中に通った局面のみに伝播される．UCT-RAVE では，プレイアウト中に選択された行動全てを現在局面でも選択されたものとして，現在局面からそれらの行動によって推移する全ての局面に対して報酬を伝播する (図 2.4) ．これによって，1 回のプレイアウトで，自分が選択した行動の回数分の勝率の更新が可能である．しかし，これは局面によって選択される合法手や局面に与える影響は変わらないといった前提においてのみ正しい．実際のゲームではその前提は成り立たないが，プレイアウト回数が多くない時の指標としては用いることができる．よって RAVE によって得られた勝率 $\bar{X}_{\mathrm{RAVE}}$ と通常の UCT によって得られた勝率 $\bar{X}_{\mathrm{move}}$ との加重平均を UCT-RAVE における評価値 $\bar{X}_{\mathrm{value}}$ とする．

$$\bar{X}_{\mathrm{value}} = \beta \bar{X}_{\mathrm{RAVE}} + (1-\beta)\bar{X}_{\mathrm{move}} \tag{2.10}$$

$$\beta = \sqrt{\frac{k}{3s+k}} \tag{2.11}$$

ここで s はその合法手を実際に探索した回数であり，k は重みが等しくなるために必要な探索回数である．この重みはその合法手の探索回数が多くなるほど小さくなり，探索回数が少ない初期では RAVE による勝率を重視し，探索回数が多くなれば実際の勝率を重視することを示す．

この手法はゲームの知識を用いていないが，ゲームの性質に大きく依存する．この手法の利用できるゲームとは,「局面によって合法手に大きな違いが無い」・「局面によって合法手が局面に与える影響にほとんど違いが無い」といった条件を満たすものである．そのようなゲームは囲碁や五目並べといったものであり，そうでないゲームと比べて非常に少ない．

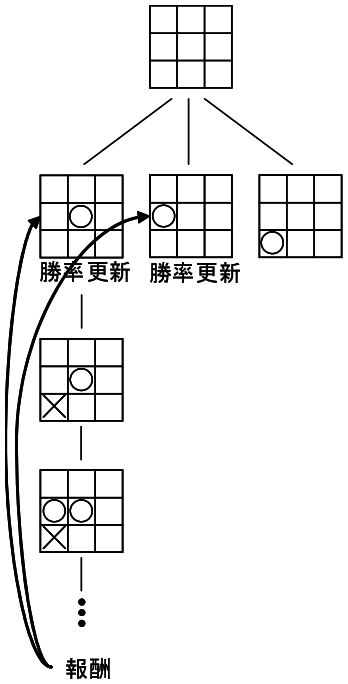

図 2.4: RAVE の報酬の与え方

表 2.3: UCT-RAVE の GnuGo に対する勝率 (S. Gelly et al. 2007.)

アルゴリズム	勝率
UCT	23.88 ± 0.85%
UCT-RAVE	60.47 ± 0.79%

2.3.7 確率的な試行回数削減

但馬らの研究 [15] では，UCT 探索においてそれぞれの合法手が選択されるのに必要な残り探索回数から，最も高い勝率とならない合法手を推定することでプレイアウト回数の削減を行った．その結果，そのような合法手削減に対して削減を行ってもプレイヤの強さに変化が現れないことを示した．

図 2.4 に文献 [15] による削減時の対戦結果を示す．ここでは対象となったゲームはブロックスデュオであり，表中の UCB1 とは第一層のみを UCB 値で，残りはランダムでプレイアウトを行うノード選択する手法である．

表 2.4: 削減を行った時の対戦結果 (但馬ら．2007.)

先手	後手	UCB1 削減法	UCB1	分け
UCB1 削減法	UCB1	64	33	4
UCB1	UCB1 削減法	33	63	5
計		97	96	9

第3章　提案手法

2章にも示されるように，モンテカルロ木探索の効率を改善するには，選択される可能性の低い合法手に対するプレイアウトの回数を少なく抑えることで，選択される可能性のある合法手に対するプレイアウト回数を多くすることが必要である．

本研究では残りプレイアウト回数に着目することで選択される可能性が低い合法手を見つけ出し，ゲームの知識やゲームの性質に依存しない枝刈りを行うことを目的とする．

3.1　有限計算量の下での探索の最適化

3.1.1　計算量の有限性と不要な探索

UCT 探索の目的は，与えられた局面に対して最善である合法手を選ぶことである．UCT 探索では，多腕バンディット問題における UCB1 アルゴリズムが基礎となっているために，無限に続く探索を行ったときに最善である合法手を見つけることを保証した手法である．しかし実際のゲームで UCT 探索を行う場合，プレイアウトの回数や探索に費やした時間等の何らかの条件により探索を打ち切り，その時点で最善と判断した合法手を選択することになる．このように使うことのできる計算量が有限であるために，残りの全プレイアウトを一つの合法手に対して集中的に行って全勝したとしても，最善であると判断されることの無い合法手が存在する可能性がある．2.3.7 節にも示されるように，そのような合法手に関しては計算を行わなかったとしても，得られる結果に影響を与えることは無い．よって，そのような合法手は枝刈りをすることが可能である．しかし，実際にはそのような合法手は，探索が終わる直前になり残り全プレイアウト回数が極端に少なくなった場合にならないと現れない可能性が高い．そのような状況で枝刈りを行ってもプレイヤとしての性能に与える影響は大きくないために，このような手法のみでは不十分である．

3.1.2　探索不要な合法手の推定

有限時間内で探索を打ち切る UCT 探索は確率的に最善である合法手を求める手法であり，真の最善手を得られるとは限らない．一般にゲームでは，プレイヤが最善であると判断した一つの合法手のみを選択するために，真の最善手や最善に近い合法手を見つける可能性を高めることが重要である．そのため評価値が低い合法手の評価値には高い精度は必要なく，評価値が高い合法手の評価値に高い精度が必要である．よって高い確率で不要である合法手の探索を放棄することで，真の最善手や最善に近い合法手を見つける可能性は上がる．以下，この方針を用いることで，残り全プレイア

ウト回数から選択される可能性の低い合法手を推定し，探索を放棄する手法について述べる．本論文では，残り全プレイアウト回数は推定できるものとし，N で表す．

3.2 不要な探索の推定

3.2.1 到達勝率

ある程度のプレイアウトが行われたノードにおいて，合法手 i の現在のプレイアウト回数を s_i，得られた報酬の和を X_i とする．また合法手 i が探索終了時までに行う残りプレイアウト回数を e_i とし，その残りのプレイアウトによって将来得られる報酬の平均を $\bar{Y_i}$ とする．この時，探索終了時での合法手 i の勝率 $\bar{P_i}$ は次の式で表される．

$$\bar{P_i} = \frac{X_i + Y_i}{s_i + e_i} \tag{3.1}$$

$$Y_i = \bar{Y_i} e_i \tag{3.2}$$

ここで Y_i は将来得られる報酬の和を表す．このように合法手 i の残りプレイアウト回数 e_i と将来得られる報酬の平均 $\bar{Y_i}$ を推定することができれば，探索終了時での勝率を推定することが可能である．

仮に探索終了時での勝率 $\bar{P_i}$ を完全に推定することができれば，残りの探索をする必要は無い．しかし現実には完全に推定することは不可能であるために，探索終了時に勝率が到達する区間の推定をする必要がある．本研究では，将来得られる報酬の区間と残りプレイアウト回数とを推定することで探索終了時に到達することのできる区間を推定することで枝刈りを行う．

3.2.2 残りプレイアウト回数推定

ある程度のプレイアウトが行われたノードにおいて，合法手 i の選択確率を q_i とする．ここでは q_i には真の値が存在し，不変であると仮定する．このとき残りのプレイアウトで合法手 i が選ばれる回数は，選択確率 q_i の二項分布で表される．選択確率 q_i を求めることができれば合法手 i の残りプレイアウト回数の推定をすることが可能である．よって q_i の期待値として $\bar{q_i}$ を現在までに合法手 i でプレイアウトが行われた割合と枝刈りの影響を考慮することで以下のように求める．

$$\bar{q_i} = \frac{s_i}{\sum_{j \in B} s_j} \tag{3.3}$$

ここで B は今回のプレイアウトで枝刈りされることなく探索対象となった合法手の集合である．しかし q_i は今回の探索の枝刈りに用いられているために，q_i を求めないと B を求めることはできな

い．よって B を求めることは困難である．ここで前回のプレイアウトの枝刈りされた合法手は今回のプレイアウトでも枝刈りされる可能性が高く，前回のプレイアウトの枝刈りされなかった合法手は今回のプレイアウトでも枝刈りされる可能性は低いことから，B は前回のプレイアウトで枝刈りされることなく探索対象となった合法手の集合で近似する．

$\bar{q}_i$ は合法手 i でプレイアウトが行われる確率の期待値であり，実際の選択確率 q_i は $\bar{q}_i$ よりも高い可能性がある．選択確率 q_i を低く見積もると残りプレイアウト回数の推定値も低く見積もられる．残りプレイアウト回数の推定値を低く見積もると，後述する探索終了時での勝率の区間を必要以上に狭く見積もり，危険な枝刈りを行ってしまう可能性がある．よって合法手 i の選択確率は正規分布に従うと仮定し，区間推定を行うことで上側信頼限界から q_i を以下のように見積もる．

$$q_i = \bar{q}_i + r\frac{\rho_i}{\sqrt{\sum_{j \in B} s_j}} \tag{3.4}$$

ここで ρ_i はプレイアウト毎に見た合法手 i が今までに選択された回数の標準偏差である．

残り n 回のプレイアウトが選択確率 q_i の二項分布に従うとすると，今後合法手 i に対してプレイアウトが行われる回数の期待値 $\bar{e}_i$ と標準偏差 b_i は以下のように求めることができる．

$$\bar{e}_i = q_i n \tag{3.5}$$

$$b_i = \sqrt{nq_i(1 - q_i)} \tag{3.6}$$

この二項分布の標準偏差によるバイアスを考慮することで，残りプレイアウト回数の推定値は現実のものより小さくなる可能性は低くなる．よって推定プレイアウト回数 e_i は最終的に以下のように見積もる．

$$e_i = \bar{e}_i + \alpha b_i \tag{3.7}$$

ここで α はバイアスの倍率であり，大きければ大きいほど推定プレイアウト回数を多めに見積もる．

なおルートノードからの合法手に対する残りプレイアウト回数の期待値を求める時には $n = N$ を用い，それ以外のノードからのプレイアウト回数の期待値を求める時には，そのノードの推定残りプレイアウト回数から $n = e$ を用いる (図 3.1) .

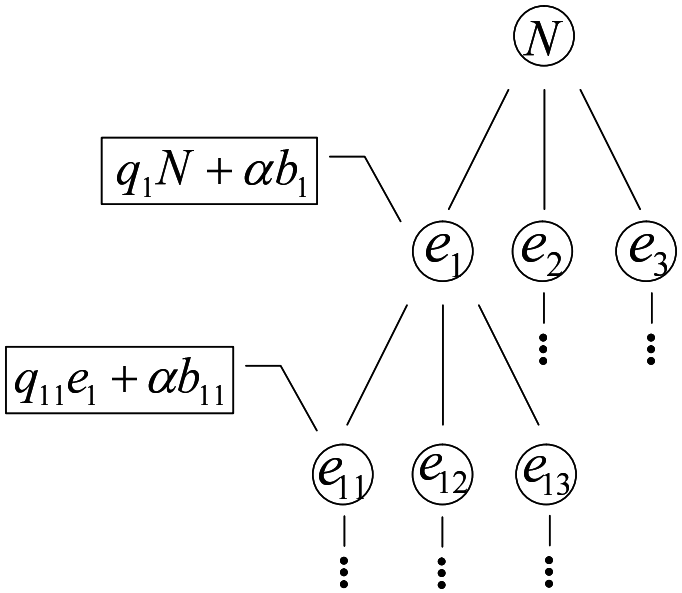

図 3.1: 残りプレイアウト回数

3.2.3 将来報酬推定

将来得られる報酬を，2.2.1 と同様に区間推定を用いることによって推定する．それぞれの合法手が将来得られる報酬の平均値は正規分布に従うと仮定し，勝率 $\bar{X}_i$ と得られた報酬の標準偏差 σ_i から将来得られる報酬の平均値が取り得る値の区間を推定する．ただし，一度のプレイアウトによって得られる報酬は $[0, 1]$ なので，最大値と最小値を与えることで将来得られる報酬の区間 $[\bar{Y}_{Li}, \bar{Y}_{Ri}]$ を以下のように与える．

$$\bar{Y}_{Li} = \max(0, \bar{X}_i - r\frac{\sigma_i}{\sqrt{s_i}}) \tag{3.8}$$

$$\bar{Y}_{Ri} = \min(1, \bar{X}_i + r\frac{\sigma_i}{\sqrt{s_i}}) \tag{3.9}$$

3.2.4 到達勝率区間推定

式 3.1・3.2 に式 3.7・3.8・3.9 を代入することによって，合法手 i が探索終了時に到達する勝率の区間 $[\bar{P}_{Li}, \bar{P}_{Ri}]$ を推定する．

$$\bar{P}_{Li} = \frac{X_i + Y_{Li}}{s_i + e_i} \tag{3.10}$$

$$Y_{Li} = \bar{Y}_{Li} e_i \tag{3.11}$$

$$\bar{P}_{Ri} = \frac{X_i + Y_{Ri}}{s_i + e_i} \tag{3.12}$$

$$Y_{Ri} = \bar{Y}_{Ri} e_i \tag{3.13}$$

図 3.2 に本提案手法による枝刈りのイメージ図を示す．図の横軸は勝率を表し，それぞれの合法手に対し $\bar{X}$ は現在の勝率を，$\bar{Y}_L, \bar{Y}_R$ は今後得られる報酬の平均値の区間を表す．これは 2.2.1 の Progressive Pruning における $\bar{X}_L, \bar{X}_R$ とほとんど同一のものである．

$\bar{P}_L, \bar{P}_R$ は探索終了時に到達する勝率の区間である．$e_i \to 0$ の時に $\bar{P}_L = \bar{P}_R = \bar{X}$ であり，$e_i \to \infty$ の時に $\bar{P}_L = \bar{Y}_L$，$\bar{P}_R = \bar{Y}_R$ となる．その間は e_i に対して単調増加・単調減少をするために，$\bar{Y}_L \leq \bar{P}_L \leq \bar{X} \leq \bar{P}_R \leq \bar{Y}_R$ が保証されている．よって Progressive Pruning よりも多くの合法手に対して枝刈りをすることができる．例えば，図 3.2 においては Progressive Pruning では合法手 k を枝刈りすることはできないが，本提案手法では枝刈りをすることができる．

本提案手法では，プレイアウトされた回数が少ない合法手であっても推定残りプレイアウト回数が少なくなることで枝刈りが進むことになる．よってプレイアウトの回数に偏りのできる UCT 探索においても枝刈りの効果を期待することができる．

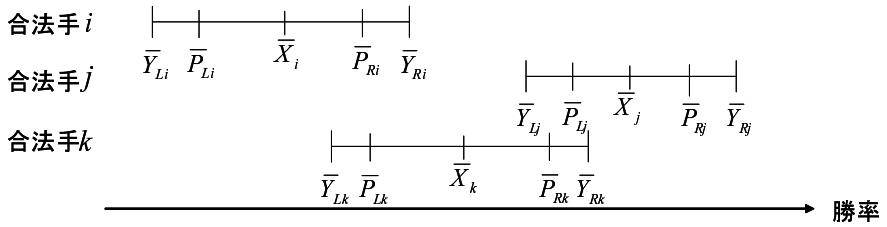

図 3.2: 提案手法による枝刈り

3.2.5 枝刈り条件

合法手 i と合法手 j に関して，$\bar{P}_{Ri} < \bar{P}_{Lj}$ である時，合法手 i の勝率が合法手 j の勝率より高くなる可能性は低い．よって合法手 i が選択される可能性は低いために，これ以上探索しても探索結果に影響を与える可能性は低く，枝刈りをすることができる．

枝刈り条件としては全ての合法手の $\bar{P}_L$ の中の最大値よりも $\bar{P}_R$ が低い合法手は枝刈りをしてもよいと言える．よって，そのノードの合法手数が m の時，合法手 i の枝刈り条件は以下のように表すことができる．

$$\bar{P}_{Ri} < \max(\bar{P}_{L1}, \bar{P}_{L2}, \cdots, \bar{P}_{Lm}) \tag{3.14}$$

3.2.6 枝刈りの流れ

前節までの議論により，投入計算量の有限性を考慮した枝刈り条件を得ることができた．本節では，その条件を用いることで具体的に枝刈りを行う手法について述べる．

図 3.3 にプレイアウト中の子ノード選択の流れを，図 3.4 に枝刈りの流れを示す．

3.2.2 と 3.2.3 で示すように，それぞれの合法手でプレイアウトが行われる確率や得られる報酬の平均値が正規分布に従うと仮定することで，残りプレイアウト回数推定や将来報酬推定を行っている．この仮定は，それぞれの合法手のプレイアウト回数がある程度多くないと成り立たない．よって，それぞれの合法手で枝刈りを行うのに必要な最低プレイアウト回数を $s_{\min}$ を定める．

プレイアウト中の子ノード選択では，枝刈りされる可能性のある合法手が存在しないなら通常の UCB 値を用いた子ノード選択を，枝刈りされる可能性のある合法手が存在するのなら枝刈り判定を行い，合法手の集合から枝刈りにあった合法手を除いた後に UCB 値を用いることで子ノード選択を行う．枝刈りされる可能性のある合法手が存在しないというのは，全ての合法手のプレイアウト回数が $s_{\min}$ 未満であることを示す．

枝刈りでは，$\max(\bar{P}_{L1}, \bar{P}_{L2}, \cdots, \bar{P}_{Lm})$ を計算し，$s_i \leq s_{\min}$ である合法手に対して式 3.14 を満たしているかを判定し，その条件を満たす合法手を合法手の集合から除く．

以上の手法をプレイアウト中に訪れた全てのノードに対して行うことで，本提案手法の枝刈りを実現する．

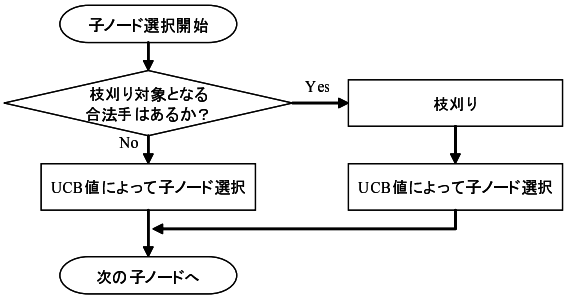

図 3.3: 子ノード選択の流れ

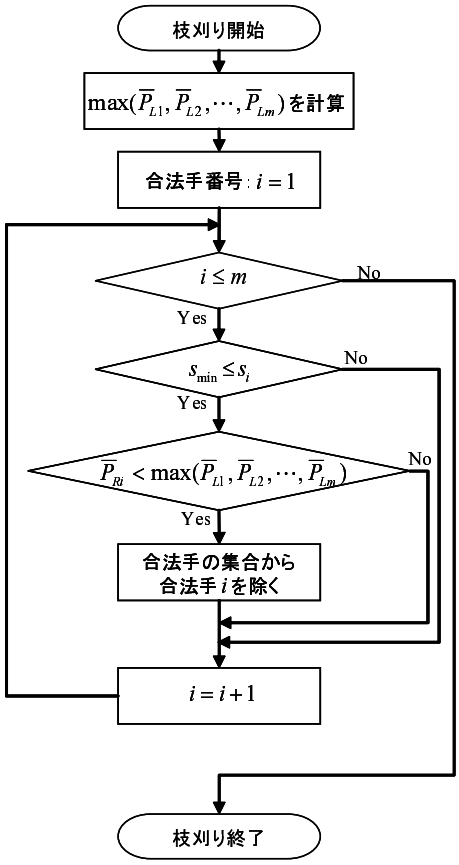

図 3.4: 枝刈りの流れ

第4章　実験と評価

本章では，3 章の提案手法の実験結果について述べる．4.4 で知識表現の難しいゲームとして囲碁に対して提案手法の性能評価を行った．

実験では通常の UCT 探索に提案手法を用いたプレイヤの性能評価を行った．また本手法が他の UCT 探索の改善手法と同時に用いることが可能なことを示すために，2.3.3 の UCB1-Tuned と提案手法を同時に用いたプレイヤ (以下，UCT-Tuned) の性能評価も行った．

4.1　実験環境

実験は以下の環境で行った．

- OS Ubuntu 8.04
- CPU AMD Opteron 2.1GHz QuadCore
- メモリ 3GB
- 実装言語 C++

4.2　実験の概要

本章では，提案手法の評価として，大きく分けて以下の 3 種類の実験を行った．

- 推定精度の実験
  (4.4.1 〜 4.4.3)
- 枝刈りに関する実験
  (4.4.4 〜 4.4.6)
- 対戦実験
  (4.4.7 〜 4.4.11)

推定精度の実験では，提案手法における残りプレイアウト回数や将来の報酬の推定がどの程度正しく行えているのかを評価した．4.4.1 節では 3.2.2 節に示した手法により残りプレイアウト回数の

推定実験を行った．4.4.2 節では 3.2.3 節に示した手法により将来得られる報酬の推定実験を行った．4.4.3 節では 3.2.4 節に示した手法によりそれぞれの合法手が到達する勝率の推定実験を行った．

枝刈りに関する実験では，提案手法による枝刈りの探索時のプレイアウトに与える影響や，枝刈りすることによって起きる合法手選択に関する変化について評価した．4.4.4 節では提案手法による枝刈りの進み方に関する実験を，4.4.5 節では提案手法による各合法手のプレイアウト回数の変化に関する実験を行った．4.4.6 節では枝刈りによる最善手を選ぶ可能性の変化に関する実験を行った．

対戦実験では，主に提案手法を組み込む前と後とでのプレイヤとしての強さの変化を評価した．そのために提案手法を組み込む前と後とでの直接対決による勝率や，同じ対戦相手に対しての勝率の変化について実験を行った．4.4.7 節ではプレイアウト回数を固定した時の対戦結果を，4.4.10 節では探索時間を固定した時の対戦結果を示した．最後に，4.4.11 節に，それぞれのアルゴリズムの GnuGo に対する対戦結果を示した．

4.3 実験のパラメータ

本節では，提案手法を囲碁の 9 路盤の UCT 探索に対して実装し，その性能評価を行った．

提案手法中に現れる各種パラメータは以下のように設定した．

- $r = 1.96$
- $\alpha = 3.0$
- $s_{\min} = 16$

r は信頼区間 95% の信頼係数から [19]，α はチェビシェフの不等式から，$s_{\min}$ は文献 [11] から得た．

4.4 囲碁による実験結果

4.4.1 残りプレイアウト回数推定実験

図 4.1・4.2 に 3.2.2 節に示した手法を用いることで，UCT 探索・UCT-Tuned それぞれにおいての第一層の残りプレイアウト回数の推定と実際の結果を示す．これらの図は最初に 10,000 回プレイアウトを行った時点でそれぞれの合法手に対して残りプレイアウト回数の推定を行い，その後 10,000 回のプレイアウトでのそれぞれの合法手のプレイアウト回数の実際の結果を示したものである．横軸は前半 10,000 回のプレイアウトを行った時点でのそれぞれの合法手のプレイアウト回数であり，縦軸は後半 10,000 回のプレイアウトによるプレイアウト回数の推定と実際の結果である．図中の result は後半 10,000 回のプレイアウトによる実際の結果，prediction が式 3.7 で求められる枝刈りに用いる残り推定プレイアウト回数 e である．また推定プレイアウト回数の平均値 $\bar{\epsilon}$ として，mean は式 3.5 の q に 3.3 の $\bar{q}$ を代入することで以下のように求めた．

$$\bar{\epsilon_i} = \bar{q}n \tag{4.1}$$

表 4.1 に上記と同様の実験を 100 回繰り返した時の，平均推定失敗率を示す．推定失敗率とは，実際のプレイアウト回数が推定プレイアウト回数を超えた合法手の割合であり，この値が高いと勝率の推定に失敗する可能性が上がる．

これらの結果から探索途中のある時点でのプレイアウト回数とその後のプレイアウト回数には相関性があると言える．またプレイアウト回数は，ある程度の推定ができている．しかし UCT-Tuned では，ある時点でのプレイアウト回数と今後のプレイアウト回数の相関性が弱いため推定失敗率が高くなっている．

残りプレイアウト回数推定に用いた全ての仮定が正しいとすると，用いたパラメータの値から推定失敗率は 5.5% 程となるはずである．UCT 探索・UCT-Tuned 共にこの値を大きく超えているので仮定に正しくない点が存在すると考えられる．その最も大きな原因は，式 3.4 によるそれぞれの合法手でプレイアウトが行われる確率の推定であると考えられる．ここでは不変である真のプレイアウト確率が存在することを仮定しているが，実際にはプレイアウトの結果によってその確率は変化するものである．プレイアウト確率の推定に，最近のプレイアウト確率を重視するなどの補正を用いることで，この問題点を解決することができる可能性がある．

表 4.1: 残りプレイアウト回数の平均推定失敗率

アルゴリズム	推定失敗率
UCT	11.7%
UCT-Tuned	18.7%

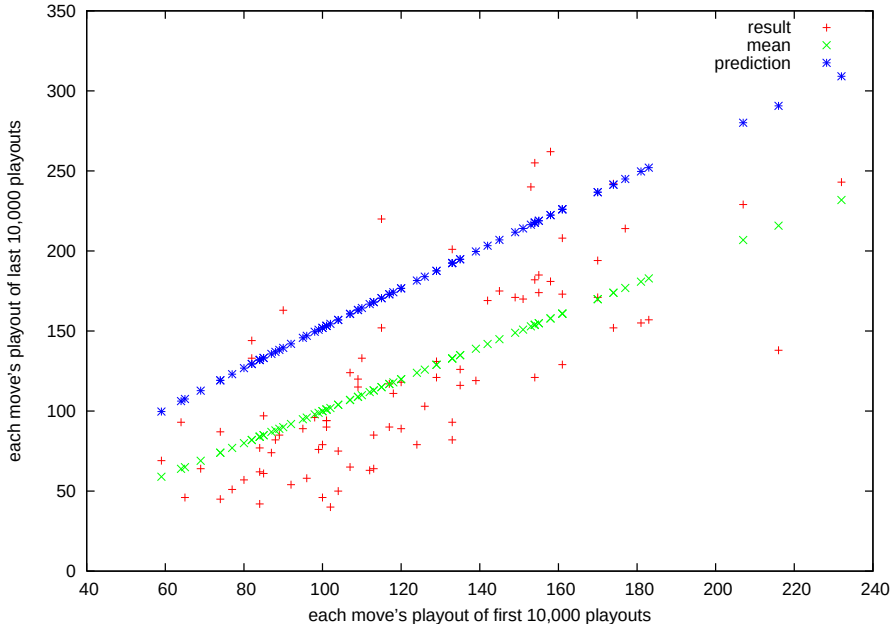

図 4.1: UCT 探索での残りプレイアウト回数推定

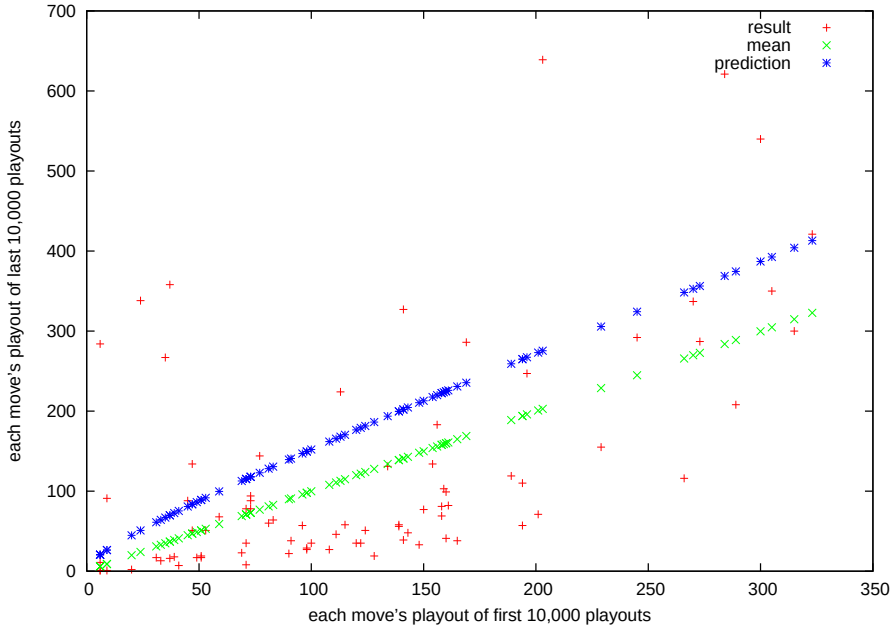

図 4.2: UCT-Tuned での残りプレイアウト回数推定

4.4.2 将来報酬推定実験

図 4.3・4.4 に 4.4.2 節に示した手法を用いた UCT 探索・UCT-Tuned それぞれにおいての第一層の将来報酬推定とその結果を示す．これらの図は最初に 10,000 回プレイアウトを行った時点でそれぞれの合法手に対して将来得られる報酬の平均値の推定を行い，その後 10,000 回のプレイアウトで得られた報酬の平均値を示したものである．横軸は前半 10,000 回のプレイアウトを行った時点でのそれぞれの合法手の報酬の平均値であり，縦軸は後半 10,000 回のプレイアウトによる報酬の平均値の結果と推定である．図中の result が実際の結果であり，mean は前半 10,000 回のプレイアウトの時点での報酬の平均値 $\bar{X}$ であり，upper は式 3.9 で求められる報酬の平均値の上側信頼限界 $\bar{Y}_R$，lower は式 3.8 で求められる報酬の下側信頼限界 $\bar{Y}_L$ である．

表 4.2 に上記と同様の実験を 100 回繰り返した時の平均推定失敗率を示す．表中の上側推定失敗率は報酬平均推定の上側信頼限界 $\bar{Y}_R$ よりも実際の報酬平均が大きくなった合法手の割合であり，下側推定失敗率は報酬平均推定の下側信頼限界 $\bar{Y}_L$ よりも実際の報酬平均が小さくなった合法手の割合である．また合計推定失敗率は推定区間 $[\bar{Y}_L,\bar{Y}_R]$ に実際の報酬平均が入らなかった割合である．

実験で与えたパラメータは信頼区間 95% の時の信頼計数なので，得られる報酬の平均が不変であれば合計推定失敗率は 5% となるはずである．しかし，UCT 探索・UCT-Tuned 共に合計推定失敗率は 5% を大きく上回っている．これは図 4.3・4.4 及び表 4.2 から分る通り，下側推定失敗率が高いためである．これには次のような原因が考えられる．UCT 探索・UCT-Tuned では探索の前半ではそれぞれのノードは何も知識を持っていないプレイヤであり，合法手選択はほぼランダムに近い．しかしプレイアウトが進むにつれ，それぞれのノードは過去のプレイアウトによる知識を持つために勝つ可能性の高い合法手を選択するようになる．この影響を最も受けやすいのはプレイアウトが行われる回数から考えて深さ 2 のノードであり，それらのノードは相手プレイヤを表す．つまり対戦すると想定している相手プレイヤの強さがプレイアウトの前半と後半で違うということである．よって後半の相手プレイヤは前半の相手プレイヤよりも強いために，前半の相手プレイヤとのプレイアウトによる報酬から後半の相手プレイヤとのプレイアウトによる報酬を推定すると，実際には推定よりも低い報酬しか得られないということが多く発生すると考えられる．

表 4.2: 将来報酬の平均推定失敗率

アルゴリズム	上側推定失敗率	下側推定失敗率	合計推定失敗率
UCT	1.7 %	21.6 %	23.3 %
UCT-Tuned	2.6 %	29.4 %	32.0 %

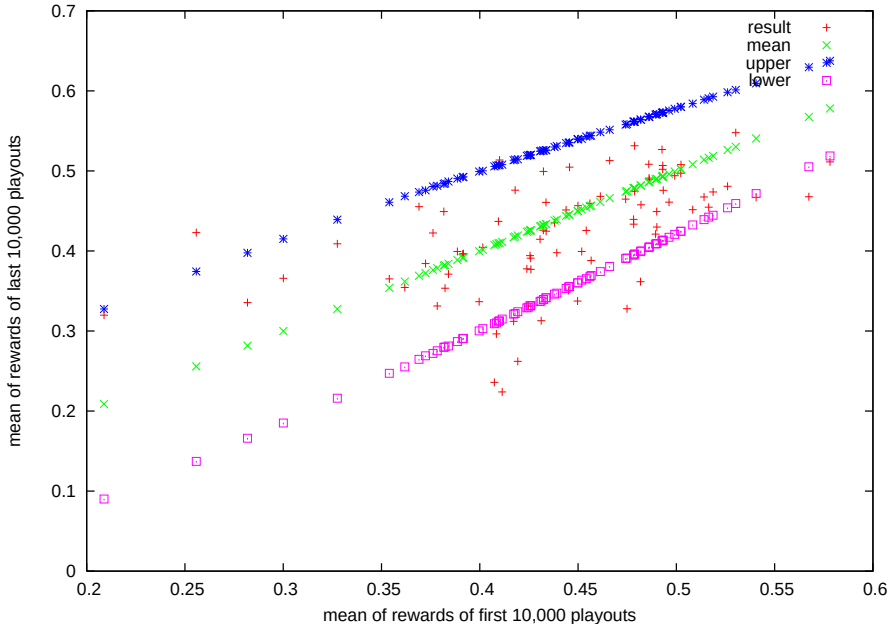

図 4.3: UCT 探索での将来報酬推定

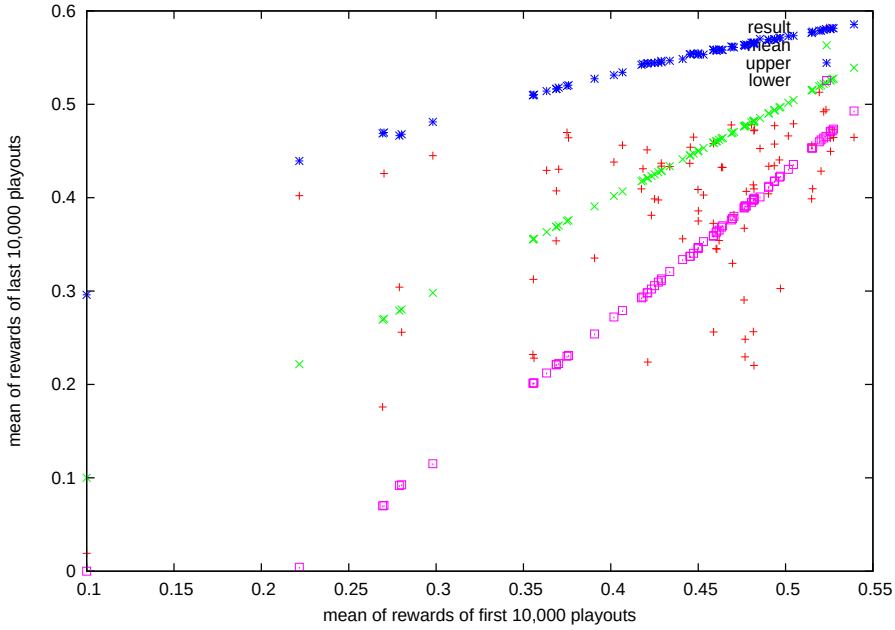

図 4.4: UCT-Tuned での将来報酬推定

4.4.3 到達勝率推定実験

図 4.5・4.6 に 4.4.3 節に示した手法を用いた UCT 探索・UCT-Tuned それぞれにおいての第一層の到達推定とその結果を示す．これらの図は最初に 10,000 回プレイアウトを行った時点でそれぞれの合法手に対して将来到達する勝率の推定を行い，その後 10,000 回のプレイアウトを行った時に到達した勝率を示したものである．横軸は前半 10,000 回のプレイアウトを行った時点でのそれぞれの合法手の勝率であり，縦軸は後半 10,000 回のプレイアウトを行った時に到達した勝率の結果と推定である．図中の result が実際の結果であり，mean は前半 10,000 回のプレイアウトの時点での勝率 $\bar{X}$ であり，upper は式 3.12 で求められる到達勝率の上限 $\bar{P}_R$，lower は式 3.10 で求められる到達勝率の下限 $\bar{P}_L$ である．

表 4.3 に上記と同様の実験を 100 回繰り返した時の平均推定失敗率を示す．表中の上側推定失敗率は到達勝率推定の上限 $\bar{P}_R$ よりも実際の勝率が大きくなった合法手の割合であり，下側推定失敗率は到達勝率推定の下限 $\bar{P}_L$ よりも実際の報酬平均が小さくなった合法手の割合である．また合計推定失敗率は推定到達勝率区間 $[\bar{P}_L,\bar{P}_R]$ に実際の報酬平均が入らなかった割合である．

表 4.3 から到達推定の推定失敗率は，残りプレイアウト回数と将来報酬の推定失敗率よりも低いことが分る．また図 4.5・4.6 から到達勝率の推定に失敗した合法手に関しても推定値と比べ，大きく外れてはいないことが示されている．これは残りプレイアウト回数と将来報酬の推定が完全に将来得られるものだけを推定しているのに対し，到達勝率の推定では推定時までに得られた報酬やプレイアウト回数も考慮されるために，既に確定した情報も用いているためである．

表 4.3 からは UCT 探索よりも UCT-Tuned での推定失敗率が低く，枝刈りしてはいけない合法手に枝刈りしてしまう可能性が低いように見える．しかし式 3.14 より，枝刈りをする時に $\bar{P}_L$ は全合法手のうち最大である一つのみが用いられるのに対し，$\bar{P}_R$ は全合法手のものが用いられる．そのために $\bar{P}_R$ の推定失敗率である上側推定失敗率の大きさが枝刈りしてはいけない合法手に枝刈りしてしまう可能性に依存していると考えられる．よって UCT 探索よりも UCT-Tuned の方が枝刈りしてはいけない合法手に枝刈りしてしまう可能性は高いと考えられる．

表 4.3: 到達勝率の平均推定失敗率

アルゴリズム	上側推定失敗率	下側推定失敗率	合計推定失敗率
UCT	1.6 %	6.7 %	8.4 %
UCT-Tuned	3.5 %	2.7 %	6.2 %

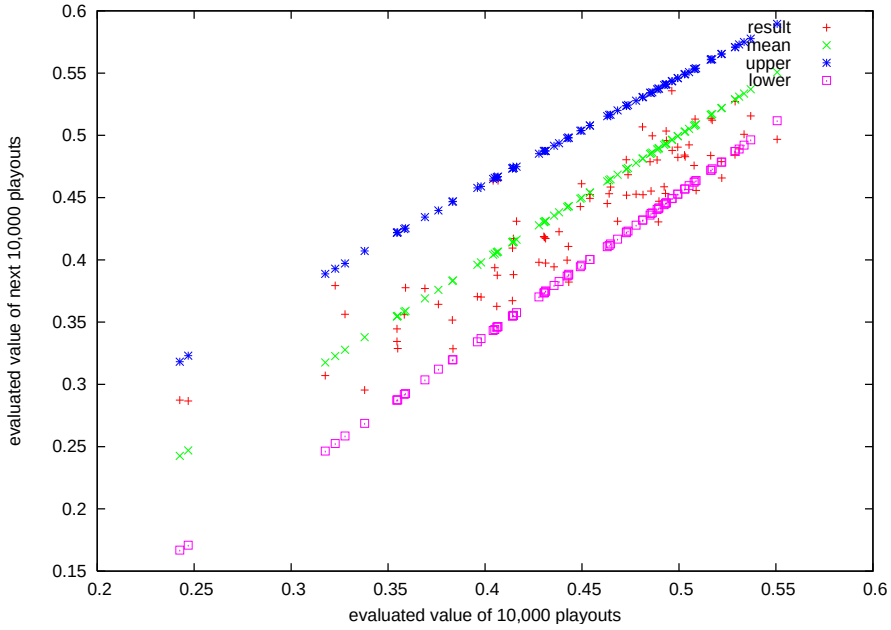

図 4.5: UCT 探索での到達勝率推定

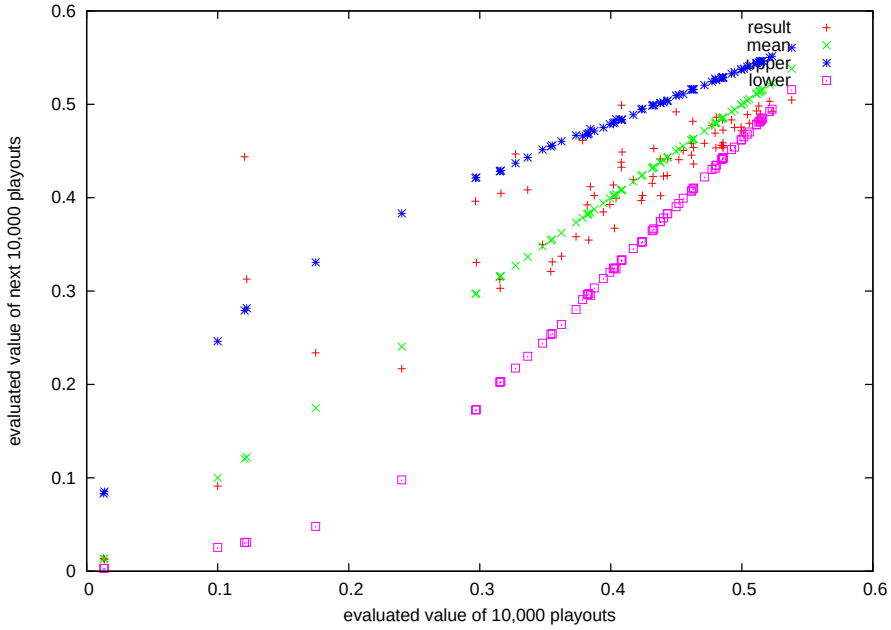

図 4.6: UCT-Tuned での到達勝率推定

4.4.4 枝刈りの進行

図 4.7 に UCT 探索の，図 4.8 に UCT-Tuned の枝刈りの進行の様子をプロットしたグラフを示す．これらのグラフは初期局面に対しての第一層での枝刈りの様子を示している．また探索打ち切り条件を全体のプレイアウト回数が 20,000 となった時とした．グラフの横軸が全体のプレイアウト回数で，縦軸が全合法手に対する枝刈りされた合法手の割合である．

UCT 探索も UCT-Tuned も共に枝刈りされた合法手の割合が下がることがあるが，それは探索が進むことで $\bar{P}_L$ の最大値が下がり，式 3.14 に示す枝刈り条件が緩和されたためである．しかし多少の上下はあるものの，全体としてみれば残りプレイアウト回数が少なくなることで枝刈りは進行している．

UCT-Tuned の方が UCT 探索よりも枝刈りの進行が遅いことが読み取れる．これは UCT-Tuned では，勝率の低い合法手はプレイアウトが行われる回数が少なく，そのような合法手では推定した到達勝率区間が狭くなりにくいためである．しかし全体の残りプレイアウト回数が少なくなるにつれ，プレイアウト回数の少ない合法手であっても推定した到達勝率区間は狭くなることで枝刈りが進み，UCT 探索の場合と似たような挙動を示している．

4.4.5 プレイアウト回数の割合

前節では，残りプレイアウト回数が少なくなる毎に枝刈りが進行することを示した．しかし UCT 探索・UCT-Tuned 共に，評価値の低い合法手はプレイアウト回数が少なくなるようなアルゴリズムであるため，枝刈りにどのような効果があるのか判断するのは難しい．この節では，それぞれの合法手のプレイアウト回数の割合からその点の評価を行う．

表 4.4 に，それぞれのアルゴリズムが最善と判断し，選択した合法手がプレイアウトされた回数の割合を示す．この値が大きいと，選択された合法手に関しての評価値の精度は高くなる．この表はランダムに作成した 5 手目の局面 100 個に対して 20,000 回のプレイアウトを行った時の平均の結果である．また図 4.9 に各順位毎のプレイアウト回数の割合のグラフを示す．横軸が評価順位であり，縦軸がその順位の合法手のプレイアウト回数の割合である．

UCT 探索・UCT-Tuned 共に提案手法を組み込むことで，選択された合法手のプレイアウト回数の割合は同程度上昇している．また UCT 探索に関しては 9 位以上，UCT-Tuned に関しては 4 位以上の評価順位の合法手に関して提案手法を組み込むことでプレイアウト回数は上昇している．このことから少なくともそれらの合法手の中から最善の手を選ぶ可能性は上がっていると言える．

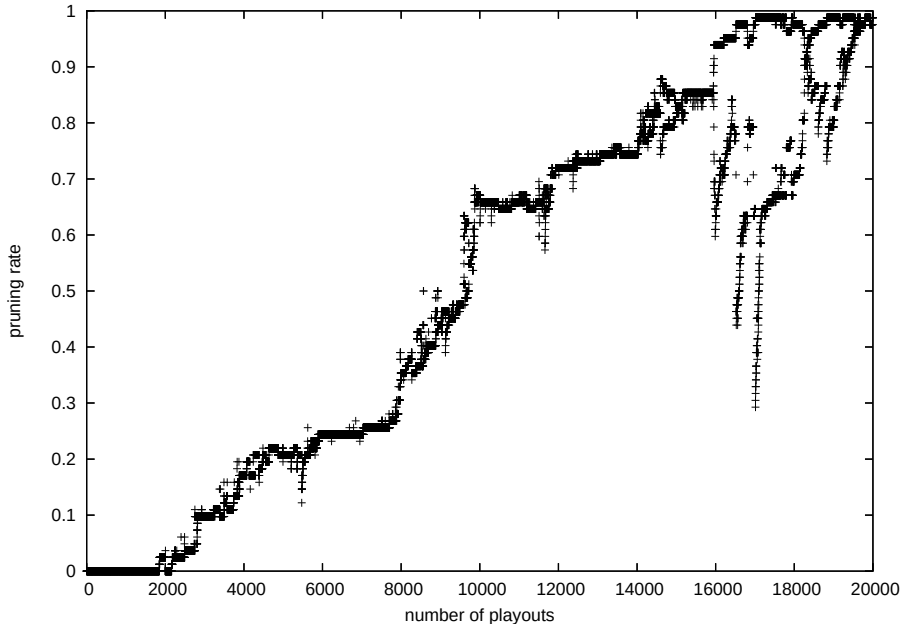

図 4.7: UCT 探索での枝刈りの進行

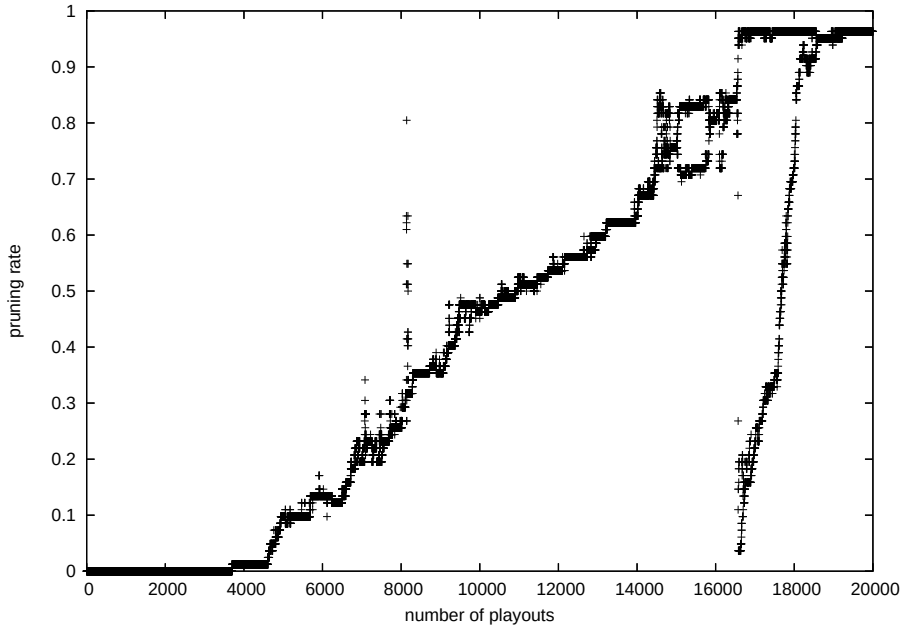

図 4.8: UCT-Tuned での枝刈りの進行

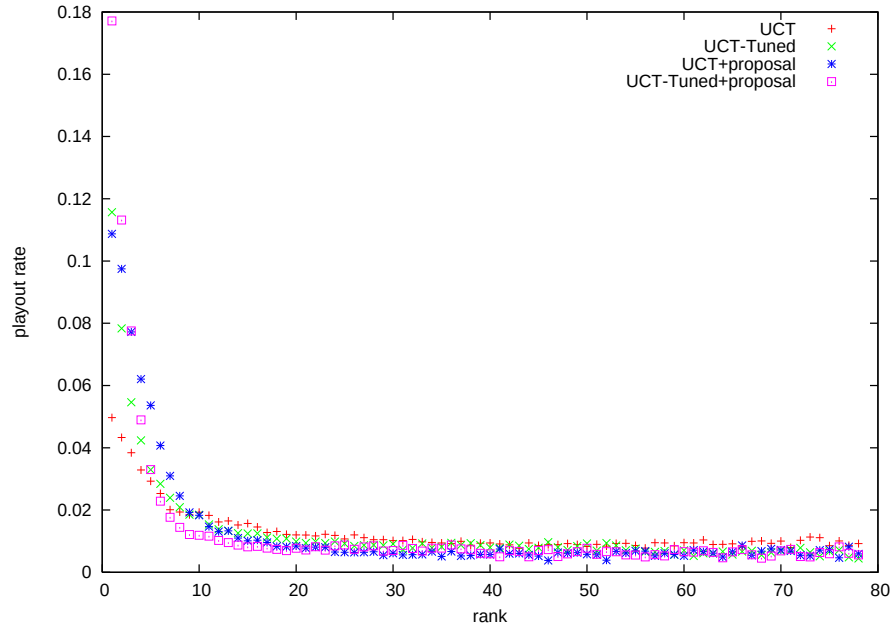

図 4.9: 合法手の評価順位毎のプレイアウト回数の割合

表 4.4: 選択された合法手のプレイアウト回数の割合

アルゴリズム	プレイアウト回数の割合
UCT	4.5 %
UCT+提案手法	10.9 %
UCT-Tuned	11.6 %
UCT-Tuned+提案手法	17.7 %

4.4.6 Fuego との合法手選択比較

前節より，評価順位が上位の合法手に関しては提案手法を組み込むことでプレイアウト回数の割合が上昇することを示すことができた．しかし仮に上位の合法手に本来の最善手が含まれる可能性が低いとすると，提案手法を組み込むことでそれぞれのアルゴリズムを用いたプレイヤは弱くなってしまう可能性がある．この節では本来の最善手を定めることで，その点に関する評価を行う．

前節と同様のランダムに作成した 100 局面に対して，Fuego (version 0.2.2) [20] を用いることで最善手を求めた．Fuego は様々な UCT 探索の拡張が行われた囲碁プレイヤである．9 路盤の CGOS[21] (Internet Go Server) でのレーティングは 2500 程であった．また ICGA Tournaments 2008[22] の 9 路盤部門では世界 4 位となっている．Fuego で 100,000 回のプレイアウトによって選択した合法手を，それぞれの局面に対する本来の最善手とした．Fuego は世界トップクラスの囲碁プログラムであるため，この合法手が最善手や最善手に近い合法手である可能性は高い．

20,000 回のプレイアウトによる各種アルゴリズムのそれぞれの局面の最善手に対する評価を調べることで，枝刈りの影響に関する評価を行った．

表 4.5 に各種アルゴリズムによる，本来の最善手の平均評価順位と標準偏差を示す．また表 4.6 に各種アルゴリズムが選択した合法手が Fuego の最善手と一致した割合を示す．図 4.10 に，各種アルゴリズムの最善手の評価順位のグラフを示す．横軸は評価順位で，縦軸は最善手がその評価順位以上に含まれている割合である．例えば，横軸が 20 の点は，評価順位 20 位以上に最善手が含まれる割合を示している．

UCT 探索・UCT-Tuned 共に提案手法を組み込むことで，最善手の平均評価順位が悪化し，標準偏差も増えている．また UCT 探索は提案手法を組み込むことで，本来の最善手が評価順位の上位に含まれる割合は上がっているが，評価順位 12 位以上からは本来の最善手が含まれる割合は逆転されている．UCT-Tuned に提案手法を組み込むことでは，評価順位 1 位以外では本来の最善手が含まれる割合は下がっている．これらは最善手に対して枝刈りが発生し，最善手に対して極端に悪い順位を付けてしまうことがあるということを示している．しかし提案手法を組み込むことで実際に最善手を選択する割合 (評価順位 1 位の割合) は改善されている．これは最善手に枝刈りが発生しなかった時には，評価順位上位の合法手の中から最善手を見つけ出す可能性が高くなっていることを示している．また，そのことから最善手に枝刈りをしてしまった場合でも，評価順位が上位の合法手の中では最善に近い合法手を見つけ出すことができる可能性も高くなっていると考えられる．

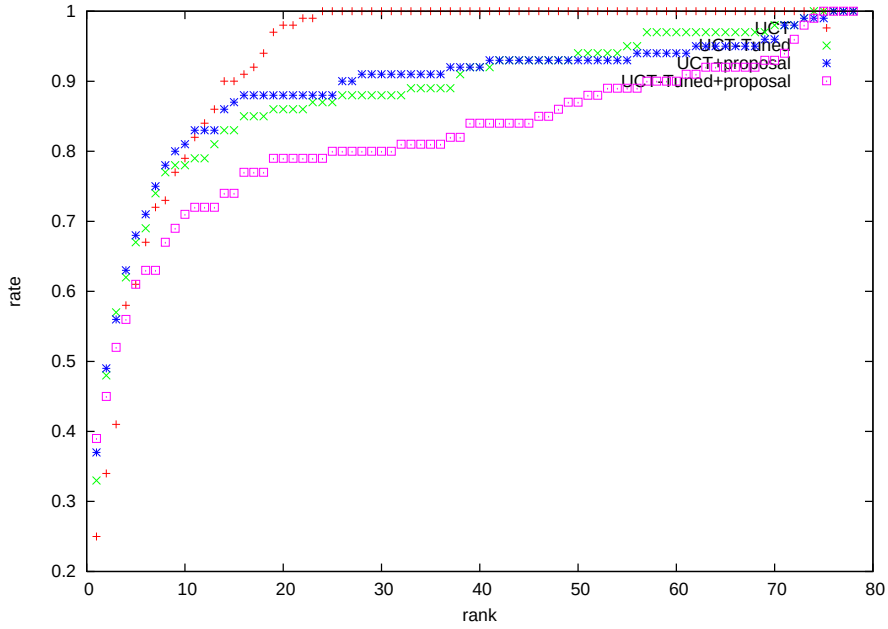

図 4.10: 評価順位

表 4.5: Fuego の最善手の平均評価順位

アルゴリズム	平均順位	標準偏差
UCT	6.13	5.72
UCT+提案手法	9.64	17.68
UCT-Tuned	10.06	16.96
UCT-Tuned+提案手法	14.99	22.74

表 4.6: Fuego との合法手選択一致率

アルゴリズム	一致率
UCT	25 %
UCT+提案手法	37 %
UCT-Tuned	33 %
UCT-Tuned+提案手法	39 %

4.4.7 プレイアウト回数固定による対戦結果

本節では環境や実装などに影響されないように，プレイアウト回数を固定することで対戦を行う．

UCT 探索・UCT-Tuned を用いるプレイヤとしては，一手につき 10,000 回，20,000 回，40,000 回，80,000 回のプレイアウトを行うものを用意した．それらのアルゴリズムに提案手法組み込んだとプレイヤして，一手につき 10,000 回，20,000 回，40,000 回のプレイアウトを行うものを用意した．それぞれ提案手法を組み込んでいないプレイヤと組み込んだプレイヤを対戦させることで，提案手法の有効性を示す．

対戦は黒白 50 戦ずつの計 100 戦を行った．

提案手法を組み込んだプレイヤのプレイアウト回数が，提案手法を組み込んでいないプレイヤのプレイアウト回数より多い場合は勝ち越すことが自明であるために省略した．

表中の括弧内はプレイアウトの回数であり，計の項目で * 印が付いているものは有意水準 5% の二項検定で有意に勝ち越したことを示す．

4.4.7.1 UCT 探索 + 提案手法 の対戦結果

表 4.7・4.8・4.9 に UCT 探索を用いたプレイヤと UCT 探索に提案手法を組み込んだプレイヤとの対戦結果を示す．

提案手法を組み込んだプレイヤは，同じプレイアウト回数を用いたプレイヤと比べ全てにおいて有意に勝ち越している．また 2 倍のプレイアウト回数を用いたプレイヤに対しても有意に勝ち越している．またプレイアウト回数が 10,000 回と 20,000 回の提案手法を組み込んだプレイヤは 4 倍のプレイアウト回数を用いたプレイヤと比べても勝ち越している．

表 4.7: UCT + 提案手法 (10,000) 対 UCT の 勝利数

黒	白	UCT+提案手法	UCT
UCT + 提案手法	UCT (10,000)	39	11
UCT (10,000)	UCT + 提案手法	38	12
計		77*	23
UCT + 提案手法	UCT (20,000)	42	8
UCT (20,000)	UCT + 提案手法	32	18
計		74*	26
UCT + 提案手法	UCT (40,000)	34	16
UCT (40,000)	UCT + 提案手法	29	21
計		63*	37
UCT + 提案手法	UCT (80,000)	23	27
UCT (80,000)	UCT + 提案手法	17	33
計		40	60*

表 4.8: UCT + 提案手法 (20,000) 対 UCT の 勝利数

黒	白	UCT+提案手法	UCT
UCT + 提案手法	UCT (20,000)	45	5
UCT (20,000)	UCT + 提案手法	39	11
計		84*	16
UCT + 提案手法	UCT (40,000)	41	9
UCT (40,000)	UCT + 提案手法	37	13
計		78*	22
UCT + 提案手法	UCT (80,000)	32	18
UCT (80,000)	UCT + 提案手法	21	29
計		53	47

表 4.9: UCT + 提案手法 (40,000) 対 UCT の 勝利数

黒	白	UCT+提案手法	UCT
UCT + 提案手法	UCT (40,000)	48	2
UCT (40,000)	UCT + 提案手法	44	6
計		92*	8
UCT + 提案手法	UCT (80,000)	39	11
UCT (80,000)	UCT + 提案手法	37	13
計		76*	24

4.4.7.2 UCT-Tuned + 提案手法 の対戦結果

表 4.10・4.11・4.12 に UCT-Tuned を用いたプレイヤと UCT-Tuned に提案手法を組み込んだプレイヤとの対戦結果を示す.

提案手法を組み込んだプレイヤは，同じプレイアウト回数を用いたプレイヤと比べ全てにおいて有意に勝ち越している．しかし 2 倍のプレイアウト回数を用いたプレイヤに対しては負け越している．特にプレイアウト回数 10,000 の時には有意な差が出ている．

表 4.10: UCT-Tuned + 提案手法 (10,000) 対 UCT-Tuned の勝利数

黒	白	UCT-Tuned+提案手法	UCT-Tuned
UCT-Tuned + 提案手法	UCT-Tuned (10,000)	33	17
UCT-Tuned (10,000)	UCT-Tuned + 提案手法	28	22
計		61*	39
UCT-Tuned + 提案手法	UCT-Tuned (20,000)	13	37
UCT-Tuned (20,000)	UCT-Tuned + 提案手法	18	32
計		31	69*
UCT-Tuned + 提案手法	UCT-Tuned (40,000)	5	45
UCT-Tuned (40,000)	UCT-Tuned + 提案手法	10	40
計		15	85*
UCT-Tuned + 提案手法	UCT-Tuned (80,000)	2	48
UCT-Tuned (80,000)	UCT-Tuned + 提案手法	3	47
計		5	95*

黒	白	UCT-Tuned+提案手法	UCT-Tuned
UCT-Tuned + 提案手法	UCT-Tuned (20,000)	35	15
UCT-Tuned (20,000)	UCT-Tuned + 提案手法	33	17
計		68*	32
UCT-Tuned + 提案手法	UCT-Tuned (40,000)	25	25
UCT-Tuned (40,000)	UCT-Tuned + 提案手法	21	29
計		46	54
UCT-Tuned + 提案手法	UCT-Tuned (80,000)	8	42
UCT-Tuned (80,000)	UCT-Tuned + 提案手法	11	39
計		19	81*

表 4.11: UCT-Tuned + 提案手法 (20,000) 対 UCT-Tuned の勝利数

表 4.12: UCT-Tuned + 提案手法 (40,000) 対 UCT-Tuned の勝利数

黒	白	UCT-Tuned+提案手法	UCT-Tuned
UCT-Tuned + 提案手法	UCT-Tuned (40,000)	32	18
UCT-Tuned (40,000)	UCT-Tuned + 提案手法	28	22
計		60*	40
UCT-Tuned + 提案手法	UCT-Tuned (80,000)	21	29
UCT-Tuned (80,000)	UCT-Tuned + 提案手法	22	28
計		43	57

4.4.7.3 プレイアウト回数固定による対戦結果のまとめ

UCT 探索・UCT-Tuned 共に同じプレイアウト回数のプレイヤに対しては，提案手法を組み込んだプレイヤは有意に勝ち越すことができた．しかし UCT-Tuned による実験では UCT 探索による実験に比べて勝率が低かった．また UCT 探索では 2 倍のプレイアウト回数のプレイヤに対しても有意に勝ち越せたのに対し，UCT-Tuned では勝ち越すことができなかった．これは UCT-Tuned では勝率が極端に低い合法手のプレイアウト回数が非常に低くなり枝刈りに大きな効果が無いことと，4.4.3 節に示したように到達勝率推定の正確さが影響していると考えられる．

4.4.8 提案手法のコスト

前節までではプレイアウト回数を固定することでの対戦結果を示したが，本提案手法ではプレイアウト毎に枝刈り条件を更新するために，従来手法と比べて計算量が増えてしまう．特に囲碁ではプレイアウトに必要な計算量のコストが少ないために，枝刈り条件の計算に必要なコストは無視することができない．

表 4.13 に実験に用いたプログラムでの，100,000 回のプレイアウトを初期局面に対して行った時の各種アルゴリズムの 1 秒間のプレイアウト回数を示す．

提案手法を組み込むことで，元のアルゴリズムより 20% 程 1 秒間に行うことのできるプレイアウトの回数は少なくなっている．実際のプレイヤでは探索時間により探索を打ち切ることが多いために，実装の方法によるものの，提案手法を加えることでプレイアウト回数が減り，元のアルゴリズムよりも弱くなる可能性がある．よって次節では探索時間を固定したプレイヤによる対戦結果を示す．

表 4.13: 初期局面での 1 秒間のプレイアウト回数

アルゴリズム	playout/s
UCT	9900
UCT-Tuned	8700
UCT + 提案手法	7600
UCT-Tuned + 提案手法	7000

4.4.9 残り全プレイアウト回数の推定

本提案手法では，残り探索時間内の残り全プレイアウト回数が必要となる．探索打ち切り条件を探索時間とした場合，この値は自明ではない．よって以下の式によってある時点での残り全プレイアウト回数 N を推定した．

$$N = \frac{u}{t}(T - t) \tag{4.2}$$

ここで T は探索打ち切り時間，t は経過探索時間，u はそれまでのプレイアウト回数である．

4.4.10 探索時間固定による対戦結果

本節では枝刈り条件の計算によるコストを考慮して，探索時間を固定することで対戦を行う．

UCT 探索・UCT-Tuned を用いるプレイヤとしては，一手につき 1 秒，2 秒，4 秒，8 秒のプレイアウトを行うものを用意した．それらのアルゴリズムに提案手法組み込んだとプレイヤしては，一手につき 1 秒，2 秒，4 秒のプレイアウトを行うものを用意した．それぞれ提案手法を組み込んでいないプレイヤと組み込んだプレイヤを対戦させることで，時間固定であっても本提案手法が有効であることを示す．

対戦は黒白 50 戦ずつの計 100 戦を行った．

表中の括弧内は探索時間であり，計の項目で * 印が付いているものは有意水準 5% の二項検定で有意に勝ち越したことを示す．

4.4.10.1 UCT + 提案手法 の対戦結果

表 4.14・4.15・4.16 に UCT 探索を用いたプレイヤと UCT 探索に提案手法を組み込んだプレイヤとの対戦結果を示す．

提案手法を組み込んだプレイヤは，同じ探索時間を用いたプレイヤと比べ全てにおいて有意に勝ち越している．また 2 倍の探索時間を用いたプレイヤに対しても有意に勝ち越している．しかし 4 倍の探索時間を用いたプレイヤに対しては勝ち越してはいるものの，有意な差は現れなかった．

表 4.14: UCT + 提案手法 (1 秒) 対 UCT の勝利数

黒	白	UCT+提案手法	UCT
UCT + 提案手法	UCT (1 秒)	43	7
UCT (1 秒)	UCT + 提案手法	36	14
計		89*	11
UCT + 提案手法	UCT (2 秒)	32	18
UCT (2 秒)	UCT + 提案手法	33	17
計		65*	35
UCT + 提案手法	UCT (4 秒)	30	20
UCT (4 秒)	UCT + 提案手法	27	23
計		57	43
UCT + 提案手法	UCT (8 秒)	24	26
UCT (8 秒)	UCT + 提案手法	18	32
計		42	58

表 4.15: UCT + 提案手法 (2 秒) 対 UCT の勝利数

黒	白	UCT+提案手法	UCT
UCT + 提案手法	UCT (2 秒)	44	6
UCT (2 秒)	UCT + 提案手法	39	11
計		83*	17
UCT + 提案手法	UCT (4 秒)	39	11
UCT (4 秒)	UCT + 提案手法	33	17
計		72*	28
UCT + 提案手法	UCT (8 秒)	33	17
UCT (8 秒)	UCT + 提案手法	22	28
計		55	45

表 4.16: UCT + 提案手法 (4 秒) 対 UCT の勝利数

黒	白	UCT+提案手法	UCT
UCT + 提案手法	UCT (4 秒)	46	4
UCT (4 秒)	UCT + 提案手法	38	12
計		84*	16
UCT + 提案手法	UCT (8 秒)	42	8
UCT (8 秒)	UCT + 提案手法	32	18
計		74*	26

4.4.10.2 UCT-Tuned + 提案手法 の対戦結果

表 4.17・4.18・4.19 に UCT-Tuned を用いたプレイヤと UCT-Tuned に提案手法を組み込んだプレイヤとの対戦結果を示す.

提案手法を組み込んだプレイヤは，同じ探索時間を用いたプレイヤと比べ全てにおいて有意に勝ち越している．しかし 2 倍の探索時間を用いたプレイヤに対しては全てにおいて有意に負け越している.

表 4.17: UCT-Tuned + 提案手法 (1 秒) 対 UCT-Tuned の勝利数

黒	白	UCT-Tuned+提案手法	UCT-Tuned
UCT-Tuned + 提案手法	UCT-Tuned (1 秒)	30	20
UCT-Tuned (1 秒)	UCT-Tuned + 提案手法	29	21
計		59*	41
UCT-Tuned + 提案手法	UCT-Tuned (2 秒)	13	37
UCT-Tuned (2 秒)	UCT-Tuned + 提案手法	19	31
計		32	68*
UCT-Tuned + 提案手法	UCT-Tuned (4 秒)	7	43
UCT-Tuned (4 秒)	UCT-Tuned + 提案手法	12	38
計		19	81*
UCT-Tuned + 提案手法	UCT-Tuned (8 秒)	1	49
UCT-Tuned (8 秒)	UCT-Tuned + 提案手法	2	48
計		3	97*

表 4.18: UCT-Tuned + 提案手法 (2 秒) 対 UCT-Tuned の勝利数

黒	白	UCT-Tuned+提案手法	UCT-Tuned
UCT-Tuned + 提案手法	UCT-Tuned (2 秒)	34	16
UCT-Tuned (2 秒)	UCT-Tuned + 提案手法	27	23
計		61*	39
UCT-Tuned + 提案手法	UCT-Tuned (4 秒)	18	32
UCT-Tuned (4 秒)	UCT-Tuned + 提案手法	19	31
計		37	63*
UCT-Tuned + 提案手法	UCT-Tuned (8 秒)	5	45
UCT-Tuned (8 秒)	UCT-Tuned + 提案手法	8	42
計		13	83*

表 4.19: UCT-Tuned + 提案手法 (4 秒) 対 UCT-Tuned の勝利数

黒	白	UCT-Tuned+提案手法	UCT-Tuned
UCT-Tuned + 提案手法	UCT-Tuned (4 秒)	32	18
UCT-Tuned (4 秒)	UCT-Tuned + 提案手法	28	22
計		60*	40
UCT-Tuned + 提案手法	UCT-Tuned (8 秒)	22	28
UCT-Tuned (8 秒)	UCT-Tuned + 提案手法	13	37
計		35	65*

4.4.10.3 探索時間固定による対戦結果のまとめ

UCT 探索・UCT-Tuned 共に同じ探索時間を用いたプレイヤに対しては，提案手法を組み込んだプレイヤは有意に勝ち越すことができた．しかし UCT-Tuned による実験では UCT 探索による実験に比べて勝率が低かった．また UCT 探索では 2 倍の探索時間のプレイヤに対しても有意に勝ち越せたのに対し，UCT-Tuned では有意に負け越した．これらはプレイアウト回数を固定した時と比べ，提案手法による性能上昇が少なくなっていることを示している．4.4.8 節に示したように，枝刈り条件判定によるプレイアウト回数の減少と，式 4.2 を用いた，残り全プレイアウト回数の推定の誤差がこの原因であると考えられる．

4.4.11 GnuGo との対戦結果

各種アルゴリズム用いたプレイヤを同一のプレイヤと対戦させることで，それぞれのアルゴリズムの性能差を示す．ここでは対戦相手として GnuGo (3.7.12) の level 0 と level 10 を用いた．各種アルゴリズムはプレイアウト回数 20,000 で固定した．

GnuGo は評価関数を利用した従来型のプログラムであり，9 路盤の CGOS[21] (Internet Go Server) でのレーティングは 1,800 程であった．

表 4.20 に各種アルゴリズムと GnuGo level 0 との対戦結果を，4.21 に各種アルゴリズムと GnuGo level 10 との対戦結果を示す．

GnuGo level 0 と対戦した時と level 10 と対戦した時の両方で，勝率の高いものから順に UCT-Tuned + 提案手法 > UCT-Tuned > UCT + 提案手法 > UCT となった．つまり UCT からの勝率上昇度で言えば，提案手法よりも UCT-Tuned の方が効果が高いと言える．しかし UCT-Tuned と提案手法を同時に用いることで勝率は更に上がっているので，本提案手法の有効性は示すことができている．

表 4.20: 各種アルゴリズム 対 GnuGo level 0 の勝利数

黒	白	各種アルゴリズム	GnuGo
UCT	GnuGo level 0	16	34
GnuGo level 0	UCT	17	33
計		33	67*
UCT + 提案手法	GnuGo level 0	27	23
GnuGo level 0	UCT + 提案手法	25	25
計		52	48
UCT-Tuned	GnuGo level 0	31	19
GnuGo level 0	UCT-Tuned	37	13
計		68*	32
UCT-Tuned + 提案手法	GnuGo level 0	38	12
GnuGo level 0	UCT-Tuned + 提案手法	34	16
計		72*	28

表 4.21: 各種アルゴリズム 対 GnuGo level 10 の勝利数

黒	白	各種アルゴリズム	GnuGo
UCT	GnuGo level 10	7	43
GnuGo level 10	UCT	9	41
計		16	84*
UCT + 提案手法	GnuGo level 10	21	29
GnuGo level 10	UCT + 提案手法	24	26
計		45	55
UCT-Tuned	GnuGo level 10	29	21
GnuGo level 10	UCT-Tuned	27	23
計		56	44
UCT-Tuned + 提案手法	GnuGo level 10	30	20
GnuGo level 10	UCT-Tuned + 提案手法	31	19
計		61*	39

第5章　結論

5.1　まとめ

本研究では，3 章に提案したように，探索時間の有限性を利用することで UCT 探索の枝刈りを行い，コンピュータゲームプレイヤの性能向上を行った．そのために 3.2.2 節に示した今後のプレイアウト回数の推定と，3.2.3 節に示した将来得られる報酬の区間の推定を行った．そして，それらを用い 3.2.4 節に示した手法により，探索終了時にそれぞれの合法手の勝率が到達する区間を推定した．その結果，3.2.5 に示した手法により今後プレイアウトを行ったとしても選択される見込みの無い合法手を見つけ出し，その合法手に対するプレイアウトを放棄することで枝刈りを実現した．

本研究では，囲碁の 9 路盤に対しての UCT 探索を用いたプレイヤと UCT-Tuned を用いたプレイヤに提案手法を実装することで評価を行った．その結果，実験内の全ての環境で，提案手法を組み込んだプレイヤは組み込んでないプレイヤに対して有意に勝ち越すことができた．

それぞれの推定実験では，提案手法に対する以下の知見が得られた．

- 残りプレイアウト回数推定と将来報酬推定の推定失敗率と比べると，到達勝率推定の推定失敗率は低い．
- UCT 探索よりも UCT-Tuned の方が推定の失敗率は高くなる．
- 推定の失敗により，枝刈りされてしまった本来の最善手も少なくない．
- 本来の最善手が枝刈りされなかったら，評価順位上位の合法手の中から見つけ出す可能性は上がる
- 全体としては，本来の最善手を見つける可能性は上がる．

以上の結果から，UCT-Tuned は UCT 探索よりも，本来の合法手に対して枝刈りをしてしまう可能性が高いということが言えた．また本来の最善手に対して枝刈りが発生し低い評価順位を付けてしまうことはあるものの，全体としては提案手法を用いることで本来の最善手を選択する可能性は上がるということが言えた．しかし本来の最善手に枝刈りが発生した場合であっても，評価順位上位の合法手の中では一番良い合法手を選択する可能性が高くなると考えられる．

対戦結果としては，提案手法を組み込む前と後のプレイヤに同じプレイアウト回数を与えて対戦させたところ，UCT 探索では 8 割前後，UCT-Tuned では 6 割以上勝ち越すことができた．また同

じ探索時間を与えて対戦させたところ，UCT 探索では 8 割以上，UCT-Tuned では 6 割前後勝ち越すことができた．

GnuGo に対する勝率を調べる実験によって，提案手法を単体で用いたものは UCT-Tuned よりも性能が低いという結果が得られた．しかし UCT-Tuned に対して提案手法を用いることで，それぞれ個々に利用していた時よりも更に性能が向上していた．このことから提案手法は他の UCT 探索の改良アルゴリズムと同時に用いることが可能であることが示された．

以上により，投入計算量の有限性を意識することでアルゴリズムの改善ができることが示された．冒頭にも示したように，このアプローチはゲーム木探索に限ったものではなく，様々なアルゴリズムに適用できる可能性がある．これから計算機を用いて必ずしも精度の高い解が得られるとは限らないような非常に難しい問題を解こうとする場合や，組み込みのための CPU で複雑な処理を実行する場合など，投入計算量の有限性を意識したアルゴリズムは必ず必要になる。本研究はそのようなアルゴリズムのひとつの指針を与えるものである．

5.2 今後の課題

本提案手法を組み込むことで，本来枝刈りをしてはいけない合法手に対して枝刈りが発生し，最善手を見逃してしまうことがあった．この問題の改善には以下の手法が考えられる．

パラメータの動的制御 本提案手法では枝刈りの判定を左右する 3 つのパラメータ r，α，$s_{\min}$ が存在する．本論文の実験では，このパラメータは過去の実験や分布の性質より尤もらしい定数を与えた．しかしゲームの進行度や局面の情勢によってこのパラメータを動的に制御することで，推定失敗率を改善することが期待できる．

局面評価の初期値付与 枝刈りをしてはいけない合法手に枝刈りが発生した最大の原因は，プレイアウト回数があまり多くない時期に低い勝率となってしまい，将来性の無い合法手であるとみなされてしまったことである．評価関数などを用いて局面に評価値の初期値を与え，ある程度はプレイアウト序盤の評価はその値に左右されるようにする．このようにすることで評価値の初期値の高い合法手に枝刈りが発生し難くなり，この問題は改善されると考えられる．

プレイアウトの時期による重み付け UCT 探索では，プレイアウトの前半と後半では探索の挙動や得られる報酬の質が違う．そのために 2.3.4 節の Discounted UCB のような考え方を，本提案手法の 3.2.2 の残りプレイアウト回数推定と 3.2.3 の将来報酬の推定にも用いることで推定精度の向上が期待できる．残りプレイアウト回数や将来報酬は過去のプレイアウトによって得られた結果よりも，最近のプレイアウト結果によって得られた結果の方に左右されやすいと考えられる．このことは Discounted UCB を UCT 探索に組み込むことで，プレイヤの性能が上がることからも示されている．よって最近のプレイアウトの結果を重視するように，プレイアウトの時期による重み付けを推定に対しても行うことによって，推定精度が向上し，本来枝刈りをしてはいけない合法手に対する枝刈りが減少すると考えられる．

また本提案手法の根本となっている「時間の有限性によるアルゴリズムの最適化」という考え方は，ゲーム木探索に対してだけではなく，他の分野にも応用できる可能性がある．現在では携帯電話や自動車など様々な製品に対して組み込みのための CPU が搭載されている．これらの CPU はサーバや家庭用 PC の CPU と比べて処理性能が低く，製品の機能として搭載されたアルゴリズムの実行に時間がかかるといった問題が発生する可能性がある．このような時に「時間の有限性によるアルゴリズムの最適化」という考え方を用いることで製品の機能を向上させることができる可能性がある．そのような利用のためにも，ゲーム木探索以外のアルゴリズムでも時間の有限性を用いることで，そのアルゴリズムの性能が向上することを示すことが必要であると考えられる．

参考文献

[1] R. Coulom. Computing elo ratings of move patterns in the game of Go. In Computer Games Workshop, 2007.

[2] S. Gelly, Y. Wang, R. Munos, O. Teytaud. Modification of UCT with Patterns in Monte-Carlo Go. RR-6062-INRIA, pp.1–19, 2006.

[3] S. Gelly, D. Silver. Combining online and offline knowledge in UCT. Proceeding sof the 24th International Conference on Machine Learning, pp. 273 – 280, OmniPress, 2007.

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

[5] L. Kocsis, C. Szepesvári. Bandit based monte-carlo planning. In Enropean Conference on Machine Learning, pp. 282 – 293, 2006.

[6] L. Kocsis, C. Szepesvári. DiscountedUCB. 2nd PASCAL Challenges Workshop, 2006.

[7] M. Buro. Experiments with Multi-Prob Cut and new high-quality evaluation function for Othello. Technical Report No.96, NEC Research Institute, 1997.

[8] M. Buro. From Simple features to sophisticated evaluation functions. In H. J. van den Herik and H. Iida, editors, Proceedings of the First International Conference of Computers and Games (CG-98), Vol. 1558, pp. 126 – 145, 1998.

[9] G. Tesauro. Temporal Difference Learning and TD-GAMMON. Communications of the ACM, Vol. 38, No. 3, pp. 55 – 68, 1995.

[10] K. Tournavitis. MOUSE (μ) : A Self-teaching Algorithm that Achieved Master-Strength at Othello. Computers and Games (2003) , pp. 11 – 28, 2003.

[11] B. Bouzy. Move-Pruning Techniques for Monte-Carlo Go. CG 2005 LNCS, vol. 4250, pp. 104 – 119, Springer, Heidelberg, 2006.

[12] B. Bouzy, B. Helmstetter. Monte-Carlo Go Developments. In 10th Advances in Computer Games, pp, 159 – 174, 2004.

[13] P. Auer, N. Cesa-Bianchi, P. Fischer. Finite-time analysis of the multiarmed bandit problem. Machine Learning, 47 (2/3), pp. 235 – 256, 2002.

[14] 三輪誠. ゲーム知識を表現する語彙の棋譜データからの自動獲得. 博士論文, 東京大学, 2008.

[15] 但馬康宏, 小谷善行. UCT アルゴリズムにおける確率的な試行回数削減方法. 情報処理学会ゲーム情報学研究会報告, vol.2008, no.59, 2008-GI-20, pp.23–30, 2008.

[16] 大崎泰寛, 柴原一友, 但馬康宏, 小谷善行. TDMC(λ) に基づく評価関数の調整. 第 13 回ゲームプログラミングワークショップ 2008, pp. 73 – 79, Nov. 2008.

[17] 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 2006, pp. 78 – 83, Nov. 2006.

[18] 北川竜平, 三輪誠, 近山隆. 麻雀の牌譜からの打ち手評価関数の学習. 第 12 回ゲームプログラミングワークショップ 2007, pp. 76 – 83, Nov. 2007.

[19] 北川竜平, 栗田哲平, 近山隆. 投入計算量の有限性に基づく UCT 探索の枝刈り. 第 13 回ゲームプログラミングワークショップ 2008，pp. 46 – 53，Nov. 2008.

[20] Fuego. http://fuego.sourceforge.net/

[21] Computer Go Server. http://cgos.bordspace.net/

[22] ICGA Tournaments. http://www.grappa.univ-lille3.fr/icga/

発表文献

査読付会議論文

[1] 北川竜平, 三輪誠, 近山隆．麻雀の牌譜からの打ち手評価関数の学習．第 12 回ゲームプログラミングワークショップ 2007, pp. 76 – 83, Nov. 2007.

[2] 北川竜平, 栗田哲平, 近山隆．投入計算量の有限性に基づく UCT 探索の枝刈り．第 13 回ゲームプログラミングワークショップ 2008，pp. 46 – 53，Nov. 2008.

謝辞

本研究を進めるにあたり，大変多くの方にお世話になりました．

近山隆教授，田浦健次郎准教授には，論文作成やプレゼンテーションなど要所で的確なご指導，ご教授を頂きました．個々の論文に関することから研究者としての考え方まで様々な面で多くの知識を伝授して頂きました．

元近山研究室博士の三輪誠さんには，研究の進め方や論文の書き方等，日頃から多くの助言を頂きました．博士過程を修了されて，辻井研究室の特任研究員になられてからも，勉強会等を通じてご意見ご指摘を頂きました．

他の近山・田浦研究室の皆様にも公私にわたり多くの助言を頂きました．

近山研究室に入るまではプログラミングも研究の進め方も全く分らない状態であった自分が本研究を進めることができたのも，ご協力いただいた皆様のおかげであると考えています．

ここに，心より感謝の意を表します．

平成 21 年 1 月 27 日