博士論文

モンテカルロ木探索の改善に関する研究

今川　孝久

目 次

図 目 次

表 目 次

概 要

本論文は二人零和完全情報ゲームを主な対象とした，モンテカルロ木探索（MCTS）の研究について記したものである．ゲームにおいては，全ての局面を読み切ることは出来ないことが一般的である．それは，一般に，ゲームは探索空間が広く，かつ手を打つまでの時間が限られているためである．従って，探索では，効率的に最善手を見つけることが要請される．MCTS は，モンテカルロシミュレーションを繰り返し，その結果が良い選択肢を優先的に探索するアルゴリズムである．MCTS では，シミュレーション結果に対し，利得（良さ）を与え，各候補手を利得の平均値で評価し，評価の高い手を優先して，先読みとシミュレーションを行う．最善手の判別のためには，各候補手を選んだ場合の真の利得の期待値を推定する必要がある．代表的な MCTS では，多腕バンディット問題（MAB）における，累積リグレット（累積的な損失）の最小化のためのアルゴリズムを，シミュレーションを開始する節点の決定に応用している．MCTS に累積リグレットの最小化のためのアルゴリズムを応用することで，MAB に同様に，徐々に最善手を多く選ぶようになり，シミュレーションの平均的な利得が，最善を尽くした場合の利得に近づくと期待される．

MCTS は汎用なアルゴリズムであり，General Game Playing や，General Video Game Playing という初見のゲームを総合的に上手くプレイすることが求められるドメインで成果を上げてきた．また，囲碁や Hex といった個々のゲームでも主流な探索手法である．特に囲碁では，初めてプロ棋士に勝ったプログラム AlphaGo での探索に採用され，その勝利に貢献した．

しかしながら，MCTS は常に優れた性能を発揮出来る訳では無く，比較的苦手なドメインがあることが知られている．そして，MCTS の性能とドメインの特徴の関係はまだ不明な点も多く，その関係性を明らかにする必要がある．また，代表的な MCTS ではシミュレーションを開始する節点を選ぶことを MAB とみなして，MAB のアルゴリズムを用いているが，MAB と木探索では異なる点があり，MAB の性質は成り立たない．そのため，MAB より木探索の性質に近い仮定とその仮定に基づくアルゴリズムについて調べる価値がある．

本研究では，MCTS に関して，ドメインの種類とそこでの性能，理論的な仮定といった観点から議論し，既存手法の問題点を指摘し，その改善策を提示した．具体的には，四つの点に着目して研究を行った．

第一点目として，MCTS が比較的苦手とされるドメインの特徴の一つとして，最善手を選び損ねた場合の不利益の偏りという特徴に新たに着目する．そして，その特徴と MCTS の性能の関係性について分析する．MCTS が成果を上げてきたドメインの一つである，囲碁の中でも，攻め合いの局面は比較的苦手であると知られている．本研究で扱う特徴を持つ局面は，攻め合いの局面と共通の難しさを有している．本研究では，シミュレーションの結果，囲碁での目数の差といったゲーム上のスコアが手に入るという

仮定の下で，理論的な分析に基づく利得の与え方を提案する．そして，既存手法が性能を発揮しづらい特徴を持つ局面で，提案手法が効果的であることを示した．

第二点目として，MCTS における探索資源の割り振りに着目し，直接的に単純リグレット（最終的な手の選択での損失）の最小化を目指した MCTS を提案する．探索の目的は最善手を判別することであり，これは単純リグレットを最小化することである．しかしながら，探索において，MAB のアルゴリズムを使い，単純リグレットを直接最小化することは難しいため，累積リグレットの最小化のための手法が MCTS に応用されてきた．本研究では，探索での単純リグレットの直接的な最小化に向け，分布を使った MCTS を提案する．提案手法は，扱う分布を真の利得の主観確率の分布と仮定すると，最善手の判別のために，探索資源の割り振りが効果的になされると期待できる手法である．実験の結果，提案手法の性能は，深さが一様な木では，既存手法を超えないものの，非一様な木では上回った．

第三点目として，MCTS における勝敗の確定に着目する．探索木中のある節点が終端（ゲームの終わり）の場合，その節点での勝敗が確定する．これは，シミュレーションによる確率的な評価しか得られない MCTS において，例外的に確実な評価が得られるという意味で重要である．しかしながら，既存手法では，その情報を枝刈りにしか利用していないという問題点がある．本研究では，勝敗確定の情報を枝刈りだけでなく，推定値の修正とそれに応じた探索の優先度度合いの調整に利用する方法を提案する．計算機実験により，提案手法は既存手法よりも効果的であることを示した．

第四点目として，MCTS における候補手の利得の推定値に着目する．最善手の判別のためには，最善を尽くした時の利得を正確に推定する必要がある．特に，最善手は以後最善を尽くすと仮定した場合での最も良い利得の手であるので，期待利得の最大値の推定が重要である．まず，MAB の設定下で，利得の期待値の最大値に対する，新しい推定量 Simplified Weighted Estimator (SWE) を提案する．各選択肢の試行回数が予め決まっているという仮定のもとで，SWE について理論的な分析を行い，試行回数が増えるにつれて，推定値が正しい値に収束することを示した．さらに，累積リグレット最小化のためのアルゴリズムに基づいて試行回数が決まる場合での，推定値の正確さについて，実験を行い検証した．その結果．設定次第では，既存手法の性能が SWE を超える場合もあるが，設定に対する推定量の性能の安定性という観点で SWE が優れていることを示した．特に，MCTS で行われている，平均による推定値は，正しい値と比べて過度に低くなることを示した．加えて，SWE を MCTS に応用し，実験で性能を確かめる．実験は，終端節点で利得が確率的に得られる木と確定的に得られる木の 2 種類で行った．提案手法は後者の木では，既存手法を超えないものの，前者の木では上回る性能を示した．

以上のように本研究は，複合的なアルゴリズムである MCTS について，利得の与え方，候補手の評価の仕方，評価に対する探索資源の割り振り方という三つの要素技術のそれぞれの観点から議論し，改善策を提示したと言える．

第1章　序論

1956 年ダートマスにて，マッカーシーらによって会議が行われた．いわゆるダートマス会議である．そこで Artificial Intelligence （人工知能）という言葉が初めて使われ，人工知能という分野が生まれたと言える．その 3 年後にはサミュエルによってチェッカーのプログラムに関する論文 [60] が発表されたことに代表されるように，人工知能研究の初期からゲームを題材に研究が行われてきた．ゲームを上手くプレイ出来るということは，相手の行動を予測し，その行動に対して自分の効用を大きくするという戦略的な行動を効率的に見つけられるということであり，それは，人間の持つ知性の一側面である．ゲームを上手くプレイする方法（探索アルゴリズム）について研究することは，間接的に人間の知性を理解することに繋がるという意義がある．また，ゲームはルールが明確なため，研究対象として扱い易いという良い性質もある．

ゲームの内でも特にチェスは，人間のプレイヤーも多く，長い歴史があり，人間のプレイが洗練されているということもあり，コンピュータプログラムがチェスで人間のトッププロに勝つことを目標に効率的な探索についての研究がなされてきた．1997 年にプログラムが，初めて人間の世界チャンピオンであるカスパロフに 6 戦中 2 勝 1 敗 3 引き分けで勝ち越した際には，ミニマックス探索を基本とする技術が用いられた [12]．チェスでトッププロに勝った後は，より探索空間が広く，なおかつプロのプレイヤーが存在するゲームである，囲碁でトッププロに勝つことが新たな目標の一つとなった．

ミニマックス探索は，ゲームの探索木（節点が局面，辺が指し手に対応する）を展開し，葉の局面の良し悪しを判定し，そして，互いに最善を尽くすという前提で指し手の優劣を判定する方法である．もし葉がゲームの終局であったなら，ゲームの勝敗を局面の良し悪しの評価に使える．しかし，例えば，囲碁やチェスの状態空間の大きさはそれぞれ $10^{172}, 10^{47}$ [31] であると推定されているように，一般にゲーム木は大きく，終局まで探索木の葉が到達することは現実的では無い．そのような状況下で，効率的にミニマックス探索するために，終局でない局面の良し悪しを正しく評価する関数，評価関数が必要である．チェスや将棋では良い評価関数が作られて，プログラムの強さも人間のトップレベル達したが，囲碁で信頼に足る評価関数が作るのは難しいと考えられていた [1]．実際に，囲碁でミニマックス探索を用いたプログラムの強さは，弱いアマチュアの段階にとどまっていた．

しかし，2006 年にモンテカルロ木探索（MCTS）が提案され，ミニマックス探索に

[1] その後，2016 年に囲碁プログラム AlphaGo に関する論文 [61] で示されたように，深層学習で作った評価関数だけを用いてアマチュア 4 ，5 段の程度のレーティングであることから，十分に信頼に足る評価関数は作れていると言える．

代えて用いられるようになって以来，CrazyStone をはじめとした囲碁プログラムの強さは急激に向上した [17]．また，近年では MCTS を深層学習と組み合わせ AlphaGo が囲碁のトッププロに勝ったことでも MCTS は注目されている [61]．MCTS はモンテカルロシミュレーションを行い，その結果で各手を評価し，その評価が良い手を優先して先読みする手法である．これは，シミュレーション結果をもとにオンラインで良い手を学習しながら探索する手法と解釈が出来る．このため，予め学習等で評価関数を用意しなくても，それなりに上手く探索出来，囲碁以外にも幅広い応用先を持つ．

MCTS の応用先として，例えば，General Game Playing（GGP）が挙げられる．GGP では，複数の初見のゲームを万遍なく上手にプレイする AI を目指している．GGP の大会において，MCTS を用いたプレイヤーが優勝したことから，MCTS の有効性が明らかになった [8]．また，類似のものとして，テレビゲームを題材とした General Video Game Playing（GVGP）が挙げられる．GVGP でも MCTS アルゴリズムが好成績を収めている [55]．さらに，GVGP では，MCTS を使ったエージェントがサンプルとして提供されているように [24,55]，標準的なアルゴリズムとなっている．MCTS は他にも，プランニング [20] や 2.2.4 節で紹介するマルコフ決定過程だけでなく，状態が部分的にしか観測出来ない部分観測マルコフ決定過程 [65] にも応用されている．

MCTS の一種 UCT [42] は，多腕バンディット問題（MAB）のアルゴリズム UCB1 [3] を MCTS に応用したものであり，代表的なアルゴリズムである．MCTS が囲碁プログラム CrazyStone に採用された 2006 年に UCT は提案されたという意味で，初期から存在し，また，UCB1 に基づく理論的な解析がなされたことや，UCB1 の簡便さや高い実用性から広く用いられている．しかし，UCT をはじめとして MCTS はチェスや将棋で性能が限定的であり [2]，囲碁でも置き碁や攻め合いなどが苦手であると経験的に知られているように，MCTS が有効に働くためのゲームの性質は明らかでない．また，代表的な MCTS ではシミュレーションを開始する節点を選ぶことを MAB とみなして，MAB のアルゴリズムを用いているが，MAB と木探索では異なる点があり，MAB の性質は成り立たない．そのため，MAB より木探索の性質に近い仮定とその仮定に基づくアルゴリズムについて調べる価値がある．また，後に記すように改善の余地があることが分かってきた．

本研究では，二人零和有限確定完全情報ゲームを主な対象とし，MCTS の内，特に UCT に関して，その欠点を明らかにするとともに，改善策について議論した．大きく分けて 2 つの観点から研究を行った．1 つ目は主流の MCTS アルゴリズムの性能とゲームの特徴の関係を分析することである．主流の MCTS が上手く性能を発揮出来ない局面の特徴が分かれば，より効果的な MCTS，あるいは新たな探索手法の考案の手がかりになる．また，そのような効果的な探索手法が考案出来ない場合でも，その特徴を持つ局面では，旧来のミニマックス探索を用いる等という対策が可能である．つまり，そ

[2]囲碁プログラム AlphaGoZero [63] を一般化し，囲碁以外のゲームでも学習出来るようにした AlphaZero [62] では，自己対戦により学習し，囲碁だけでなく，チェス，将棋でも手法の有効性を示した．そして，AlphaZero で使われている探索は MCTS と言及されている．しかしながら，局面を評価する際に，モンテカルロシミュレーションを行う代わりに評価関数を使う点で，従来の一般的な MCTS とは異なる．

の特徴を見極める手法を考案出来れば，旧来の手法を合わせて用いることで全体として性能が改善するという期待が持てる．２つ目は MCTS 自体の改善である．２つ目に関して，主に３つの観点から研究を行った．１つ目は最善手を見分けるという観点，２つ目は終局時の情報の扱いという観点，３つ目はシミュレーション結果からの手の評価という観点である．MCTS の全般的な性能の改善には大きな意義がある．しかし，特定の局面でしか改善しなかったとしても，前述の通り，その特徴を持つ局面を見つけ出すことが可能な手法と組み合わせることで，全体として性能を改善できると期待される．また，特定の局面で何故改善したのかを突き詰めることで，理解を深め，より優れた MCTS に向けた知見が得られるという意義もある．

尚，全般的な性能を改善することについての反論として，No Free Lunch (NFL) 定理が有名である [75] が，MCTS の全般的な性能の改善は可能だと思われる．大雑把には，NFL 定理は，あらゆる問題に対しての平均的な性能を比べると，どのアルゴリズムも同じになるという定理である．つまり，全般的な性能の改善することは不可能ということである．しかしながら，NFL 定理は繰り返しのある問題（例えば多腕バンディット問題）には適用出来ないため [76]，MCTS についても同様であると予想される．また，Blackbox Optimization の分野で信じられているように，現実の問題に限って考えた場合，問題の分布は偏ったものになり，たとえ繰り返しの無い問題でも，NFL 定理での前提を満たさないと予想される．

1.1　本論文の構成

2 章には，本研究の背景について記す．本研究での探索対象のゲームや，モンテカルロ木探索（MCTS），それから MCTS に関連が深い多腕バンディット問題（MAB）とそのアルゴリズム，また，MCTS と組み合わせて使われる拡張手法について記す．

3 章では，主に MCTS の分析の観点から有利不利が偏っている局面での MCTS の改善について記す．より具体的には,「最善手を互いに選び続ければ引き分けになるが，次善手を選んだ場合の不利益の度合いが手番によって異なる」という状況を表すゲームを提案し，分析する．このゲームは UCT では性能が出にくく，また，MAB における他のアルゴリズム KL-UCB，Thompson Sampling [1,40,69] を MCTS に応用しても改善はほとんど無いという意味で MCTS とは相性が悪いということを示す．加えて，このゲームでは性能を改善する鍵が，終局時のスコアを利得に変換する過程にあることを示す．

4 章には，多腕バンディットの知見に基づき，最善手を見分けるアルゴリズムと最善をより多く選ぶことで評価を正確にするアルゴリズムとを適切に使い分けることで MCTS の性能の改善を目指した手法について記す．実験の結果，提案手法は近くに終局がある場合に効果的であった．

5 章では，終局が近くにあると判明した場合に，その情報をより積極的に用いる手法を提案する．4 章の内容から，既存手法は終局の情報を上手く活かせていないと予想さ

れるためである．提案手法は 4 章の手法と比べて，計算コストが少ない手法である．

6 章には，期待値の最大値の推定量の研究について記す．5 章の手法は効果的ではあったが，探索局面の近くに終局が無いと効果が無い．推定量を改善することにより一般的な状況での MCTS の改善が期待される．6 章にはその端緒として，多腕バンディット問題での推定量の改善について記す．

7 章には，6 章の推定量を実際に MCTS に応用する研究について記す．

8 章には本論文の結論を記す．

第2章　モンテカルロ木探索とその関連手法

本章では本研究で扱う，モンテカルロ木探索（MCTS）を中心に，MCTS の改善のために考案された手法や MCTS と関連が深い多腕バンディット問題，探索の対象としてのゲーム等についての研究を紹介する．

2.1　グラフ

本研究では，モンテカルロ木探索という木探索アルゴリズムの一種について扱う．木はグラフの一種であり，木探索アルゴリズムは木を調べるアルゴリズムである．本章では，本研究で用いる，グラフ理論の用語について，簡単に説明する．

まずは基本的な定義を紹介する．グラフ G（特に無向グラフ）とは形式的には，集合 V と集合 $E \subset [V]^2$ に対する，組 (V, E) のことである．ここでの $[V]^2$ は元を 2 つ含む V の部分集合族である．グラフ理論では，V と E の元をそれぞれ節点，辺と呼ぶ．E の元は V の元を 2 つ含んでいるので，V の 2 つの元を結んでいるという解釈が可能である．例えば，$a, b \in V$ かつ $\{a, b\} \in E$ なら，節点 a と節点 b の間に「道」が存在するという解釈である．グラフ理論では，これを一般化し，道や閉路を定義する．道 P とは $P = (V', E')$ となるグラフである．ここでの V', E' は $V' = \{v_0, v_1, \ldots, v_k\}$，$E' = \{\{v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_{k-1}, v_k\}\}$ であり，$v_0, v_1, \ldots v_k$ は全て異なる節点である．特に v_0, v_k を道 P の端点と呼ぶ．k を v_0 と v_k の距離と呼ぶ．閉路とは，端点 2 つを含む辺を道に加えたグラフである．より正確には，上記の V', E' に対し，$(V', E' \cup \{\{v_k, v_0\}\})$ となるグラフである．

次に，上記の定義を使って，本研究での使うグラフ理論の用語を紹介する．木はどの 2 節点にも道が存在するグラフで閉路が存在しないグラフである．木の内，根付き木は，木の節点の内の 1 つを根としたグラフである．根という特別な節点を導入することで，根との距離の大小から,「親」,「子」等の概念を導入することが出来る．$r \in V$ を根とした根付き木 $T = (V, E)$ を考える．ある節点 n, m $(\{n, m\} \in E)$ に対し，n と r の距離の方が m と r の距離よりも小さい時，節点 n を節点 m の親と呼び，逆に節点 m を節点 n の子と呼ぶ．ある節点 n, m が同じ親を持つ時，節点 n と m は兄弟と呼ぶ．ある節点 n, m に対し，n と r を端点とした道 (V', E') を考える．$V' \ni m$ の時，節点 m を n の先祖と呼び，節点 n を m の子孫と呼ぶ．子を持たない節点のことを葉と呼ぶ．

2.2 ゲームとゲーム木

本研究では，二人零和有限確定完全情報ゲームを主な対象として探索アルゴリズムを評価する．これは囲碁やチェス等のゲームが属するゲームの種類である．この用語の意味について簡単に説明する．二人とは，エージェント（プレイヤー）が二人（Max と Min とする）であることを意味する．零和とは，プレイヤーの利得の総和が 0 になることを意味し，特に，二人ゲームの場合，相手の利得を下げ（上げ）た分だけ，自分の利得が上がる（下がる）ことを意味する．有限とは，ゲームで可能な行動（手）の列が有限であることを意味する [1]．確定とは，確定的なゲームを意味する．非確定的なゲームでは，ある状態（局面）である手を選ぶとゲームのルールから次の局面の確率分布が定まり，その確率分布に従い次の局面が決まる．確定的なゲームはその特別な場合で，手を選ぶと次の局面が一意に決まるゲームである．完全情報とは，各プレイヤーがゲームの局面についての情報を全て観測可能であることを意味する．例えば多くのカードゲームに存在する手札は所有するプレイヤーだけが観測可能な情報である．このような情報が無いゲームが完全情報ゲームである．

ゲーム木とは，ゲームの局面等の状態を節点に，そこから可能な行動（手）を辺に対応させた木 [2] であり，特に，ゲームの初期局面を根とした根付き木である．非確定的なゲーム木では，状態を対応させた節点の他にチャンスノードと呼ばれる節点を追加することが多い．チャンスノードは確率的な状態遷移に対応した節点で，状態と行動の対に対応する．その対から遷移確率が正の状態へと辺を結ぶことで，非確定的な遷移をゲーム木で表現出来る．

本節では，MCTS やその他の探索アルゴリズムの評価に用いられてきた二人零和完全情報ゲームのゲーム木モデルや general video game playing 等について紹介する．尚，本研究では，確定的なゲームを主な対象としているため，行動・手・辺を選ぶことと次の状態・次の局面・子節点を同一視している．

2.2.1 ゲーム木モデル

二人零和有限確定完全情報ゲームのゲーム木についての研究として，古くは Pearl の研究 [52] がある．そこでは，確率 P で終端節点にランダムに勝敗を割り当てるというゲーム木のモデルを考え，P の大小と木の深さを深くした場合のゲームの理論値（根から最善を尽くした場合の勝敗）との関係について調べられている．このようなゲーム木モデルは一般的なゲームの性質の解析だけでなく，探索アルゴリズムの性能の評価にも用いられてきた．通常のゲームでは手に入らない値，例えば，最善手がどれか，現局面プレイヤーの優勢度合い等の情報が解析者には手に入るという利点があるためである．

[1] チェスは 3 回同じ局面になった場合，プレイヤーの指摘により，引き分けとなるため，実質的に有限である．また，囲碁でも，3 コウ等の同一局面になり得る局面が存在するが，ルールにより，プレイヤー合意のもとで引き分け（正式には無勝負）となるため，実質的には有限である．

[2] 多くのゲームでは，同一局面になるような異なる手順が存在するため，厳密には木でない場合も多いが，異なる手順で至る局面は異なるとすれば木になる．

ゲーム木モデルを使った他の研究として，木探索で深く読むほど手の評価が不確かになる病的な状況について調べた研究 [49,50] がある．そこでは，P-game（Pearl's game）モデルと N-game（incremental game）モデルが用いられた．この N-game モデルでは，各節点での各行動に対し -1 か 1 の値をそれぞれ確率 $1-p$，p で割り当てられる．根では Max プレイヤーの手番とする．ゲーム木の分枝数（合法手の数）はどの節点でも一定とし，規定深さの節点を終端節点とする．初期局面（根節点）から終端局面（終端節点）に至るまでの道の辺の値の総和（ゲームのスコア）が正だとその終端局面は Max プレイヤーの勝ち，負だと負け，0 だと引き分けとして勝敗を決める．N-game モデルではこのように勝敗を決めるため，兄弟間の勝ち負けは同じになりやすいという特徴がある．また，同じ N-game モデルでも，ランダムに辺に値を割り当てることで様々なゲーム木ができるという特長がある．このようなモデルを使うことで，個々のゲーム木に着目して性能を調べることや，個々のゲーム木での性能を平均することで，全体的な性能について評価することも出来る．Smith は N-game モデルに基づき生成したゲーム木等を使って，前向き枝刈りの性能を確かめた [66]．

Kocsis は N-game モデルを一般化したゲーム木モデルを MCTS の評価に利用した [42][3]．Kocsis のゲーム木モデルについて図 2.1 に例を示した．このゲーム木モデルでの Max（Min）の手には $[0,127]$（$[-127,0]$）を一様な確率でランダムに割り当てる．この辺の値はその辺を辿る（手を選ぶ）ことで Max プレイヤーがどれだけ勝ちに近づくかを表す値であり，根から各節点までの道の辺の値の総和はその節点でのプレイヤーの優勢度合いに相当する．Kocsis はこのモデルで MCTS とアルファベータ探索を行い，最善手を選べたかの割合を調べ，MCTS が優れていることを示した．この木の内部節点のゲームのスコアの理論値（互いに最善を尽くした場合のゲームのスコア）は木全体に渡ってミニマックス探索することで判明する．

しかしながら，木全体に渡って探索するのは時間がかかる．Furtak は Kocsis のモデルの変種として，より簡便にゲームのスコアの理論値が分かるゲーム木モデルを提案した [23]．この変種では，Max（Min）節点から子節点への辺の内，一つだけ値を 0 として他は負（正）の値をランダムに割り当てる．そのため，値を 0 にした辺が最善手に対応し，各節点のゲームのスコアの理論値は，根からその節点までの道の辺の値の和になり，根節点の理論値は引き分けとなる．また，一方のプレイヤーが常に最善をとる場合，もう一方のプレイヤーが一度でも次善手をとるとそのプレイヤーの負けになる．そのため，最善手を選ぶ必要性が高いゲーム木モデルといえる．それに対して，N-game モデルや Kocsis のゲーム木モデルに従ってゲーム木を作る場合，一方のプレイヤーがどんな手を選んでも勝ちになるゲーム木が出来うる．

Furtak のモデルは，前述のとおり，終端まで読むことなく，最善手が予め分かる．そのため，最善手を選べたかという絶対的な評価を探索アルゴリズムの評価に使えるという長所がある．一般にチェスや囲碁等の局面数が膨大なゲームでは，最善手は不明で，

[3]文献 [42] では "P-game" と言及されているが，ここで扱われているのはむしろ N-game モデルの変種である．

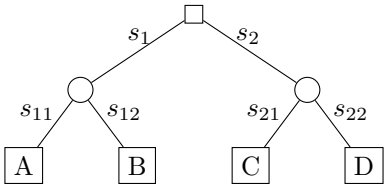


図 2.1: Kocsis のゲーム木モデルの例（分枝数と深さは 2）．四角と丸はそれぞれ Max，Min プレイヤーの節点を意味する．この例では，終端節点が A，B，C，D の 4 つ存在する．終端節点のゲームのスコアは根からその節点までの辺の値の和である．例えば，終端節点 A のゲームのスコアは $s_1 + s_{11}$ である．節点 A で Max の勝ちか負けかはこのゲームのスコアが正か負かによって決まる．Max プレイヤーの手に相当する辺の値は $s_1, s_2 \sim \mathcal{U}([0, 127])$ かつ Min プレイヤーの手に相当する辺の値は $s_{11}, s_{12}, s_{21}, s_{22} \sim \mathcal{U}([-127, 0])$ である．$\mathcal{U}(\cdot)$ は一様分布を意味する．N-game モデルは Kocsis のゲーム木モデルの内，$[0, 127]$（$[-127, 0]$）の範囲の値を一様な確率で割り当てる代わりに -1 か 1 の値をそれぞれ確率 $1 - p$，p で割り当てたモデルである．

現実的な時間では見つけることが出来ない．加えて，局面の優勢度合いについても，エキスパートがいる，もしくは，それに準じる評価ができる評価関数がないと判断できない．そのため，それらのゲームで探索アルゴリズムの性能を評価する場合，対戦実験を繰り返し行い，勝率をもとに評価をするのが一般的である．しかしながら，勝率は，対戦相手と比べた場合の相対的な評価である．つまり，対戦相手がそれなりに妥当な手を打つという期待がなければ，評価が難しい．その一方で，Furtak のモデルでは，最善手を選べたかどうかによる絶対的な評価が出来る．特にこのモデルで生成されるゲームの局面は根で最善手を選べないと理論値が引き分けから負けに変わるため，最善手を選べるかが重要な局面である．

他にも長所として，ゲーム木モデル全般に言えることであるが，分析の際に探索空間の大きさの変更が容易であることが挙げられる．囲碁では 9 路盤，13 路盤，19 路盤等，盤面の大きさを変えることである程度可能だが，多くのゲーム，例えばチェスでは難しい．

本研究では，Furtak のゲームを特に incremental random game（そのゲーム木を incremental random tree）と呼び，探索の主な対象とし，探索アルゴリズムの評価に用いた．

ゲーム木モデルを使った研究が他にも存在する．Ramanujan は Furtak のモデルで節点の種類として，トラップを定義し，解析した．トラップである節点は，その節点から数手で終局する一方で兄弟節点ではまだ終局しない，かつ，ゲームの理論値が親手番の負け（つまりその節点を選ぶと数手で負けてしまう）という節点である．トラップの図を図 2.2 に示した．Ramanujan はゲーム木モデルの優勢度合いにノイズを加えた値が評価関数として手に入る場合のミニマックス探索と MCTS を比較し，木に多くのトラップがある場合では MCTS の性能はミニマックス探索と比べて下がることを示し

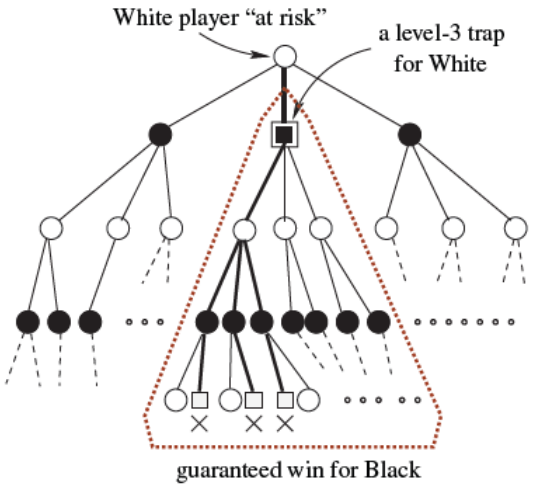


図 2.2: 文献 [58] からのトラップの図（figure 1）の抜粋．この例では，白と黒のプレイヤーが存在し，白が根で手番となっている．点線で囲まれた節点の内，四角の節点は終端節点であり，黒の勝ちの節点である．根で白が 3 つ手の内で真ん中の手を選び，黒が最善を尽くした場合，その後の白の手によらず，白の負けになる．真ん中の手がトラップである．

た [58, 59]．

2.2.2 Finnsonn の仮想ゲーム

Finnsson の研究 [22] もまた，ゲーム木モデルを使った研究の一つと位置づけられる．Finnsson の研究では二人零和有限確定完全情報ゲームに属するいくつかのゲームを提案し，そのゲームの種類と探索空間の大きさと MCTS の性能の関係を示した．Finnsson のゲームの一つは 2.2.1 節でのゲーム木モデルで，ランダムに割り当てる辺の値を予め決めたルールで決定的に割り当てる場合に相当する．実験を通じて，MCTS の性能にとって，古典的なミニマックス探索で重要と知られている探索空間の大きさよりも，次善手の値の割り当て方法の方が重要であると示した．

2.2.3 General Video Game Playing

General Video Game Playing AI（GVG-AI）は，初見の様々なゲームをプレイし，上手くプレイするエージェントを作るということを目標としたドメインである．この研究には，汎用人工知能に向けた知見を得るという意義がある．

GVG-AI コンペティションでは，複数のゲームで，複数エージェントが競い合う．ゲームには勝利（敗北）条件と，スコアとタイムリミットがある．タイムリミットが来たらゲームは終了し，勝利（敗北）条件を満たしているかが判定され，勝利か敗北かのどちらかに決まる．タイムリミットが来ない場合でも，勝利（敗北）条件を満たすとゲーム

が終了し，その結果が勝利（敗北）となる．各エージェントがプレイしたゲームの結果は，勝利したか否か，ゲーム上のスコア，かかった時間で順で優劣をつける，そして，各ゲーム毎に順位に応じてポイントがエージェントに与えられ，得たポイントの総和が最も高いエージェントが優勝となる．

コンペティションの内，single player planning track では，一人ゲームをそれぞれのエージェントがプレイする．エージェントは，ゲーム中の行動を決める際に forward model と呼ばれる，状態と行動に対し，次の状態（確率的な遷移をする場合は，確率的にその遷移先の一つを）返す関数を利用出来る．他にも，NPC，壁等のオブジェクトの情報（位置，オブジェクトの種類の ID 等）を利用出来る．エージェントはこれらをもとに良い行動を見つけることが求めれられる．

2017 年現在では，single player planning track の他にも，2-player planning track（協力するゲームも含む二人ゲーム部門），level generation track（ゲームの生成部門），single player learning track（forward model 無しの部門）がある．

2.2.4 マルコフ決定過程（MDP）

本研究では扱わない対象の内，代表的な探索対象としてはマルコフ決定過程が挙げられる．マルコフ決定過程（MDP）は一人ゲームで，非確定性完全情報のゲームに対応するドメインに相当し，強化学習での主要な対象である．MDP は 4 つ組 (S, A, T, R) で表せる．S は取りうる状態の集合，A は可能な行動の集合である．$T : S \times A \times S \to [0, 1]$ は状態遷移確率で，状態 s と行動 a と状態 s' に対し，状態 s で行動 a をとった時に状態 s' に至る確率を返す．$R : S \times A \times S \to [0, 1]$ は利得関数で，状態 s で行動 a をとり状態 s' に至った時の利得，あるいは利得の期待値を返す．利得は，$[0, 1]$ に限らない実数をとる場合もある．

MDP には終端状態が存在するものとしないものがあり，特に前者についてより詳しく記す．終端状態はゲームでの終局に相当するものであり，そこに至ると MDP は終了し，それ以上利得は得られないという状態である．終端状態が存在する MDP での状態行動価値関数 $Q : S \times A \to \mathbb{R}$ を

$$Q(s,a) := \begin{cases} 0 & (s \text{ は終端節点}), \\ \sum_{s' \in S} T(s,a,s')(R(s,a,s') + \max_{a' \in A} Q(s',a')) & (s \text{ は終端節点でない}) \end{cases}$$

と定義する．状態行動価値関数を使って，この MDP における状態 s での最善の行動を式で表すと $\arg\max_{a \in A} Q(s,a)$ である．

Kocsis は MDP に属する sailing domain で，既存手法に対する UCT の状態空間の大きさに対する性能の良さを示した [42]．

2.3 強化学習と探索

強化学習と探索には密接な関わりがある．あるドメイン（例えば MDP(S, A, T, R)）を考えて，強化学習する際，その目的は現在の状態から可能な行動をとり，それに対する報酬（利得）を得ることを繰り返し，最も未来の期待利得の総和が高いという意味で最善の方策（状態と行動に対しその確率を返す関数 $\pi : S \times A \to [0, 1]$）を学習することである．その一方で探索の目的は，探索対象の局面の最善手を見つけることである．最善手は対象の局面に対し最善の方策が返す行動（手）である．

強化学習のアルゴリズムはドメインのモデルの存在の有無の観点で分類すると model-free，model-based の 2 種類の学習が存在する．本研究の主題であるモンテカルロ木探索は model-based の学習アルゴリズムの一種として位置づけられる．model-free ではエージェントはドメインのモデル（遷移確率と利得関数）を持たない仮定のもとで，最善の行動を学習する [67][4]．model-free のアルゴリズムはモデルが手に入りにくい状況でも使えるという長所がある．一方，model-based ではモデルが手に入るという仮定のもとでそのモデルを利用し，先読みして方策を学習する．こちらは，モデルを使う分だけ観測したデータを効率的に扱えるという長所がある．尚，モンテカルロ木探索の代表的なアルゴリズム UCT は遷移確率が不明でも generative model（2.7 に記す）があれば実行可能である．

探索と学習について別の位置づけも可能である．文献 [64, 67] での Dyna アーキテクチャに従って位置づけると，探索は学習したモデルをもとに，実際の行動を決める方法で，学習は方策の更新方法ではなく，その実際の行動とその結果に基づきモデルを更新する方法である．

2.4 バンディット問題とそのアルゴリズム

本節では，MCTS と密接な関わりのある多腕バンディット問題とそのアルゴリズムについて紹介する．また， 6 章で議論する，期待値の最大値の推定量について紹介する．

2.4.1 多腕バンディット問題（MAB）

多腕バンディット問題（MAB） [44] は強化学習の最も単純なドメインの一つである．MAB を 2.2.4 節のマルコフ決定過程（MDP）の文脈で解釈すると，根から一つの行動を選ぶと終端状態に至るような MDP であり，MAB でのアームの選択は MDP での行動の選択に対応する．典型的な MAB の設定では，アームは K 本あり，各アーム $1 \leq i \leq K$（i は整数）には利得の分布が割り当てられる．エージェント（プレイヤー）は繰り返しアームを引く．アームを引くとそのアームに割り当てられた分布に従って独立に利得が得られる．利得は $[0, 1]$ の間の値をとる．しかしながら，エージェントに

[4] モデルを学習しないことが model-free という定義も存在する（文献 [38]p251）．この定義だと例えば，モデルを推定するアルゴリズムは model-free ではない．

とって利得の分布は不明で，そのため，アーム i の期待値 μ_i や分散 σ_i^2 も不明である．エージェントはアームを引くことで，利得一つ一つから分布に関する情報を得る．この分野の主な研究課題は，アームを引く数を決められたもとで，累積的な利得を最大化すること [3] や，期待値最大のアームを見つけ出すこと [2] 等がある．

詳しい内容に移る前に記号について定義しておく．$X_{i,t}$ をアーム i が t 回目に引かれた時の利得とする．N_i をアーム i が今までに引かれた回数とする．n をアームを引かれた回数の総和 ($n := \sum_{i=1}^{K} N_i$) とし，$\bar{X}_{i,N_i}$ をアーム i の平均利得 ($\bar{X}_{i,N_i} := \sum_{t=1}^{N_i} X_{i,t}/N_i$) とする．利得の期待値が最大のアームを $* := \arg\max_{1\leq i\leq K} \mu_i$ とし，利得の期待値の最大値とアーム i の利得の期待値との差を $\Delta_i := \mu_* - \mu_i$ と表記する．

多腕バンディット問題は単純なため，理論的な解析が進んでいる．この問題で理論的な性能を議論するためにリグレットという量がよく用いられる．リグレットには 2 種類あり，累積的な利得を最大化する観点では，累積リグレット，期待値最大のアームを判別する観点では単純リグレットが使われる．累積リグレットは利得の期待値が最大（最善）のアームを引き続けた場合と比べて，利得の観点でどれだけ損をしたかを表す量である．具体的には，

$$\mu_* n - \sum_{1\leq i\leq K} \mu_i \mathbb{E}[N_i(n)]$$

この累積リグレットが小さくなるアルゴリズムが望ましい．

単純リグレットは最善のアームとアルゴリズムが最善として選んだアーム（J_n とする）との期待値の差

$$\sum_{1\leq i\leq K} \mathrm{P}(J_n = i)\Delta_i$$

である．ここでの $\mathrm{P}(J_n = i)$ はアーム i を J_n として選ぶ確率である．

以下，累積的な利得を最大化するアルゴリズムを中心に紹介する．累積的な利得を最大化するために必要なことは，利得を得るために期待値が高そうなアームを引くことと，より良いアームを探すために情報の少ないものを引くことのバランスを上手くとることである．以下で紹介する UCB1 と KL-UCB は利得の推定値とその不確かさに，Thompson Sampling は各アームについての期待値の主観確率分布にそれぞれ基づいて，バランスをとっている．本節では，他にも強化学習で広く使われている epsilon-greedy, softmax についても紹介する．

2.4.2 UCB1

UCB1 [3] はその単純さのため，広く用いられている手法であり，利得の推定値とその不確かさを組み合わせた評価基準である UCB 値を定義し，それが最大となるアームを選ぶ．UCB1 での UCB 値は具体的には

$$\bar{X}_{j,N_j(t)} + \sqrt{\frac{2\ln t}{N_j(t)}} \tag{2.1}$$

である．UCB 値は平均利得の項（第一項）により，有望さを，それに加えられる補正項（第二項）により，相対的な不確かさを評価している．UCB1 は，まず始め全てのアームを一度引き，その後は上記の UCB 値が最大のアームを引く．

UCB1 アルゴリズムでは，アームを引く回数の期待値について

$$\mathbb{E}[N_i(n)] \leq \frac{8\ln n}{\Delta_i^2} + 1 + \frac{\pi^2}{3} \tag{2.2}$$

という不等式が成り立つと示されている [3]．

2.4.3 KL-UCB

KL-UCB も UCB1 と同様に，UCB 値に基づきアームを選ぶ方法であるが，UCB1 よりも小さな累積リグレットの上界が示されており，実験においても UCB1 よりも優れた性能を示した手法である [13]．KL-UCB では，ベルヌーイ分布間の KL 距離を元にして UCB 値を定める．具体的には，２つベルヌーイ分布 p, q 間の KL 距離

$$d(p,q) := p\log\frac{p}{q} + (1-p)\log\frac{1-p}{1-q}$$

を用いて

$$\arg\max_{j\in\mathbb{K}} \max\left(q_j \in [0,1] : d(\bar{X}_{j,N_j(t)}, q_j) \leq \frac{\ln t + c\ln(\ln t)}{N_j(t)}\right) \tag{2.3}$$

のアームを選ぶ．ここでの c は定数である．この式は有望さと KL 距離を使った相対的な不確かさを評価していると言える．例えば，あまり引かれていないアームでは KL 距離の離れ具合の上界 $\frac{\ln t + c\ln(\ln t)}{N_j(t)}$ は上昇するので，より高い UCB 値となる．UCB1 と異なる点は，単純に訪問数の大小ではなく，ベルヌーイ利得間の KL 距離を用いているため，例えば，平均利得が低いほど，相対的不確かさの変化に対して，より鋭敏に UCB 値が変動する等の特徴がある．KL-UCB では，アームを引く回数の期待値について

$$\lim_{n\to\infty} \sup \mathbb{E}[N_i(n)] \leq \frac{\ln n}{d(\mu_i, \mu^*)}$$

という上界が示されている [13]．次善のアーム $i \neq *$ について考えると，この上界は UCB1 のものよりも低い．

2.4.4 Thompson Sampling

Thompson sampling [1, 69] は得られた利得をもとに，ベイズ推定をしながら，確率的にアームを選ぶアルゴリズムである．各アームの利得はベルヌーイ分布に従うと仮定し，各アームの期待値は，ベルヌーイ分布の共役事前分布である，ベータ分布に従うとする．各アーム j について，利得が $1, 0$ であった回数をそれぞれ α_j, β_j とする．つまり平均利得を用いると

$$\begin{aligned} \alpha_i &= \bar{X}_{i,N_i} N_i(t), \\ \beta_i &= (1 - \bar{X}_{i,N_i}) N_i(t) \end{aligned} \tag{2.4}$$

である．このとき各アーム j に対し，その期待値の主観確率分布はベータ分布 $B(\alpha_j + 1, \beta_j + 1)$ である．Thompson sampling では各アーム j に対し，ベータ分布 $B(\alpha_j + 1, \beta_j + 1)$ からサンプルした値が最も高いアームを引く．尚，ベータ分布は α_j（β_j）が大きいほど利得が 1 (0) の方に歪んだ分布となり，また，α_j，β_j ともに大きいと尖度が高い分布となる．累積リグレットの上界は KL-UCB と同等であると証明され，また，KL-UCB と同様の実験で Regret が KL-UCB や UCB1 より低いことが示されている [40].

2.4.5 Epsilon-greedy

epsilon-greedy 法 [67] は確率的にアームを選ぶ手法で，強化学習で広く使われている．具体的には，確率 $1-\varepsilon$ で $\arg\max_i \bar{X}_{i,N_i}$（推定最善）を確率 ε で一様にランダムにアームを選ぶ.

2.4.6 Softmax

softmax 法 [67] もまた，確率的にアームを選ぶ手法で強化学習で広く使われている．softmax 法では，各アーム i を確率 $\frac{\exp(\bar{X}_{i,N_i}/T)}{\sum_j \exp(\bar{X}_{j,N_j}/T)}$ で選ぶ．ここでの T は温度パラメータであり，高い（低い）温度では，高い平均利得のアームがより少なく（多く）選ばれる

2.5 期待値の最大値の推定量

本節と 6 章では， 2.4.1 節で紹介した MAB の設定のもと，アームを引いて，得られた利得の集合（サンプル）から $\max_i \mu_i$ を推定する推定量について考える．そして推定量を 7 章で MCTS に応用する．推定量の性能は推定値 [5]$\hat{\mu}$ の平均自乗誤差に基づいて評価することが自然である．平均自乗誤差はバイアスの自乗とバリアンス（分散）に分解出来るので，バイアスと分散の観点からも評価する．バイアスは $\mathrm{Bias}(\hat{\mu}) = \mathrm{E}[\hat{\mu} - \mu_*]$，分散は $\mathrm{Var}(\hat{\mu}) = \mathrm{E}[(\hat{\mu} - \mathrm{E}[\hat{\mu}])^2]$ である．バイアスは正に大きいほど過大評価，負に大きいほど過小評価することを意味し，分散が大きいほど評価が安定しないことを意味する．簡単のため，最善のアームはただひとつだけ存在すると仮定し議論するが，この議論は一般化可能である．理論的な解析では，N_i は確率変数でないと仮定して行う．この仮定は先行研究の類似の解析の際の仮定に習っている [18, 72]．しかしながら，エージェントが UCB1，epsilon-greedy，softmax 等の実際的なサンプリングアルゴリズムを使う場合，平均利得 $\bar{X}_{i,N_i}$ に基づいてアームを引くため，N_i は確率変数である．そのため， 6.3 節では，実際に，理論での仮定が満たされない状況での性能について計測している．

[5]推定値は推定した具体的な値で，推定量は推定値を出力する関数である．

6，7 章で扱う期待値の最大値の推定量に関連して，先行研究の 5 つの推定量，AVerage Estimator (AVE), Maximum Estimator (ME), Double Estimator (DE), Weighted Estimator (WE), 及び MixMax estimator (MM) について紹介する．

2.5.1 Average Estimator (AVE)

AVE は観測した全利得の平均をとることで μ_* を見積もる．

$$\hat{\mu}_{AVE} := \sum_{i=1}^{K} \frac{N_i}{n} \bar{X}_{i,N_i} = \frac{1}{n} \sum_{i=1}^{K} \sum_{t=1}^{N_i} X_{i,t}.$$

UCT では，行動を UCB1 [3] に従って選ぶが，その行動の価値 $\bar{X}_{i,N_i}$ は子の推定値をもとに AVE で推定されている．

AVE は各アームをどれだけ引くかに強く依存する手法である．その結果，全てのアームの期待値が等しい時を除き，アームを引く回数が増えても，アームの引き方次第では推定値が正しい値に収束しないという欠点がある．例えば，各アーム i を引いた回数 N_i が $\Omega(n)$ となる時，n が大きくても AVE のバイアスは 0 に収束しない．

全てのアームに対して，最善のアームを引く回数の割合が 1 に収束するような場合に AVE による推定は収束性の面で妥当である．例えば，UCB1 で引くアームを決める場合である．その場合，次善 [6] のアームを引く回数の期待値 $O(\log n/n)$ で抑えられる．このことは，n が増えるにつれて，平均的には最善のアームを引く割合が 1 に収束することを意味する．しかしながら，収束する前については，少数のアームが高い期待値であり，その他のアームの期待値低い場合に，AVE による推定は過度に低い値となる．また，MCTS の文脈では，どこからシミュレーションするかの選択と行動の価値の推定は分けた方が良いという報告もある [21].

$\Delta_i := \mu_* - \mu_i$ とすると AVE のバイアスと分散は以下の性質を持つ．

$$\mathrm{Bias}(\hat{\mu}_{AVE}) = -\sum_{i=1}^{K} \frac{N_i}{n} \Delta_i,$$

$$\mathrm{Var}(\hat{\mu}_{AVE}) = \sum_{i=1}^{K} \frac{N_i}{n^2} \sigma_i^2.$$

2.5.2 Maximum Estimator (ME)

ME は μ_* を平均値の最大値によって見積もる．$m := \arg\max_j \bar{X}_{j,N_j}$ とすると

$$\hat{\mu}_{ME} := \bar{X}_{m,N_m}$$

である．ME のバイアスは正であり，上界があることが知られている [4].

$$0 \leq \mathrm{Bias}(\hat{\mu}_{ME}) \leq \sqrt{\frac{K-1}{K} \sum_{i=1}^{K} \frac{\sigma_i^2}{N_i}}.$$

[6] 本論文では，次善とは最善でないという意味で使う．

特に，全てのアームの期待値が等しい時，バイアスは大きくなる．分散の上界についても，先行研究により

$$\mathrm{Var}(\hat{\mu}_{ME}) \leq \sum_{i=1}^{K} \frac{\sigma_i^2}{N_i}$$

と示されている [72].

2.5.3 Double Estimator (DE)

DE [71] は強化学習での Q 学習の改善のために提案された推定量である．Q 学習では，各推定値は推定量 ME で更新されるが，ME は過度に大きく推定するために上手く行かないドメインがある．そこで提案されたのが DE であり，DE ではその問題は生じない．

DE は利得の集合を重複のない 2 つのサブセットに分ける．一つのサブセットを使い，各アーム i の平均利得 $\bar{X}_{i,N_i}^{(1)}$ を計算して，期待値 μ_i を推定をする．もう片方のサブセットを使い，$m^{(2)} = \arg\max_j \bar{X}_{j,N_j}^{(2)}$ を計算して，$*$ を推定する．これをサブセットを入れ替えてもう一度行い $\bar{X}_{i,N_i}^{(2)}$ と $m^{(1)}$ を計算する．DE はこれらの推定を統合し，

$$\hat{\mu}_{DE} := \frac{1}{2}(\bar{X}_{m^{(1)},N_{m^{(1)}}}^{(2)} + \bar{X}_{m^{(2)},N_{m^{(2)}}}^{(1)})$$

により最大値を推定する．$N_i^{(1)}, N_i^{(2)}$ をそれぞれ 1 ，2 つ目のサブセットでのアーム i を引いた回数とすると，バイアスと分散については以下の通り，負のバイアスを持ち，分散に上限があることが知られている [72].

$$-\frac{1}{2}\left(\sqrt{\sum_{i=1}^{K}\frac{\sigma_i^2}{N_i^{(1)}}} + \sqrt{\sum_{i=1}^{K}\frac{\sigma_i^2}{N_i^{(2)}}}\right) < \mathrm{Bias}(\hat{\mu}_{DE}) \leq 0,$$

$$\mathrm{Var}(\hat{\mu}_{DE}) \leq \frac{1}{4}\sum_{i=1}^{K}\left(\frac{\sigma_i^2}{N_i^{(1)}} + \frac{\sigma_i^2}{N_i^{(2)}}\right).$$

2.5.4 Weighted Estimator (WE)

WE [18] は μ_* を各アームの平均利得の重み付き和で推定する．重みは各アーム i が最善であるかどうかの主観確率で決まる．D を利得の集合とすると，WE の推定値は

$$\hat{\mu}_{WE} := \sum_{i=1}^{K} \mathrm{P}(i = *|D)\bar{X}_{i,N_i}$$

である．バイアスについて，WE は ME（正）と DE（負）の間であるという望ましい性質を持つ [7].

$$\mathrm{Bias}(\hat{\mu}_{DE}) \leq \mathrm{Bias}(\hat{\mu}_{WE}) \leq \mathrm{Bias}(\hat{\mu}_{ME})$$

[7] これはバイアスについて，ME と DE の内 0 から遠い方と比べて，WE の方が 0 に近いことを意味し，望ましい性質である．しかし，ME と DE の内 0 に近い方と比べて，WE が 0 に近いかは場合による．

また，分散についても以下が成り立つ．

$$\mathrm{Var}(\hat{\mu}_{WE}) \leq \sum_{i=1}^{K} \frac{\sigma_i^2}{N_i}$$

しかしながら WE は最大値についての分布を複数の分布から計算するため，計算コストが高い．そのため，計算コストが低い推定量を 6.2 節で提案する．

2.5.5 Mixmax Estimator (MM)

MM [36] は AVE と ME の推定値の重み付き和をとるという推定量である．

$$\hat{\mu}_{MM}^{\lambda} := (1-\lambda)\hat{\mu}_{AVE} + \lambda\hat{\mu}_{ME}$$

ここでの $\lambda(0 < \lambda < 1)$ は重みである．MM は UCT で Mario AI を動かす際に，過度に慎重になるという傾向を改善するために提案された手法である．UCT で用いられている推定量 AVE でゲームの局面を評価する場合， 2.5.1 節に記したように，その局面から悪い行動をとった場合の評価により，その局面の評価が過度に低く見積もられる．その結果として，上手く良い行動を選び続ければ，敵にぶつからずに進めるが，敵にぶつかってしまう行動が多い局面で，本来は敵を避けてすぐに進むべきであるのに，立ち止まってしまうということが観測されていた．その悪さを軽減するために提案された推定量である．MM のバイアスもまた，AVE（負）と ME（正）の間にあるという望ましい性質がある．しかしながら，MM には AVE と ME の悪さが λ の大きさに応じて存在する．つまり MM にのバイアスについて，AVE と同様にサンプルのとり方次第で最適な値に収束せず，λ が 1 に近いほど，バイアスは正に外れる．

2.6 モンテカルロ木探索（MCTS）

木探索とは外界のモデルを使って，現在の状態からあり得る状態を計算していき，最善の行動を見つける方法であり，人工知能における主題の一つである．MCTS [10] は木探索の一種で，その中でも最良優先探索の一つである木探索アルゴリズムでは，探索木と呼ばれる，ゲーム木の部分木を作り，探索のための各種の値を保持する．最良優先探索は何らかの評価基準に従って，葉節点の良さを定義し，最も良い葉節点を優先して展開する方法で，MCTS の他に A*探索やダイクストラ法が相当する．MCTS は，良さの評価をランダムシミュレーション（ランダムに終局まで手を選ぶ）で行い，その結果が良いところを優先して展開する．最良優先探索アルゴリズムは図 2.3 の 4 つのステップを繰り返す．

1. 葉の選択：探索木の葉の内，最も優先度の高い葉を選ぶ．
2. 展開：必要であれば，葉が展開される．展開では，その葉の子節点を新たに探索木の葉とし，もとの葉は探索木の内部節点とする．展開したら，新たに探索木に加えた葉の内から葉を選ぶ（例えば，MCTS では新たに加えた葉を 1 つ選び，A* 探索では新たに加えた葉を全て選ぶ）．
3. 評価：選んだ葉を評価する（例えば，MCTS ではシミュレーションして評価し，A*探索ではヒューリスティック関数を使って評価する）
4. 更新：評価した値を木全体で共有する．

図 2.3: 最良優先探索の概要

MCTS の研究において，シミュレーション，プレイアウト，ロールアウトは同じ意味で使われることもあるが，本研究では，探索木の葉から終端節点まで，ランダムに手を選ぶことをシミュレーションと呼び，選択のステップで探索木の根から葉まで選ぶこととそこからシミュレーションすることを合わせてプレイアウトと呼ぶことにする（図 2.4）．尚，一般にターン制ゲームのプログラムでは，相手のターンでの思考時間も無駄にしないように探索するということが行われている．その場合，根では相手の手番である．簡単のため，本研究では根で手番のプレイヤーの探索方法に限って述べている．

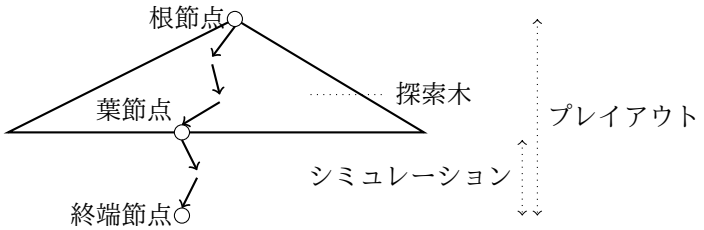


図 2.4: プレイアウトとシミュレーション

2.6.1 UCT

UCT はモンテカルロ木探索（MCTS）の代表的なアルゴリズムである．UCT では，ゲームの手を多腕バンディット問題（MAB）でのアームとしてモデル化 [3] し，手 i の利得はその手の利得分布に従うとみなしている．以下，UCT のアルゴリズムについて図 2.3 に従って説明する．

図 2.3 の 1 つ目のステップである葉の選択は，各節点で MAB のアルゴリズム UCB1 [3] に従い UCB 値を計算し，その値が最大の手を選ぶことを繰り返して行う．プレイアウト t 回行った際，今までに節点 j を選んだ回数（訪問数と呼ぶ）を $N_j(t)$ とする．プレイアウト回数を明示的に示さなくても，説明に不都合がない場合，簡単のため N_j と略

記する．$\bar{X}_{j,N_j}$ を節点 j の利得（j の親節点の手番から見た利得）の推定値（UCT では平均利得），とすると UCB 値

$$\bar{X}_{j,N_j} + 2C_p\sqrt{\frac{\ln n}{N_j}} \tag{2.5}$$

が最大の子節点を選ぶ．ここでの C_p は十分大きな n に対して，任意の $\delta > 0$ で

$$\mathrm{P}(n\bar{X}_{j,n} \geq E[n\bar{X}_{j,N_j}] + C_p\sqrt{n\ln(1/\delta)}) \leq \delta, \tag{2.6}$$

$$\mathrm{P}(n\bar{X}_{j,n} \leq E[n\bar{X}_{j,N_j}] - C_p\sqrt{n\ln(1/\delta)}) \leq \delta \tag{2.7}$$

を満たすパラメータである．$C_p = \frac{1}{\sqrt{2}}$ の時，式 (2.5) の UCB 値は式 (2.1) の UCB1 での UCB 値と同じになる（利得を $[0,1]$ でなく，$[-1,1]$ とすることも出来る．その場合はスケールが 2 倍になるので，$C_p = \sqrt{2}$ の時に UCB1 と同じである）．C_p を小さくすると，推定値がその期待値から離れることはまれという想定のもとで探索することになる．これは，利得の推定値が良い子を優先的に選ぶという UCB1 の性質の優先度合いを高めることに繋がり，結果として，利得が高いところでより重点的にシミュレーションを行うことになる．各手を MAB のアームとみなして考えると，MCTS では，その先でどこからシミュレーションするかはその時々で異なるため，アームの利得分布が非定常である MAB に相当する．そのため，MCTS では $\bar{X}_{j,N_j}$ の期待値 $E[\bar{X}_{j,N_j}]$ まわりの挙動について，MAB より性質の悪い問題を扱っているといえる．

図 2.3 の 2 つ目の展開のステップでは，葉を訪問したら展開する方法 [6,8,17,19] と展開閾値を設け，訪問数が展開閾値になったら展開する方法がある．尚，前者で展開する場合と後者で展開閾値を 2 とした場合とでは多少異なる挙動となる．前者では，展開されるのが二度目の訪問時の場合（新たに葉になった時に選ばれた場合）と一度目の訪問時の場合（新たに葉になった時には選ばれなかった場合）があり，後者では二度目の訪問時の時のみ展開するためである．一般に展開閾値を高く設定することで，探索木の成長を遅らせて，メモリを節約することが出来る一方で，展開していれば保持していたはずの情報を活用出来ないというトレードオフが存在する．本論文では，特に断らない限り，前者を採用した． また，探索の始め，探索木は根とその子節点だけを含んだ状態とした．

図 2.3 の 3 つ目の評価のステップでは，シミュレーションを行う．シミュレーションではゲーム木の終端節点まで一様な確率で手を選び，到達した終端節点の勝ち負けに応じて利得を得る．利得は勝ち，引き分け，負けでそれぞれ $1, 0.5, 0$ とした（7 章ではプログラムの都合上，利得はそれぞれ $1, 0, -1$ としている．この場合はスケール 2 倍になるため，C_p の値も 2 倍するのが自然で，本研究でもそうした．）利得は離散値 $1, 0.5, 0$ をとるとしているが，代わりに $[0,1]$ の任意の値を使うことも可能である．オセロや囲碁等のいくつかのゲームではゲームのスコア（オセロでの黒・白の枚数，囲碁での目数）を勝ち負け以外に利用することも出来る．歴史的には，多くの囲碁プログラムが線形にスコアを利得に変換して，勝ちや負けの中でもより良いものとそうでないものを

区別しようとしてきた．しかしながら，対局での強さは離散値（勝ちと負けがはっきりと違う値）を使うことで，大いに改善した．最近では，多くの囲碁プログラム（例えばfuego [19]）では離散値を使っている．本研究では，これに習い，ゲームのスコアが分かる場合でも，勝ち負けに応じて利得を与えた．また，シミュレーション中の方策（手の選び方）について，一様な確率で手を選ぶのではなく，ドメイン固有の知識を使い，確率を偏らせた方が性能が上がるという報告もあるが，本研究ではドメインの知識は使わず一様な確率で手を選ぶとした．

図 2.3 の 4 つ目の更新のステップでは，各節点の平均利得と訪問数を更新する．UCTでは各節点で，その節点自身と子孫から行ったシミュレーションの平均利得と訪問数を管理する．尚，平均利得と訪問数の代わりに勝ち数，負け数，引き分け数を管理しても等価である．更新は節点 j が選ばれた場合，シミュレーションでの利得（j の親で手番のプレイヤーにとって）を r とすると

$$\bar{X}_{j,N_j+1} \leftarrow \frac{\bar{X}_{j,N_j} N_j + r}{N_j + 1} \tag{2.8}$$

$$N_j \leftarrow N_j + 1 \tag{2.9}$$

と行う．節点自身とその子孫から行ったシミュレーションの平均利得は，子の値を使っても更新が可能である．UCT における，内部節点 j の値の更新は，C_j を節点 j の子，r_j と n_j をそれぞれ，節点 j から行われた（節点 j が葉であった時に行われた）シミュレーションの平均利得とその回数とする．節点 j の子 C_j の訪問数と平均値は更新済みであること，及び，手番が変わることによる利得の反転（プレイヤーにとっての利得が r の時，その相手にとっての利得は $1-r$）に注意すると，更新式は

$$N_j \leftarrow \sum_{k \in C_j} N_k + n_j \tag{2.10}$$

$$\bar{X}_{j,N_j} \leftarrow 1 - \frac{\sum_k \bar{X}_{k,N_k} N_k + r_j n_j}{N_j} \tag{2.11}$$

と表すこともできる．つまり（r_j と n_j を除いて）その節点の子の推定値を子の訪問数で重み付けした和（ 2.5 節で紹介した AVE）に相当する．

UCT はこれらの 4 つのステップを繰り返し，各手の評価の精度を高めるアルゴリズムである [42]．UCT では，根の最善手として，最大訪問数の手，平均利得が最大の手，平均利得の lower confidence bound が最大の手を選ぶ等の方法があるが，本研究では，最大訪問数の手を最善とした（5 章では訪問数の修正する関係から平均利得最大の手を最善とした）．UCT はこのように徐々に精度を高めていくアルゴリズムであるから，いつ探索を打ち切っても，最善手としてそれなりに良い手を返すことが出来る，anytime アルゴリズムである．

UCT は上記の通り，UCB1 を葉の選択で用いているため，UCB1 の性質により，利得が高いところでより重点的にシミュレーションを行う．そのため，各節点の利得を平均で推定しても，シミュレーション数に応じて正確になると期待される．実際に十分に

探索木が成長した状態で十分にシミュレーションすれば利得の推定値が利得の理論値（互いに最善を尽くした場合の利得）に収束するということが示されている [42].

しかしながら，多くの場合，時間等の計算資源に制限があるという前提のもとで最善手を判別する必要がある．最善がどの手かはその手の利得の推定値をもとにして判断する必要があるため，利得の理論値を正確に素早く推定するのが肝要である．しかし，どのような値を利得の推定値として用いるべきかは，まだ十分に理解されているとは言えず，議論の余地がある．これについては 7 章で議論する．

尚，MCTS の研究の中には，単純化のために探索木は固定して，解析を行って有効性を示した手法もあるが [39,68,70]，この設定での実験結果だけでは，性能の評価としては不完全であると考える．なぜなら，典型的なゲームでは，局面の数は膨大な一方で，探索資源は限られており，全て展開済みの状態から探索することは難しいため，探索の過程での葉の展開を組み込むことが重要であると考えるためである．

囲碁等，多くのゲームには，指し手の順序が異なっても同じ局面に至る場合がある．この場合，古典的なミニマックス探索では，同じ局面の評価を繰り返すことを避けるため，transposition table と呼ばれる表で，局面の評価を保持するという工夫をするのが一般的である．しかしながら，MCTS の場合，指し手の順序が異なる局面は，同じ局面であっても異なる局面として扱うことが一般的である．但し，指し手の順序が異なっても同じ局面に至るという状況の扱いを工夫し，効果を上げた手法も考案されている [15].

2.6.2 MCTS-Solver

MCTS-Solver は探索木中の終局（終端節点）での勝ち・負けの情報を有効に活用して，無駄な探索を減らす方法で，もとは Lines of Action というゲームのために考案された手法である [74]．プレイアウト結果の更新は通常の UCT と同様に行うが，加えてゲームの理論値が勝ち・負け・引き分け・不明かを更新する．具体的には，もし，Max (Min) 手番の節点 i のある子節点が，終端節点で，Max (Min) プレイヤーの勝ちとなるなら，節点 i は実質上，終端節点で Max (Min) プレイヤーの勝ちである．また，Max (Min) 手番の節点 i の全ての子節点で，Max (Min) プレイヤーの負けであるなら，その節点は実質上，終端節点で Max (Min) プレイヤーの負けである．このようにして，各節点から互いに最善を尽くすと勝ちか負けか引き分けか（勝ち確定か負け確定か引き分け確定か）を計算する．そして，手番にとっての負けに至る手は探索中選ばないようにしている．このようにすることで，探索の効率化ができるとともに，負けに至る手の訪問数が他の手と比べて小さくなるため，通常の UCT と比べて，利得の推定値が素早く真の値（理論値）に近づくことを期待できる．尚，文献 [74] では手の選択の際，UCB 値に手のカテゴリに基づいた値を付加したものを用いるが，本研究のゲーム木では，特に手にカテゴリはないので，式 (2.5) をそのまま用いた．

MCTS-Solver を使った場合，葉以外の更新を UCT と同様に式 (2.11) で表せる．し

かし，最善手の値を利用して親，そして祖先の評価をすることが理想的である一方で，MCTS-Solver では，手番の負けが確定した手，つまり，最善で負けの手しかない場合を除き，最善でないと確定した手の古いプレイアウト結果が親・祖先の評価に引き続き使われてしまっている．ここに改善の余地があると考えられ，この観点から 5 章では改善を試みている．

2.6.3 Accerated UCT

Accerated UCT は UCT の変種で，最近行ったプレイアウト結果をより重要視する手法である．囲碁，オセロ，havannah で応用され，UCT よりも勝率の面で優れた結果を示した手法である [29]．最近行ったプレイアウトを優先する理由は，探索が進むにつれて，UCB1 の性質から推定値が良いところをより訪問することと，真の最善手に対する推定値が他の手の推定値より良くなると期待されることの二点による．Accerated UCT では UCT のプレイアウト結果の更新の部分を変更し，親の利得の推定値を算出する際に子の推定値の重み付き和をとる．$t+1$ 回目のプレイアウト終了後，まず，葉の値を式 (2.8)(2.9) で更新する．$t+1$ 回目のプレイアウトで節点 i が訪問されたとすると，i の子 j が選ばれたなら 1 そうでないなら 0 を意味する項 $\mathbb{I}_{t+1}\{\text{j is chosen}\}$ を用いて，子 j の重みは v_j を

$$v_j \leftarrow v_j\lambda + \mathbb{I}_{t+1}\{\text{j is chosen}\}$$

という形で更新する．ここでの $0 < \lambda \leq 1$ は重みの減り方を調整するパラメータである．従って，節点 i を訪問したにもかかわらず，子 j が訪問されないとその子 j の重みが下がる．

Accerated UCT では子節点だけでなく，展開前（その節点が探索木の葉であった時）のシミュレーション結果に対しても，重みを考慮する．方法としては，探索木の葉 i の展開の際，子の他に仮想的な子を作り，i から行われた今までのシミュレーションの平均利得 $\bar{X}_{i,N_i}$ を仮想的な子 c_i に入れて，その子の重みを $v_{c_i} = N_i$ にしておく．尚，2.6 節の探索木の成長の仕方では N_i は 1 または 0 になる．この仮想的な子 c_i は訪問されないため，徐々に重みが減るという特徴がある．節点 i の子の集合を C_i とすると，子の重み v_j を使い，i の利得の推定値は

$$\bar{X}_{i,N_i} \leftarrow 1 - \frac{\sum_{j\in C_i\cup\{c_i\}} v_j\bar{X}_{j,N_j}}{\sum_{j\in C_i\cup\{c_i\}} v_j} \tag{2.12}$$

という形で更新される．この更新では相手番にとっての利得とするために正負反転し 1 を加えている．尚，この更新は $\lambda = 1$ の時には UCT と同じになる．加えて，Accerated UCT では平均利得の調整だけを行うので，UCB 値の第二項はそのまま用いる．文献 [29] では Accerated UCT での終端節点の扱いについて特に述べられていない．本研究では，終端の節点扱いとして，$\lambda = 1$ の時に MCTS-Solver を UCT と組み合わせたものに一致するように，勝ち負けが確定しても葉の値は，古いプレイアウト結果を利用し，式 (2.8)(2.9) で更新した値を用いた．

2.7 その他の MCTS アルゴリズム

本節では，UCT ほど実用で使われていないが，理論的な性能が比較的明らかになっている点で優れている MCTS について紹介する．理論的な性質の解析は，ドメインによらない性能を保証することに繋がるため，重要である．それに加えて，MCTS に対する深い理解を得られる，しかしながら，理論的に優れた手法が，実験的な性能も優れている訳では無いと経験的に知られている．理論的に優れ，実験的にも優れた MCTS を考案することは今後の課題である．

2.7.1 Sparse Sampling (SS)

Sparse Sampling (SS) [41] は，MDP での状態価値推定（そこから最善を尽くした場合の利得の期待値）を目的としたアルゴリズムである．SS は最良優先探索ではないが，UCT の元となったアルゴリズムの一つである．

SS では各状態で各行動を規定回数（C 回）を再帰的に行う．具体的には根から実行し，行動をとって至った次の状態でも C 回実行するというように再帰的に実行する．そのようにして，状態遷移確率，利得の推定を行う．この手法で特筆すべきなのは，十分な精度で状態価値推定を行うのに必要なサンプルサイズ C は，MDP の状態数には依存しないということが理論的に示されている点である．これは，滅多なことでは遷移しないような状態が多数存在しても，SS では，それらをあまり考慮しないことにより効率的に状態価値が推定できるということである．例えば，古典的な方法の一つである value iteration では，全状態を辿る必要があるため，状態数に比例する計算時間がかかる．

SS では，状態遷移確率が手に入らなくても，状態，行動対に対し，次の状態がサンプル出来れば良い．つまり，状態遷移確率の代わりに次の状態を生成するためのモデル（生成モデル）があれば良い．この利点は，例えば，複数個の確率変数の最大値をパラメータとして次の状態の確率分布が定まる場合に活かせる．なぜなら，この場合，最大値の分布の計算にはコストがかかるために，状態遷移確率を計算するのは容易ではない一方で，最大値の分布からサンプリングするのは容易なためである．この生成モデルさえあれば実行可能という良さは，SS だけでなく，UCT を含む多くの MCTS アルゴリズムに共通している．

2.7.2 BRUE

BRUE [21] は UCT とは異なり，シミュレーションをどこから実行するのかと手の評価を分けるアルゴリズムであり，単純リグレットが定数 b に対し $\mathrm{O}(\exp(-bn))$ で抑えられるという収束性能の保証を持つ．木での単純リグレットは MAB の単純リグレットを一般化したもので，根から最善を尽くした場合の利得の期待値とアルゴリズムが選ぶ根の行動の後，最善を尽くした場合の利得の期待値の差である． 2.2.4 節で紹介した

MDP の記法を用いて，根の状態を s_0，アルゴリズムが選ぶ行動を a_0 とし，単純リグレットを式で表すと $\max_a Q(s_0, a) - E[Q(s_0, a_0)]$ である．尚，単純リグレットの式では，行動 a_0 は探索の結果（確率的）により変わるため，期待値をとっている．

UCT の単純リグレットの上界について，プレイアウト数が十分に大きい時の減り方と，特定のゲーム木で減るまでにかかる平均的なプレイアウト数の両方が解析されているが，それらの観点では BRUE の方が優れている．前者については，UCT の Theorem 6 [42] で次善手を選ぶ割合は多項式のオーダーであると示されている．そのため，単純リグレットの減り方も多項式のオーダーであり，前述の上界が示されている BRUE の方が良い．後者について，UCT で最善な終端節点（根から最善手を選び続けて至る終端節点）を初めて訪問するまでに平均的には木の深さの hyper-exponential のオーダーのプレイアウトが必要な木が示されている [16]．そのため，少なくとも hyper-exponential なオーダーでプレイアウトを行わない限り，UCT の単純リグレットの上界は小さくならない．これに対して BRUE では，指数オーダーのプレイアウトで済むということが示されており，UCT で必要なプレイアウト数よりずっと小さいという利点を持つ．

しかしながら，BRUE の解析では，探索木を固定した状況を考えており，現実的な想定ではない．また，BRUE は一様な確率でのプレイアウトを行い，現状での推定最善の方策をとった場合の利得をサンプルするというアルゴリズムである．一様な確率でのプレイアウトというのは，UCT の様に，今までに得た利得に基づき，プレイアウトの方策を有望そうなところを重視するということを行わないということである．そのため，二人零和ゲームを始めとしたゲームでの実際的な性能（限られたプレイアウト回数での性能）はあまり期待できない．但し，シミュレーションと評価を分けるという方法は有効である可能性がある． 7 章では，UCT での手の期待利得の推定をシミュレーションの平均利得で行う代わりに， 6 章での推定量を使うという形で，シミュレーションと評価を分けた MCTS を提案する．

2.7.3 単純リグレットの最小化に向けた研究

上記の他にも，最近では，MCTS 探索での単純リグレットを減らす方法について理論面で研究が進んでいる．文献 [25] では，二人零和完全情報ゲームでの MCTS の探索木を深さ 2 で固定した場合に相当する，maximin のバンディット問題を考案し，その問題でのアルゴリズムを提案し，解析した．より具体的には，maximin のバンディット問題は，2 つの行動 $i \in \{1, \ldots, K\}, j \in \{1, \ldots, K_i\}$ を選ぶとその i, j に応じた分布に従い利得を得る設定のもと，目的は

$$\arg \max_{i \in \{1,\ldots,K\}} \min_{j \in \{1,\ldots,K_i\}} \mu_{i,j}$$

を見つけることである．ここでの $\mu_{i,j}$ は行動列 i, j に対する分布の期待値である．文献 [25] では，提案したアルゴリズムについて，δ-PAC（$\varepsilon \geq 0$ と $\delta \in (0, 1)$ に対して，アルゴリズムで選んだ手の利得の期待値の理論値と，最善手のものとの差が ε 以下にな

る確率が $1-\delta$ 以上）となるのに必要なプレイアウトの数について理論的に解析している．δ-PAC は言い換えると木における単純リグレットが ε 以下になる確率が $1-\delta$ 以上になることである．

文献 [39] では，maximin のバンディットを拡張し，2 つの行動ではなく，いくつかの行動を選ぶと確率的に利得が得られる木でのアルゴリズムを提案し理論的な解析をしている．提案されたアルゴリズムは各葉で平均利得の不確かさの上界，下界を考え，それに基づき探索をする．このアルゴリズムでも，上記と同様（(ε, δ)-correct と言及されている）探索における単純リグレットが確率 $1-\delta$ 以上で ε 以下になるという条件を満たすために，必要なプレイアウト数について理論的に解析している．この論文の設定では，MCTS の探索木を成長させない場合に相当する．もし仮に木を成長させていく場合，葉からの開始したシミュレーションの数がそれほど大きくなる前に葉が展開される．そのため，葉の平均利得の不確かさが常に大きい値になってしまう．従って，文献 [39] のアルゴリズムは，探索木の成長を考えると，非常に緩い上界と下界に基づき，探索することになるという弱点がある．

2.8 探索ヒューリスティックの動的な調整

本研究では，MCTS における「次善手の偏り」という難しさを 3 節で扱う．3 節で用いる Dynamic komi という手法を中心に，他の難しさについてどのように対処しているかについて紹介する．一般に，可能な行動の数，行動後の遷移先の数が大きい場合に最善の行動を見つけるのは難しい．

2.8.1 Dynamic Komi

3 章と関連の深い Dynamic komi という手法を紹介する．Dynamic komi [5] はゲームのスコアがエージェントに手に入るゲームを対象とした手法で，MCTS アルゴリズムにおけるシミュレーションの結果を適切に補正することで，より効果的に探索することを目的としている．尚，囲碁において，ゲームのスコアは目数とコミの和に相当する．

Dynamic komi が提案された背景として，MCTS はハンディキャップをもらってもそれを上手く活かせないということがある．例えば，囲碁で，平手（ハンディキャップ無し）の場合と比べて，置き碁（ハンディキャップ有り）が苦手であると知られている．置碁のように一方のプレイヤーが有利な状況では，有利な側は勝ちをより確実なものにする手を，不利な側は不利な状況を拮抗した状態に近づける手を指すことが求められる．しかし，UCT で手を決定すると有利な状態から互角に，不利な状態からより不利な状態になってしまうことがある．このことの主な原因は，有利な場合はどんな手でもプレイアウト結果はほぼ勝ちで，不利な場合はその逆となり，各手の評価に差が付かず，その結果，悪い手を選びやすくなってしまうことにあると考えられる．

Algorithm 1 Score Situational（一部簡略化）

Require: $\overline{\text{score}}$ is the average score of all playouts

phase $\leftarrow$ BoardOccupiedRatio $+s$

rate $\leftarrow 1/(1+\exp(c \cdot \text{phase}))$

threshold $\leftarrow$ threshold + rate $\cdot\overline{\text{score}}$ **return** threshold

Dynamic komi [5] は勝ち負けの境界（普通はゲームのスコアが 0 が境界）を調整することで，この悪さを和らげる手法である．通常の UCT では，ゲームのスコアを勝ち負けに変換し，勝ち負けに応じて利得を決める．文献 [5] では，Score Situational，Value Situational，Linear Handicap という 3 種類の Dynamic komi が提案手法されている．Score Situational（Algorithm 1）は境界を平均スコアへ移動させる手法である．c と s はパラメータである．“BoardOccupiedRatio” は囲碁ではゲームの進行度合いを表しているため，今までの手数を対応させることで一般のゲームでも適応できる．Value Situational（Algorithm 2）は全シミュレーションでのエージェントの平均利得（$\bar{X}$）が $[X_{\text{lo}}, X_{\text{hi}}]$ に収まるように境界を移動させる．境界の位置を安定させるため，“Ratchet” という変数が導入されている．“Ratchet” は始めは ∞ である．Linear Handicap は囲碁のルールに依存するため，一般化が難しく，本研究の対象とはしていない．

文献 [5] 中の両アルゴリズムでは，それぞれ $s = 0.75, c = 20$, red $= 0.45$, green $= 0.50$ が用いられている．尚，前者では，s, c を十分大きく設定し，後者では，red を 0 に green を 1 に設定することで通常の UCT と同じ振る舞いになる．また，両アルゴリズムでは，初期局面から 20 手未満の局面で探索を行う時，例外として置き石に応じて線形にコミを加える．しかし，本論文で扱う状況では置き石に相当するものは無いので，省いた．

一般に利得は必ずしも離散値 $0, 1$ である必要は無く，ゲームのスコアに対し線形に利得を与えることも可能である．しかしながら，囲碁では，スコアではなく勝ち負けを扱った方がより効果的であると知られている [6,19]．その一方で勝ち負けの質に対するさらなる調査がなされ，$1, 0.5, 0$ 以外の利得を試した研究もある [54]．3 章では前者の立場から勝ち負けの境界の調節の方法について議論している．

2.8.2 シミュレーションからの学習

UCT を始めとした MCTS では，シミュレーション方策は一様な確率で手を選ぶだけであった．しかしながら，シミュレーション結果を元にシミュレーション方策を学習することも可能であり，実際に General Game Playing (GGP) を対象とした研究 [8] や囲碁を対象とした研究 [27] が存在する．これにより，探索木が成長しきらないような条件（時間的，メモリ的な制約）でも，良い行動がとれるようになる可能性がある．GGP を対象とした研究については特に比較はなされていないが，囲碁を対象とした研究については，シミュレーション方策の学習を行う方が，ランダムなシミュレーション方策より

Algorithm 2 Value Situational（一部簡略化）

Require: $\overline{X}$ is the observed winning rate.

```
if X̄ < X_lo then
    if threshold > 0 then
        Ratchet ← threshold
    end if
    threshold ← threshold −1
else
    if X̄ > X_hi and threshold < Ratchet then
        threshold ← threshold +1
    end if
end ifreturn threshold
```

効果的であったという報告されている．

2.9 ハイブリッドな探索手法

本節では MCTS におけるハイブリッドな手法を紹介する．ハイブリッドな手法では，根（付近）で，単純リグレットを最小化する手法により探索資源を分け，そしてそれを正確に行うために，累積リグレットを最小化する手法を使い，各節点の評価の精度を高める．

2.6 節で紹介した通り，MCTS の代表的なアルゴリズム UCT は MAB での累積リグレットを最小化を目指したアルゴリズム UCB1 を MCTS に応用したものである．そのため，推定値の良い手を優先し，シミュレーションを行う葉を選択する．しかしながら，探索の目的は最善手を判別すること（ゲーム木での単純リグレットを最小化すること）である．従って，最善手の判別に役立つ葉からシミュレーションを行い，木での単純リグレットを小さくするアルゴリズムが理想的である．

しかしながら，木での単純リグレット最小化は簡単ではない．簡単な方法として，例えば，UCT での UCB1 の代わりに MAB での単純リグレット最小化のアルゴリズムを使う方法が考えられるが，その方法では，単純リグレットの最小化は難しいと予想される．理由は以下による．まず，前提として，MAB の単純リグレットの最小化アルゴリズムで，木の単純リグレットを最小化するためには，根の子節点以下は MAB での場合と近い，つまり，根の子節点の利得の推定値は平均的には利得の理論値に近くなければならない．しかしながら，単純リグレット最小化のアルゴリズムは（少なくとも累積リグレット最小化のアルゴリズムよりも）次善手を多く引く傾向にあるため，根の子節点の推定値（平均利得）が不正確になると予想される．

根の子節点の利得のより正確な推定のためには，MAB での単純リグレット最小化のアルゴリズムよりも累積リグレット最小化のアルゴリズムを使う方が優れている．それ

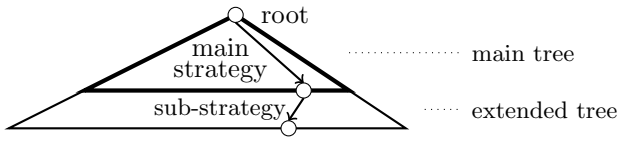


図 2.5: ハイブリッドな MCTS の探索木

は，一般の木の場合でも MAB と同様に，最善の子を多く選ぶことにつながると期待されるため，その結果として，各節点での利得をシミュレーションの平均利得によって推定する際により正確な推定が出来ると期待されるためである．従って，節点の利得を正確に推定するために UCT が役立つと期待される．

一般に MAB では，単純リグレットの最小化と累積リグレットの最小化は両立できないことが知られている [11]．そのため，MCTS でも，別のアルゴリズムを使ったハイブリッドな MCTS が提案された．ハイブリッドな方法では，探索の方法を 2 つ（primary strategy と sub-strategy と呼ぶ）に分け，単純リグレットを減らすための primary strategy を根あるいは木の浅い方（main tree と呼ぶ）で用い，累積リグレットを減らすための sub-strategy を残りの木（extended tree）で用いる（図 2.5）．

近年の研究はハイブリッドな方法が効果的であると示唆している．sub-strategy として UCT を用い，情報の価値 (VOI) を近似的に求め，それが最大の手を選ぶという primary strategy を用いた MCTS SR+CR という方法が効果的であったという報告 [70] や，primary strategy に SHOT [14] を用いた H-MCTS が効果的という報告もある [53]．しかしながら，SHOT は予め，シミュレーションの数を固定しておかなければならないという制約があるため，H-MCTS でも同様の制約がある．4 章では，他の primary strategy を使った，anytime アルゴリズムを提案する．

2.9.1 利得分布のミニマックス的な更新

MCTS では，通常，各節点で平均利得だけが保持される．しかしながら，もし，利得のヒストグラムが保持されていれば，局面について，平均利得と比べてより多くの情報が引き出せると期待できる．例えば，文献 [28] では，囲碁の攻め合いの局面を，目数のヒストグラムから推定する手法を提案している．この手法の意図は，MCTS が比較的苦手とされる攻め合いの局面を見つけ，探索に役立てるということである．尚，利得の分布はミニマックス（ネガマックス）の考えに基づく方法で更新される [7,68]．そのため，推定値が正しい値に収束するのが速いと期待される．

Bayes-UCT [68] は UCT のベイズ的な拡張で，Tesauro は 2 種類提案した．どちらの Bayes-UCT も子の中で，それぞれの Bayes-UCT の値が最大の子を選ぶ．

$$\text{Bayes-UCT1} := \hat{\mu}_i + \sqrt{2\ln t / N_i(t)},\ \ \text{Bayes-UCT2} := \hat{\mu}_i + \sqrt{2\ln t}\sigma_i, \tag{2.13}$$

上記の各変数について説明する．$\hat{\mu}_i$ は主観確率に基づく子 i の利得の期待値で，$\hat{\mu}_i = \int_x (1-x)p_i(x)$ であり，ここでの $p_i(x)$ は節点 i の手番のプレイヤーにとっての利得の理

表 2.1: 類似の手法との比較．表の上半分にに既存手法をまとめた．“minimax backup” の列は各節点の分布がどのように表現されるかを示し，“-” は利得の分布のミニマックス的な更新が行われないことを表す．“primary strategy” の列は展開やプレイアウトのために葉を選ぶ方法 primary strategy を表す．“sub-strategy” の列はハイブリッドな方法が使われている場合，木のより深いところで使われている探索方法 sub-strategy を意味し，“-” はハイブリッドな方法を使っていないことを表す．表の下半分には 4 章での提案手法をまとめた．提案手法は，離散分布でミニマックスの考えに基づく更新をし，UCT と組み合わせたハイブリッドな方法である．

既存手法	minimax backup	primary strategy	sub-strategy
Baum and Smith [7]	discrete	QSS	-
BayesUCT [68]	Beta distribution	UCB (Bayes)	-
H-MCTS [53]	-	SHOT [14]	UCT
MCTS SR+CR [70]	-	ϵ-greedy, UCB$_{\sqrt{(\cdot)}}$, VOI	UCT
Yokoyama and Kitsuregawa [78]	discrete [7]	QSS [7], UCT	$\alpha\beta$ search
4 章での提案手法			
HB+E$^{\text{Expected}}$	discrete [7]	E$^{\text{Expected}}$	UCT
HB+E$^{\text{Robust}}$		E$^{\text{Robust}}$	
HB+E$^{\text{Terminal}}$		E$^{\text{Terminal}}$=QSS [7]	
HB+BayesUCT1		$\hat{\mu}_i + \sqrt{2 \ln t / N_i}$ [68]	
HB+BayesUCT2		$\hat{\mu}_i + \sqrt{2 \ln t} \sigma_i$ [68]	

論値が x である確率であり，σ_i は確率分布の標準偏差である．一手打てば手番が変わるため，節点での利得 $1-x$ はその親節点での利得 x に相当する．Bayes-UCT では，UCT で平均利得 $\bar{X}_{i,N_i}$(式 (2.5)) を用いていたところを $\hat{\mu}_i$ に置き換える．実験により，Bayes-UCT の $\hat{\mu}_i$ はより速く節点 i の利得の理論値に収束することが示されている．本研究での手法（4 章）との間には，事前分布を使うか，離散分布を使うか，ハイブリッドにするか等の違いがあるが，本研究では，他の手法に合わせて $\hat{\mu}_i = \sum_{x \in \{0,0.5,1\}} (1-x) p_i(x)$ を使った．

他にも分布を使った探索手法としては Baum のアルゴリズム [7] がある．このアルゴリズムは最良優先探索の一種で最善手とその他を見分けるのに最も展開する意味があると期待される葉を展開していく．Baum の方法は，節点に対し，確率分布を返す評価関数の存在を仮定している．

4 章では，UCT により分布を作成し，その分布に基づくハイブリッドな MCTS を提案する．そのため，提案手法は Baum の方法とは異なり，評価関数が無くても探索が可能である．また，4 章では Baum の方法を MCTS に相応しく改良する．表 2.1 の上半分に既存手法をまとめた．

第3章　モンテカルロ木探索と有利不利が偏った局面

モンテカルロ木探索（MCTS）は様々なドメインで応用され，成果を上げてきた．古典的なミニマックス探索では，確かな評価関数があるという仮定のもとで，その性能が主に探査空間の大きさに依存すると知られている．それに対し，MCTS では，どのようなゲームの特徴が性能を左右するのかは，明らかではない．MCTS に関する先行研究では，次善手の分布や，木の形の非一様さといった特徴が状態空間の大きさという特徴より，MCTS の性能に大きな影響を及ぼすと示されている．本章では，新たな特徴として，次善手の偏りを提案し，この特徴もまた，MCTS の性能に大きな影響を及ぼすことを示す．加えて，その特徴を持つ局面をより上手く扱う，MCTS の簡単な拡張手法も提案する．次善手の偏りがあるゲーム木は，根のゲームの理論値が引き分けで，一方のプレイヤーの次善手はもう一方のプレイヤーの次善手と比べて悪いというゲーム木である．本章では，多腕バンディット問題のアルゴリズムの MCTS への応用として，最も広く用いられている UCT だけでなく，様々なアルゴリズムを MCTS に応用し，実験を行い性能を確かめる．その結果，次善手の偏りにより全てのアルゴリズムで性能が低下すること，拡張手法を使うことにより，その特徴による性能の低下を抑えられることを示した．

尚，本章の内容は発表済みの論文 [33] に加筆，編集したものである．

3.1　背景

モンテカルロ木探索 (MCTS) は囲碁を始めとしたドメインで，成功したアルゴリズムである [26]．また，General game playing [22] や不完全情報ゲーム [37] やリアルタイムゲーム [56] を含む，様々なドメインにも応用されている．しかしながら，MCTS がどのドメインに対し効果的でであるかについてはまだ研究の途上である．チェス等のいくつかのドメインでは，確かな評価関数を利用した古典的なミニマックス探索が MCTS を凌駕する性能を持つと知られている．また，単純な MCTS の代わりに，MCTS にミニマックス探索して得た評価値を組み合わせたアルゴリズムを使うことで改善するドメインもあると報告されている [45]．その一方で，Lines of action では評価関数を組み合わせることで MCTS はミニマックス探索に並んだという報告がある [73].

各ドメインのどのような特徴が MCTS の性能に支配的な影響を及ぼすのかをよりよく理解するため，人工的なゲーム木を使って研究が行われてきた．Finnsson は，探索

空間の大きさはミニマックス探索の性能にとって第一義的な意味をもつ一方で，探索空間の大きさよりも次善手の分布が MCTS にとってより重大であることを示した [22]. Ramanujan は節点の種類の一つとして，トラップという，数手で負けとなる節点であるが，兄弟を選ぶとゲームが続くという節点を考え，実験的に解析した．その結果．MCTS は一様な分枝数と深さの木で効果的で，トラップがあると性能が落ちることを示した [59]．また，Long は不完全情報ゲームで，disambiguation factor と呼ばれる，木の深さに対する情報集合に含まれる節点の数の減り具合を定義し，不完全情報ゲームにおいて，あり得る世界を完全情報ゲームとして探索する場合に disambiguation factor と葉の価値との相関が性能に大きな影響を及ぼすことを示した [47].

本章では，新しい特徴として,「次善手の偏り」を定義し，その存在が MCTS アルゴリズムの性能を下げるという意味で重要な特徴であることを説明する．そして，ゲーム木の根のゲームの理論値は引き分けであるが，その一方で，片方のプレイヤーの次善手は，他方のプレイヤーの次善手より悪いことが多いとしたゲーム木に焦点を当てる．これは，ゲームの理論値の面では局面は互角であるが，最善手を選べなかった場合の損失の面では有利不利が偏っている状況である．例えば，チェスでチェックメイトに至る長い手筋が存在し，正しい手を選び続ければ勝てるが，正しい手を一度でも選べないと負けが決まるような状況である．また，人間対 AI という形式のゲーム (e.g. ミズ・パックマン [32]) では，人間のプレイヤーは勝つことは可能であるが難しい局面から始まることもある．囲碁での攻め合いの局面の一部は，次善手の偏りがある局面と類似の特徴を持つ．Graf が提示した局面 [28] では，多くのプレイアウトは大差をつけて黒の勝ちであるが，そこから互いに最善を尽くすと引き分けか僅差で白の勝ちである．そのため，その局面及びその先の局面の白の次善手には，黒の次善手より悪い手がより多く含まれていると予想される．加えて，MCTS では，囲碁のハンディキャップ戦（置き碁）で上手くハンディキャップを活かせないことが知られている [5]．置き碁はゲームの理論値は有利なプレイヤーの勝ちと予想されるという点で本章で対象とした性質とは異なるものの，一方のプレイヤーが有利であるという点では共通する特徴を持つ局面である．

「次善手の偏り」の存在が MCTS アルゴリズムの性能を下げることの大雑把な説明は，自分と相手の手をいろいろ試してみても勝つ確率がほとんど変わらないなら，次善手の中から最善手を見つけることの難しさは増大するということである．この問題を解決するため，勝ち負けの基準をプレイアウトでのゲームのスコアの頻度に基づいて調整するという単純な解決策を提示する．実験では，incremental random game モデルとその変種を用い，提案手法を組み合わせる探索アルゴリズムとして，UCT の他に，MCTS に KL-UCB [13]，Thompson sampling [1] を応用したものを用いた．ゲーム木モデルとその変種は標準的なモデルであり，アルファベータ探索や MCTS ゲーム木探索アルゴリズムの性能を評価するのに用いられている [23, 42, 43, 49, 59] ものを使った．その結果，次善手の偏りが実際に全てのアルゴリズムでの性能を下げること，提案手法により性能の低下が改善されたことを示した．また，incremental random game モデルでも評価し，実際に改善したことも示した．

尚，UCB1 等のバンディット問題でのアルゴリズムを MCTS に応用した場合の比較について，同時着手ゲームである Tron ですでになされているが [56]，決定的なゲームではなされておらず，知りうる限りこの研究が初めてである．

3.2 Incremental Random Game モデルでの次善手の偏り

本節では，次善手の偏りという特徴を表現するため， 2.2 節で紹介した，標準的なゲーム木モデルである incremental random game モデルに従って，いくつかのゲーム木モデルを紹介する．

incremental random game でのゲーム木は，一様な分枝数を持ち，その深さは偶数と仮定する．木において，整数値が各手（辺）に与えられ，そして，最善手に対応する辺では値が 0 で，残りの手は，Max (Min) プレイヤーの手番の節点では負（正）の値を割り当てる．終端節点でのゲームのスコアは根からその終端節点までの値の和である．もし，スコアが境界（通常は 0）より大きい，小さい，等しい場合，それぞれ，Max プレイヤーの勝ち，負け，引き分けを意味する．そのため，それぞれの節点にはちょうど 1 つだけ最善手（辺の値が 0）が存在する．そして，文献 [23] と同様に，互いに最善を尽くして終局するとゲームのスコアは 0 となる．また，簡単のため，根は Max プレイヤーの手番とする．

3 つのゲーム木モデル Constant, Constant', Random について次善手の偏りを導入する．パラメータ $0 < a \leq b$ に対し，Constant(a, b) は Max（Min）プレイヤーの次善手の辺の値が $-a$ (b) と決められた値となる木である．Constant'(a, b) は Min プレイヤーの全ての次善手の辺の値は最後（終局前）の手を除き a となる変種で，最後の手の内，次善手の辺の値は b となる．a と b の違いが次善手の偏りを表している．$a = b$ となる特別な場合では，次善手の偏りは無くなり，Finnsson の扱ったモデルと同じになる [22]．パラメータ $0 < p$ と $0 \leq q$ に対し，Random(p, q) はランダムな木の一つで，Max（Min）プレイヤーの次善手の値は範囲 $[-p, -1]$ $([2^q, 2^q + p - 1])$ の内の整数値を一様な確率で割り当てる．この木では，パラメータ q が次善手の偏りを表している．$q = 0$ という特別な場合では偏りがなくなり，木は incremental random game のものと同じになる．図 3.1 に木の例を示す．後の実験（図 3.3(a)，3.8(a)）で示されるように，ゲーム木モデルにおける次善手の偏りは MCTS の性能を劇的に下げると言える．

3.3 利得の差の最大化

本節では，最善手の利得とその他の手の利得との平均的な差を最大化することに基づいた MCTS の拡張手法を示す．

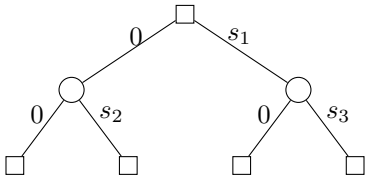


図 3.1: 分枝数と深さが 2 のゲーム木モデルの例．四角と丸はそれぞれ Max，Min プレイヤーの節点を意味する．それぞれの辺は手を表し，その値は最善手で 0，次善手はモデル次第である．Constant(a, b) では， $s_1 = -a$ で $s_2, s_3 = b$ である．Random(p, q) では，$s_1 \sim \mathcal{U}([-p, -1])$ かつ $s_2, s_3 \sim \mathcal{U}([2^q, 2^q + p - 1]$ であり，$\mathcal{U}(\cdot)$ は一様分布を意味する．

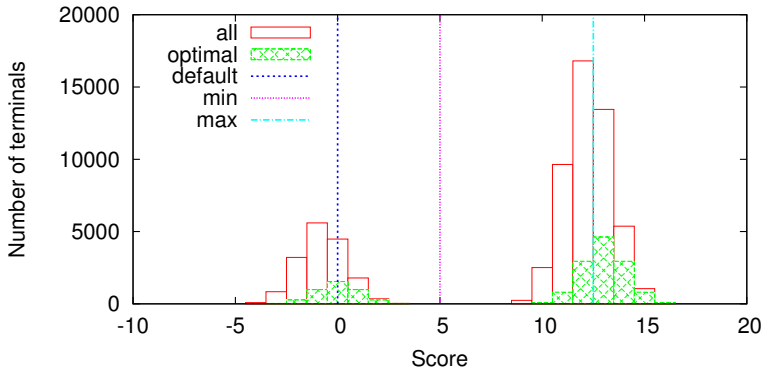


図 3.2: Constant'(1,13) で，終端節点でのゲームのスコアのヒストグラム

$\mathbb{K}$ を根の局面での合法手（その局面で可能な手）の集合とする．μ_i を手 $i \in \mathbb{K}$ からシミュレーションを開始した際の利得の期待値とし，特に μ_* を最善手からシミュレーション開始した場合の利得の期待値とする．最善手の期待値と次善手の内で最も有望に見える手の期待値との差を

$$\Delta := \mu_* - \max_{i \in \mathbb{K} \setminus \{*\}} \mu_i$$

と定義する．概ね，大きな Δ の方がより簡単に最善手が見つかると期待される．例えば， 2.4.1 節の式 (2.2) で紹介したように，UCB1 では $\Delta_i := \mu_* - \mu_i$ とすると，次善手を引く回数の期待値について

$$\mathbb{E}[N_i(n)] \leq \frac{8 \ln(n)}{\Delta_i^2} + 1 + \frac{\pi^2}{3}$$

という不等式が成り立つ．多腕バンディット問題と深さ 1 の木探索との類似性のため，木探索でも同様であると期待される．

3.3.1 Constant モデルの一手読み探索での分析

一手読みでの Constant と Constant' を分析することで，次善手の偏りによる難しさを示す．ここでの一手読みとは，MCTS におけるゲーム木の深さを 1 で固定し，それ

以上の展開をしない場合のことで，この時，MCTS は MAB と同じになる．Δ の大小と問題の難しさへの影響について，図 3.2 を例に示す．これは，Constant'(1,13) で (分枝数，深さ) が (4,8) とした時の全ての終端節点でのゲームのスコアをヒストグラムにしたものである．図で “all” は全ての終端節点，“optimal” は根での最善手の子孫の終端節点だけを抜き出したものである．この設定では，根の次善手の辺の値は全て同じなので，“all” から “optimal” を除いたヒストグラムが，次善手のヒストグラムである（これを次善手の数である 3 で割れば “optimal” とスケールが同じになる）．

図 3.2 は，終端節点の多くが正のスコアを持つことを示している．つまり，通常の勝ち負けのスコアの境界（$= 0$ であり，図中の “default”）では多くの終端節点が Max プレイヤーの勝ちの節点である．一般に，次善手の偏りが増えると，利得の期待値がより非対称になる．それは，根からのシミュレーションでは終端節点が一様な確率で訪問されるためである．図中では，全体での利得の期待値は 0.798 であり，最善手からシミュレーションした場合の利得の期待値 0.875 に近い値となっている．次善手からシミュレーションした場合の利得の期待値は 0.772 と計算でき，$\Delta = 0.103$ と小さな値を持つ．もし，勝ち負けの境界がスコア 4 から 8 にある（図中の “min”）と， 全体と最善手の利得の期待値は 0.75 であり，Δ は 0 となる．このような小さな Δ は MCTS での最善手を見つけることの難しさの原因になる．

しかしながら境界を 12 から 13（図中の “max”）にすると，最善手の利得の期待値は 0.515 となり，全体の利得の期待値は 0.234 となる．次善手の利得の期待値は 0.14 となるため，Δ は最大値 0.375 をとる．一般に良い境界は Δ を大きく出来，最善手を見つけることに貢献する．

Dynamic komi の Score Situational はこの Constant' には向かない手法である．Score Situational は今までに観測されたスコアの平均値を 0 に近づける．そのため，境界が平均には，図 3.2 で “all” や “optimal” の頻度が低い，スコア 9 から 10 になり，その場合 Δ はほぼ 0 である．

プレイヤーには得られないが，全ての終端節点のスコアを使えば，Δ を最大化するような境界が判明する．良い境界の調整方法を考察するため，実験では，比較対象として，Δ を最大化する境界も用いた．Constant(a, b) と Constant'(a, b) では，勝ち負けの境界が t の時の Δ の値は下記のような簡単な式で求まる [79].

$$\Delta(t) = \sum_{t<s<t+a} P[s] + 0.5P(t) + 0.5P(t+a), \tag{3.1}$$

ここでの s は整数値で，0.5 は引き分けの利得である．$P[x]$ は最善手から始まるシミュレーションの利得が x となる確率で，$P(x)$ は x が整数値の時 $P[x]$ そうでない時には 0 となる関数である．Constant(a, b) と Constant'(a, b) では，ある節点を根とした部分木とその兄弟を根とした部分木はその親節点までの辺の値の総和を除いて，同じであるという特徴がある．そのため，$P[x + a]$ は次善手から始まるシミュレーションのスコア

Algorithm 3 Maximum Frequency method

Histogram[score] += 1 **return** $\arg\max_t$ Histogram[t]

の確率を表していることに注意されたい．等式

$$\sum_{t<s\leq t+a} P[s] = \sum_{t<s<t+a} P[s] + P(t+a)$$

を使うと式は

$$\Delta(t) = \mu_* - \mu = \left(\sum_{s>t} P[s] + 0.5P(t)\right) - \left(\sum_{s>t} P[s+a] + 0.5P(t+a)\right)$$

である．

3.3.2 最大頻度法と UCB

実際的な状況での Δ を増やすため，Algorithm 3 に示した，頻度が最大なスコアに境界を移すという最大頻度法を提案する．提案手法ではプレイアウト毎に，それぞれのスコアの頻度を管理し，それに基づき，最も良い境界を返す．式 (3.1) で示したように $a = 1$ かつ 1 手読みを仮定した場合で s を $P[s]$ を最大化するスコアとすると，$\Delta(t)$ を最大化するという意味で最良な t は $t = s - 0.5$ である [1]．最善手 $*$ や各手の期待利得 μ_i は不明であるから，スコア s が最善手からのシミュレーションで出る確率 $P[s]$ をスコア s が観測された頻度で近似する．

妥当な多腕バンディットのアルゴリズムでは，次善手を引く数の期待値には上界がある．例えば UCB1 では 3.3 節で再掲した 2.4.1 節の式 (2.2) のように $O(\ln(n))$ である．そのためプレイアウトが増えるにつれて，この近似の誤差も小さくなると期待される．Random では，Constant と同様の解析は出来ないため，実際に実験して最大頻度法の効果を確かめた．

また，境界が変化した際の MAB のアルゴリズムの不確かさの評価の修正も合わせて提案する．境界が変化した時，期待利得は今までのものとは大きく変化する可能性がある．探索を安定させるため，評価を修正する．t を現在の時刻， t_c を最後に境界が変化した時刻，t' をその差分つまり $t' = t - t_c$ とする．また， $N'_i(t)$ を t_c の後に手 i を訪問した数，つまり $N'_i(t) = N_i(t) - N_i(t_c)$ とする．UCB1，KL-UCB，Thompson sampling で t の代わりに t' を使い，それぞれ，式 (2.1)，(2.3)，(2.4) を式 (3.2), (3.3), (3.4) へと変更する．

$$\overline{X}_{i,N_i(t)} + \sqrt{\frac{2\ln t'}{N'_i(t)}}, \tag{3.2}$$

$$\max q_i \in [0,1] : d(\overline{X}_{i,N_i(t)}, q_i) \leq \frac{\ln t' + c\ln(\ln t')}{N'_i(t)}, \tag{3.3}$$

$$\alpha_i = \overline{X}_{i,N_i(t)} N'_i(t),\ \beta_i = (1 - \overline{X}_{i,N_i(t)}) N'_i(t). \tag{3.4}$$

[1] スコアの細かさを除いて Constant(a,b) と Constant$(1,b/a)$ はほぼ同じである．

$t_c = 0$ のとき，つまり，境界が変化していない時，この式は元の式と同じである．加えて，探索の始め，探索を安定化するため．境界はプレイアウト数が規定の回数に達するまで，固定した．実験では 10 とした．

3.4 実験

本節では，実験を通じて，次善手の偏りが MCTS の性能を著しく下げ，また，境界の調整により，性能の低下を軽減出来ることを示す．本実験では，境界の調整の効果は UCT にとどまらず，一般的であることを示すため，UCB1 だけでなく，KL-UCB 並びに Thompson sampling を MCTS に応用する．図中では，それぞれ “Score” と “Value” は Dynamic komi の Score Situational と Value Situational（2.8.1 節）を意味する．“MaxFrequency” は提案手法である最大頻度法を意味し，“Plain” は境界の補正なしの MCTS であり，“Static” は式 (3.1) のように Δ を最大化する静的な補正を行う．“Static” は境界の補正で得られる効果の上限を調べるための手法であり，プレイヤーには分からない，全終端節点でのスコアの情報を特別に利用して補正値が計算する．

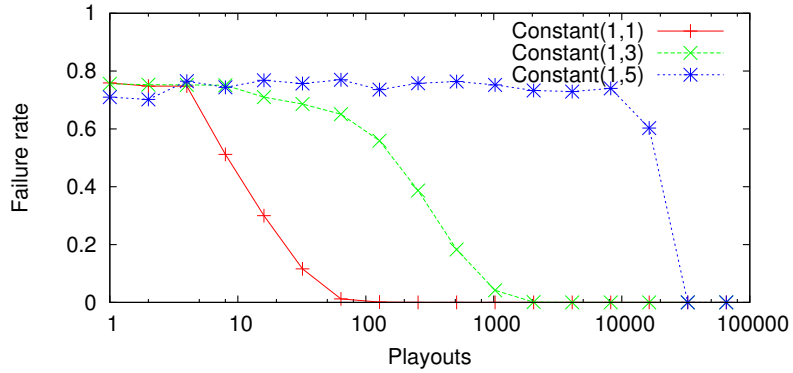
3.4.1 実験設定

文献 [42] の MCTS での性能の評価にならって，各手法の誤答率をもとに性能を計測する．誤答率はアルゴリズムが最善手を選べなかった割合である．それぞれの木に対し，それぞれのアルゴリズムが，各時刻 $t = 2^n$ で誤答したかを数え，誤答率を算出した．尚，最善手は最大訪問数をもとにして選ぶとした．木で平均を取ることで，それぞれのアルゴリズムの時刻 t での誤答率が得られる．

本節では，(分枝数，深さ) が (4,8) の結果を示す．本節と同様の結果が木の大きさを変えた場合にも報告されている [79]．全ての MCTS のアルゴリズムで葉節点は二回目に訪問した時に展開した．Dynamic komi のパラメータは予備実験によりプレイアウト数 625 時点での誤答率が最小になるように調整した．Score では，s は 0 から 1 まで 0.1 毎に，c は 0 から 30 まで 1 毎に試した結果，最善の s と c はそれぞれ 0.2 と 24 であった．Value では X_{lo} と X_{hi} はそれぞれ 0.05 と 0.9 であった．

2.6 節で説明したように，MCTS の探索木はプレイアウト数が増える毎に成長するため，いずれは探索木の葉が終端節点に達する．終端節点では，勝ち負けというゲームの理論値を利用できる．実験では，最大頻度法や Dynamic komi といった，勝ち負けの境界を補正する方法を用いたとしても，終端節点では勝ち負けの境界は常に 0 として，勝ち負けを決めている．これは探索結果をゲームの理論値へ収束させるために妥当な設定である．終端節点以外についても，MCTS-Solver [74] と同様に，ゲームの理論値も計算した．ゲームの理論値がすでに分かっている場合，UCT の選択の過程で勝ち（負け）確定の手は常に選ばれる（無視される）．また，根の負け確定の手が存在する場合，その手の訪問数が多くても，誤答率の計算の際に最善手としては選ばないとした．

(a) UCT



(b) UCT + MaxFrequency

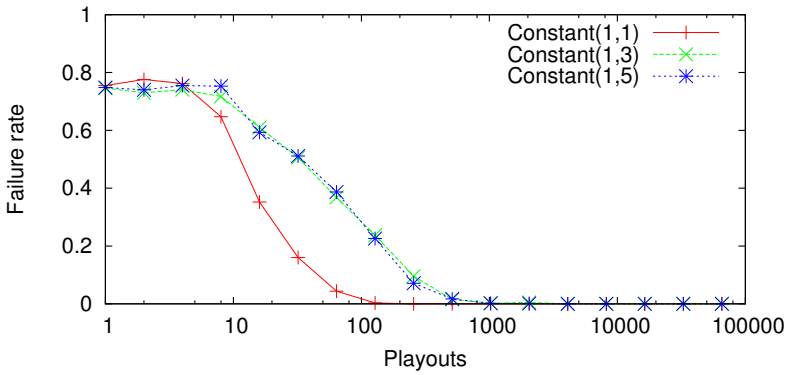


図 3.3: Constant での (a) 通常の UCT (Plain) と (b) UCT に最大頻度法を使用したものとの比較

3.4.2 Constant Trees

Constant での実験では，誤答率は異なる乱数シードでの 1 000 回の探索の平均をとって計測した．Constant(a, b) では偏りのパラメータ b により次善手の偏りが調整出来る．次善手の偏りに応じて，性能の低下が生じることを確かめた．

図 3.3(a) は UCT，つまり UCB1 に基づく MCTS の誤答率を図示したものである．本節の図は縦軸が誤答率で，横軸がプレイアウトの数である．誤答率は始め，0.75 であった [2]．それから，プレイアウトを重ねる毎に，基本的には下がり，終端節点まで探索木が成長する頃には 0 に収束した．しかしながら，誤答率の収束の速度は木の種類に大きく依存した結果となった．Constant$(1, 1)$ つまり，偏りの無い木ではおよそ 100 回プレイアウトで誤答率が収束した一方で，Constant$(1, 5)$ では， 10 000 回でもまだ誤答率が高くなった．Constant$(1, 5)$ は非常に難しい状況であることに注意されたい．それは，プレイアウトがほとんど全て Max プレイヤーの勝ちになるためである．木の深さが 8 の時の Constant$(1, 5)$ は最も極端な場合で，Min プレイヤーが，次善手（値 $+5$ の手）を一回でも選ぶと，Max プレイヤーが勝つ．それは，Max プレイヤーが次善手を選び続けたとしても，その値は -1 であり，Max プレイヤーの手番は 4 回しかないので，合計で -4 にしかならず，ゲームのスコアは正であることが確定するためである．

[2] 仮に，手を一様な確率で選んだ場合，誤答率の期待値は 0.75 であり，実験の結果は想定通りである．

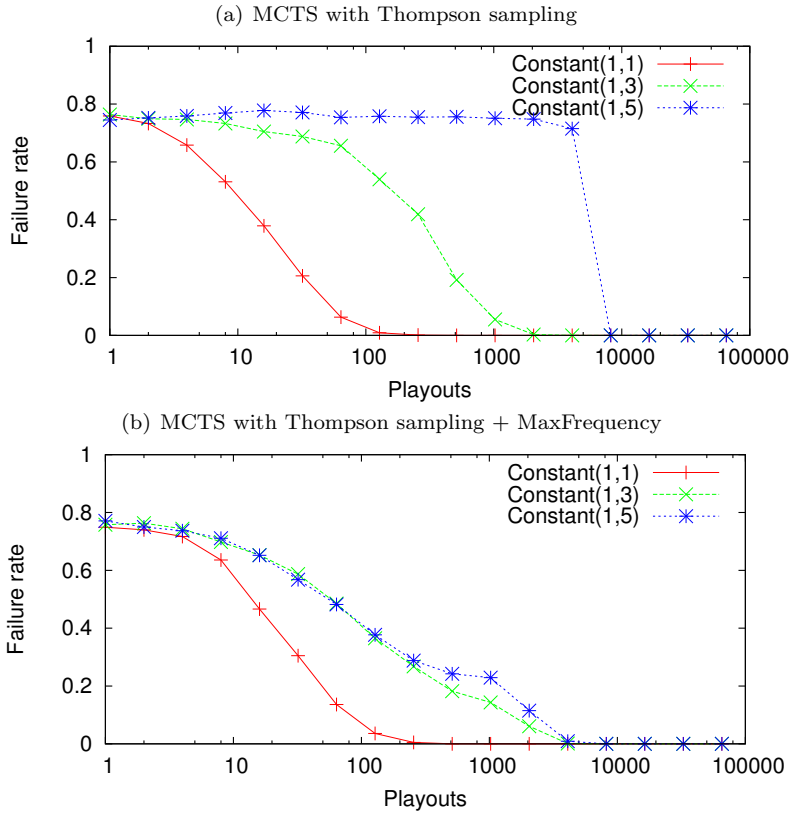


図 3.4: Constant での (a) MCTS に Thompson sampling を応用したもの (Plain) と (b) 最大頻度法を加えたものとの比較

UCT の性能は次善手の偏りが増えるほど落ちることを示した．

図 3.3(b) は UCT に最大頻度法を組み合わせた時の誤答率である．Constant$(1,3)$ や Constant$(1,5)$ では最大頻度法により改善が見られた．また，誤答率の収束までに必要なプレイアウト数はおよそ 1/8 であった．しかしながら，補正なしの場合の方が，Constant$(1,1)$ では多少良かった．それ故，偏った局面で上手くいくことと偏っていない局面で上手くいくこととの間にはトレードオフが存在すると言える [3]．ウェルチの t 検定（両側，有意水準 0.05）を行った結果，プレイアウト数が少ない場合とプレイアウト数が多く，誤答率がほぼ 0 になった場合を除き，多くのプロットポイントで誤答率の差は有意であるという結果となった．

図 3.4 は同じ実験を，Thompson Sampling を MCTS に応用したものに対して行った結果である．UCT での結果と同様に，MCTS の性能は Thompson Sampling を使っても，より偏っている場合に下がり，また，最大頻度法を用いることで，収束の速度は改善した．KL-UCB での結果（図 3.5）も同様の傾向であった．

Constant$(1,5)$ と Constant'$(1,13)$ で実験を行った．図 3.6(a) は Constant$(1,5)$ で拡張手法を組み合わせた UCT での誤答率を計測したものである．実験結果は，Plain は誤

[3]次善手の偏りが無い状況での最大頻度法の悪さは境界をより安定させることで，和らげることが出来る．また，トレードオフがあっても，最大頻度法は 複数エージェントで合議をして手を決める [48, 51] 際に役立つと言える．それは，エージェントの多様性が強さに貢献すると知られているためである [48].

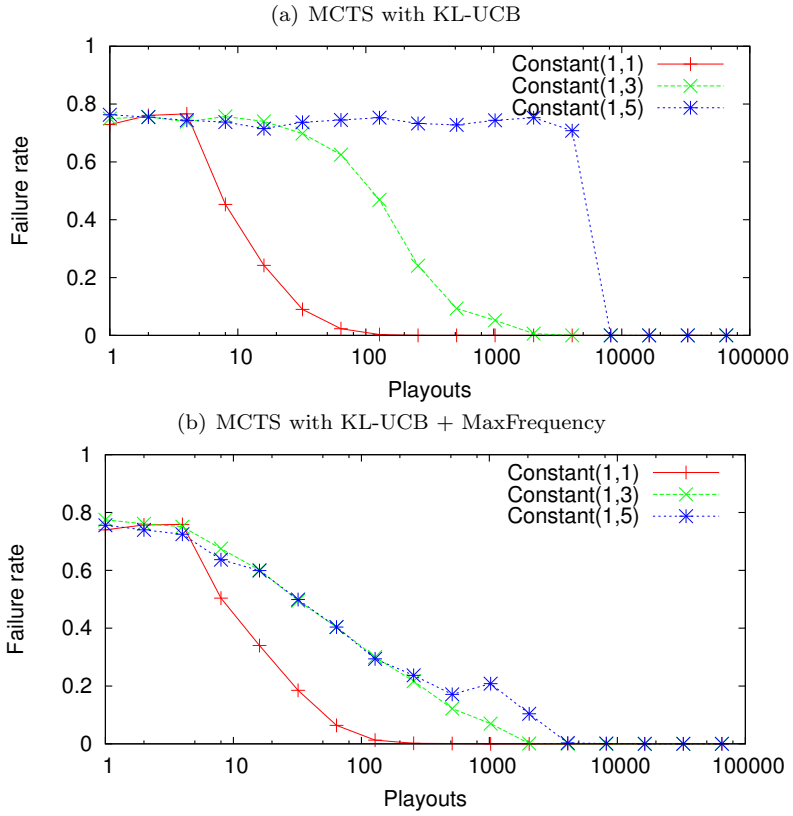


図 3.5: Constant での (a) MCTS に KL-UCB を応用したもの (Plain) と (b) 最大頻度法を加えたものとの比較

答率が 10000 回プレイアウトまで高いが，拡張手法により，誤答率が効果的に下がることを示した．また，MaxFrequency と Static は最も効果的であった．図 3.6(b) と 3.6(c) は同一の設定下，KL-UCB や Thompson sampling を MCTS に応用した場合の結果である．結果は UCT と同様に拡張手法を適用することで誤答率が下がった．但し，拡張手法を適用した場合の誤答率は概ね UCT の方が，KL-UCT や Thompson sampling を MCTS に応用したものより，低くなった．Score（図 3.6(a)）と Static（図 3.6(b) と 3.6(c)）で不安定な挙動が観測されている．探索木が成長した際に，終端節点での勝ち負けだけは，境界を 0 とした場合の勝ち負けであるのに対し，他の節点では異なる境界に基づく勝ち負けであることにより，この不安定さが引き起こされたと予想される．同様の比較を Constant'$(1, 13)$ にて行った．結果を図 3.7 に示す．MaxFrequency と Static は効果的であった一方で，Score と Value はほぼ全ての時刻で Plain よりも高い誤答率となった．これは図 3.2 で示したように，境界の位置により $\Delta = 0$ となるようなことが多かったためであると予想される．また，Plain の誤答率は収束前に上昇した．この理由は，偏った木のため，Plain では利得の推定値が全体的に高めであることと十分探索が進み，終端節点まで，探索木の一部が成長し，最善手の値が正しい値に収束する際にその値が大きく下がることの 2 点にあると推察される．

3.4.3 Random Trees

Incremental random tree での実験では，200 本の木を生成し，各木に対し，200 回，各アルゴリズムを実行した．そのため，誤答率は 40 000 回の探索の平均である．これは，文献 [42] と同じ設定である．

図 3.8 と 3.9 は Random$(4, q)$ において，次善手の偏りが UCT と Thompson sampling に基づく MCTS の性能を下げることを示している．ここでの q は偏りを示している．Constant と同様に，誤答率の収束の速さは図. 3.8(a) や 3.9(a) に示した通り，特に境界の調整をしない場合，木の種類に強く依存する．偏りが無い Random$(4, 0)$ では，およそ 1 000 プレイアウトで誤答率が 0 に収束したのに対し，Random$(4, 3)$ では，10 000 プレイアウトでもまだ誤答率は高かった．図 3.8(b) と 3.9(b) に示した通り，MaxFrequency は Plain よりも Random$(4, 2)$ 及び Random$(4, 3)$ で良い結果となった．その一方で Plain の方が Random$(4, 0)$ と Random$(4, 1)$ では良くなり，Constant で観測されたトレードオフが Random でも観測された．UCT で拡張手法あり，なしの誤答率の差はプレイアウト数 $[16, 1024]$ の時，統計的に優位（有意水準 0.05 で両側のウェルチの t 検定）であった．プレイアウト数が大きく，誤答率がほぼ 0 に収束した場合では，その差は有意では無かった．KL-UCB を MCTS に応用した結果（図 3.10）は同様であった．

各拡張手法を用いた場合の様々な木での誤答率を比較するため，Random$(4, 3)$ での結果を図 3.11 に示した．MaxFrequency は安定して，改善したものの，Constant$(1, 5)$ での改善ほど劇的ではなかった．また，Value は効果的であった．Score は Plain よりも高い誤答率となることもあった．

3.5 まとめ

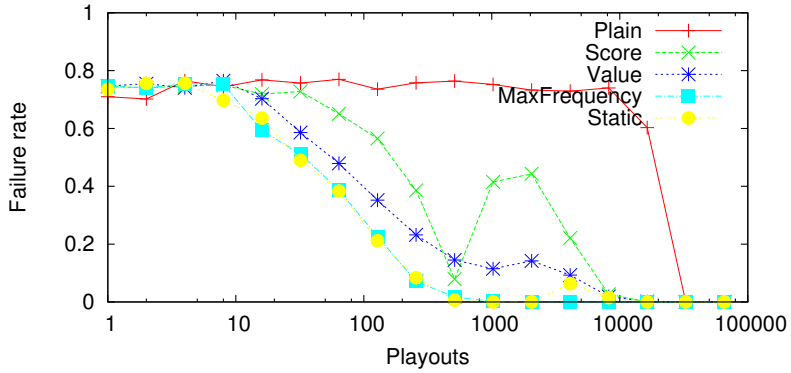
本節では「次善手の偏り」という特徴が MCTS の性能を劇的に下げることを示した．この特徴量は，最善手を逃した場合の損失の片方のプレイヤーともう片方のプレイヤーとの差である．実験では，Constant と Random と大きく分けて二種類の木を使った．そして，UCB1 を MCTS に応用した UCT だけでなく，KL-UCB や Thompson Sampling を MCTS に応用したものも利用した．KL-UCB と Thompson Sampling は多腕バンディットで UCB1 よりも効果的であると確認された手法である．それらの木に対して，次善手の偏りが大きいほど，UCT だけでなく，他の MCTS の性能も下がることを観測した．

また，本章では，囲碁等のゲームのスコアをエージェントが利用出来るゲームを対象として，プレイアウトでの勝ち負けの境界を調整する手法の一つである最大頻度法を提案し，それにより，性能の低下が和らぐことが明らかにした．この手法は最善手と次善手の利得の期待値の差を最大化することに基づく手法である．実験を通じて，最大頻度法により，Constant と Random の木の探索での性能の低下が抑えられることを示した．

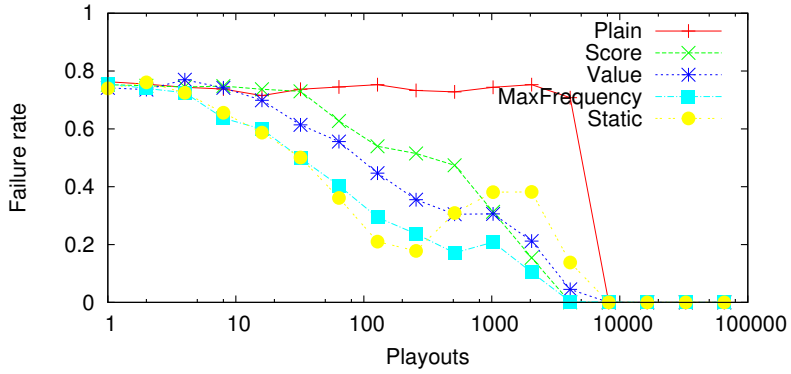
Random の木では，一般に，平均的なスコアが次善手の方が高い場合もあり得るた

め，最善手と次善手の利得の期待値の差を最大化することは難しい．そのような状況でも最大頻度法により改善することを示した．但し，その度合いは Constant の木ほど劇的な改善では無く，また，最大頻度法でも，偏りによる性能の低下は観測された．そのため，次善手の偏りという特徴を持つ局面は MCTS には難しい局面であると言える．

(a) UCT



(b) MCTS with KL-UCB



(c) MCTS with Thompson sampling

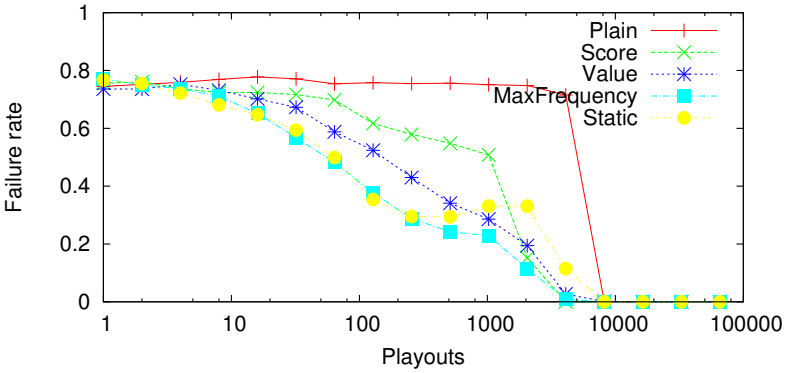


図 3.6: Constant(1,5) での誤答率の比較（拡張手法あり，なし）

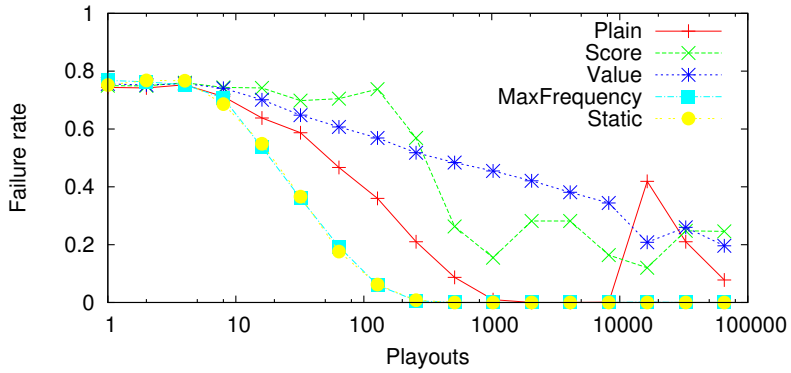
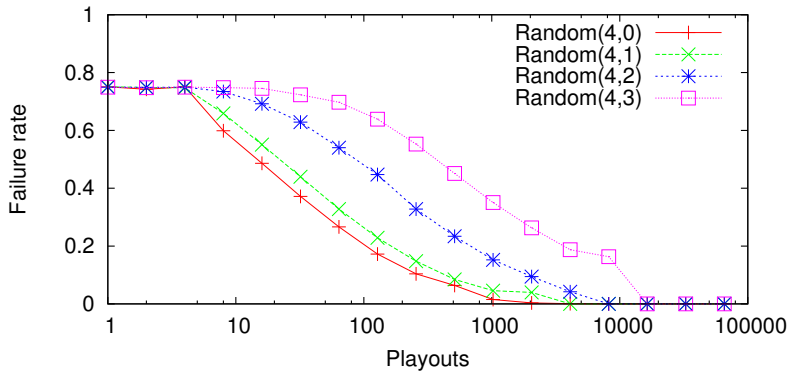


図 3.7: UCT, Constant'(1,13) での誤答率

(a) UCT



(b) UCT + MaxFrequency

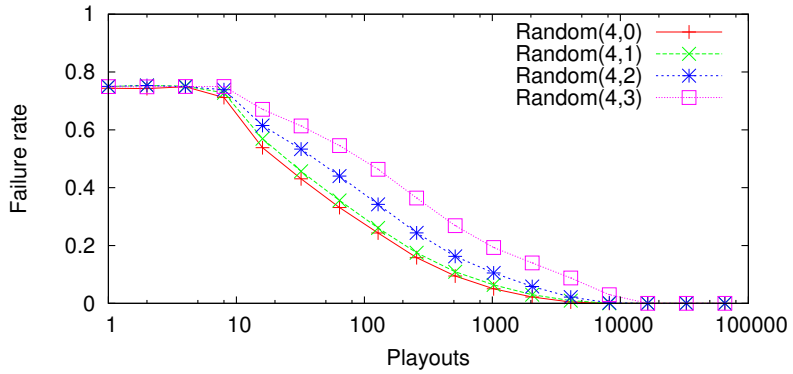
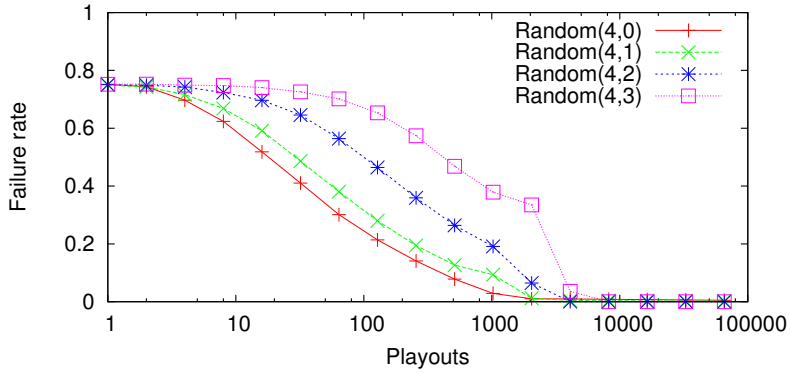


図 3.8: (a)UCT と (b) 最大頻度法を使った UCT の Random での比較.

(a) MCTS with Thompson sampling



(b) MCTS with Thompson sampling + MaxFrequency

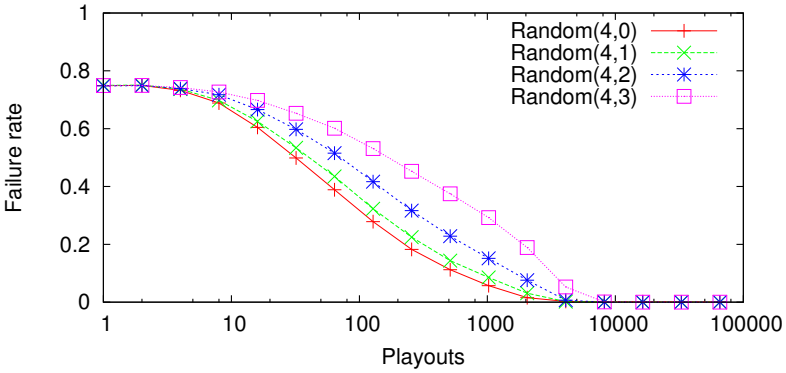
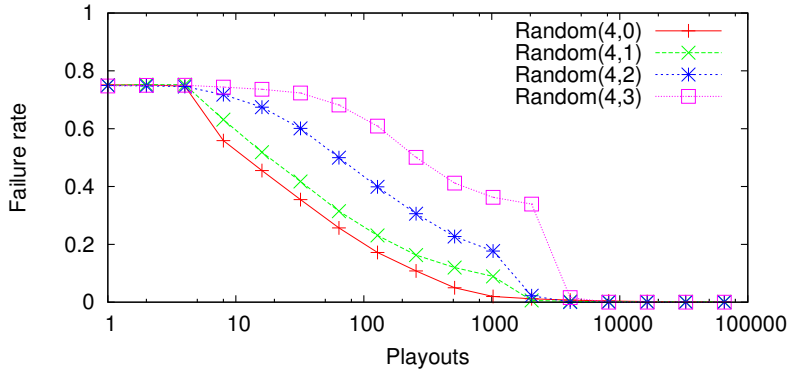


図 3.9: (a)Thompson sampling を使った MCTS と (b) 最大頻度法を組み合わせたものとの比較

(a) MCTS with KL-UCB



(b) MCTS with KL-UCB + MaxFrequency

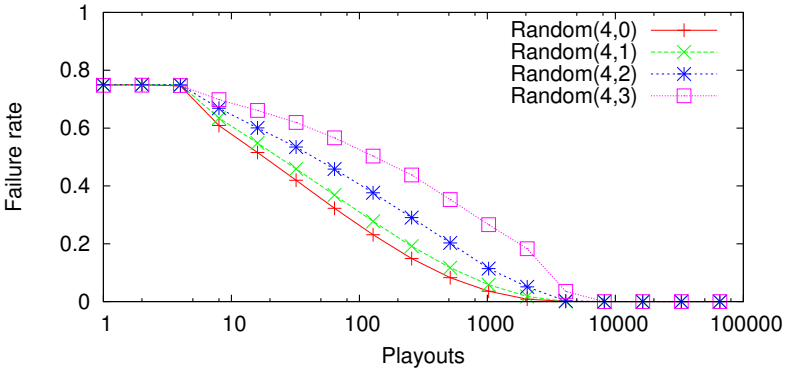
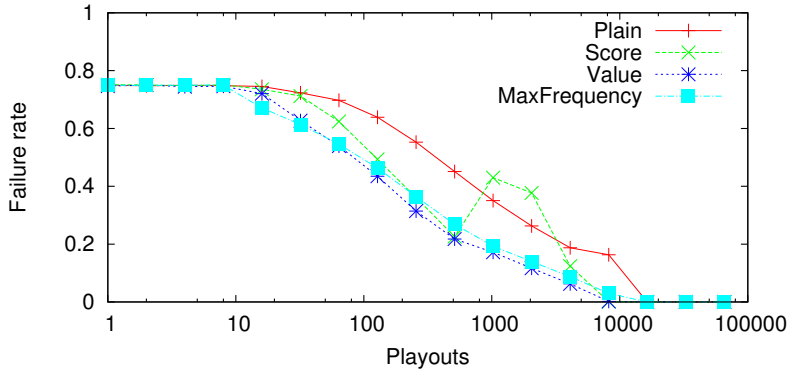
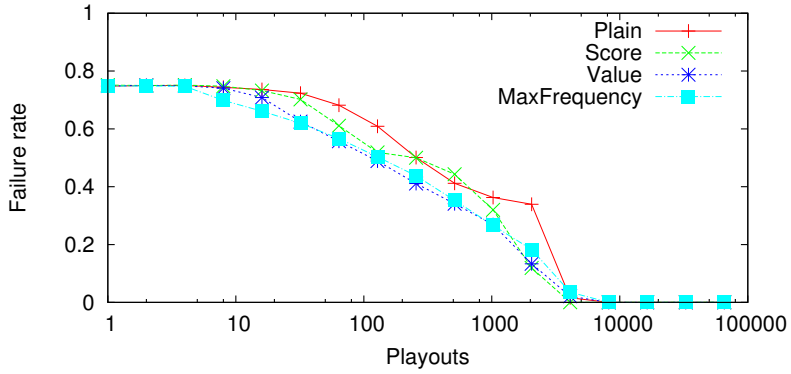


図 3.10: (a)KL-UCB を使った MCTS と (b) 最大頻度法を組み合わせたものとの比較

(a) UCT



(b) MCTS with KL-UCB



(c) MCTS with Thompson sampling

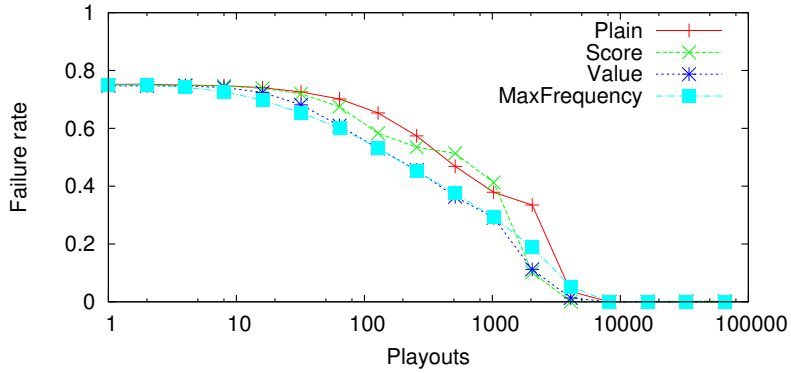


図 3.11: Random(4, 3) での各 MAB アルゴリズムに基づく MCTS の比較

第4章　分布を使ったハイブリッド MCTS

モンテカルロ木探索（MCTS）の一種，UCT は，多腕バンディット問題（MAB）における累積的な利得を最大化することを目的とした UCB1 アルゴリズムに基づいている．しかしながら，MAB の最近の研究で，累積的な利得を最大化することと最善のアームを見分けることのためには，それぞれ異なるアルゴリズムを使う方が効率的であることが明らかになっている．本章では，MCTS において，最善手を見つけるという観点から，平均だけでなく，分布を使った手法を提案する．提案手法はハイブリッドな MCTS で，木の浅い方ではネガマックスの考えに基づいた更新を行い，残りは UCT を適用する．実験により，提案手法は，一様な分枝数，深さの木を除き，UCT を上回ることが確かめられた．

尚，本章の内容は発表済みの論文 [34] に加筆，編集したものである．

4.1　背景

UCT は 2.6 節で論じたように，累積リグレットの最小化のアルゴリズムである UCB1 に基づいている [3, 42]．これにより，平均利得が良い節点からのシミュレーションが優先されるため，各節点の利得の期待値を平均利得で推定するのに相応しいと言える．しかしながら，ゲームをプレイする状況では，利得の期待値を推定するよりも，根での最善手を見つけることが重要である．それらの目標は相互に関係しているものの，異なる点がある．MTD(f) の研究 [57] では $\alpha\beta$ 探索の文脈でこの違いについて扱った．同様の観点から，MAB での累積リグレットの代わりに単純リグレットに基づくアルゴリズムを MCTS に応用することで性能は改善することが示されている [46, 53, 70]．しかしながら，木探索では直接的に単純リグレットを最小化することは難しい．

本章では，利得の分布を利用した新しい MCTS を提案する．提案手法は既存研究の２つのアイデアの組み合わせに基づいている．１つは 2.9 節で紹介した，探索木を根付近（main tree）とそれ以外（extended tree）に分けた，ハイブリッド MCTS である [53, 70]．もう１つは，利得の分布に基づく，最善手の信頼度合いの評価である [7]．提案手法では，extended tree を UCT で探索することで，main tree の葉の利得の分布を得て，分布に基づき，最善手の信頼度合いを評価する．そのようにして，最善手の信頼度合いに最も影響を与えるという意味で，main tree の最も重要な葉が分かる [7]．提案手法は，最も重要な葉を UCT で探索をするということを繰り返し，最善手の判別を

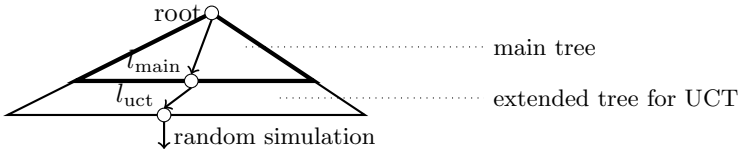


図 4.1: 提案手法で扱う探索木の二つの部分

目指した手法である．

4.2 信頼度合いの修正に基づく探索

本節では，徐々に根での選択の信頼度合いを修正していく新しいハイブリッド MCTS を提案する．提案手法は図 4.1 のように探索木を二つに分けて，primary strategy のために main tree を保持し，sub-strategy のために extended tree を保持する．そして，最良優先探索の様に，逐次的に探索木を展開し，評価を改訂する．概要は図 4.2 に示した．ステップ 1 から 4 は primary strategy に従い，ステップ 2 (a) から 2 (d) は sub-strategy に従う．提案手法では sub-strategy として UCT を用いる．まずは 4.2.2 節で紹介する primary strategy を使い，main tree の l_{main} を決める．それから UCT に従い，*budget* 回（通常は 1 回）プレイアウトを行う．新たに作られた葉は始め，extended tree（UCT の探索木）に含まれるが， 4.2.3 節で説明する状況を満たせば，main tree に含まれるようになる． 2.9 節の表 2.1 の下半分に primary strategy と sub-strategy をまとめた．

4.2.1 利得の分布の更新

提案手法では，利得として，勝ち，引き分け，負けに対応する 3 種類 $\{0, 0.5, 1\}$ を用い，探索木のそれぞれの節点で，サイズ 3 の配列で利得の分布を保持する．分布の更新方法として，main tree と extended tree での異なる方法を導入する．この更新の枠組みを hybrid backup (HB) と呼ぶ．

main tree の各節点では，利得の分布の更新方法として，ネガマックスの考え方に基づいた Baum の方法 [7] を採用する．この方法では，内部節点 j の確率分布は子孫の節点の確率分布に従って決まる．$\underline{c}^{(i)}(x)$ と $\overline{c}^{(i)}(x)$ をそれぞれ，利得の理論値が x 以下，以上である確率とする．すなわち，$p_i(x)$ を節点 i の利得の理論値が x である確率とすると

$$\underline{c}^{(i)}(x) := \textstyle\sum_{k:k\le x} p_i(k),\ \overline{c}^{(i)}(x) := \sum_{k:k\ge x} p_i(k) \tag{4.1}$$

である．特に $\underline{c}^{(i)}(x)$ は $p_i(x)$ の累積密度分布（CDF）である．逆に，確率 $p_i(x)$ は簡単に $\underline{c}^{(i)}(x)$ や $\overline{c}^{(i)}(x)$ から計算できる．main tree の節点 j の CDF $\underline{c}^{(j)}(x)$ は

$$\underline{c}^{(j)}(1-x) = \textstyle\prod_{c\in\text{successor}(j)} \overline{c}^{(c)}(x) \tag{4.2}$$

で更新する．尚，分布は兄弟間で独立と仮定する．

1. 葉の選択： 4.2.2 節で定義した方法に基づき，最も価値のある main tree の葉 l_{main} を選ぶ．

2. 評価： 葉 l_{main} の評価を改善するため，以下のように，l_{main} を根とした UCT のプレイアウトを *budget* 回行う：

   (a) 選択： 最も価値のある葉 l_{uct} を l_{main} から UCB 値最大の子を逐次的に選ぶことで特定する．

   (b) 展開： 2.6 節で記した，MCTS の展開の状況を満たすと，葉 l_{uct} を展開し，新たな葉の中から一様な確率で l_{uct} を選ぶ．

   (c) 評価： 葉から始めるシミュレーションによって，葉を評価する．

   (d) 更新： シミュレーションで得た利得を l_{main} と l_{uct} の間の各節点でヒストグラムとして保持する．

3. 展開： 評価の後，UCT の木の節点で， 4.2.3 節で記す条件を満たすものがあれば，main の木に組み入れる．

4. 更新： l_{main} と根との間の各節点利得の分布を式 (4.2) で更新する．

図 4.2: 提案手法の概要

extended tree の各節点では利得 $\{0, 0.5, 1\}$ のヒストグラムを保持し，それを正規化し利得の確率分布とする．プレイアウトが終了後，その結果に対応して，利得のヒストグラムを更新する．そして，main tree の葉（extended tree の根）l での利得の確率分布を，l での利得の理論値の主観確率分布 $p_l(\cdot)$ とする．

提案手法では，main tree でのネガマックスに基づく更新をすることで，UCT のように頻度分布を更新する場合と比べて，利得の推定値が速く正しい値に収束すると期待できる．

4.2.2 探索の手法

本節では，main tree の l_{main} を選ぶための primary strategy を紹介する．l_{main} は，図 4.2 のステップ 1 で示した様に，UCT 探索を開始する節点である．根を R とする．節点 m_0 を現在までの探索によって推定された，根での最善手 $\arg\max_{c \in \text{successor}(R)} \hat{\mu}_c$ とする．$\hat{\mu}_c$ は節点 c の分布から計算される利得の平均，つまり $\sum_x (1-x) p_c(x)$ である [1]．根で手番のプレイヤー視点での，節点 R と m_0 の利得の推定値をそれぞれ $\hat{\mu}'_R$ $(= 1 - \hat{\mu}_R)$ とと $\hat{\mu}'_{m0}$ $(= \hat{\mu}_{m0})$ とする．不確かさ U を根 R の利得の推定値と m_0 の利得の推定値との差分，つまり

$$U := \hat{\mu}'_R - \hat{\mu}'_{m_0} \tag{4.3}$$

[1] 節点 c の親節点で手番となるプレイヤーにとっての利得であるため，x ではなく $1 - x$ である．

とする．一般に $U \geq 0$ である．

U は，下記の理由により，探索を続けることの効用を表していると解釈出来る．$\hat{\mu}'_R$ は，main tree の全ての葉の真の利得がそれぞれの分布に従って独立に決まると仮定した場合の，根 R の利得の理論値の期待値である．また，$\hat{\mu}'_{m_0}$ は，同様の仮定での推定最善手 m_0 の理論値の期待値である．言い換えると，　$\hat{\mu}'_R$ は葉の利得の理論値が決まってから，根の利得の理論値（つまり根の最善手の利得の理論値）の期待値である．そして，$\hat{\mu}'_{m_0}$ は先に最善手 m_0 を決めてから，葉の利得の理論値を決まったとした場合の m_0 の理論値の期待値である．つまり，U は，葉の分布に従って利得が決まると仮定した場合に，葉の利得の理論値が全て明らかになった後に最善手を選んだ時の期待値と，全て明らかになる前に推定最善手を選んだ時の期待値の差ということである．従って，U の値が小さいことは，推定最善手に対して，本当に最善であると確信出来ることに対応する．また，探索の目的は最善手を判別することであるので，その意味では，U の値が大きいほど探索を続けるべきであると解釈出来る．

提案手法では，根の各手の利得の分布の分散を最小化するのではなく，U を最小化することで最善手を見分けることを目指す．目標は，最善手を見分けるために，最も優先して探索すべき葉 l を見つけることである．提案手法において，ステップ 1 で葉 l を選んだとする．U_l を葉 l を UCT で探索（ステップ 2）後の U の値とする．この値は実際にステップ 2 が完了するまでは不明である．U_l とステップ 2 の前の U 値との関係性について，下記が言える．もし，$U_l \ll U$ なら，それは，他の手と最善手が明確に区別出来るようになったということを意味する．もし，$U_l \gg U$ なら，それは，今までの U の値が不正確で，修正する必要があると分かることを意味する．そのため，文献 [7] の議論と同様に，どちらの場合も l は高い優先度を持つと解釈する．

HB+E$^{\mathbf{Terminal}}$

葉 l を選択する妥当な方法の一つは，差の絶対値 $|U_l - U|$ の期待値を考え，最大の l を選ぶことである．この方法では，葉 l を訪問すると l の利得の理論値 r が明らかになると仮定する．$\hat{U}_l^{=r}$ は葉 l が終端節点で，その理論値が r だと確定した時の U_l とする．他の葉の分布は l の探索により，変化しないと仮定すると，$\hat{U}_l^{=r}$ を直接的に計算出来る．そして，探索後に l の理論値が r だと確定する確率を l の利得の理論値が r である主観確率 $p_l(r)$ として，HB+E$^{\mathrm{Terminal}}$ では，絶対値の差 $|\hat{U}_l^{=r} - U|$ をその確率で重みをつけて和，つまり期待値をとる．そして，その値が最大の葉を選ぶ．

$$\arg\max_l \sum_{r\in\{0,0.5,1\}} p_l(r)|\hat{U}_l^{=r} - U| \tag{4.4}$$

この方法は，MCTS による分布に代えて，予め学習した評価関数による分布を使うと QSS [7] と等価である．

HB+E$^{\mathbf{Expected}}$

通常のゲームでは，探索中に終端節点を訪問することは，終端でない節点を訪問ことと比べて稀であると想定されている．この想定にもとづき，葉 l を訪問し，l からプレイアウトした結果，利得 r が出る確率を $p_l(r)$ とし，l の理論値は不明と仮定するモデルを提案する．葉 l の分布が新たなプレイアウト結果 r により変化し，その結果として変化した U_l の値を $\hat{U}_l^{+r}$ とする．HB+E$^{\mathrm{Expected}}$ では，差 $|\hat{U}_l^{+r} - U|$ の期待値が最大の葉 l を選ぶ．式で表すと以下となる．

$$\arg\max_l \sum_{r\in\{0,0.5,1\}} p_l(r)|\hat{U}_l^{+r} - U| \tag{4.5}$$

HB+E$^{\mathbf{Robust}}$

HB+E$^{\mathrm{Robust}}$ は，各葉 l で U_l を最大化するという意味で最も運の悪いプレイアウト結果を考え，それに基づき葉を決める main strategy である．U_l が現在の U より大きいことは，現在の推定の誤差が大きいことを示唆する．この考えを発展させ，HB+E$^{\mathrm{Robust}}$ では $\hat{U}_l^{+r}$ を全て考え，U_l を最大化する l を選ぶ．式で表すと

$$\arg\max_l \max_{r\in\{0,0.5,1\}} \hat{U}_l^{+r} \tag{4.6}$$

である．全ての葉 l において，$\max_{r\in\{0,0.5,1\}} \hat{U}_l^{+r} \geq U$ であるため，

$$\arg\max_l \max_{r\in\{0,0.5,1\}} \hat{U}_l^{+r} = \arg\max_l \max_{r\in\{0,0.5,1\}} |\hat{U}_l^{+r} - U| \tag{4.7}$$

である．U の値が増え得る l を選んで，数回のプレイアウト結果で実際に U の値が増えるなら，現在の U の値は不安定と言える．この strategy は U の値に大きな影響を与える節点を先に探索することで，不安定さを改善すると期待される．U_l^{+r} は根での推定最善手 m_0 に対して，最善である確信度合いが下がるほど大きな値になる．そのため，推定最善手 m_0（推定最善以外の手）の子孫の葉では r が 0 (1) の時に最大化される．ここでの r は根でのプレイヤー視点の利得である．葉 l で手番のプレイヤーが根で手番のプレイヤーと異なるなら，根での利得 r は葉では $1-r$ である．

HB+BayesUCT

HB+E$^{\mathrm{Terminal}}$，HB+E$^{\mathrm{Expected}}$，HB+E$^{\mathrm{Robust}}$ は最終的に不確かさ U を最小化することを目標として葉 l を選択しているが，代わりに，BayesUCT [68] を適用して葉 l を選ぶことも出来る．HB+BayesUCT は根から葉まで Bayes-UCB の値 (2.13)

$$\text{Bayes-UCT1} := \hat{\mu}_i + \sqrt{2\ln t/N_i(t)},\ \ \text{Bayes-UCT2} := \hat{\mu}_i + \sqrt{2\ln t}\sigma_i, \tag{4.8}$$

が一番大きな節点を選ぶ．2 種類の値があるが，本研究では，それぞれ primary strategy に応用した．

4.2.3 実装の詳細

提案手法では，extended tree の節点が十分に探索されると図 4.2 のステップ 3 で記したように main tree の節点になる．実験では，main tree の全ての葉 l の子節点は l を 20 回以上訪問し，かつ，l の子節点について，最小でも 2 回以上訪問した時のみ，子を全て同時に main tree の節点とした．訪問数の条件は特に HB+$\mathrm{E}^{\mathrm{Terminal}}$ や HB+$\mathrm{E}^{\mathrm{Expected}}$ では必要である．もし，仮に，ある葉節点の分布で頻度が正となる利得が 1 つしかない場合，これらの strategy では，そのような葉節点は最小の優先度を持ち，ほとんど選ばれないということが起こりえる．例えば，もし，節点 l でのプレイアウトの結果が常に勝ちならば，$\mathrm{E}^{\mathrm{Terminal}}$ と $\mathrm{E}^{\mathrm{Expected}}$ では同じプレイアウト結果が出ると見なされ，その結果 $U_l = U$ となる．HB+$\mathrm{E}^{\mathrm{Robust}}$ はこのような問題は生じない．尚，事前分布（例えば，予め頻度分布を増やしておく等）を考えることで上記の問題を避けることが出来るが，その場合，増やす量が微小な場合，上記の問題はほとんど解決しない一方で，増やす量が多い場合，探索自体に大きな影響があると予想される．適切な増やし方は不明であり，本研究では，事前分布を用いる代わりに訪問数の条件を設けた．

HB+$\mathrm{E}^{\mathrm{Terminal}}$，HB+$\mathrm{E}^{\mathrm{Expected}}$，及び HB+$\mathrm{E}^{\mathrm{Robust}}$ では各葉の優先度の計算は，それぞれの葉の分布の変化の影響を分布の更新時に合わせて計算することで，効率化できる [7]．提案手法でもこれに従った．優先度の計算量は main tree の節点の数に比例する．しかしながら，main tree の節点の数は，extended tree のものよりもずっと少ないため，計算コストは普通は目立たない．また，primary strategy と UCT で計算コストは budget によって調整できる．budget は l_{main} を選ぶ毎に行うプレイアウトの数である．budget が 2 以上なら，U_l を複数回のプレイアウト後に見積もることになる．HB+$\mathrm{E}^{\mathrm{Robust}}$ では，budget が 2 以上でも，U_l は計算コストが増えることなく見積もれる．それは，HB+$\mathrm{E}^{\mathrm{Robust}}$ の性質から，各葉 l で $\hat{U}_l^{+r}$ の最大値を与える利得 r は budget によらず同じであるためである．加えて，HB+$\mathrm{E}^{\mathrm{Terminal}}$ ではプレイアウトが複数回でも 1 回でも U_l は同じと見なされるので，HB+$\mathrm{E}^{\mathrm{Robust}}$ と同様にして，複数回プレイアウトでも計算コストは増えない．しかしながら，HB+$\mathrm{E}^{\mathrm{Expected}}$ では U_l は複数回プレイアウトに応じて計算する必要があり，計算コストが増えるという欠点がある．

勝ち負けが確定した節点の扱いについては，MCTS-Solver [74] に倣った．main tree では，ネガマックスに基づく分布の更新を行うので，勝ち負けが確定した節点の利得の分布は自然と，正しい値に収束する．しかしながら，extended tree では UCT を用いているため，勝ち負けが確定したかの計算は必要である．また，提案手法の main tree では，引き分けが確定した節点は選ばれないが，その一方で UCT では，平均利得の安定のため，選ばれ得る点に注意されたい．

4.3 実験

様々なアルゴリズムの性能を比較するため，incremental random tree で実験を行った．全ての木は基本的に一様な幅と深さを持つように生成したが，幾つかのサブツリーは幅と深さにばらつきを持たせるため，枝刈りした．枝刈りの過程では，それぞれ内部節点を確率 P で終端節点にした．そして，その節点より下のサブツリーは枝刈りされる．尚，そのようにしても根の利得の理論値は変わらない．本研究では，スコアの偏りのパラメータ Bi を導入し，ゲームにバリエーションを持たせるため，全ての葉のゲームのスコアに Bi を加える．それぞれの辺の値が整数のため，Bi が非整数値（例えば 0.5）の場合，終端節点のゲームの結果は引き分けには成りえず，勝ち負けだけになる．

4.3.1 誤答率の比較

4.2 節で導入した HB+E$^{\text{Terminal}}$, HB+E$^{\text{Expected}}$, HB+E$^{\text{Robust}}$, HB+BayesUCT1, 及び HB+BayesUCT2 という 6 つのアルゴリズム（ 提案手法の 5 つの変種と UCT）で実験を行った．実験では budget は 1 で固定し，全てのアルゴリズムで MCTS-Solver [74] を使い，勝ち負けの確定を扱った．探索のはじめ，全ての手の利得の平均値は同じため，推定最善手をランダムに初期化した．推定最善手は，現在の推定最善手の平均値を他の手の平均値が真に上回った時のみ，置き替える．

ゲーム木自体は予め用意するが，それぞれのアルゴリズムの探索木は根とその子のみから始まり，徐々に節点を増やしていく（ 4.2 節）．(分枝数，深さ) が (4, 12) と (8, 8) の 2 つの設定で実験を行った．加えて，終端確率 P は 0, 0.1，0.2 とした．それぞれの木の設定で 400 本の木を作り，それぞれの木に対し，各アルゴリズムで 200 回探索した．

本実験では，文献 [42] にならい，誤答率で各アルゴリズムの性能を計測した．誤答率とは，最善手を選べなかった割合である．

結果を表 4.1 にまとめた．提案手法 HB+E$^{\text{Robust}}$ は終端確率 P が 0 でない時に，一番良い結果となった．その一方で，UCT は P が 0 の時に最も良い結果となった．これらの結果は，一様な分枝数でない時，trap という負けの手がある時に UCT の性能は下がるという既存研究 [58, 59] の知見に矛盾しない．HB+E$^{\text{Robust}}$ と UCT の性能はスコアのバイアスが木に加わると改善した．その一方で BayesUCT の性能は他と比べて良くなかった．ヒストグラムと BayesUCT の原論文で用いられているベータ分布の違いが性能に大きな影響を与えたと予想される．また，原論文では探索木は成長させない設定である一方で，本研究では探索の間に探索木が成長するということもまた大きな違いである．図 4.3 の結果は，終端確率が正の時，HB+E$^{\text{Robust}}$ は UCT や他の手法よりも誤答率が速く減ることを示している．

表 4.1: 4000 プレイアウト時の誤答率．B は分枝数，D は最大深さ，P は終端確率，Bi はスコアのバイアスである．それぞれの設定で，誤答率最小の場合赤字で表示した

B - D	P	Bi	Hybrid Backup (HB)					UCT
			$E^{Terminal}$	$E^{Expected}$	E^{Robust}	BayesUCT1	BayesUCT2	
4-12	0.0	0	0.079025	0.078487	0.078575	0.037750	0.129050	0.013088
		0.5	0.060987	0.058975	0.029712	0.057512	0.085550	0.016987
	0.1	0	0.001275	0.001212	0.000188	0.005525	0.017175	0.040137
		0.5	0.000163	0.000238	0.000000	0.001225	0.000988	0.000213
	0.2	0	0.000000	0.000000	0.000000	0.000650	0.000450	0.002288
		0.5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	0.0	1	0.050350	0.046487	0.025087	0.066387	0.145387	0.023013
	0.1		0.001213	0.001225	0.000325	0.034000	0.028837	0.007738
	0.2		0.000000	0.000000	0.000000	0.001088	0.002325	0.000025
	0.0	2	0.008500	0.008138	0.001512	0.003912	0.067475	0.000000
	0.1		0.000113	0.000100	0.000000	0.001438	0.007950	0.000013
	0.2		0.000000	0.000000	0.000000	0.000000	0.000450	0.000000
8-8	0.0	0	0.264350	0.251387	0.301200	0.777625	0.174887	0.016825
	0.1		0.090300	0.074125	0.046825	0.093362	0.068025	0.098437
	0.2		0.007125	0.004425	0.001187	0.036912	0.014563	0.067225

表 4.2: プレイアウト一回当たりにかかる平均時間（ミリ秒）

	Budget	P = 0.0	P = 0.1	P = 0.2
HB+E^{Robust}	1	0.151095	0.079921	0.041580
	10	0.021505	0.015718	0.011130
	20	0.016167	0.012834	0.008930

	Budget	P = 0.0	P = 0.1	P = 0.2
UCT	1	0.012968	0.008496	0.004249

4.3.2 Budget を増やすことでの計算の効率化

実験で最も良かった HB+E^{Robust} の計算コストについて調べた．primary strategy として HB+E^{Robust} を使った場合，UCT と比べて計算コストが大きい．しかしながら，main tree の葉を一回選ぶ毎のプレイアウト回数である budget を大きくすることで，プレイアウト一回当たりの計算コストを減らすことが出来る．その一方で，プレイアウト数当たりの性能が下がる可能性もある．本節では，budget を増やした時の性能を明らかにする．

実験では，かかった時間全体を計測し，プレイアウト数で割ることにより，プレイアウト一回辺りの平均的な消費時間を算出した．分枝数－深さが 4 － 1 2 の木に対し 4000 プレイアウトを行った．つまり，main tree で 4000/budget 回葉を選ぶということである．ハイブリッドアルゴリズムでの UCT 探索中に，extended tree の根すなわち main tree の葉の理論値が分かる場合がある．そのような場合，それ以上 UCT 探索しても意味が無いため，プレイアウト数が budget に達する前に，UCT 探索は止めて，main tree

の分布を更新する．実験時の環境として，AMD Opteron Processor 6274, 2.2 GHz を使用し，Linux 上で実行した．

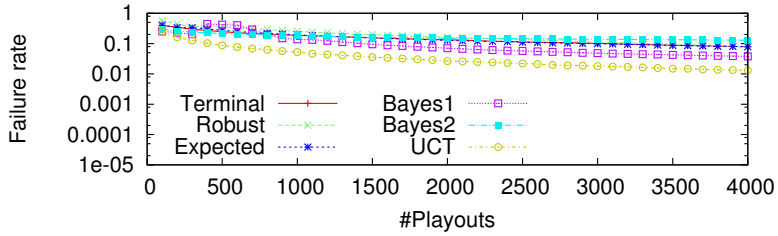
図 4.4 に示したように，budget を増やすことで，誤答率は少し増加した．しかしながらその差は限定的であった．表 4.2 に HB+E^{Robust} と UCT でのプレイアウト一回の平均消費時間を示した．計算効率は budget を増やすことで，改善した．UCT は budget 20 での HB+E^{Robust} よりも速いが，プレイアウト一回当たりおよそ 1 ms かかるという典型的な状況ではその差はほとんど無視できる．

4.4 まとめ

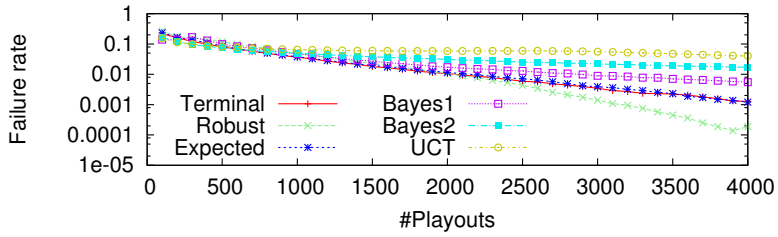
本節では，最善手に対する確信度合いに基づく，探索を続けることの効用から，シミュレーションを開始する節点を決めるという新しい MCTS を提案した．最善手に対する確信度合いは各節点の利得の分布に基づき推定される．提案手法では探索木を根付近の探索木 main tree とそれ以外の探索木 extended tree に分け，それぞれの探索木で別のアルゴリズムを使う．main tree での各節点の利得の分布を，main tree の葉の利得の分布からネガマックスの考え基づく更新をして推定し，main tree の葉の分布は extended tree を UCT で探索することによって推定する．各イテレーションでは，最善手の推定に資する main tree の葉を UCT で探索する．実験では，提案手法は一様な深さで一様な分枝数の木を除いて，UCT を超える性能を示した．提案手法の幾つかの strategy の内，最も運の悪いプレイアウト結果を想定して葉の優先度を決める方法が最も良い結果となった．

一様な木では UCT よりも性能が悪く，そうでない場合に良いということは，本質的なトレードオフが存在する可能性もあるが，提案手法では，終局の情報が無いとシミュレーションをどこから開始するのかの割り振りは上手く行かない一方で，終局の情報が有るとそれを活かして割り振りが上手く出来るという解釈が可能である．それは逆に，UCT と MCTS-Solver では，提案手法ほど，終局の情報を上手く活かせていないということであり，改善の余地があるということでもある．5 章では，終局の情報が手に入る場合に絞って，UCT と MCTS-Solver の改善策を議論する．

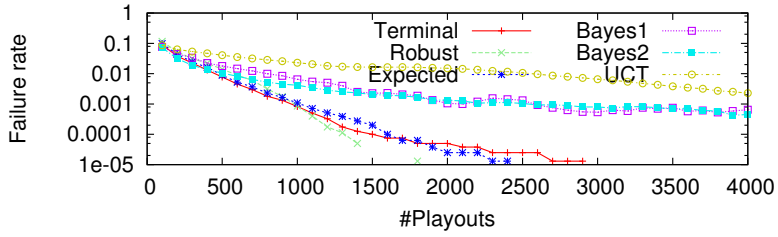
(a) （分枝数，深さ）$(4, 12)$，終端確率 0.0



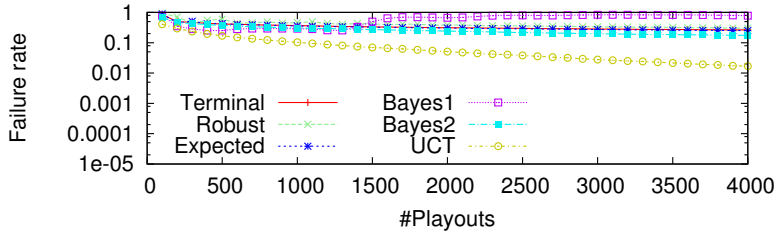
(b) （分枝数，深さ）$(4, 12)$，終端確率 0.1



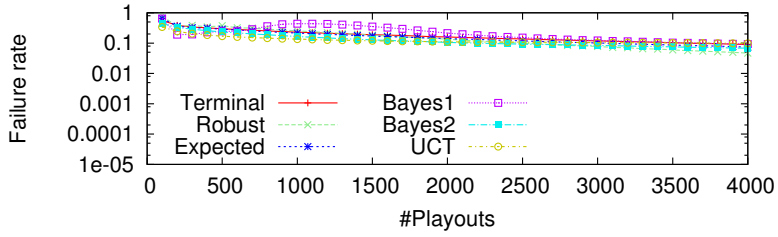
(c) （分枝数，深さ）$(4, 12)$，終端確率 0.2



(d) （分枝数，深さ）$(8, 8)$，終端確率 0.0



(e) （分枝数，深さ）$(8, 8)$，終端確率 0.1



(f) （分枝数，深さ）$(8, 8)$，終端確率 0.2

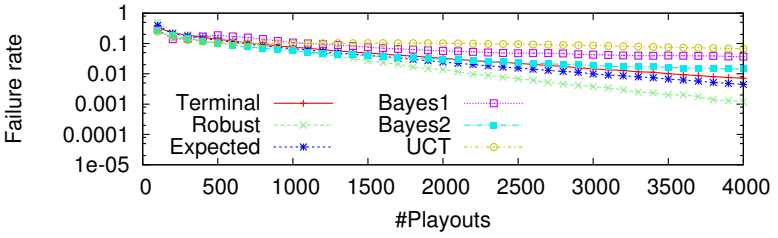
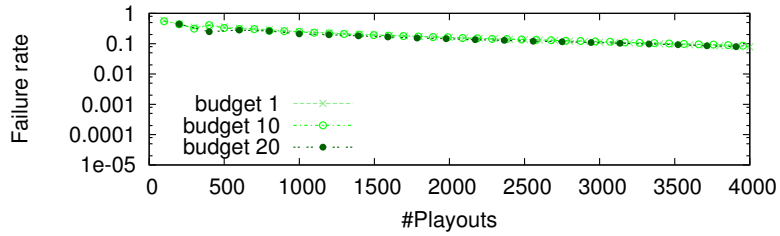


図 4.3: 誤答率の比較

(a) 終端確率 0.0



(b) 終端確率 0.1

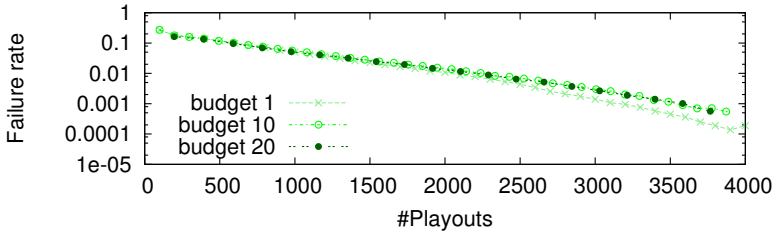


図 4.4: HB+E$^{\mathrm{Robust}}$ での誤答率．budget はそれぞれ 1, 10, 20.

第5章　子孫の勝敗確定時のシミュレーション結果の修正

ゲームにおいて終局は，重要である．終局では，勝ち負けが確定し，そして，終局の前の局面でも，手番次第で，最善を尽くした場合の勝ち負けが確定するためである．しかし，モンテカルロ木探索（MCTS）における終局の扱いをどうすべきかについてはまだ明らかでない．本章では MCTS での終局の扱い方について議論し，最善を尽くした場合の勝ち負けの計算とともに，勝ち負けが確定した手とその祖先の評価の調整を行う手法を 3 種類提案した．実験の結果，提案手法は既存手法より優れた性能を示した．また，提案手法について最善手の評価の観点から分析した．

尚，本章の内容は発表済みの論文 [80] に加筆，編集したものである．

5.1　背景

2.6 節で紹介した通り，MCTS の代表的なアルゴリズム UCT は探索木中の手の評価を，シミュレーションで行い，それを繰り返すことで各手の評価の精度を高めていく．UCT は，十分に探索すると最善手を選ぶ確率が 1 に収束することが示されている [42]．

しかしながら，実際にゲーム等で用いる際には，多くの場合，限られた時間の中で各局面と可能な手を調べ，最善手を選ぶ必要がある．その観点から考えると，効率的な探索のために終局の情報は非常に重要である．終局では勝ち負けが確定し，子の結果と手番次第では，親自身についても最善での勝ち負けが確定するというように，再帰的に最善でない手が判明する可能性があるためである．

MCTS での終局の扱いについての先行研究としては文献 [82] や，Monte-Carlo Tree Search Solver (MCTS-Solver) が挙げられる．MCTS-Solver は探索木中の終局（終端節点）での勝ち・負けの情報を用いて，無駄な探索を減らす方法である．手番にとっての負けに至る手は探索中選ばないようにして効率化している．

しかしながら，この方法は改善の余地があると考えられる．MCTS-Solver を使った探索の場合，ある節点 i が最善で勝ち，もしくは負けと確定したとしても，それまでのシミュレーション結果が i の先祖で保持されたままになっている．つまり節点 i の最善の結果とは異なるシミュレーション結果が先祖で保持されているということである．これは望ましくないと考えられる．

本章では，以上の観点から，MCTS-Solver による勝敗の確定の計算とシミュレーション結果の調整を組み合わせることで，MCTS の性能の改善を目指す．本章の手法は，終

局が探索木中にある場合に限った改善であるので，この改善が役立つ局面は限られる反面，4 章での提案手法のように一様な木ではっきりと悪くなることも無い．また，シミュレーション結果を調節するだけであるので，簡便で，計算コストも低いという長所もある．

5.2 提案手法

本節では，MCTS-Solver の拡張として，3 種類の方法を提案する．提案手法では，シミュレーション終了後，通常の UCT と同様に利得の推定値と訪問数を更新するだけでなく，最善を尽くした場合の勝ち負けを計算し，勝ち負けが確定した際に，訪問数及び利得の推定値を修正する．そして手番にとっての負けの手はそれ以降の探索中に訪問しない．その際の修正の対象では，負けが確定した手がある節点だけを考えれば十分である．もし，節点 j に勝ちが確定した手があった場合，節点 j は j の手番の勝ちが確定するため，節点 j に至る手は j の親 i の手番の負けが確定した手である．そのため，その手 j [1] は探索中に二度と訪問しないので，節点 j の手については修正を考える必要はなく，節点 i の負けの手 j についての修正を考えれば十分である．

提案手法では，葉節点では，まず，式 (2.8) と (2.9)（ 2.6 節参照）で値を更新する．さらに，負け確定の手があれば，その都度，後述の手法により，訪問数，利得の推定値を修正する．そして，葉節点の祖先の更新は式 (2.10) と (2.11) を用いる．これらの式により，負け確定の手がある節点での値の修正が，自然な形で祖先の値に反映される．

5.2.1 利得の推定値の修正：Replacement MCTS-Solver

互いに最善を尽くした場合の勝ち負けが判明した場合の単純な修正方法を提案する．MCTS において，最善の勝ち負けが判明する前の節点の推定値は，ランダムシミュレーションの結果である．Replacement MCTS-Solver (Replacement) では，最善の勝ち負けが判明した場合，その値で推定値を置き換える．これは，節点で保持している，今までのシミュレーションの結果を，その節点以降，常に最善手を選ぶというシミュレーションの結果で置き換えることに相当する．Replacement はもしある手 j が負けと確定した場合，$\bar{X}_{j,N_j} \leftarrow -1$ と修正する．Replacement では訪問数は特に修正を行わないため，UCT と同様に増える．

5.2.2 訪問数の修正：All Removal MCTS-Solver

前節の手法 Replacement では，負けと確定した手をその訪問数分だけ負けが出たとして計算するため，シミュレーションがある程度行われた後で負けが確定した場合は大

[1] 節点 j とその親節点 i について，確定的なゲームを対象としているので，便宜上，節点 j に至る節点 i の手と節点 j を同一視しているが，以下，勝ち負けについては，手 j と記した場合は，節点 i で手番のプレイヤーにとっての，節点 j と記した場合は，節点 j で手番のプレイヤーにとっての勝ち負けを意味する．

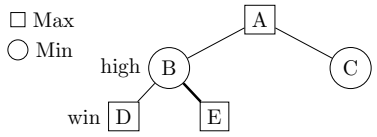


図 5.1: Replacement の問題点．節点 D が Max の勝ちと判明し，兄弟節点 E の勝ち負けは不明とする．この時，節点 B から節点 D に至る手が（少なくとも単独では）節点 B の最善手となることは無い．しかしながら，Replacement では，勝ち負けが判明した節点 D とその先祖 B の推定値を書き換えるため，節点 D の影響で節点 A から節点 B に至る手は過度に高い推定値になる．特に，節点 D の訪問数が大きい時に顕著である．

きく推定値が変化し得る．その結果，まだ勝ち負けが確定していない，確定した節点に至る手が，それ以降しばらくの間探索されないということが起こり得る．これは MCTS の性質として望ましくないと予想される．また，上記と関連して，負けが確定した手は（全ての手が負けの場合を除き）最善手では無いにもかかわらず，その手の推定値を親の推定値を算出するのに使っているため，負け確定の手をもつ節点の推定値は過度に高くなる（図 5.1）．これもまた望ましくないと予想される．そこで，負けが確定したら，その訪問数も調整するとともに，負け確定の手の値を使わないという手法を提案する．このように，UCB 値の第 2 項も合わせて調整することで，負けが確定した影響で，その先祖の推定値が大きく下がり，しばらく訪問されないという状況を避けられると期待出来る．この手法を All Removal MCTS-Solver (ARemoval) とする．ARemoval では，もしある手 j が負けと確定した場合，ε は十分小さい値として，$N_j \leftarrow \varepsilon$, $\bar{X}_{j,N_j} \leftarrow -1$ とする．

5.2.3 利得の推定値の変化に応じた訪問数の修正：Inconsistency Removal MCTS-Solver

前節の手法 ARemoval では，特に探索の始めで，近くに終端節点がある場合に探索が偏りがちになると予想される．具体的には，ある手を先に探索した結果，その子孫の手で負け確定の手があって，その結果訪問数が減って，再びその手を探索するということが起こり得る．また，訪問数が少なくなる分，まだ勝ち負けが確定していない手のシミュレーション結果の違いに大きく影響されてしまい，推定がばらつくことも予想される．これらもまた，MCTS の性質として望ましくないと考えられる．推定値を安定させることとそれを正しい値に近づけることを両立させるのが理想的である．節点 i の手 j で負けが確定した場合に Replacement のような修正がどれだけ影響を及ぼすかを考えると，今までの推定値が負けの利得 -1 から離れているほど，節点 i とその先祖の推定値に大きな変化を与え得る．そこで，提案手法では，負けが確定した際の推定値の修正による変化が大きいほど訪問数を減らし，推定値の修正の影響を和らげることを目指す．この手法を Inconsistency Removal MCTS-Solver (IRemoval) とする．具体

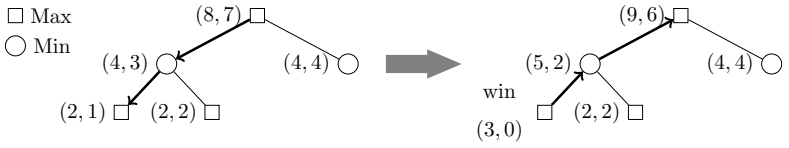


図 5.2: 提案手法 IRemoval の更新の例左下の節点の子孫でのシミュレーション結果が Max プレイヤーの勝ちとなり，かつ左下の節点が Max プレイヤーの勝ちと確定した時の提案手法の更新の例．各節点の (a, b) は節点の Max プレイヤーの勝ち数，負け数がそれぞれ a, b を意味する．

的には，IRemoval は $t+1$ 回目のシミュレーションである手 j が負けと確定した場合，まず，シミュレーションで負けが確定しない場合の推定値を式 (5.1) で求める．そして，式 (5.2) でその値が負けに対応する値 -1 から離れているほど j の訪問数を減らす．更に，式 (5.3) で推定値を正しい値である -1 にする．

$$\bar{X}_{j,N_j(t+1)} \leftarrow \frac{\bar{X}_{j,N_j(t)} N_j(t) + r}{N_j(t) + 1} \tag{5.1}$$

$$N_j(t+1) \leftarrow \frac{(1 - \bar{X}_{j,N_j(t+1)})}{2} (N_j(t) + 1) \tag{5.2}$$

$$\bar{X}_{j,N_j(t+1)} \leftarrow -1 \tag{5.3}$$

シミュレーション結果を保持する方法として，シミュレーションの（勝数，負数）と（勝率，訪問数）は一対一に対応する．利得に言い換えるとそれぞれ，シミュレーションの利得が（1 の回数，0 の回数）と（平均利得，訪問数）である．勝率と訪問数の代わりに勝数と負数を保持する方法で，IRemoval の修正を行う場合，手 j で負けが確定すると手 j のシミュレーション結果の内，手番にとっての勝ちの数を 0 にし，取り除いた分だけ祖先のシミュレーション結果も取り除く．例を図 5.2 に示す．例では簡単のため手番ではなく Max プレイヤーにとっての勝ちと負けを考えている．

このようにすることで，ARemoval より訪問数が減りにくく，ARemoval で予想される，探索初期での推定値の不安定さを，和らげることが出来ると期待される．また，Replacement で手 j で負けが確定した際の 2 つの懸念である，大きな変化と節点 j の親 i の推定値が過度に大きくなることによる不具合も Replacement ほどではないと期待される．ARemoval は，節点 i の利得の推定値を算出するために，負け確定の手 j の推定値を使っているという意味では Replacement と同じではあるが，負け確定の手の訪問数は，負け確定時の調整の分だけ，Replacement に比べて少なくなるためである．例えば，手 j が終端節点が近い場合，負け確定するまでの訪問数は少なくなる．また，j が終端節点が遠い場合でも，手 j が負け確定するまでには，子孫で順に負けが確定していく必要があり，その際に徐々に訪問数が減ると期待される．そのため，いずれにせよ，手 j が負け確定した際の訪問数は少なく，その結果として，手 j の修正が節点 i の推定値にもたらす影響も少ないと期待される．

5.3 実験

5.3.1 実験設定

実験で用いたゲーム木について説明する．本実験では 2.2 節で紹介した incremental random tree に，ランダムで終局を割り当てることでゲーム木を作成した．Max（Min）の次善手の値の幅 $[-a-1]$（$[1,a]$）は $a=5$ としている．本節の木は 3 章の木とは異なり，プレイヤーの有利不利が対象である．また，本節の木は 4 章と同様，終局の割り当ては，最大深さでは，確率 1 で終局，根を除いたそれ以外の節点では，既定の確率 P（終端確率と呼ぶ）で終局とした．尚，このように終局をランダムに決めても，ゲームのスコアの理論値は 0 である．実験では，引き分けも Max プレイヤーの勝ちとして扱った．UCT のパラメータ C_p は $1/\sqrt{2}$ とした．各手法では，訪問数を修正するため，最善手として選ぶのは，利得の推定値が最大の手とした．

5.3.2 実験結果

誤答率の変化

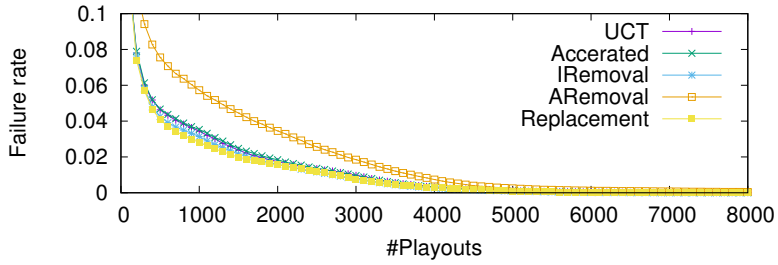
実験では，(幅，最大深さ) が $(4,12)$, $(8,8)$, $(16,16)$ のゲーム木に対し，$P=0.1,0.2$ の 2 種類の終端確率を設定し，それぞれの設定に対し，各アルゴリズム，UCT + MCTS-Solver (UCT)，Accerated UCT + MCTS-Solver (Accerated)，UCT + Replacement，UCT + ARemoval， UCT + IRemoval で探索し，利得の推定値が最も高い手を選ばせて，最善手以外を選んだ比率（誤答率）を計測した．Accerated の λ は $0.9, 0.99, 0.999$ の内，予備実験で最良の結果であった 0.999 を用いた．尚，各設定に対し，ゲーム木を 800 本生成し，それぞれの木に対し各 200 回探索した．プロットは 10 プレイアウト毎に行っている．もし，根節点が勝ちと確定した場合はそれ以上の探索は行っていない．

分枝数 4，最大深さ 12 の結果（図 5.3）は ARemoval の誤答率が UCT の誤答率と比べて有意に高くなった [2]．特に，図 5.2(a) のプレイアウト数 800 から 1200 では，UCT と比べて，IRemoval, Replacement の誤答率は有意な差となった．尚，Accerated の誤答率と UCT との誤答率にはほとんど差がつかなかった．このことは本章の他の実験でも同様であった．また，終端確率 0.2 でプレイアウト数が 5000 手前でプロットが無くなっているのは，全ての探索（800×200）でプレイアウト数 5000 までに根節点の勝ちが確定しているためである．

分枝数 8，最大深さ 8 （図 5.4），終端確率 0.1 の時の誤答率は，UCT と比べて Replacement はプレイアウト数 1000 から 2500 辺りまで，有意に低くなったものの 3000 辺りから，逆に有意に高くなった．ARemoval は全体的高く，IRemoval は低くなり，特にプレイアウト数 400 辺りから有意に差がついた．終端確率 0.2 の時の誤答率は，

[2]有意水準 0.05 のウェルチの t 検定で，両側検定を行った．

(a) 終端確率 0.1



(b) 終端確率 0.2

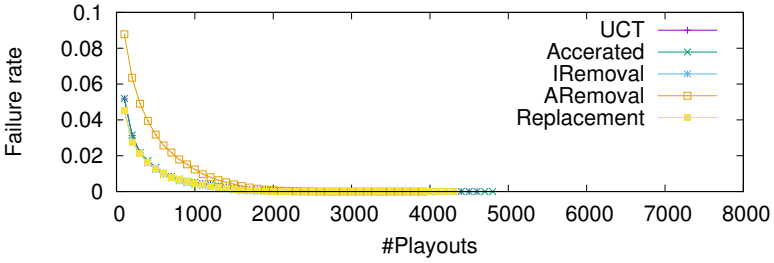


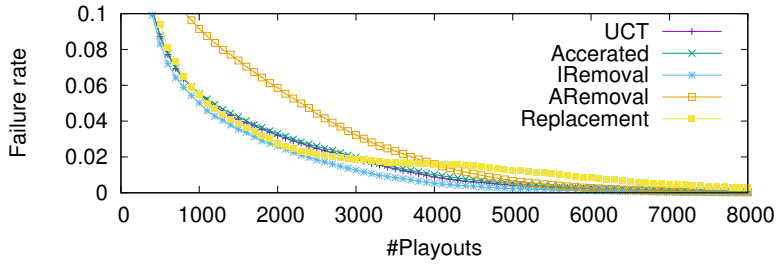
図 5.3: （分枝数，最大深さ）＝ (4, 12) での誤答率の推移

UCT よりも，ARemoval, Replacement は有意に高くなった．しかし，IRemoval はプレイアウト数 300 から 3000 辺りまでは有意に下回った．

より大きな木である，分枝数 16，最大深さ 16 の時の誤答率（図 5.5）は，終端確率 0.1，0.2 の時も同様の傾向で，プレイアウト数 200 辺りから ARemoval は有意に高くなった．また，Replacement は始め UCT より誤答率が有意に高いものの，プレイアウト数 4000 ほどから低くなり，IRemoval も UCT より有意に高くなることがあるものの，プレイアウトを重ねるとより低くなった．

今度はより難しいと考えられる，手の値の比率が異なる木を設計して，誤答率を計測した．このゲーム木モデルでは，手の値の幅のパラメータ a の値を根節点以外の手では，根節点の手の数倍にするという形で変えた．こちらの方がずっと簡単ではあるが，3 章で扱った難しさとシミュレーション結果が偏っているという点で共通した難しさを持つゲーム木モデルである．手の値の比率が異なるゲーム木モデルでは，よりはっきりとした差がつくと期待される．今までの木では，根節点の手の値が大きく異なる場合も多いと予想され，その場合は手法によらず，すぐに最善手を選べるようになり，そのような場合では差がつきにくいと予想される．その一方で，根節点の手の値に対しその他の手の値が大きい状況では，最善手を見つけるため，根節点の手の相対的に小さな違いを見分ける必要があるので難しいと考えられるためである．具体的には，根節点の手では引き続き $a = 5$ とし，それ以外では根節点の手の値の 2 倍の $a = 10$ として，実験を行った．その結果を図 5.6，図 5.7，図 5.8 に示す． 結果は手の値の比率が 1 倍の時よりも全体的に誤答率に差がついた．傾向としては比率が 1 の時と同様に IRemoval の性能が良かった．

(a) 終端確率 0.1



(b) 終端確率 0.2

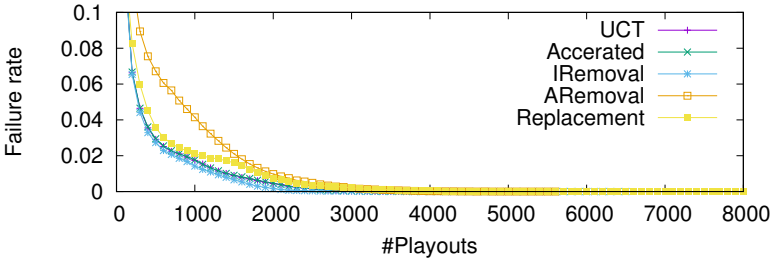


図 5.4: （分枝数，最大深さ）＝ $(8, 8)$ での誤答率の推移

最善手の推定値の分析

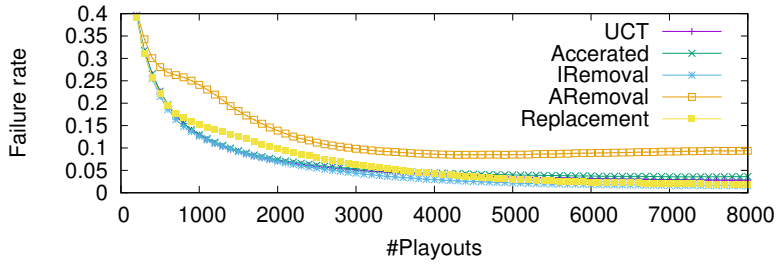
本節では，誤答率に差がついた理由についてより詳しく調べる．特に，各手法で差がついた，手の値の比率の異なるゲーム木モデルで，手の値の比率を 2 とした時の結果について分析する．分析のため，各ゲーム木における探索 200 回の最善手の利得の推定値の平均と分散を算出した．その値の全体的な傾向を見るため，800 本のゲーム木での値の平均をとった．

このゲーム木では，互いに最善を尽くすと，Max プレイヤー（根節点は Max プレイヤーの手番）の勝ちであり，互いに最善を尽くした場合の利得は 1 であるので，利得の推定値が 1 に近いほど，正確であることを意味する．そして分散が小さいほど，各探索による違い [3] によらず，推定値が安定しているということを意味する．尚，根節点の勝ちが確定すると推定値は 1 になる．以下，実験結果について提案手法毎に記す．

まず ARemoval について記す，分枝数 4，最大深さ 12，終端確率 0.1（図 5.9）の時の推定値の平均はプレイアウト数 4000 辺りから UCT が一番低くなった．しかし，誤答率（図 5.5(a)）は UCT のほうが ARemoval より低くなった．ARemoval は平均的には，最善手の利得をより正しく見積もることができているが，UCT よりも誤答率が高くなっているということである．分散は ARemoval が高くなった．つまり，プレイアウト数が同じでも，推定値が各探索で比較的大きく変わるということである．終端確率 0.2（図 5.10）の時の推定値も，UCT よりも ARemoval が高く，その一方で，分散は ARemoval が高く，特にプレイアウト数が少ない場合に顕著であった．これは ARemoval では，終端節点まで探索木が成長し，ある節点で勝ちが確定した場合，その親では負け確定の手に対する調整が行われ，訪問数が減るため，推定値が安定しないということと

[3] 乱数により，シミュレーション結果が異なり，その結果としてシミュレーションを開始する節点の優先順位が異なる可能性があり，探索結果に違いが生じ得る．

(a) 終端確率 0.1



(b) 終端確率 0.2

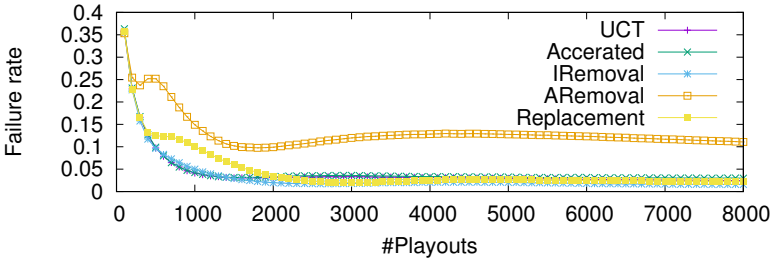
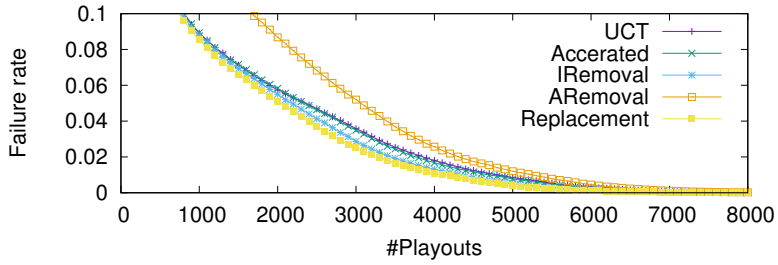


図 5.5: （分枝数，最大深さ）＝ $(16, 16)$ での誤答率の推移

予想され，特に終端確率 0.2 の時は頻繁に起きたためであると予想される．また，分枝数 8，最大深さ 8 の時（図 5.11，5.12），も同様の傾向で，UCT と比べて推定値の平均は高くなった．また，その差は，分枝数 4 の時と比べて，多少大きくなった．分散は，終端確率 0.2 でプレイアウトが少ない場合に特に高くなった．分枝数 16，最大深さ 16，終端確率 0.1 の時（図 5.9），平均値は 0.4 付近でほとんど上昇しなかった．終端確率 0.2 の時（図 5.10）も同様に，多少は上昇傾向が見られるものの，平均値はまだ 0.4 付近でほとんど上昇しなかった．これらは木が大きいため 8000 回プレイアウトしただけでは，最善手がどの手か判断するのに十分でない場合が多いということを示唆する．分枝数 4 の時より，分枝数 8 の時のほうが UCT と比べて推定値が正確であったことは，UCT の推定値が負けが確定した手の値も利用して算出するため，分枝数が大きいほど，その値に影響され，ARemoval と比べて正しい値になりづらいということを示唆する．UCT との誤答率の差について，ARemoval は分枝数 16 の木を除いて，分枝数が高いほど改善する傾向になったのはそのためであると予想される，

次に Replacement について記す．分枝数 4，最大深さ 12 （図 5.9，5.10）の時の Replacement の推定値の平均は UCT よりも少し高くなった．分散は終端確率 0.1 の時に多少 UCT と比べて上昇した．分枝数 8，最大深さ 8 の時（図 5.11 図 5.12）は，Replacement の推定値は UCT と同程度で分散については，Replacement はプレイアウトが増えるにつれて，UCT と比べて少し上昇する傾向が見られた．これは Replacement で訪問数が大きな節点の勝ち負けが確定し，推定値を書き換えたためと予想される．UCT と同様に Replacement では負けが確定した手の値も利用して節点の利得の推定値を算出するため，分枝数が大きいほど，その値に影響されやすいと予想される．そして，負け確定なら全て負けとして計算するので，負け確定でも古いプレイアウトを使う UCT と比べて，その影響は大きいと予想され，推定値の分散の結果はこれを支持している．

(a) 終端確率 0.1



(b) 終端確率 0.2

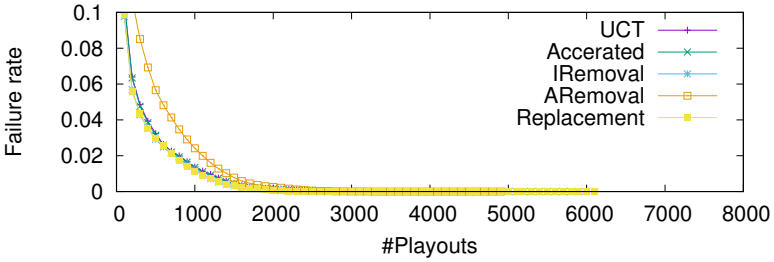


図 5.6: 手の値の比率 2,（分枝数，最大深さ）＝ $(4, 12)$ での誤答率の推移

そのような理由により分枝数 8 の時は UCT ほど誤答率が下がらなかったと考えられる．

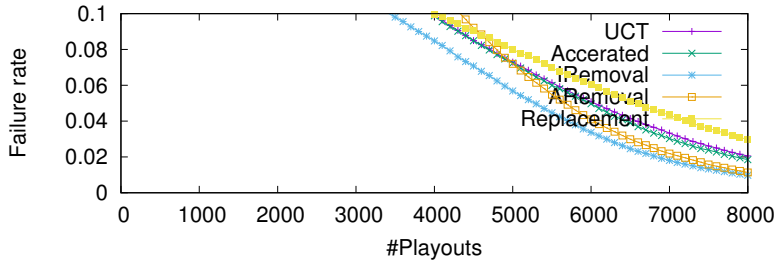
最後に，IRemoval について記す．IRemoval は ARemoval のように訪問数を調整していて，Replacement のように推定値を修正し，その値が親の推定値を計算するのに使われる手法である．最善手の推定値の平均と分散の分析を通じて，IRemoval の推定値の平均は既存手法と同程度かそれより高く，ARemoval と Replacement の高い方を超えない程度であった．しかし，分散については ARemoval のように大きくなることはなく，Replacement のようにプレイアウトが増加するにつれて分散が安定して上昇する傾向は，全ての手法で上昇傾向の分枝数 16，終端確率 0.2 の木を除けば，無かった．分散は総じて低く，UCT と同程度に抑えられていた．平均は UCT よりも高く，分散は UCT と同程度であるので，推定値の平均と分散の観点からも，IRemoval の良さを支持する結果となった．

5.4 まとめ

本章では，探索中にある手の勝ち負けが確定するような状況について，その節点を訪問した過去のプレイアウト結果をどう扱うかについて議論した．そして，3 つの提案手法に対し，その性能について，実験で評価した．その結果，特に Inconsistency Removal MCTS-Solver (IRemoval) は既存手法よりも優れた性能を示した．加えて，性能について最善手の推定値の平均と分散をとり，各手法を比較し，分析をした．その結果，既存手法と比べて IRemoval の最善手の利得を見積もりは正確であり，各探索で既存手法と同程度に安定していた．

提案手法は 4 章のものと異なり，終局が近くに無い場合は通常の UCT と変わらないため，適用しても悪影響は無い．また，分布を使わない，かつ，更新の際にプレイアウ

(a) 終端確率 0.1



(b) 終端確率 0.2

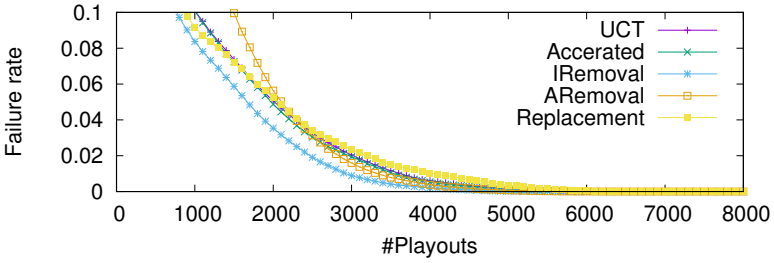
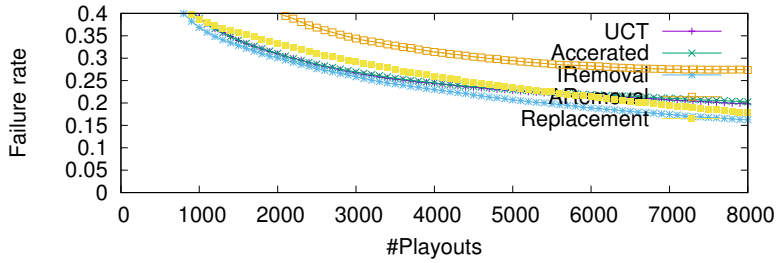


図 5.7: 手の値の比率 2,（分枝数，最大深さ）＝ $(8, 8)$ での誤答率の推移

トで辿った節点の値を子節点の推定値に基づき更新するだけであるため，計算のコストはより少なく抑えられる．

本章の結果は換言すると，終端節点（終局）が根の近くにある場合，終局での利得の理論値を上手く用いて，間違っているシミュレーション結果を取り除く等を行い，節点の推定値を正しい値に修正（加えて不確かさの評価も修正）することで改善したということである．これは，一般の場合でも成り立つ可能性がある．つまり，終端節点が根の近くに無く，理論値が分からない場合でも，節点の推定値をより正しい値に修正することで，MCTS の性能の改善出来る可能性があるということである．その立場から，6 章では推定値について，より基礎的な視点から論じ，新しい推定量を提案する．また，7 章では 6 章で提案した推定量を実際に MCTS に応用し，改善を目指す．

(a) 終端確率 0.1



(b) 終端確率 0.2

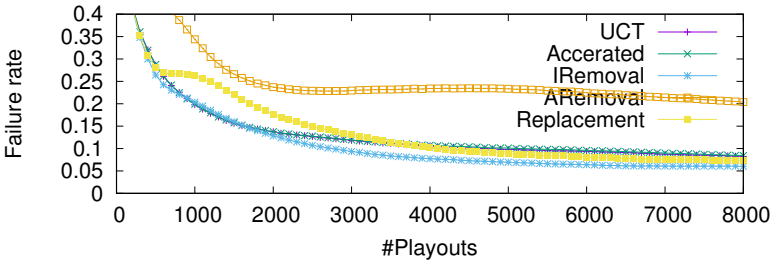
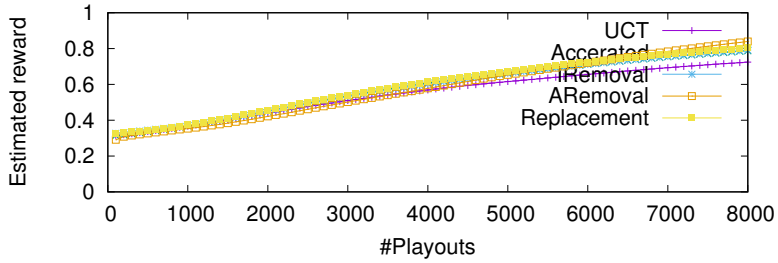


図 5.8: 手の値の比率 2,（分枝数，最大深さ）＝ $(16, 16)$ での誤答率の推移

(a) 平均



(b) 分散

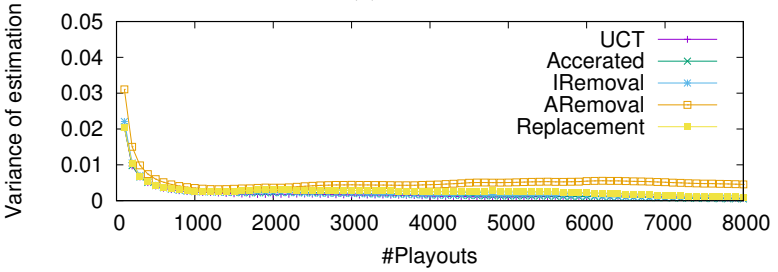
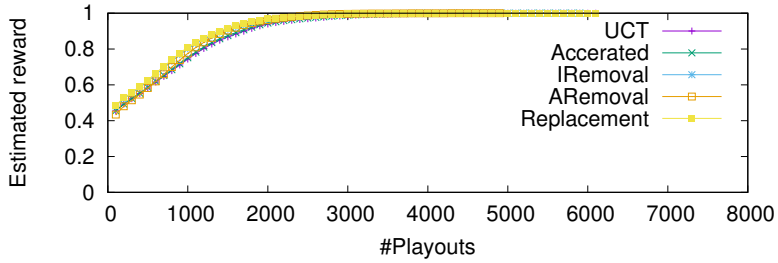


図 5.9: 終端確率 0.1，手の値の比率 2,（分枝数，最大深さ）＝ $(4, 12)$ での最善手の推定値の平均と分散の推移

(a) 平均



(b) 分散

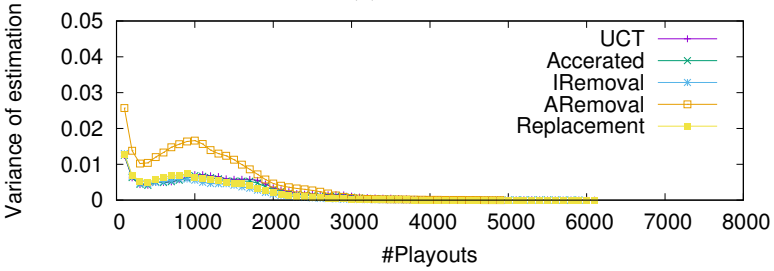
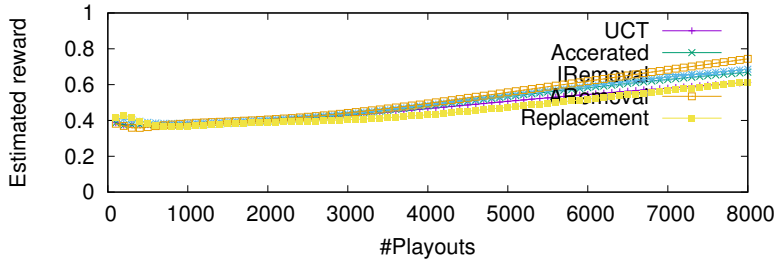


図 5.10: 終端確率 0.2，手の値の比率 2,（分枝数，最大深さ）＝ $(4, 12)$ での最善手の推定値の平均と分散の推移

(a) 平均



(b) 分散

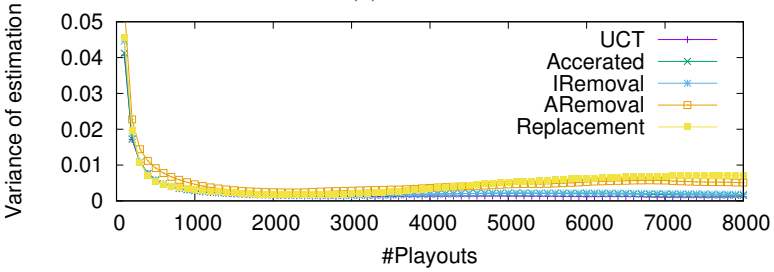
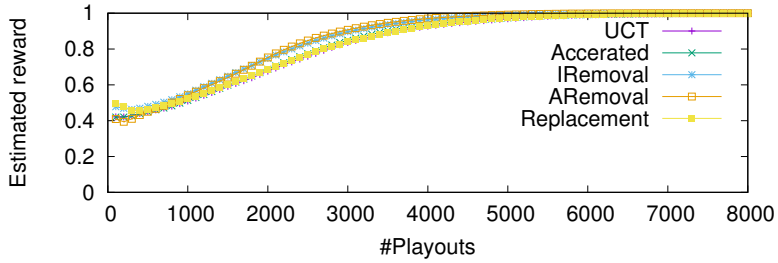


図 5.11: 終端確率 0.1，手の値の比率 2,（分枝数，最大深さ）＝ $(8, 8)$ での推定値の平均と分散の推移

(a) 平均



(b) 分散

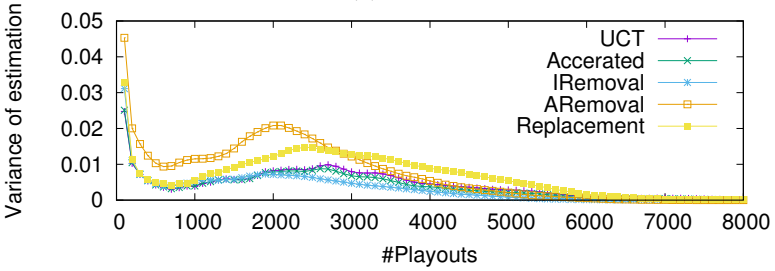
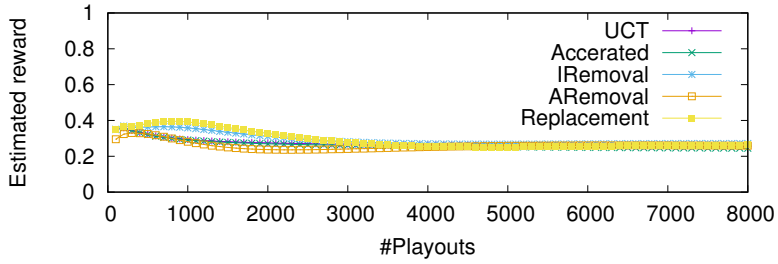


図 5.12: 終端確率 0.2，手の値の比率 2,（分枝数，最大深さ）＝ $(8, 8)$ での推定値の平均と分散の推移

(a) 平均



(b) 分散

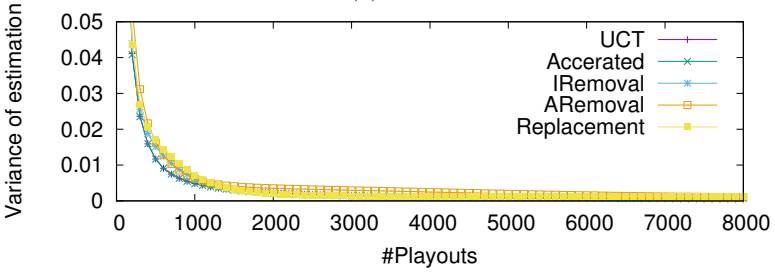
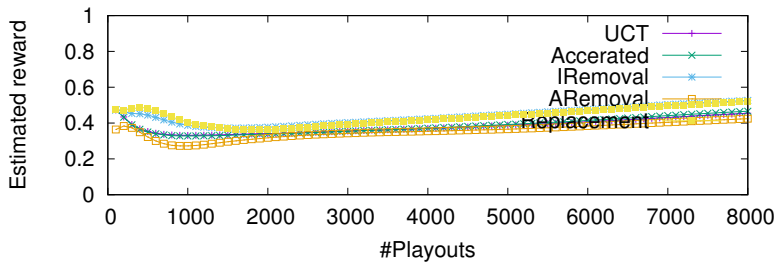


図 5.13: 終端確率 0.1，手の値の比率 2,（分枝数，最大深さ）＝ $(16, 16)$ での最善手の推定値の平均と分散の推移

(a) 平均



(b) 分散

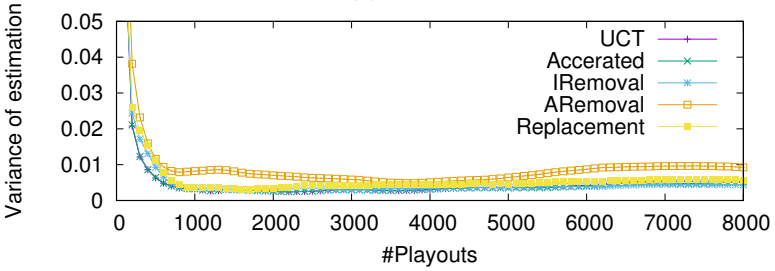


図 5.14: 終端確率 0.2，手の値の比率 2,（分枝数，最大深さ）＝ $(16, 16)$ での最善手の推定値の平均と分散の推移

第6章　期待値の最大値の推定量

複数の確率変数が存在する状況下で，その期待値の最大値を求めることは，強化学習，探索等をはじめとした分野で重要である．本章では，期待値の最大値の新しい推定量 Simplified Weighted Estimator（SWE）を提案し，多腕バンディット問題の設定で，既存の推定量と実際的，理論的側面から議論する．推定量 SWE が算出する推定値は各アームからサンプリングした値の平均の重み付き和であり，各アームの重みはアームが最善である確率をもとに決めている．理論的な分析では，新しい推定量のバイアスは妥当な仮定のもとでサンプルサイズが増加するにつれて 0 に収束することを示した．さらに，分散についても，上界を示した．実験では，強化学習でよく使われる，UCB1，softmax，epsilon-greedy で，アームを引いて集めたサンプルに対し，推定値を算出し，提案手法の性能を評価した．推定量はサンプルサイズに対する，バイアス，バリアンス（分散），平均自乗誤差の観点から評価した．様々なバンディット問題の設定の下で，提案手法は常に最良ではないものの，多くの設定で良い結果となった．

尚，本章の内容は文献 [35] として発表済みのものを編集，加筆したものである．

6.1　背景

本章では，2.4.1 節で紹介した多腕バンディット問題を題材に，複数の確率変数の期待値の最大値を，サンプリングをもとに推定する問題を考え，新たな方法を提案する．期待値の最大値を見積もるという問題は重要であることを 3 つの例を挙げて説明する．1 つ目の例は，強化学習である．強化学習での最善の方策を構成する行動は，その行動以後，最善方策をとると仮定したもとで，累積期待利得が最大の行動を定義される．従って，その最善方策を見つけるためには，期待値の最大値を見積もる必要がある．しかしながら，Q 学習で使われる，maximum estimator（ME）は正しい値より高く見積もる性質があり，そのため，学習の速度が落ちるドメインの存在が知られている [71]．2 つ目の例は，sponsored search auctions [77] である．この問題は多腕バンディット問題の一種で，期待値の最大値の見積もりの誤差が経済的な損失に繋がるという問題である．3 つ目の例は，2.6 節で紹介した，モンテカルロ木探索（MCTS）である．MCTS でも，強化学習と同様に，最善手を判別するために，期待値の最大値が必要である．MCTS の代表的なアルゴリズムである UCT [42] では，子の価値の推定を子孫から行ったシミュレーション結果の平均で行う．これは，サンプルの平均による推定量（AVE）を用いていると解釈出来る．UCT は累積的な損失を最小化することを目的とした UCB1 [3] に

基づくアルゴリズムであるから，良さそうなところを多く選ぶという特質があり，漸近的には，最善のアームを引く頻度が 1 に近づくものの，多数の子の推定値が低い場合に，AVE の推定は低くなり，推定値が正しい値に近づくまでが遅い．従って，期待値の最大値の推定方法として改善の余地がある． 2.5 節で紹介したように，他にも様々な推定方法が研究されてきたが，既存の推定量には精度や計算コストの面でやはり改善の余地がある．尚，期待値の最大値の推定量に関して，一般には不偏推定量は存在しない [72]．そのため，優れた推定量を見つけることは簡単では無く，不偏では無い推定量から，より良いものを選ぶ必要がある．

推定量の評価を行うにあたって，なるべく小さいサンプルサイズに対して，より正確に，より安定して推定値を算出することが望ましい．本章では，バイアスと分散と平均自乗誤差で推定量の性能を評価する．

本章で提案する新しい推定量は，サンプル中の利得の平均ではなく，利得の重み付き和をとる．各利得の重みはサンプルが得られたアームが最善である尤もらしさに応じて決まる．理論的な分析により，提案手法の推定値のバイアスは妥当な仮定のもとで，サンプルサイズを増やせば 0 に収束することを示す．加えて実験を行い，提案手法と既存手法の実際的な性能を確かめる．実験では，強化学習で使われる主要なアルゴリズムを用いてサンプルを集めた．様々な設定で，そのように集めたサンプルに対する推定の誤差を評価する．

6.2 Weighted Estimator Based on Upper Confidence Bound

本節では， 2.5 節で紹介した既存手法 WE のアイデアに基づき，新たな推定量 Simplified Weighted Estimator (SWE) を提案し，理論的な性質について分析する．分析では， 2.5 節と同様に，多腕バンディット問題の設定で，アーム i を引いた回数 N_i は確率変数でないと仮定して行っている．

SWE は WE を簡略化し平均利得とサンプルサイズだけを使うので．分布を使う WE よりも計算コストが少ないという長所がある．SWE は $\hat{\mu}$ を全ての利得（サンプル D の要素）の重み付き和で推定する．各アームでの平均利得の重み付き和で推定するとも言える．W_i をアーム i の平均利得の重みとした．$\sum_{i=1}^{K} W_i = 1$ となるように正規化している．推定量 SWE は

$$\hat{\mu}_{SWE} := \sum_{i=1}^{K} W_i \bar{X}_{i,N_i} \tag{6.1}$$

である．最善のアーム $*$ に対応する重み W_* が比較的大きい時にこの推定値は正確になる．SWE では，今までのサンプルに対し，アーム i が最善である尤もらしさを考え，W_i をそれに応じて決める．直感的には，アーム i の平均利得が他のアームの平均利得よりも遥かに大きいなら，アーム i は本当に $*$ であると予想される．そのため，平均利得の差に着目する．一般に，最善のアームとそれ以外のアームの平均利得について，最

善のアームの平均利得がそれ以外のアームの平均利得を下回ることは稀であり，以下の補題が成り立つ．

補題 1. 任意の $\varepsilon \geq 0$ と任意のアーム $j \in \{1 \dots K\} \setminus \{*\}$,

$$\mathrm{P}(\bar{X}_{j,N_j} - \bar{X}_{*,N_*} \geq \varepsilon) \leq 2\exp\left(-2\left(\varepsilon \frac{\sqrt{N_j N_*}}{\sqrt{N_j} + \sqrt{N_*}}\right)^2\right).$$

以下，簡単のため，任意のアーム k, l について

$$N_{k,l} := \left(\frac{\sqrt{N_k N_l}}{\sqrt{N_k} + \sqrt{N_l}}\right)^2$$

とする．

Proof. 任意の定数 a $(0 \leq a \leq 1)$ について以下の不等式が成り立つ．

$$\begin{aligned}
&\mathrm{P}(\bar{X}_{j,N_j} - \bar{X}_{*,N_*} \geq \varepsilon) \\
&\leq \mathrm{P}(\bar{X}_{j,N_j} - \mu_j \geq a(\varepsilon + \Delta_j)) \\
&\quad + \mathrm{P}(\bar{X}_{*,N_*} - \mu_* \leq -(1-a)(\varepsilon + \Delta_j)) \\
&\leq \exp(-2\{a(\varepsilon + \Delta_j)\}^2 N_j) \\
&\quad + \exp(-2\{(1-a)(\varepsilon + \Delta_j)\}^2 N_*)
\end{aligned}$$

1 つ目の不等式は，$\bar{X}_{j,N_j} - \mu_j < a(\varepsilon + \Delta_j)$ かつ $-(1-a)(\varepsilon + \Delta_j) < \bar{X}_{*,N_*} - \mu_*$ ならば $\bar{X}_{j,N_j} - \bar{X}_{*,N_*} < \varepsilon + \Delta_j - \mu_* + \mu_j = \varepsilon$ の対偶から導かれる．2 つ目の不等式は，Hoeffding の不等式 [30] から導出される．$\Delta_j \geq 0$ であることと 2 項のバランスをとるため， $a = \frac{\sqrt{N_*}}{\sqrt{N_j} + \sqrt{N_*}}$ とすると

$$\begin{aligned}
&\leq 2\exp\left(-2\left((\Delta_j + \varepsilon)\frac{\sqrt{N_j N_*}}{\sqrt{N_j} + \sqrt{N_*}}\right)^2\right) \\
&\leq 2\exp\left(-2\left(\varepsilon \frac{\sqrt{N_j N_*}}{\sqrt{N_j} + \sqrt{N_*}}\right)^2\right) \\
&= 2\exp\left(-2\varepsilon^2 N_{j,*}\right).
\end{aligned}$$

□

補題 1 は $\bar{X}_{j,N_j} - \bar{X}_{*,N_*} > \varepsilon$ が成り立つ確率の上界を与えている．つまり観測された平均利得について，最善でないアームが最善のアームを ε だけ上回っている確率である．$\frac{\sqrt{N_j N_*}}{\sqrt{N_j} + \sqrt{N_*}}$ は直感的には，引いた回数 N_j, N_* に基づいて，差の信頼度合い，つまり，差がついたのは偶然か否かを評価していると言える．尚，上記の不等式で右辺を最小化する a の値は一般には不明である．例えば，N_i と ε 次第では，2 項のバランスをとらずに，単に $a = 1/2$ をとる方が小さくなることもある．しかしながら，2 項のバランスをとることで，一方の項が相対的に指数的に大きくなることを避けることが出来，多くの場合で，右辺を小さく出来ると期待される．

サンプルの重み $\tilde{w}_i$ をこの確率の上界に基づいて定め，式 (6.1) のアームの重み W_i を

$$W_i = \frac{\tilde{w}_i N_i}{\tilde{n}} \tag{6.2}$$

として定める．ここでの $\tilde{n}$ は正規化項であり，

$$\tilde{n} = \sum_i \tilde{w}_i N_i \tag{6.3}$$

である．任意のアーム k と l に対し，$d_{k,l}$ を観測された平均利得の差 $\bar{X}_{k,N_k} - \bar{X}_{l,N_l}$ と定める．$\tilde{w}_i$ を以下のように定める．

$$\tilde{w}_i := \begin{cases} \exp\left(-2\left(cd_{m,i}\right)^2 N_{m,i}\right) & (i \neq m), \\ 1 & (i = m), \end{cases}$$

ここでの $c \geq 0$ はパラメータであり，$m := \arg\max_i \bar{X}_{i,N_i}$ である，つまり観測された平均利得が最良のアームである．

$\tilde{w}_i$ について $0 \leq c \leq 1$ の場合の意味について以下でより詳しく述べる．$\tilde{w}_i$ は既存手法 WE でのアーム i の重み $\mathrm{P}(i = *|D)$（サンプル D を観測した後の $i = *$ の主観確率）と関連がある．ベイズの定理より

$$\mathrm{P}(i = *|D) = \frac{\mathrm{P}(i = *)}{\mathrm{P}(D)} \mathrm{P}(D|i = *)$$

である．事前分布について，任意のアーム i, j について，$\mathrm{P}(i = *) = \mathrm{P}(j = *)$ と仮定すると，比について

$$\mathrm{P}(i = *|D) : \mathrm{P}(j = *|D) = \mathrm{P}(D|i = *) : \mathrm{P}(D|j = *)$$

となる．尚，サンプル D は，観測値の集合

$$\begin{aligned} D = \{&X_{1,1}, \ldots X_{1,N_1}, \\ &\vdots \\ &X_{K,1}, \ldots X_{K,N_K}\} \end{aligned}$$

である．以下では，議論の明確化のため，各アーム i の平均利得について，確率変数（X_i とする）とその具体的な値（サンプル D での平均）$\bar{X}_{i,N_i}$ とで表記を使い分ける．$i = *$ のもとで，事象 $\{\bigwedge_j X_j = \bar{X}_{j,N_j}\}$ について考えると，この事象は平均利得だけに着目しており，個々の利得には着目していないため，事象 $\{D\}$ の起こる確率は $\{\bigwedge_j X_j = \bar{X}_{j,N_j}\}$ よりも小さい，つまり

$$\mathrm{P}(D|i = *) \leq \mathrm{P}\left(\bigwedge_j X_j = \bar{X}_{j,N_j} \middle| i = *\right)$$

となる．$0 \leq c \leq 1$ とすると $\mathrm{P}(D|i=*)$ の上界について以下が成り立つ．

$$\mathrm{P}(D|i=*) \leq \mathrm{P}\left(\bigwedge_j X_j = \bar{X}_{j,N_j} \middle| i=*\right) \tag{6.4}$$

$$\leq \mathrm{P}\left(\bigwedge_j X_j - X_i = \bar{X}_{j,N_j} - \bar{X}_{i,N_i} \middle| i=*\right) \tag{6.5}$$

$$\leq \mathrm{P}\left(\bigwedge_j X_j - X_i \geq d_{j,i} \middle| i=*\right) \tag{6.6}$$

$$\leq \mathrm{P}\left(\bigwedge_j X_j - X_i \geq c d_{j,i} \middle| i=*\right) \tag{6.7}$$

$$\leq \min_j \mathrm{P}\left(X_j - X_i \geq c d_{j,i} | i=*\right) \tag{6.8}$$

$$\leq \min_j \left(2\exp\left(-2\left(\frac{c d_{j,i}\sqrt{N_j N_i}}{\sqrt{N_j}+\sqrt{N_i}}\right)^2\right)\mathbb{I}\{d_{j,i}\geq 0\} + \mathbb{I}\{d_{j,i}<0\}\right) \tag{6.9}$$

$$\leq \min_{j:d_{j,i}\geq 0}\left(2\exp\left(-2\left(\frac{c d_{j,i}\sqrt{N_j N_i}}{\sqrt{N_j}+\sqrt{N_i}}\right)^2\right)\right) \tag{6.10}$$

$$\leq 2\exp\left(-2\left(\frac{c d_{m,i}\sqrt{N_m N_i}}{\sqrt{N_m}+\sqrt{N_i}}\right)^2\right) \tag{6.11}$$

$$= 2\tilde{w}_i \tag{6.12}$$

上記の不等式で，右辺が十分に小さければ，

$$\mathrm{P}(D|i=*) \approx 2\tilde{w}_i$$

であり，

$$\mathrm{P}(i=*|D) : \mathrm{P}(j=*|D) \approx \tilde{w}_i : \tilde{w}_j$$

つまり，$\tilde{w}_i$ と $\tilde{w}_j$ の比は WE でのアーム i と j の重みの比と近いと期待できる．

$c > 1$ の時は上記の (6.7) の不等式が成り立たず，$\mathrm{P}(D|i=*)$ との関係が不明になる．しかしながら，それ以降は成り立つ．つまり，$2\tilde{w}_i$ は差についての確率 $\mathrm{P}\left(\bigwedge_j X_j - X_i \geq c d_{j,i} \middle| i=*\right)$ の上限であると言える．この確率は平均利得が全てのアーム j で i（$=*$ と仮定）を $c d_{j,i}$ 以上上回っている確率であり，この確率でアームの最善度合いを評価するのは妥当である．それは以下の理由による．もし，アーム i が本当に $*$ なら，多くのアーム j で $c d_{j,i} < 0$ になり，そうでないアームがあったとしても $c d_{j,i}$ は小さく，結果としてその確率は大きくなると期待できる．加えて，もし，アーム i が本当は $*$ でないなら，多くのアーム j で $c d_{j,i} > 0$ が成り立ち，$d_{j,i}$ は大きくなり，確率は小さくなると期待できる．以上のように，c を一般化して $c > 1$ の場合も考えるのは，WE との関連性の解釈は成り立たないものの，有効であると期待される．実験では，$c > 1$ も扱っている．

パラメータの範囲の有効性について $0 \leq c \leq 1$ だけでなく $1 < c$ にもあることを説明したが，大雑把には c が低いほど AVE に近く，高いほど ME に近い振る舞いとなる．

実際，もし，$c \to 0$ なら，$\tilde{w}_i$ は，$i \neq m$ で 0 であり，結果として，式 (6.1) の重み W_i は $\mathbb{I}\{i = m\}$ となり，SWE は ME と同じになる．同様に，もし $c = 0$ なら，W_i は N_i/n であり，この場合 SWE は AVE と同じになる．

N_i は全てのアーム i で n に関する広義単調増加関数であることに注意すると，$c > 0$ での SWE のバイアスが 0 に収束する条件について以下の定理が成り立つ．

定理 1. *SWE*のバイアスは以下の 2 条件を満たすとき 0 に収束する．つまり $\lim_{n\to\infty} Bias(\hat{\mu}_{SWE}) = 0$ が成り立つ．条件 *(1)* 任意のアーム i について $\lim_{n\to\infty} N_i = \infty$ である．条件 *(2)* 任意のアーム $j \neq *$ について，

$$\lim_{n\to\infty} \frac{3N_j}{N_*} \exp\left(-\left(\frac{c\Delta_j}{c+1}\right)^2 \frac{N_*}{2}\right) = 0, \tag{6.13}$$

上記の定理の意味することは，SWE のバイアスは次善手の引く割合が最善手のものより極端に多くても（例えば 100 倍）定数倍なら 0 に収束するということであり，さらに極端な場合として，N_j が N_* に対して指数的に増えても，c, Δ_j 次第では，0 に収束するということである．そのため，SWE は収束性に関して，AVE よりもアームの引き方に対する依存度合いが低く，よりロバストである．尚，ME はその観点では SWE よりもロバストである．ME では，全てのアーム j でサンプルが十分に存在する，つまり $\lim_{n\to\infty} N_j = \infty$ という条件さえ満たせば，バイアスが 0 に収束するためである．このことは，$|\mathrm{Bias}(\hat{\mu}_{ME})|$ は $(K-1)\mathrm{P}(m \neq *)$ で抑えられるため，定理 1 と同様にして示される．SWE はサンプルに対して重みを与えているため，各アームのサンプルサイズに多少依存してしまう．それ故，バイアスの収束性を示すためには，条件 (2) のようなサンプルサイズに関する条件は本質的に必要なものであると考えられる．

Proof. 利得は 0 以上，1 以下であるので，$|\bar{X}_{i,N_i} - \bar{X}_{*,N_*}| \leq 1$ であり，$W_* = 1 - \sum_{j\neq *} W_j$ から

$$\begin{aligned}
|\mathrm{Bias}(\hat{\mu}_{SWE})| &= |\mathrm{E}[\hat{\mu}_{SWE}] - \mu_*| \\
&= \left| E\left[\sum_{i\neq *} W_i(\bar{X}_{i,N_i} - \bar{X}_{*,N_*})\right]\right| \\
&\leq \sum_{i\neq *} E\left[W_i \cdot |\bar{X}_{i,N_i} - \bar{X}_{*,N_*}|\right] \\
&\leq \sum_{i\neq *} E\left[W_i\right]. \qquad (6.14)
\end{aligned}$$

である．$\mathrm{E}[W_i]$ の上界について，

$$\begin{aligned}
\mathrm{E}[W_i] &= \mathrm{P}(m = *)\mathrm{E}[W_i | m = *] + \mathrm{P}(m \neq *)\mathrm{E}[W_i | m \neq *] \\
&\leq \mathrm{P}(m \neq *) + E\left[\frac{\tilde{w}_i N_i}{\sum_{j\neq *} \tilde{w}_j N_j + N_*}\middle| m = *\right] \\
&\leq \mathrm{P}(m \neq *) + \frac{N_i}{N_*} E\left[\tilde{w}_i | m = *\right] \qquad (6.15)
\end{aligned}$$

である．$N_{i,j} = \left(\frac{\sqrt{N_i N_j}}{\sqrt{N_i}+\sqrt{N_j}}\right)^2$ という表記を用いると，式 (6.15) の第一項について

$$\begin{aligned}
\mathrm{P}(m \neq *) &\leq \mathrm{P}(\exists i, \bar{X}_{i,N_i} > \bar{X}_{*,N_*}) \\
&\leq \sum_{i \neq *} \mathrm{P}(\bar{X}_{i,N_i} > \bar{X}_{*,N_*}) \\
&\leq \sum_{i \neq *} 2\exp\left(-2\left(\Delta_i\right)^2 N_{i,j}\right) \qquad (6.16)
\end{aligned}$$

である．最後の不等式は，補題 1 の議論と同様に導かれる．任意のアーム j について $N_j \to \infty$ であるから，式 (6.16) は 0 に収束する．式 (6.15) の第二項についても補題 1 の議論と同様にして，アーム $i \neq *$ と定数 d $(0 < d < \Delta_i)$ について，

$$\begin{aligned}
&E\left[\tilde{w}_i | m = *\right] \\
&= E\left[(\mathbb{I}\{d_{m,i} \geq d\} + \mathbb{I}\{d_{m,i} < d\})\tilde{w}_i | m = *\right]
\end{aligned}$$

$\mathbb{I}\{d_{m,i} \geq d\} \leq 1$ かつ $\tilde{w}_i \leq 1$ であるから

$$\begin{aligned}
&\leq \exp\left(-2\left(cd\right)^2 N_{*,i}\right) + \mathrm{P}(d_{*,i} < d) \\
&\leq \exp\left(-2\left(cd\right)^2 N_{*,i}\right) + 2\exp\left(-2\left(\Delta_i - d\right)^2 N_{*,i}\right)
\end{aligned}$$

である．そのため $d = \frac{\Delta_i}{c+1}$ とすると，

$$\frac{N_i}{N_*} E\left[\tilde{w}_i | m = *\right] \leq \frac{N_i}{N_*} 3\exp\left(-2\left(\frac{c\Delta_i}{c+1}\right)^2 N_{*,i}\right)$$

である．Δ_i と c は定数であるので，もし $N_i \leq N_*$ なら，$\forall j, N_j \to \infty$ より，最後の項は 0 に収束する．もし $N_i > N_*$ なら，$N_{*,i} > N_*/4$ であることを用いて，

$$\leq \frac{N_i}{N_*} 3\exp\left(-\left(\frac{c\Delta_i}{c+1}\right)^2 \frac{N_*}{2}\right)$$

である．この項もまた，定理 1 の式 (6.13) で記した条件により，0 に収束する． □

バイアスだけでなく，分散についても理論的なことが示される．

定理 2. *SWE* の分散について，以下が成り立つ．

$$Var(\hat{\mu}_{SWE}) \leq \sum_{i=1}^{K} \frac{\sigma_i^2}{N_i}$$

この不等式は文献 [18] の Theorem 3 の証明に従って示される．この上界は少し，悲観的すぎるかもしれない．次節の実験で，分散についても調査する．

6.3 実験

本節の実験では，UCB1，softmax，epsilon-greedy といった強化学習での著名なアルゴリズムに従い，サンプルする場合の推定量のバイアス，分散，自乗誤差（MSE）につ

いて示す．これらのアルゴリズムでは，今までのサンプルの平均利得に基づいて，アームを選ぶため，各アームを引く回数は確率変数となる． 2.5 節で紹介した既存の推定量についての分析や 6.2 節の提案手法の分析では，簡単のため，各アームを引く回数は確率変数でないという仮定をおいたが，そうでない場合の性能についても概ね理論に従っていることを示す．また，参考のため，全てのアームを一様な確率で引く場合（uniform）での結果も示した．比較対象は既存の推定量 AVE，ME，DE 及びパラメータを調節した MM である．

各アームの利得は，ベータ分布 $B(2\mu, 2(1-\mu))$ に従うこととした．ここでの μ はアームの利得の期待値である．それぞれの設定では実験を 1 000 回行い平均をとった．$\bar{X}_{i,N_i}$ は始め，0 とした．softmax の温度パラメータは $T = 1$ とし，epsilon-greedy のパラメータは $\varepsilon = 0.1$ とした．MM のパラメータは $\lambda = 0.3$ と 0.7，提案手法のパラメータは $c = 1, 2, 4$ とした．

まず，バイアスや分散がサンプルが増える毎にどのように減るのかを精査した．10 本のアームがあり，2 本は高い期待値 0.8，残りは低い期待値 0.1 であるという設定で実験をした．

softmax，UCB1，epsilon-greedy に従って，サンプルを集めた場合の結果について図 6.1, 図 6.2，図 6.3 に示した．加えて，参考のため，一様な確率で各アームを引いてサンプルを集めた場合（uniform）の結果も図 6.4 に示した．全てのサンプリング方法において，理論と同様に ME と AVE はそれぞれ最も正と負の方向に大きいバイアスとなった．また，MM は ME と AVE の間であり，MM は 2.5 での分析から予想される通り，特に softmax, uniform で収束し損なっている．SWE のバイアスは MM と同様に AVE と ME の間であったが 0 に収束するという意味では，より好ましい性質がある．パラメータ c の値として，2 や 4 といったより大きな値を使うと，SWE のバイアスはより速く収束したが，分散は少し高くなった．DE の分散は他と比べて少し高くなった．これは DE がサンプルを分け，結果的に利得の推定に使えるサンプルサイズが減るためであると予想される．epsilon-greedy はこの設定にはあまり向かない．全てのアームの平均利得の初期値が 0 であり，利得はほとんど確かに正であるため，最初に引いたアームを優先するようになる．そしてそれは少なくとも他のアームが引かれる（確率 $\varepsilon\frac{K-1}{K}$）まで続くため，長い間，最初に引いたアームを引くことになる．このような状況下でも，SWE は良い推定値である．

推定量の様々な設定下でのロバストさを測るため，アームの数 10 で固定し，高い期待値（$\mu = 0.8$）のアームの数を変化させ，それぞれの設定下で各種の値を計測した．この実験では，サンプリング手法として，UCB1 を用いた．結果 (図 6.5) は高い期待値のアームの数により，ほとんど全ての推定量でバイアスが大きな影響を受けること示した．SWE で $c = 2$ や 4 としたものと DE だけが，バイアスが安定して 0 付近となった．DE の分散が他の推定量のものよりも高いため，SWE がこの実験では良い成績であったと言える．

また，高い期待値と低い期待値のアームの割合 2 : 3 に保ったまま．アームの数を増

やした場合で実験し，別の尺度での推定量のロバストさを計測する．結果は図 6.6 に示した．アームの数が増えるにつれて，多くの推定量，特に ME と AVE で，正と負の方向に大きなバイアスになる傾向が見られた．DE，MM（$= 0.7$），SWE（$c = 2$）がバイアスを 0 に近く保つことが出来た．また，DE の分散は他のものよりも高い傾向が見られた．

6.4 まとめ

本研究では，multi-armed bandit の設定で，平均利得の最大値を見積もる方法について議論し，新しい推定量 Simplified Weighted Estimator（SWE）を提案した．SWE では，アーム i が最善であるとした仮定した時，i の平均利得とその最大値との差の尤もらしさをもとに，サンプルの各要素に重みづけし，重み付き和をとり，その値を推定値とする方法である．この手法は，負の Bias である単純にサンプルの平均を推定値とする方法（AVE）と正のバイアスである平均利得の最大値を推定値とする方法（ME）の間の性質を持ち，始めは前者に近く，十分にサンプルが集まると後者に近くなるという手法であり，パラメータ c を導入し，その近づく速さを調整する．理論的な解析により，一部のアームのサンプルサイズが極端な割合で増えることがなく，全てのアームのサンプルサイズが十分に大きければ，提案手法のバイアスは 0 に収束することを示した．実験では UCB1，softmax，epsilon-greedy という 3 つの代表的なアルゴリズムに対する安定性，期待値が高いアームが増えることに対する安定性，アームが増えることに対する安定性を見た．実験で MSE を計測した結果，設定しだいで，既存手法が上回る場合があるものの，様々な設定に対する安定して MSE が低いという点で提案手法（パラメータ $c = 2$）は優れていた．

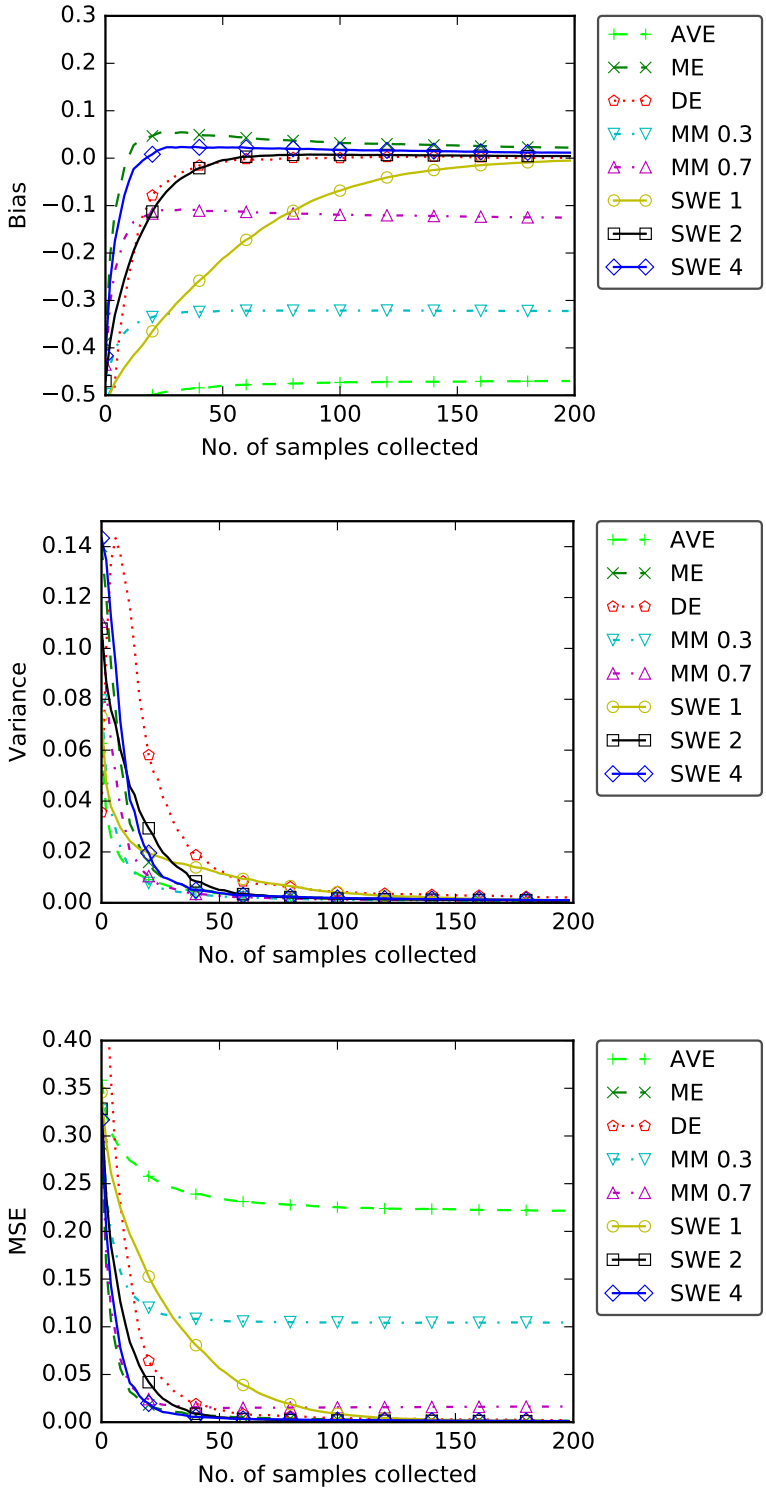


図 6.1: 左から順に Softmax に従って，サンプルサイズを増やした時の推定量のバイアス，分散，MSE．バイアスと MSE について SWE は素早く 0 に近づいていることが分かる．他の推定量では AVE $\ll$ MM $\ll 0 <$ ME という関係であった．DE はサンプルサイズが小さい時に高い分散の値であることが分かる．

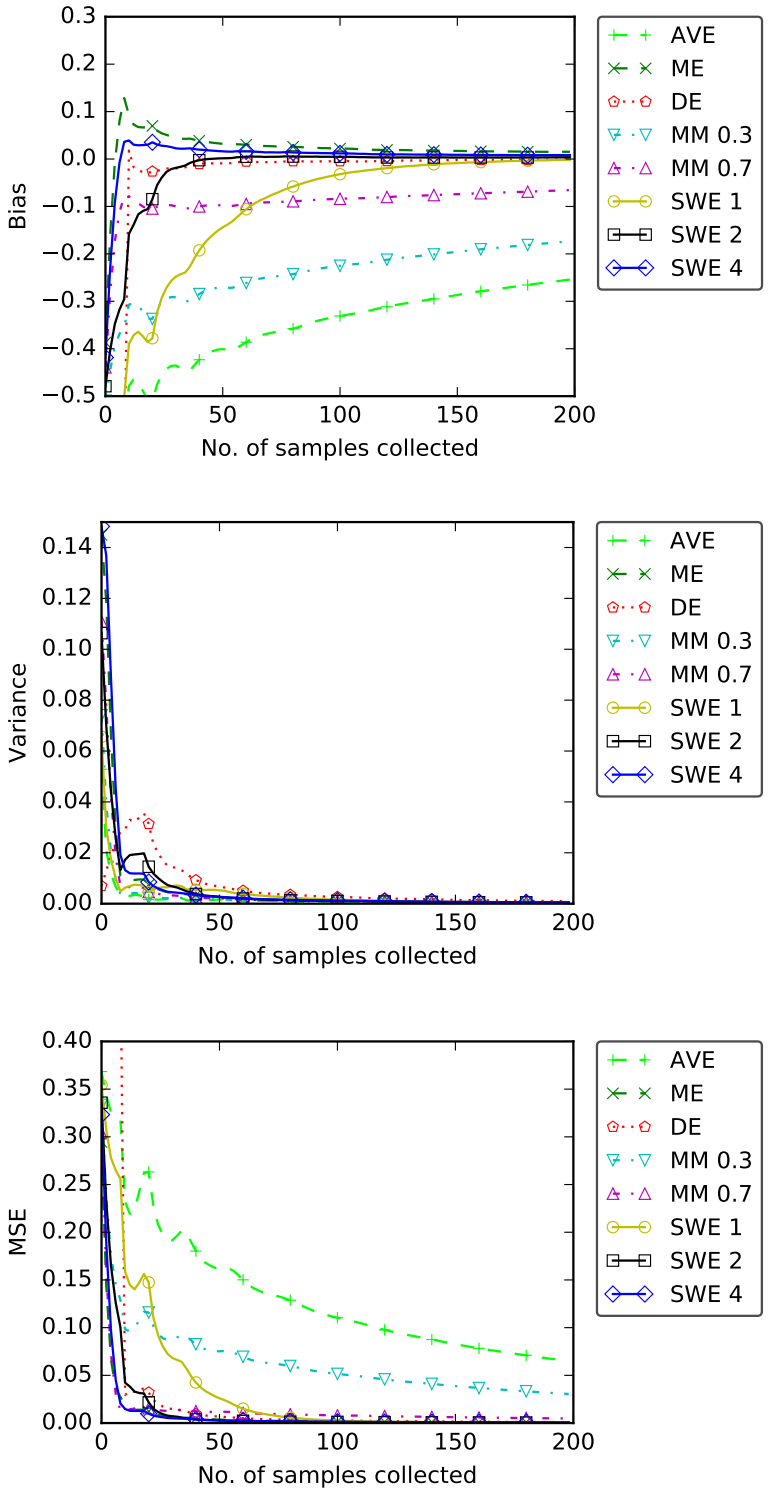


図 6.2: 左から順に UCB1 に従ってサンプルサイズを増やした時の推定量のバイアス，分散，MSE．図 6.1 と同様，AVE と MM は大きなバイアスであることが分かる．

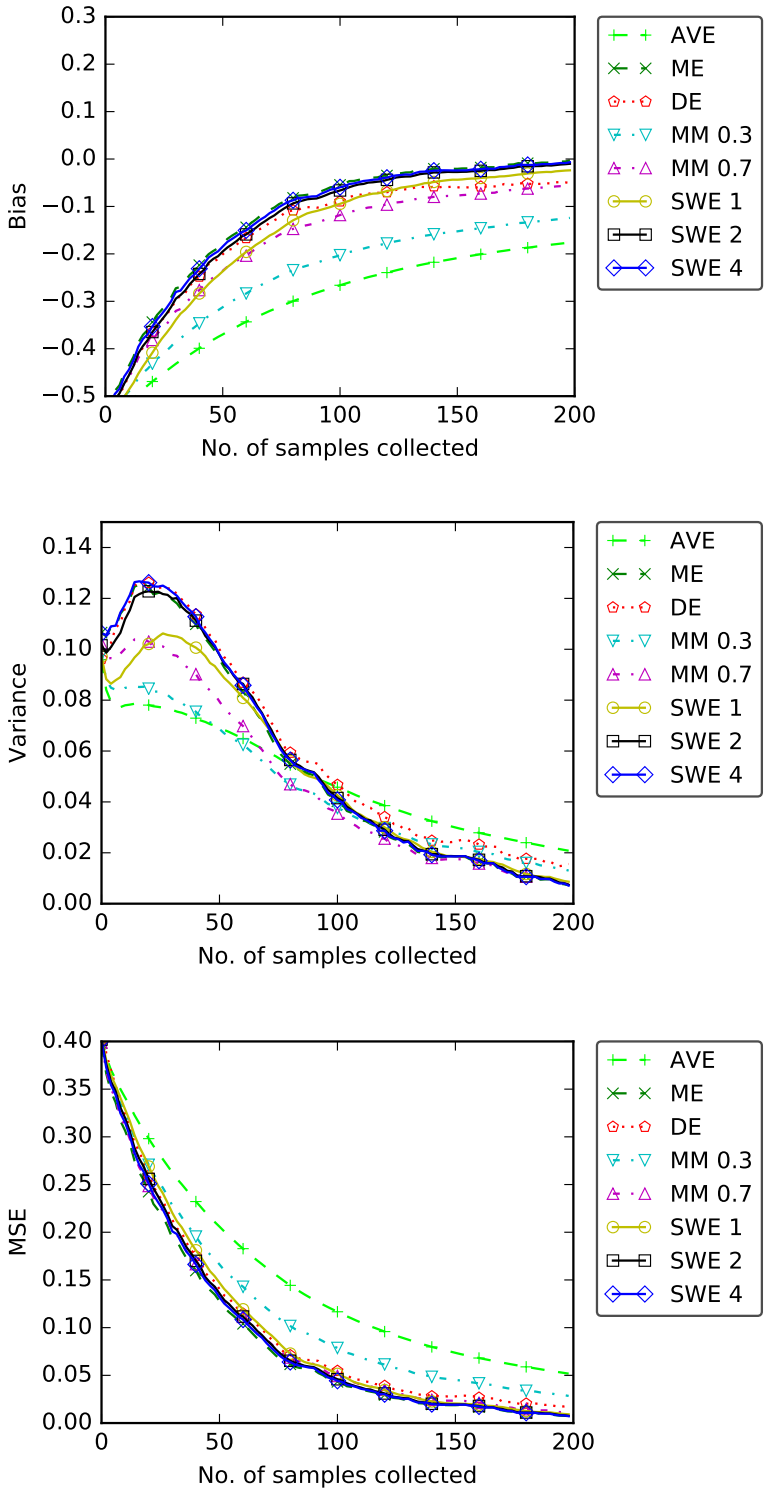


図 6.3: 左から順に epsilon-greedy に従ってサンプルサイズを増やした時の推定量のバイアス，分散，MSE.

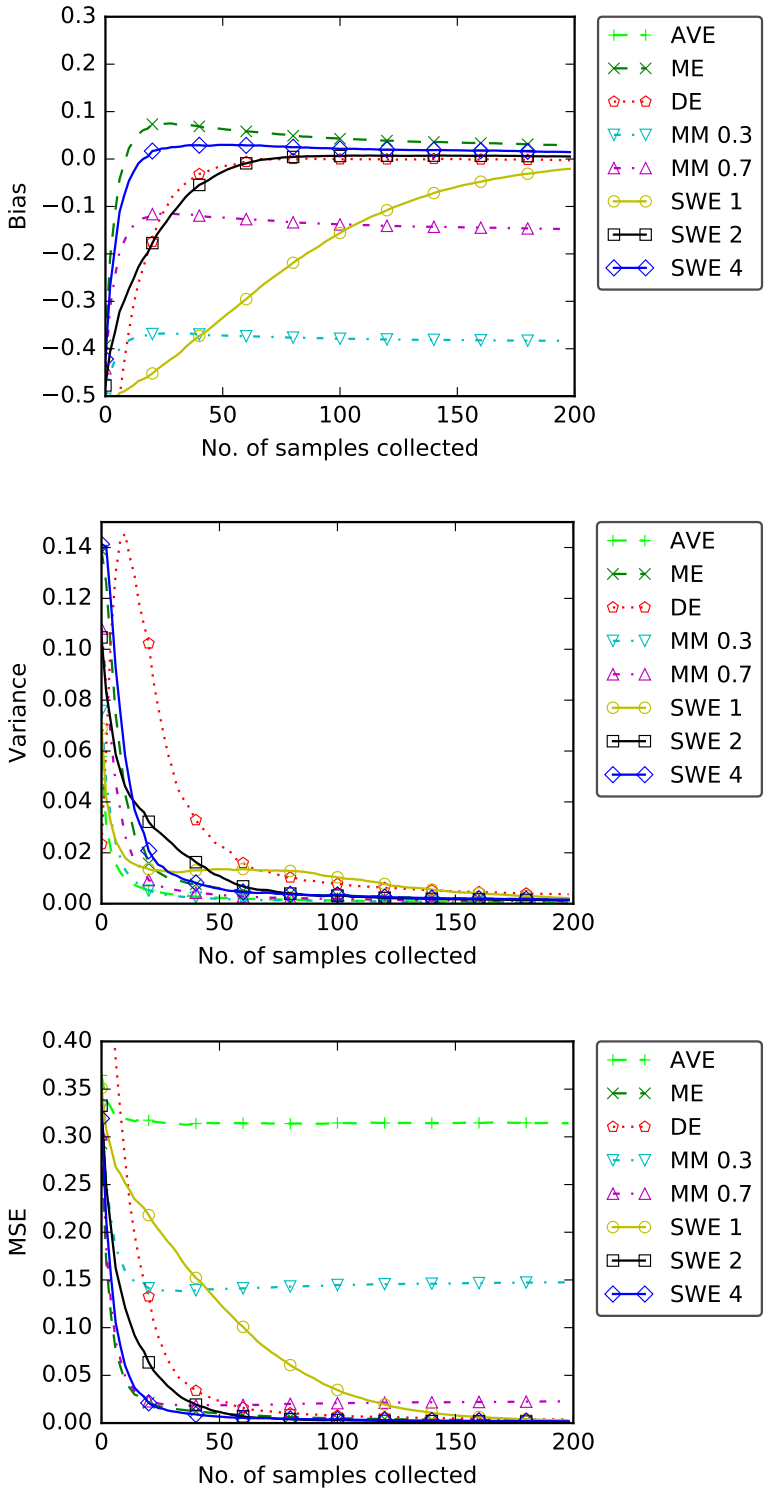


図 6.4: 左から順に uniform に従ってサンプルサイズを増やした時の推定量のバイアス，分散，MSE．図 6.1 と同様，AVE と MM は大きなバイアスであることが分かる．

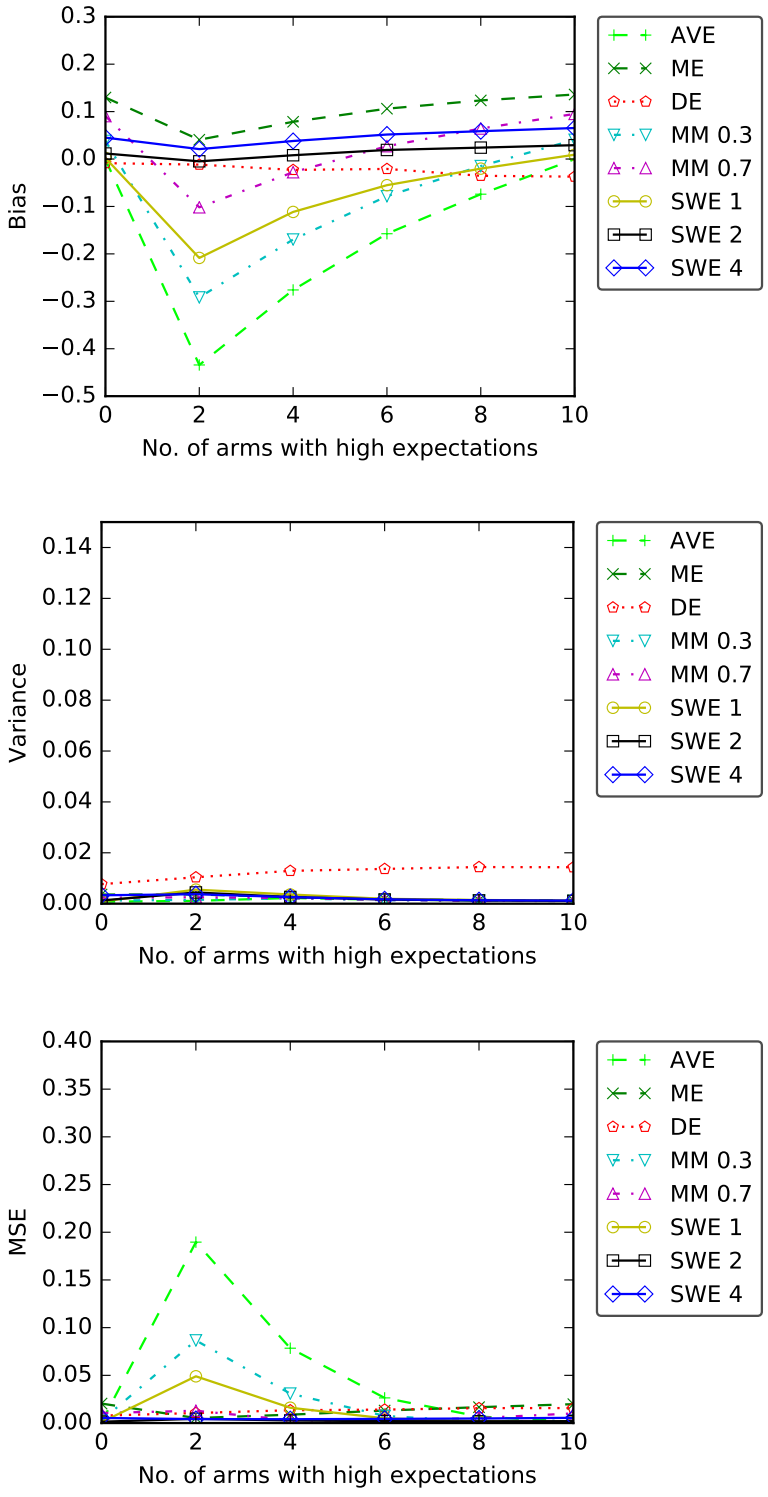


図 6.5: $\mu = 0.8$ のアームの数に対する推定値：左からそれぞれバイアス，分散，MSE（UCB1 で 40 回プレイ時）.

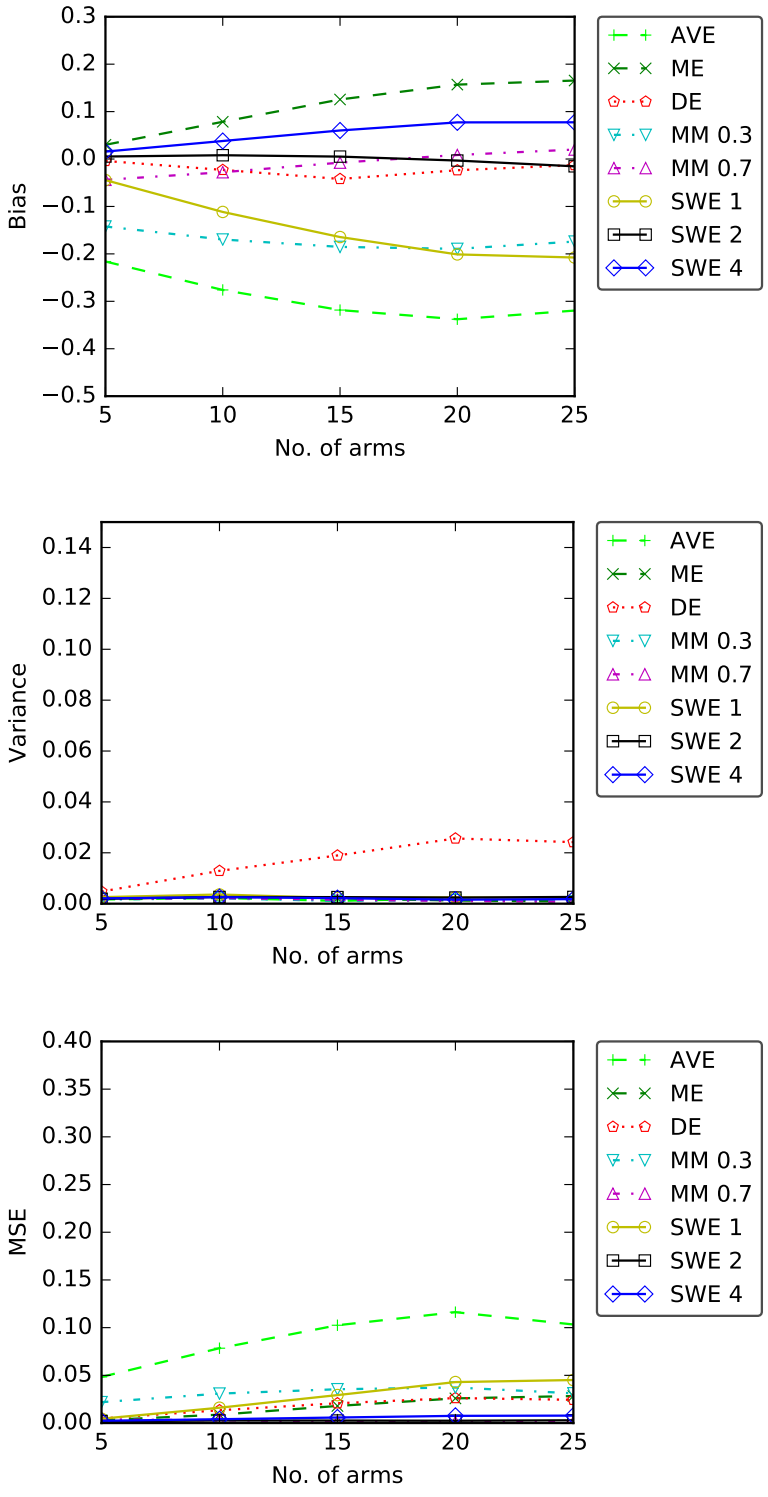


図 6.6: アームの数に対する各種の値の依存度合い．左からバイアス，分散，MSE（ UCB1 に従って，40 回プレイ時 ）．

第7章　MCTSにおける期待値の最大値の推定量の改善

MCTS の代表的なアルゴリズム UCT はシミュレーションを複数回行い，その利得の平均に基づき最善手を判別する．しかしながら，最善手を判別するために，子の局面の利得の理論値の最大値を推定し，比較する必要があり，最善の判別のための推定を，シミュレーションの利得の平均値で行うことには改善の余地がある．本章では，平均値の代わりに，6 章で提案した，期待値の最大値の推定量 Simplified Weighted Estimator を応用した探索手法を提案する．二人零和完全情報ゲームを対象に，実験を通じて，推定値の正確さ，ばらつき，最善手を選べるかの観点からその効果を議論する．実験の結果，序中盤に近い局面のモデルでは安定して効果があり，終盤に近い局面のモデルでは一部の状況で効果があった．この結果は，前者のモデルの方が MAB に近い性質を持つことから，事前の予想と整合するものである．

尚，本章の内容は発表済みの論文 [81] に記した実験の他に，追加実験を行い，加筆，編集したものである．

7.1　背景

モンテカルロ木探索 (MCTS) は，囲碁や General Game Playing 等で成功したアルゴリズム [8,61] で，ゲーム分野での主要な探索アルゴリズムの一つである．MCTS の代表的なアルゴリズム UCT はシミュレーションを繰り返し行い，その平均利得が良い手を優先して深く読む．より詳しく述べると，UCT では，各手の利得の理論値をシミュレーションの平均利得で推定するということである．しかしながら，各手の利得の推定を平均で行うことが最良かは明らかでない．

ゲームでの探索の主要な目的は良い手（特に最善手）を素早く見つけることである．最善手は，以後互いに最善を尽くすという仮定のもとでの利得の期待値が最大の手であり，それを判別するためには利得の期待値の最大値を十分な精度で推定する必要がある．UCT は，手の利得の推定を平均によって行うが，有望そうな葉からより多くのシミュレーションを行うという性質のため，漸近的には，正しい値に収束する．しかしながら，6 章の実験で確かめたように，少数の手の期待値が高く，多数の手の期待値が低い場合，多数の手の期待値の影響で推定値が低くなり，正しい値になるまでに時間がかかる [35] ため，実用的な観点から見て，改善の余地がある．

例えば，一人ゲームでは，Mario AI Benchmark において，UCT の推定値を単にシ

ミュレーションの平均利得でなく，子の内で最大の利得の値を重視するという工夫をすることで，計算資源が一定のもとで行動選択が改善したという研究がある [36]．また，UCT での節点の推定値において，分散も用いて評価し，はっきりと推定値が悪い場合に，その値を親の評価に用いないという工夫をして，Sailing [42] で改善を示した研究 [9] や，加えて，二人ゲームでは，囲碁において，探索アルゴリズムは UCT ではないものの，平均をとるよりも最大をとるほうが，シミュレーションが十分なされた場合に推定値の誤差が少ないという実験結果 [17] もある．また，最近選んだ子の推定値をより重視して親の推定値を計算するという工夫をした Accerated UCT [29] は，囲碁等のゲームで UCT より優れた結果となったという研究もある．

本章では引き続き二人零和有限確定完全情報ゲームを対象に，MCTS における利得の推定の方法とそれによる MCTS の性能の改善について議論する．特に，6 章で提案した推定量，Simplified Weighted Estimator [35] を MCTS に組み合わせた手法を新たに提案する．SWE は 6 章で確かめたように，MAB では効果があったが，木の設定では不明である．そこで MAB からの設定の離れ具合が異なる，2 種類の木で実験を行った．1 つはリーフが確率的に利得を返す設定で，現実のゲームでは序中盤に近い状況に重点をおいたモデルに対応する．もう 1 つはリーフが確定的に利得を返す設定で，終盤に近い局面のモデルに対応する．実験では，提案手法と既存手法の性能について，最善手を見つけるまでの早さと，その推定値が正確かの観点から調査する．

7.2 UCTSWE

UCT での手の利得の推定はシミュレーションの利得の平均で行っているが，より優れた最大値の推定量を使うことで改善出来ると期待される．それは，以下の理由による．簡単のため，特に局面の遷移が確定的（局面とそこで指した手に対して，次の局面が一意に決まる）ゲームを対象に説明する．まず，最善手は理論値が最も良い手であり，最善手の判別のために理論値を見積もる必要がある．そのため，手の利得の推定の目標はその手の利得の理論値を正確に見積もることである．そして，手の利得の理論値は次の局面での候補手の利得の理論値の最大値である．従って，利得の期待値の最大値を正確に見積もることが出来れば，性能を改善出来ると期待される．

本章では，6 章で提案し，多腕バンディット問題（MAB）で優れた性能を示した SWE を UCT に組み合わせた手法を提案する．具体的には，提案手法では，既存手法 UCT での手の利得の推定方法を平均利得（AVE）から，SWE に変え，子の推定値から SWE に従って，親の推定値を計算する．

incremental random tree を MCTS で探索することと 6 章での MAB の設定でアームを引くことは異なる点があり，重要な点を 2 点挙げる．1 つは，MCTS では，利得の推定値に基づいてシミュレーションする葉を決める一方で，後者では，アームの平均利得に基づいて引くアームを決めるので，平均利得はアームの選択に影響しても，平均利得から計算される，推定値の大小は影響しないという点である．もう 1 つは，MCTS

では，利得は独立同一分布からほとんど得られないということである．推定最善を重視し推定値を算出する方法として，6 章で比較したように様々な方法が考えられるが，例えば ME を使うと問題が生じる．特に本研究の葉の展開方法では，同じ葉からシミュレーションは 2 度行わないため，全ての節点で ME による推定を行うと，各節点での利得の推定値は葉でのシミュレーション 1 回の利得という，ばらつき大きい値のミニマックス値になる．それを避けるために，最初は AVE で評価し，十分に利得が集まった節点から ME で評価する等 4 章での Hybrid MCTS [34,53] の枠組みで応用することも考えられるが，その場合，その節点が Min 手番，Max 手番かによって，推定値が負，正に偏るという別の問題も生じる．具体的には，兄弟間で ME で推定されたものと AVE で推定されたものがあると推定値が著しく不均一になり安定しないという問題である．本章では，比較のため，独立同一分布で何度も利得が得られるという意味では多腕バンディット問題により近いゲーム木（ 7.3 節で導入）も扱う．

提案手法では，UCT で AVE の代わりに SWE を使い，推定最善を重視する．SWE を使うことで推定最善の重みを，訪問数が大きく（十分に利得が集まる）につれて滑らかに増やすことができる．但し，MCTS の場合，MAB とは異なり，ある手を選んだ場合に得られる利得の分布は定常ではない．そのため， 6.2 節の補題 1 が成り立たない．この点に関して，提案手法では， 2.6 節で紹介した，推定値がその期待値の付近から離れることは稀であるという，UCT における C_p に関する想定に基づき，SWE の重みの計算に適用する．具体的には，提案手法では，SWE における子節点 i の重み $\tilde{w}_i$ として

$$\tilde{w}_i = \begin{cases} \exp\left(-\left(\frac{cd_{m,i}\sqrt{N_m N_i}}{C_p\sqrt{N_m}+\sqrt{N_i}}\right)^2\right) & (i \neq m), \\ 1 & (i = m), \end{cases}$$

を使用する．UCT に SWE を適用するにあたり，このように重みを修正した．また，2.6 節で紹介した Accerated UCT と同様に，展開前のシミュレーション結果については，仮想的な子を作って利用した．

7.3 確率的な利得が得られるゲームのモデル

本節では， 2.2.1 節で紹介した incremental random tree と比べて，より多腕バンディット問題に近いと考えられるゲーム木について紹介する．このゲーム木は大雑把には，incremental random tree の終端節点で確率的に利得が得られるとしたものである．2.2.4 節の Markov decision process と比較すると，状態の遷移が確率的でないという意味では，より単純であり，二人ゲームであるという意味では，より複雑である．

考案した木についてより詳しく述べる．この木では，incremental random tree のように，各辺に値を割り当て，終端節点でのスコアを計算する．終端節点 i でのスコアを s_i とする．全ての終端節点について，取りうるスコアの最大値，最小値をそれぞれ U, L とする．終端節点 i で得られる，Max にとっての利得を r_i とすると，r_i が $-1, 1$ になる確率をそれぞれ $\mathrm{P}(r_i = 1) := \frac{s_i - L}{U - L}$，$\mathrm{P}(r_i = -1) := \frac{U - s_i}{U - L}$ で定める．

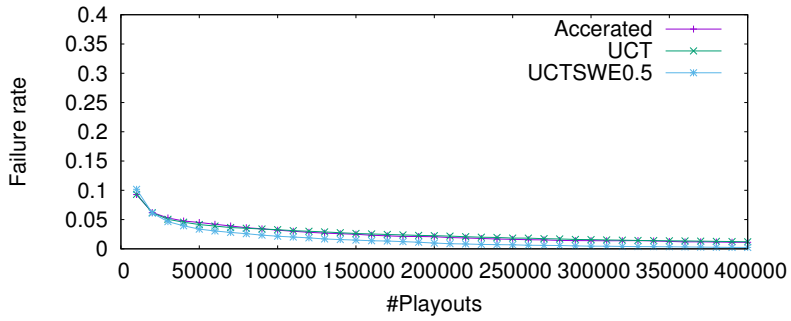


図 7.1: 確率的な利得が得られる木，$C_p = \sqrt{2}$,（分枝数，深さ）$= (8, 2)$ の誤答率の推移

木はスコアに基づき，終端節点で得られる利得の期待値を決めているため，incremental random tree と同様に，最善手が分析者に予め分かるという長所を持つ．特にこの木での根での利得の期待値の理論値は 0 である．

7.4 実験

本実験では，SWE による推定値を使った UCT (UCTSWE) について，ゲーム木モデルでの誤答率と利得の推定値を実測し，議論する．誤答率は最善手を選べなかった場合を誤答とした，全ての試行に対する誤答の割合である．推定値は全ての試行に対して，平均をとった．UCT と Accerated UCT を比較対象とした．Accerated UCT のパラメータ λ の値は原論文 [29] での囲碁，Havannah の対戦実験で最も良い成績であった $\lambda = 0.9999$ とした．各アルゴリズムで，探索終了後，最善手として推定値が最良の手を選ぶ方法も考えられるが [10]，ロバストと経験的に知られている，訪問数最大の手を選ぶという方法を採用した．また，ゲーム木は 200 本生成し，それぞれゲーム木に対し 200 回探索した．尚，本研究では，引き分けは Max（根で手番のプレイヤー）の勝ちとして，Max（Min）プレイヤーの次善手の辺の値は値域 $[-128, -1]$（$[1, 128]$）の一様分布に基づき決めている．

7.4.1 確率的な利得が得られる木

7.3 節で導入した，確率的な利得が得られるモデルで実験を行った．実験は，木の（分枝数，深さ）を (8, 2)，(8, 4)，(16, 2)，(16, 4)，(32, 2) として行った．尚，この実験では，SWE のパラメータは 0.5 とし，$C_p = \sqrt{2}$ とした．

各設定での誤答率の結果を図 7.1，7.2，7.3，7.4，7.5 に示した．縦軸を誤答率とし，横軸をプレイアウト数とした．

誤答率について，おおよその傾向として，提案手法は，プレイアウト数が少ない間は UCT 等より誤答率が高いものの，やがて提案手法の誤答率が最も小さくなった．設定に関する傾向として，分枝数が大きいほど，誤答率が下がるまでに必要なプレイアウト

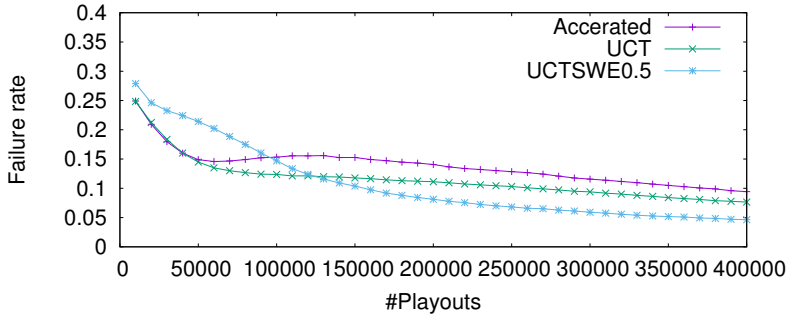


図 7.2: 確率的な利得が得られる木，$C_p = \sqrt{2}$,（分枝数，深さ）$= (8, 4)$ の誤答率の推移

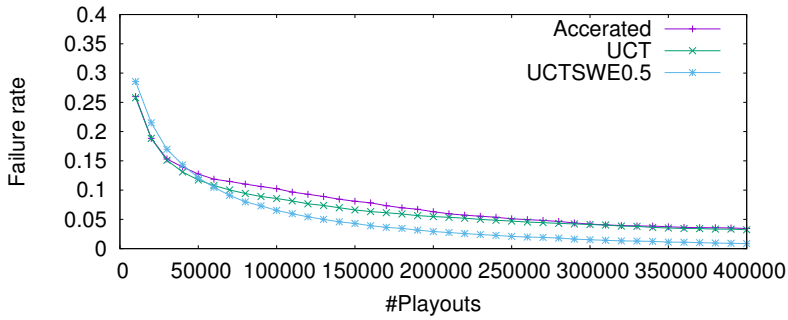


図 7.3: 確率的な利得が得られる木，$C_p = \sqrt{2}$,（分枝数，深さ）$= (16, 2)$ の誤答率の推移

数が多くなり，木の深さが深いほど，プレイアウト数が少ない時の提案手法の誤答率が高くなった．

各アルゴリズムで探索した際の，各節点の利得の推定値の推移を計測した結果を図 7.6,7.7,7.9,7.8,7.10 に示した．利得の推定値は Max プレイヤー視点での利得（Max プレイヤーが勝ちなら 1，負けなら -1）で表記した．図中の root, opt, best, best's best は各節点の推定値を表し，節点はそれぞれ，根，根の最善の子，根の推定最善（子の中で推定値最大）の子，推定最善の子の推定最善の子を意味する．縦軸を各節点での利得の推定値とし，横軸をプレイアウト数とした．尚，各節点の推定値は 0 が理論値である．全体として，提案手法，Accerated UCT，UCT の順で，推定値はより素早く正しい値に収束する傾向がみられた．

7.4.2　確定的な利得が得られる木

次に，終端節点で確定的な利得が得られる 2 種類の木で実験を行った． 1 つは incremental random tree である．もうひとつは，incremental random tree で枝刈りを行った場合に相当する木である．

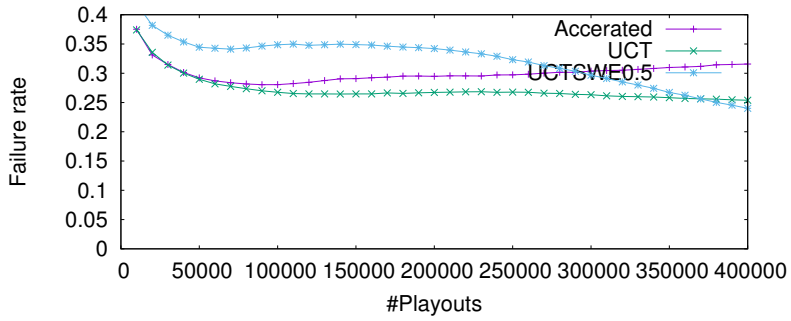


図 7.4: 確率的な利得が得られる木，$C_p = \sqrt{2}$,（分枝数，深さ）$= (16, 4)$ の誤答率の推移

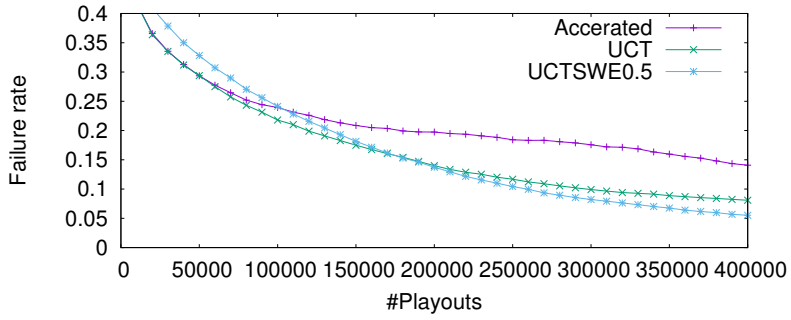


図 7.5: 確率的な利得が得られる木，$C_p = \sqrt{2}$,（分枝数，深さ）$= (32, 2)$ の誤答率の推移

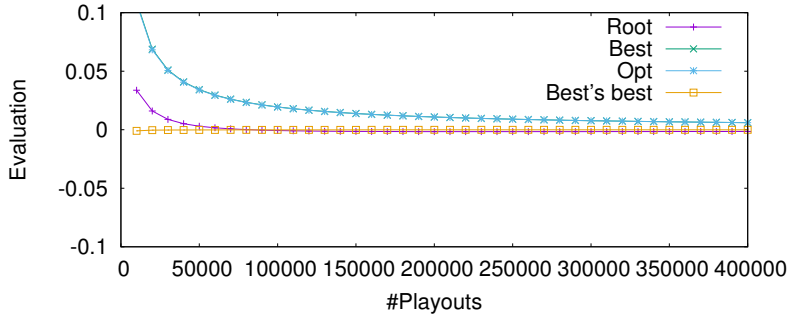
Incremental Random Tree

まず予備実験として，incremental random tree を UCT で探索し，利得の推定値を計測した．そして，推定最善の子の推定値を重視することで，推定値が真の値に近づく余地があるかを検証する．より具体的には，実験で，利得の推定値が利得の理論値からどれだけ離れているかをプレイアウト数毎に計測し，推移を見た．UCT の探索には 2.6 節で紹介した MCTS-Solver を併用した．もし，各節点の勝ち負けが確定したら，その利得の推定値を正しい値（勝ち負けならそれぞれ $1, 0, -1$）に書き換えた [1]．5 章で導入した MCTS-Solver の変種 Replacement MCTS-Solver (Replace) [80] を UCT に組み合わせた場合 (UCTReplace) の推定値も計測した．この手法では，勝ち負けが確定したら，その節点の値を正しい値に書き換えるだけでなく，書き換えた節点の訪問数分だけ，先祖の値も書き換える．

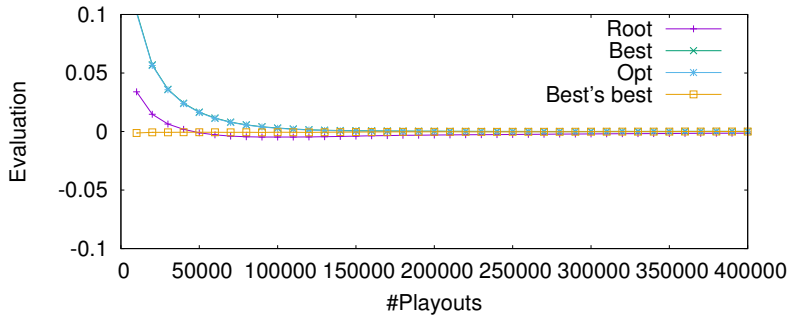
実際に UCT で探索した際の，各節点の利得の推定値の推移を計測した結果が図 7.11 である．本節の木では，各節点の推定値は 1 が理論値であり，その値により早く近づく方が望ましい．尚，利得の推定値は区間 $[-1, 1]$ 中の値をとるが，実験結果は，正の値を取ることがほとんどであったので負の部分は省いた．また，全ての試行において，8000

[1] 5 章で議論したように，この書き換えは祖先の推定値には影響しない．この書き換えの意図は節点の勝ち負けが確定が確定した際にその節点の利得の推定値として自然な値を計測することにある．

(a) UCT



(b) Accerated UCT $\lambda = 0.9999$



(c) UCTSWE $c = 0.5$

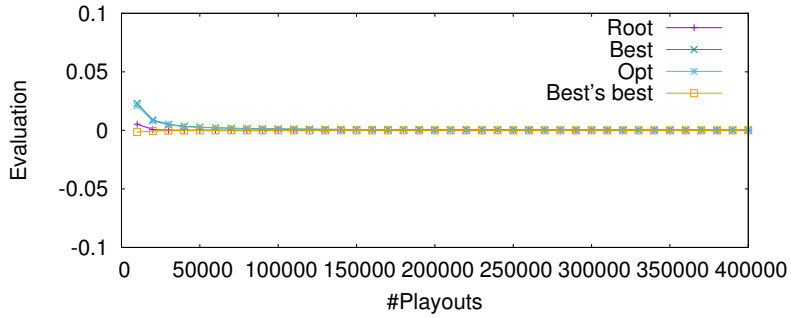


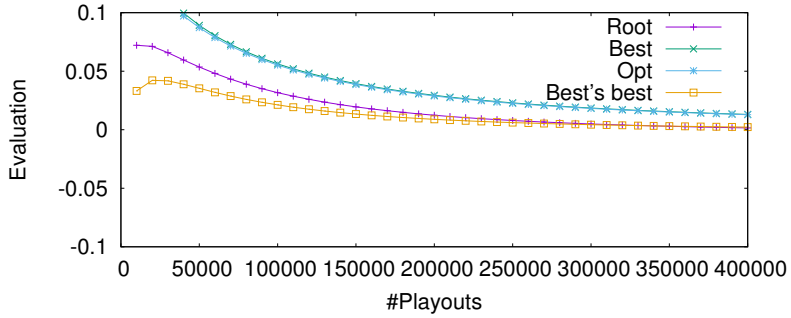
図 7.6: 確率的な利得が得られる木，$C_p = \sqrt{2}$,（分枝数，深さ）$= (8, 2)$ での利得の推定値の推移

プレイアウト時点で，MCTS-Solver により根の勝ち負けが判明することはなかった．UCT において，親の利得の推定値は子の推定値の重み付き平均であるので，基本的には推定値は best's best $\leq$ best であり，root $\leq$ best である [2]．実験の結果，MCTS-Solver の種類にかかわらず，プレイアウト数が増えると best's best の推定値は root よりも利得の理論値である 1 に近くなった．つまり，プレイアウトが多い時は推定最善の手をより重視した方が利得の理論値に近い推定値となる．これは，推定最善の子をより重視して，親の推定値を算出するという方向性が有望であることを示している．

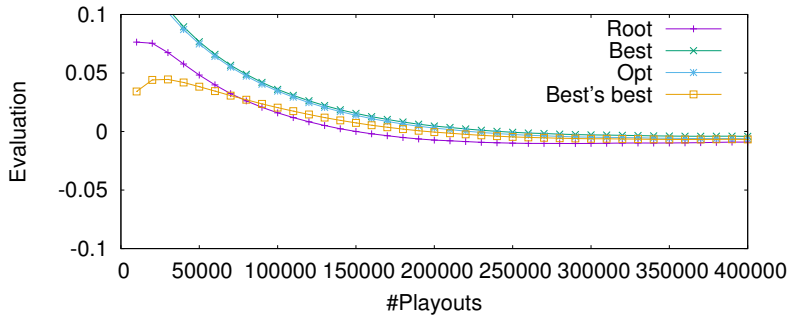
まず予備実験と同様の設定と（分枝数，深さ）を $(8, 12)$ とした設定で，各アルゴリズ

[2] best's best $>$ best となる例外は，MCTS-Solver の挙動と勝敗確定時の書き換えによって生じる．簡単のため，best $=$ opt とし，opt の子節点は A，B の 2 つだけとする．子節点 A，B でそれぞれ Min の勝ち，負けが多く出ているとする．その時，opt の推定値は B の推定値以下，かつ A が opt の推定最善の子となる．そして，A が Min の負け（利得 1）と確定した場合．opt の最善の子は B になり，前述の例外が起きる．尚，Replace では A の勝敗が確定すると opt を含む A の祖先も書き換えるため，この例外は起こらない

(a) UCT



(b) Accerated UCT $\lambda = 0.9999$



(c) UCTSWE $c = 0.5$

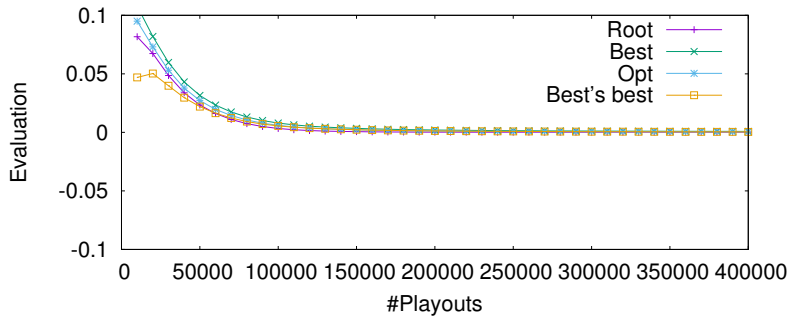


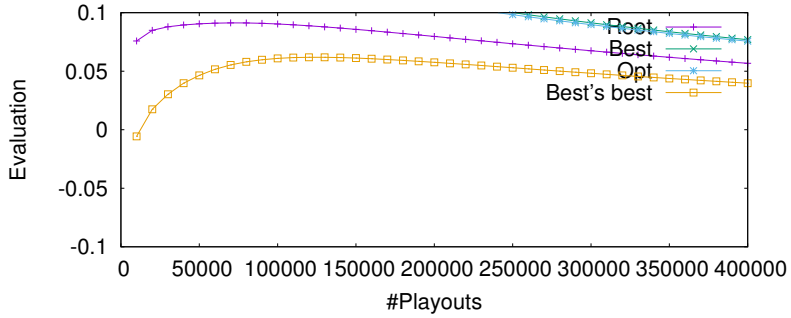
図 7.7: 確率的な利得が得られる木，$C_p = \sqrt{2}$,（分枝数，深さ）$= (8, 4)$ での利得の推定値の推移

ムの誤答率を計測した．以下の実験では，SWE のパラメータは $c = 0.01, 0.05, 0.1, 0.5$ の 4 種類とし，C_p の値を $\sqrt{2}$ を基準として $1, 1.5, 2$ 倍して計測した．尚，誤答率については，以下の全ての実験で $C_p = \sqrt{2}$ とした時が 8000 プレイアウトでの UCT の誤答率が最も低かった．また，SWE と組み合わせる MCTS-Solver として，Replace を使った．従って，$c = 0$ とすると提案手法は UCTReplace と同じになる．

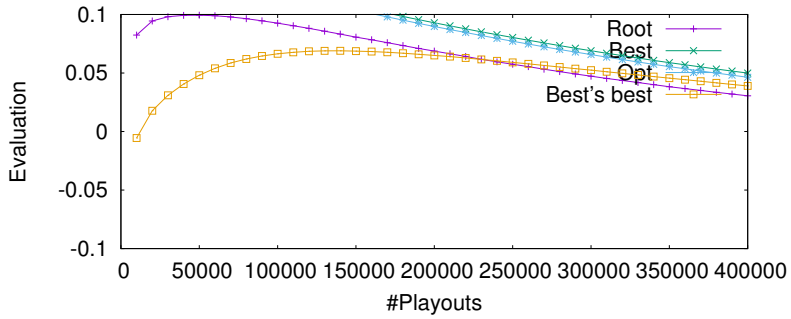
（分枝数，深さ）が (8, 6) の際の実験結果（図 7.12(a)）では誤答率は，SWE のパラメータ c を適切に調整しても，UCT の誤答率からほとんど改善しなかった．特に $c = 0.5$ では悪化した．また，確率的な利得が得られる木の実験と比べて，プレイアウトの数が少ないため，UCT と Accerated UCT の間に差はほとんどつかなかった．尚，8000 プレイアウト行うまでに，探索木の一部が終端節点まで達したが，根の勝ち負けが確定することは無かった．

次に（分枝数，深さ）を (8, 12) に設定した，より大きな木での結果を（図 7.12(b),

(a) UCT



(b) Accerated UCT $\lambda = 0.9999$



(c) UCTSWE $c = 0.5$

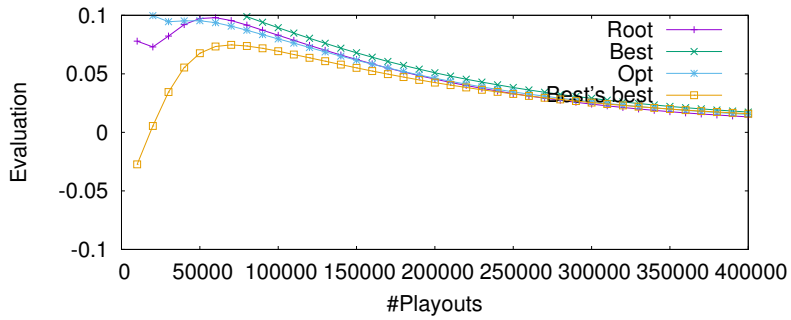
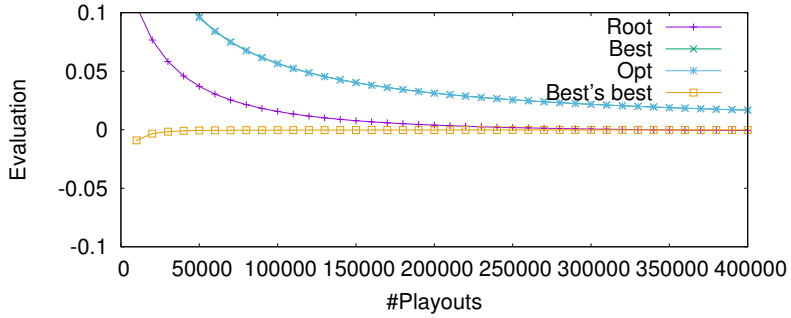


図 7.8: 確率的な利得が得られる木，$C_p = \sqrt{2}$,（分枝数，深さ）$= (16, 4)$ での利得の推定値の推移

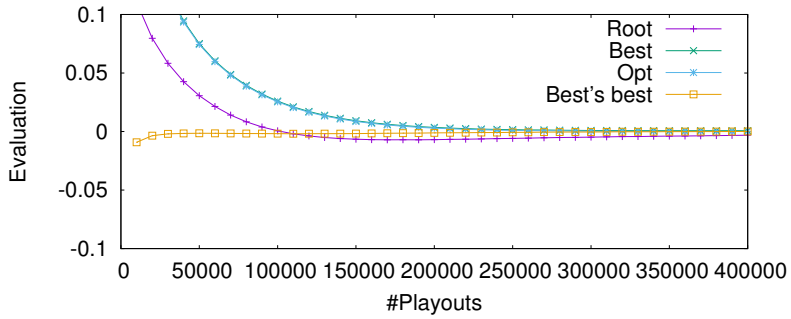
図 7.12(c)）に示した．尚，この設定では，全ての探索で，探索木の葉が終端節点まで達したことは一度もなく，従って，MCTS-Solver の影響が無いため，UCTReplace は UCT と同じである．（分枝数，深さ）が 8 − 12 での結果は C_p が 2 倍の時（図 7.11(c)）では多少改善したが，C_p が 1 倍の時（図 7.11(b)）は c を調整してもほとんど改善しない．また，$c = 0.5$ では悪化した．

つづいて，各手法での推定値について調査した．UCT との違いがよく分かるように，SWE のパラメータを調節した中で最も大きい $c = 0.5$ の結果を載せる．図 7.11 と同じ設定での UCTSWE の推定値のプレイアウト数による変化を図 7.13 に示した．図 7.13 で各推定値は UCTReplace (図 7.10(b)) のものより互いに近くなった．SWE のパラメータ c の推定値に対する影響を調べるため，プレイアウト数 4000 と 8000 での最善手の推定値と次善手の推定値についてさらに詳しく調べた．全探索中の推定値の平均値，及びそれらの標準偏差について表 7.1 に示した．パラメータ c を増やすと，4000 プレ

(a) UCT



(b) Accerated UCT $\lambda = 0.9999$



(c) UCTSWE $c = 0.5$

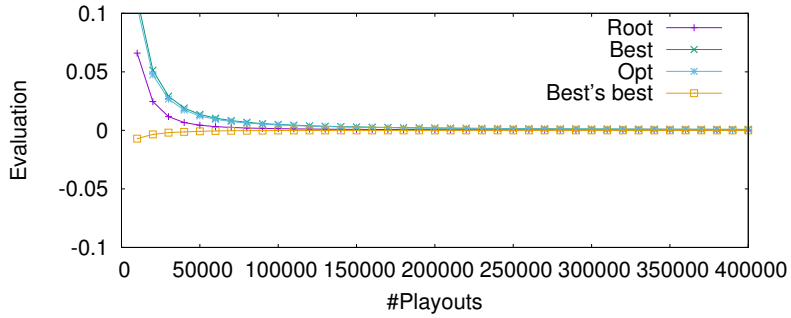
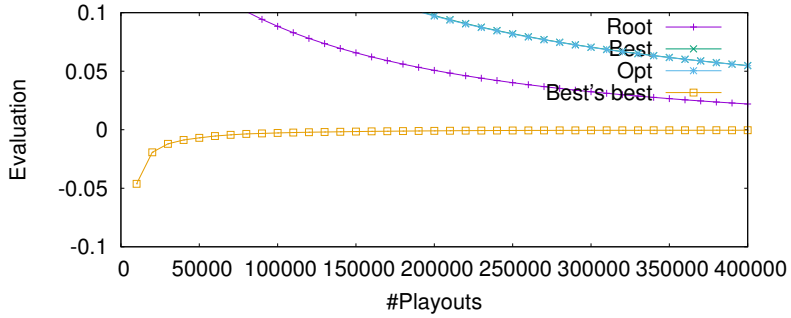


図 7.9: 確率的な利得が得られる木，$C_p = \sqrt{2}$,（分枝数，深さ）$= (16, 2)$ での利得の推定値の推移

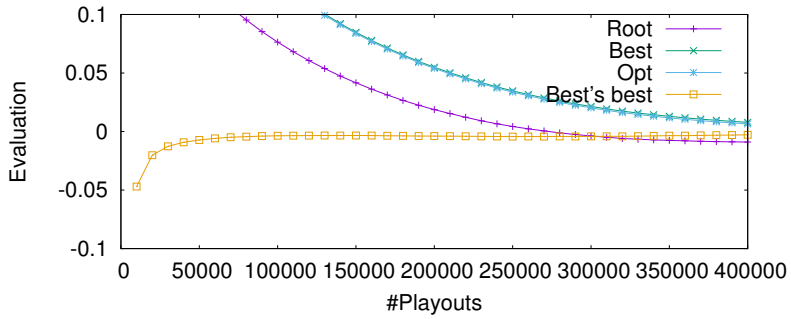
イアウト時の最善手の推定値は変わらないものの，次善手の推定値が低くなり，真の値 -1 に近づいた．8000 プレイアウト時も 4000 プレイアウトと同様の傾向であった．

次に,（分枝数，深さ）を $(8, 12)$ とした場合の推定値の推移について図 7.14 に 4000, 8000 プレイアウト時の最善手と次善手の推定値について表 7.2 に示した．図 7.14 では，図 7.11 や図 7.13 の場合と異なり，プレイアウトを増やしても，最善手の推定値の平均値がほとんど改善しなかった．表 7.2 では，c を大きくするにつれて，次善手だけでなく最善手も推定値が下がる傾向見られた．また，どちらの推定値の標準偏差も上がった．平均的には最善の推定値は次善のものよりも大きくなったが，その場合で標準偏差が大きかったということは，最善手の推定値を次善手の推定値が上回ることも多かったと予想される．そのことが提案手法の性能の悪さの一因と予想される．

(a) UCT



(b) Accerated UCT $\lambda = 0.9999$



(c) UCTSWE $c = 0.5$

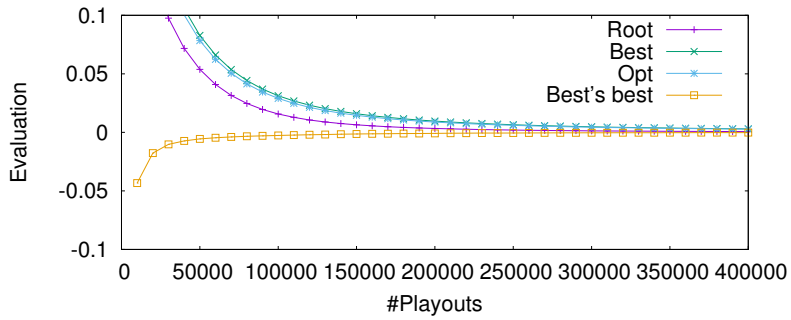
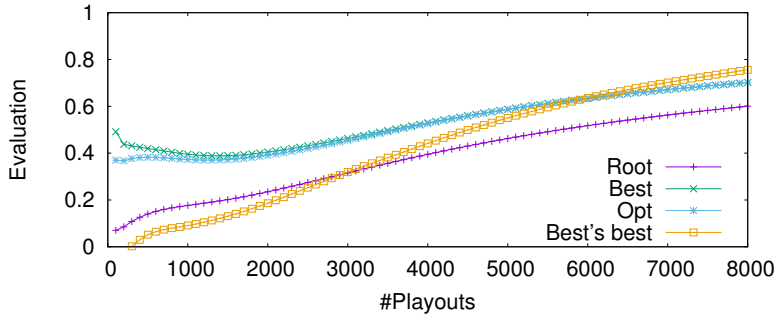


図 7.10: 確率的な利得が得られる木，$C_p = \sqrt{2}$,（分枝数，深さ）$= (32, 2)$ での利得の推定値の推移

特異節点を含む木

推定最善の評価を重視した UCTSWE での改善は incremental random tree では限定的であった．最善を重視するのが良いのはどのような場合か明らかにするため，新たなゲーム木モデルを導入する．このモデルでは， 2.2 節の incremental random tree の変種として，辺にランダムに値を割り当てるだけでなく，ランダムに特異節点を作る．特異節点以降の辺の値はどの辺も 0 とする．つまり，終端節点の勝ち負けを決める辺の値の総和は根節点から辿って特異節点まで和をとったら，それ以降どんな辺を辿っても変わらないという意味で特異節点は擬似的な終端節点である．そのため，特異節点以降でシミュレーションを開始すると互いに最善を尽くした場合の結果が必ず返ってくるという性質がある．実験では，難しさの調節のため，規定深さ以下の節点（本実験では，深さ 3，つまり根から 3 手以内の節点）では特異節点は作らず，また，最善の子節点（最善手を指した後の局面に相当する節点）は特異節点にしないという制約を加えた．それ

(a) UCT + MCTS-Solver



(b) UCT + Replacement MCTS-Solver

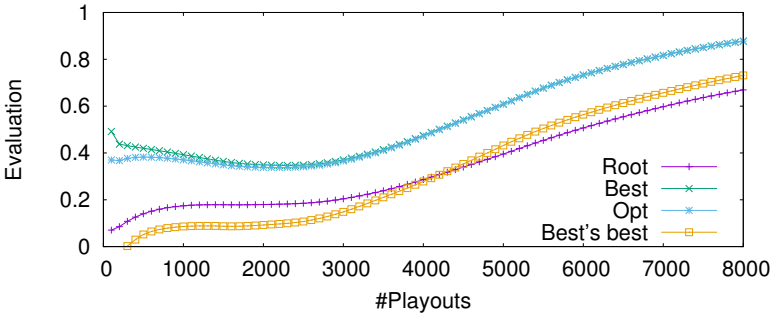


図 7.11: $C_p = \sqrt{2}$, (分枝数，深さ) = $(8, 6)$ での利得の推定値の推移

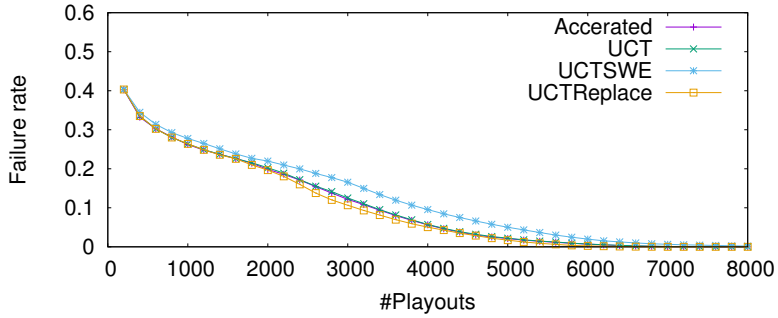
表 7.1: $C_p = \sqrt{2}$, (分枝数，深さ) = $(8, 6)$，各プレイアウト数での利得の推定値（左が最善手，右が次善手，上段が平均値，下段が標準偏差）

Playout		UCT	c = 0	c = 0.1	c = 0.5
4000	Ave	0.5265 -0.1572	0.4719 -0.1851	0.4515 -0.1892	0.4736 -0.2628
	Std. dev	0.1050 0.3092	0.1994 0.2789	0.1856 0.2780	0.2443 0.2615
8000	Ave	0.7016 -0.1585	0.8775 -0.1952	0.8685 -0.2017	0.8523 -0.2996
	Std. dev	0.0557 0.3090	0.0792 0.2691	0.0873 0.2671	0.0973 0.2289

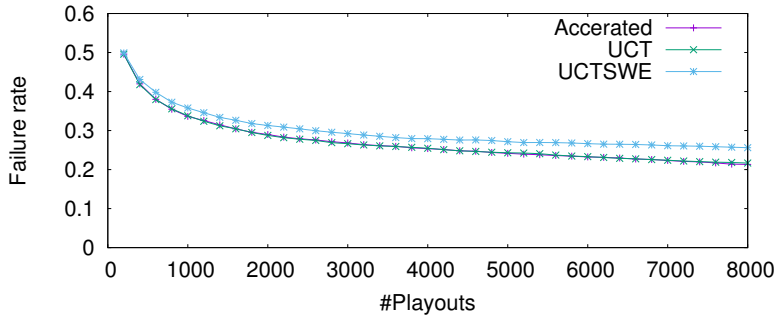
以外の節点は確率 0.1 で特異節点とした．この木を特異節点を含む木とよぶ．この木では特異節点の存在により，はっきりと悪い子節点出来得るため，その推定値に親が影響されないことが incremental random tree と比べてより重要と考えられる．

まず，特異節点を含む木での，各アルゴリズムでの誤答率を計測した．UCT での子の選択を UCB1 ではなく，一様に行う場合，つまり訪問数が同じになるように子を訪問し（ランダムにタイブレークし）た場合の誤答率も計測した．この場合は推定値最良のものを最善手として選んだ．結果を図 7.15 示す．一様に探索した場合，誤答率がほとんど減少しないのに対し，SWE を使うとより速く誤答率が下がった．尚，一様な探索での，プレイアウト数 8000 の時の探索木の葉の最大深さは 5 であり，特異節点を訪問し得る深さまで探索木が成長している．同様に，C_p が 2, 1.5 倍の時も SWE を使うことで改善した．しかしながら C_p を調整した中で UCT の結果が最も良かった $C_p = \sqrt{2}$ の場合で，$c = 0.5$ とした場合では改善しなかった．

(a) $C_p = \sqrt{2}$（分枝数，深さ）$= (8, 6)$



(b) $C_p = \sqrt{2}$（分枝数，深さ）$= (8, 12)$



(c) $C_p = 2\sqrt{2}$（分枝数，深さ）$= (8, 12)$

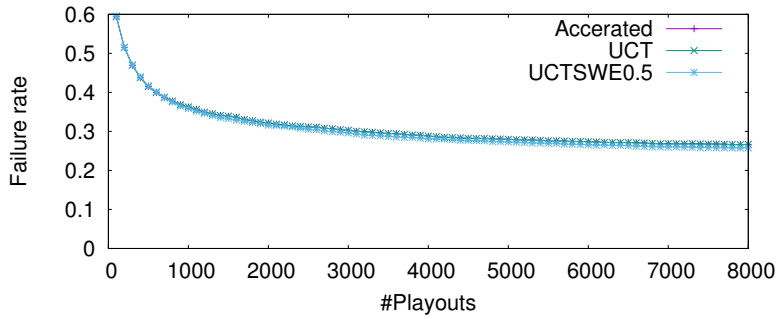


図 7.12: incremental random tree で，$c = 0.5$ とした時の誤答率の推移

前節と同様に特異節点を含む木を探索した際の推定値について調査した．結果を図 7.16 に示した．図 7.16 では，図 7.14 と異なり，best's best だけでなく，best にも緩やかな上昇傾向が見られ，正しい値 1 に近づいた．表 7.2 と同様に特異節点を含む木でも最善手と次善手の推定値を表 7.3 に示す．c が大きいほど次善手の推定値が低いだけでなく，次善手に至っては標準偏差も低く抑えられた．最善手は推定値は高くなるが，その標準偏差も高くなった．平均的には最善手も次善手もより正確な値となったと言える．これは，c が大きくなるほど最善手の推定値が下がった表 7.2 とは異なる傾向である．但し，最善手の推定値の標準偏差は incremental random tree と同様に c が大きいほど高くなった．

最後に，より大きな木として，分枝数を 16 に増やした場合の誤答率を図 7.17 に示す．分枝数，深さ 16 - 12 の木も 8 - 12 の木と同様に C_p が高い場合に SWE を用いることで誤答率が改善した．この設定では $C_p = \sqrt{2}$ で，$c = 0.5$ の時でも，2000 プレイアウト付近では悪化したものの，12000 プレイアウト付近では改善した．

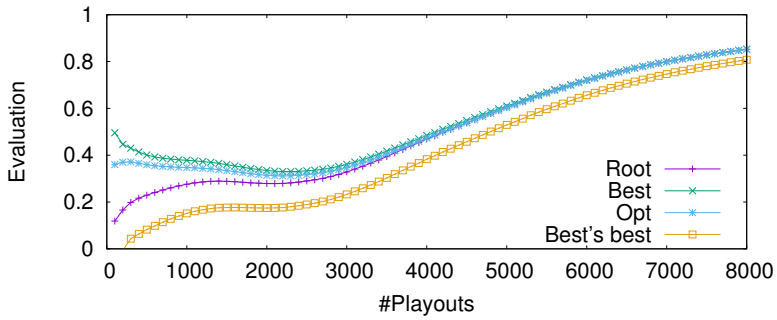


図 7.13: UCTSWE（$c = 0.5$，$C_p = \sqrt{2}$），（分枝数，深さ）$= (8, 6)$ での利得の推定値の推移

表 7.2: $C_p = \sqrt{2}$,（分枝数，深さ）$= (8, 12)$，各プレイアウト数での利得の推定値（左が最善手，右が次善手，上段が平均値，下段が標準偏差）

Playout		UCT (c = 0)	c = 0.1	c = 0.5
4000	Ave	0.2336 -0.1070	0.2322 -0.1089	0.2123 -0.1508
	Std. dev	0.0601 0.2197	0.0604 0.2196	0.0846 0.2245
8000	Ave	0.2300 -0.1127	0.2290 -0.1146	0.2127 -0.1613
	Std. dev	0.0535 0.2132	0.0541 0.2135	0.0910 0.2208

最後に補足として，C_p の設定の重要性について明らかにするため，C_p を $\sqrt{2}$ の $0.25, 0.5, 0.66, 1, 1.5, 2$ 倍した時の UCT の 8000 プレイアウト時の誤答率について表 7.4 にまとめた．C_p の調整することで性能が上がるが，最も性能を発揮するような C_p の値は木によって異なるという結果となった．

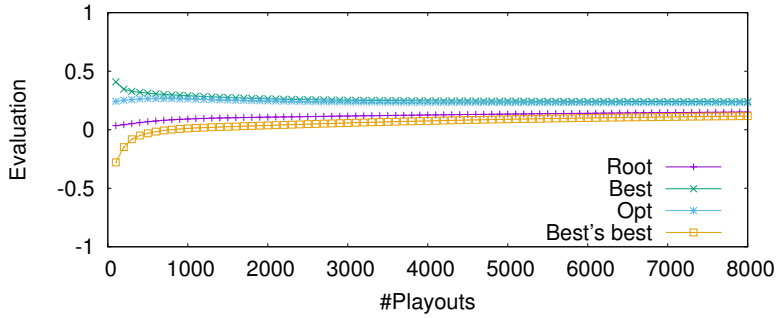
7.5 まとめ

MCTS の代表的な手法，UCT ではシミュレーションの利得の平均値で各手の評価を行っている．本研究では，MCTS における，各手の利得の推定方法を平均ではなく，推定最善の値をより重視して行う手法として，期待値の最大値の推定量である，Simplified

表 7.3: 特異節点を含む木（分枝数，深さ）$= (8, 12)$，$C_p = \sqrt{2}$ 各プレイアウト数での利得の推定値（左が最善手，右が次善手，上段が平均値，下段が標準偏差）

Playout		UCT (c = 0)	c = 0.1	c = 0.5
4000	Ave	0.1880 -0.2783	0.1906 -0.2805	0.2313 -0.3382
	Std. dev	0.0953 0.2243	0.0990 0.2227	0.1889 0.2066
8000	Ave	0.2645 -0.2842	0.2724 -0.2868	0.3603 -0.3556
	Std. dev	0.1116 0.2178	0.1158 0.2164	0.2051 0.1979

(a) UCT



(b) UCTSWE $c = 0.5$

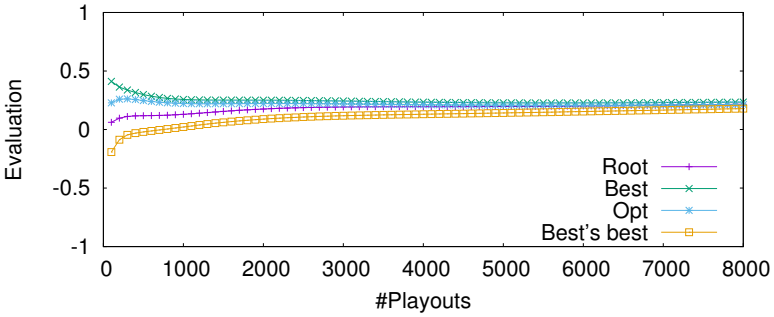


図 7.14: $C_p = \sqrt{2}$（分枝数，深さ）$= (8, 12)$ での利得の推定値の推移

表 7.4: $C_p = \sqrt{2}k$ とした時の 8000 プレイアウト時の誤答率

k	2	1.5	1	0.66	0.5	0.25
incremental random tree	0.266	0.248	0.216	0.182	0.183	0.249
特異節点を含む木	0.147	0.092	0.037	0.006	0.003	0.004

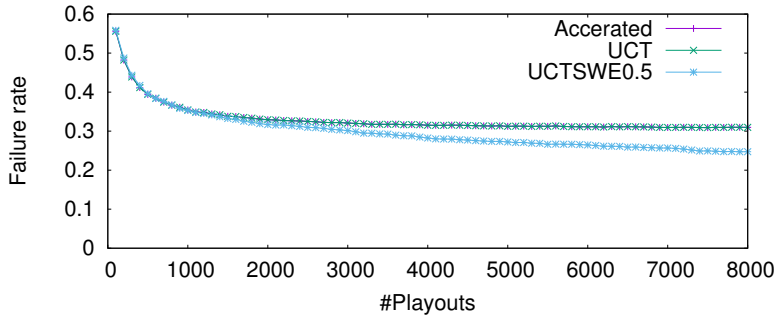
Weighted Estimator (SWE) を用いた UCT (UCTSWE) を新たに提案した．そして，誤答率，推定値の正確さの観点から改善するかについて実験を通じて調査した．実験では，確率的な利得が得られる木と確定的な利得が得られる木の大きく分けて 2 種類のゲーム木モデルを用いた．確率的な利得が得られる木は，SWE が優れた性能を示した多腕バンディット問題により近いと考えられる．実験の結果，プレイアウト数を増やした際の誤答率，推定値ともに UCT よりも提案手法の方が優れた性能を示した．確定的な利得の得られる木として，incremental random tree に加えて，特異な節点（擬似的な終端節点）をランダム生成するという工夫を加えた木を用いた．実験の結果，incremental random tree では，UCT のパラメータ C_p が $2\sqrt{2}$ と大きい時に微小ながら改善したものの．C_p が $\sqrt{2}$ と小さい場合に悪化した．特異な節点を含む木では，UCT のパラメータ C_p が $\sqrt{2}$ と小さい場合を除き改善した．推定値の分析から，悪化した原因の一つは最善手の推定値の標準偏差の高さにあると予想される．

今後の課題の一つ目は一人ゲームで試すことである．二人ゲームでは，悪い手をとっても，相手も悪い手をとれば，互いに最善をとったときと形勢がほとんど大差無いということが起こるのに対し，一人ゲームでは起こらない．そのため，平均によって利得を

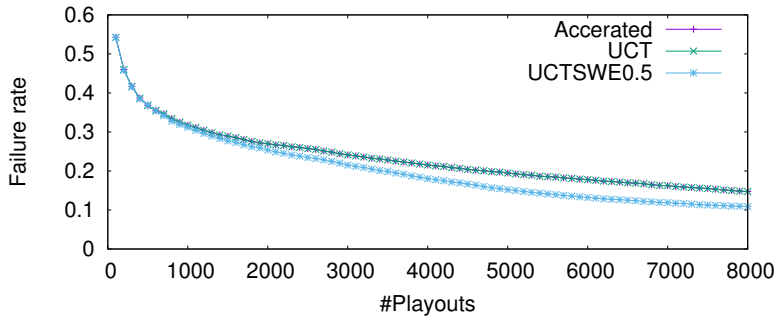
推定することでの不正確さは著しくなると予想される．

今後の課題の二つ目はSWEのMCTSへの応用の仕方の再考である．SWEでの重みの減らし方は独立同一分布と仮定した場合に基づいている．本研究では，UCTの仮定に基づきC_pを含む形で重みの減らし方を微調整したものの，この仮定は十分にシミュレーションをした場合についての仮定であり，シミュレーション数に限りがある場合の重みの減らし方についてはまだ明らかでない．

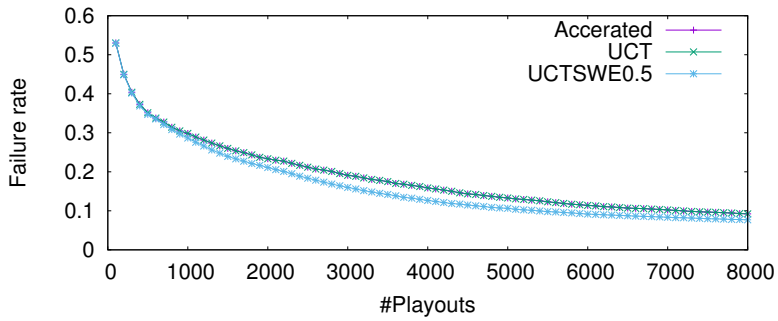
(a) 一様な探索



(b) $C_p = 2\sqrt{2}$



(c) $C_p = 1.5\sqrt{2}$



(d) $C_p = \sqrt{2}$

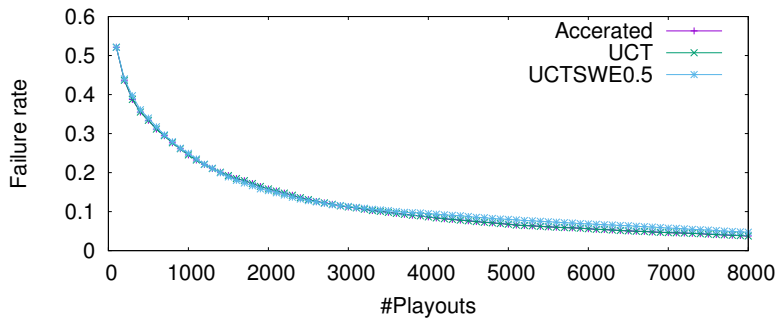
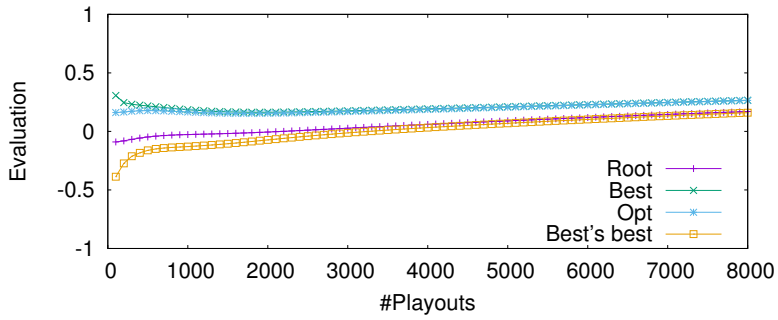


図 7.15: 特異節点を含む木（分枝数，深さ）$=(8, 12)$ で C_p を調節した場合の誤答率の推移

(a) UCT



(b) UCTSWE $c = 0.5$

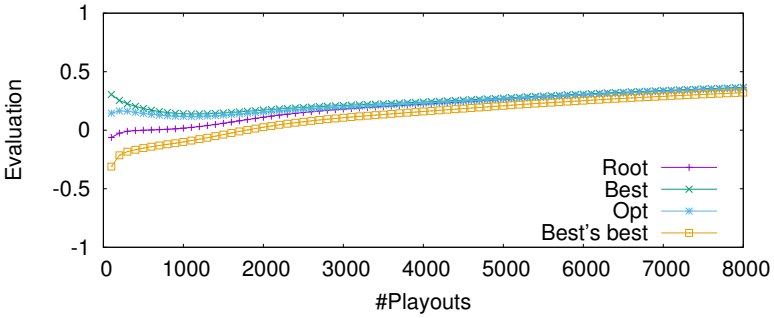
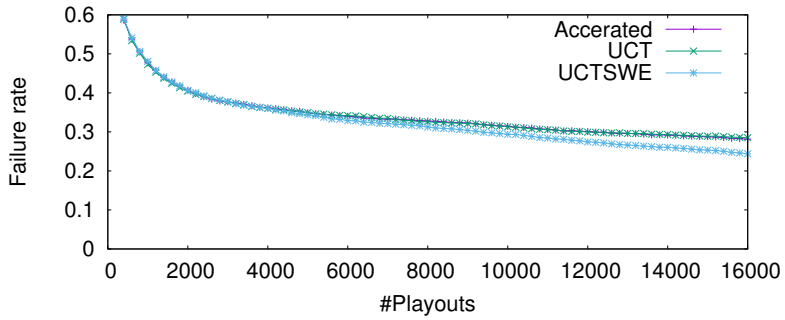


図 7.16: 特異節点を含む木，$C_p = \sqrt{2}$（分枝数，深さ）$= (8, 12)$ での利得の推定値の推移

(a) $C_p = 2\sqrt{2}$



(b) $C_p = \sqrt{2}$

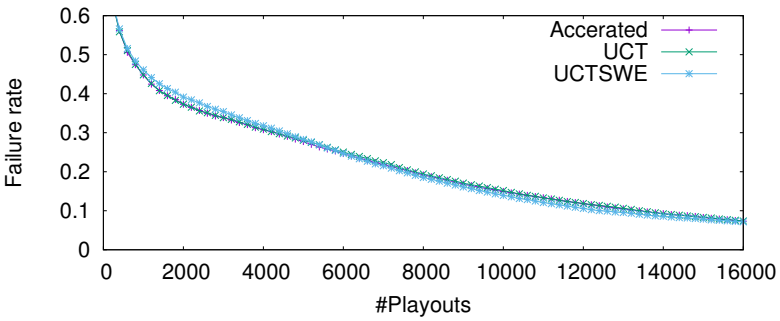


図 7.17: 特異節点を含む木（分枝数，深さ）$= (16, 12)$ で C_p を調節した場合の誤答率の推移

第8章　結論

本研究では，探索アルゴリズムの内，モンテカルロ木探索（MCTS）を対象に，改善へ向けた研究を行った．探索アルゴリズムは，外界のモデルを使って，複数の行動（候補手）の中から良い行動（手）を見つける方法であり，その効率性を改善することで，例えばゲームにおいて，良い手を効率的に見つけられるようになり，上手くプレイ出来るエージェントが作れる等の意義がある．探索アルゴリズムの中でも，MCTS は，シミュレーションに基づき，手を評価し，先読みする方法であるため，予め対象のドメインの知識を持たなくてもそれなりに上手く探索が可能であるという良い特徴を持つ．

本研究では，モンテカルロ木探索（MCTS）について，様々な仮定のもとで，どのように探索を行うことで性能を発揮出来るかを分析した．そして，既存手法の弱点とそれに対する改善策を提示した．探索の対象としたゲーム木の仮定の観点から整理すると 3 章では，一方のプレイヤが有利であるゲーム木を扱い，4，5，7 章では終端節点の深さが一様でないゲーム木を扱った．これらの MCTS での知見が比較的得られていないゲーム木のもとでの優れた MCTS について研究した．また，理論的な仮定という観点で整理すると 4 章では，節点での利得の理論値の主観確率分布が得られ，その分布が兄弟間で独立と仮定した場合に MCTS でどこから，どれだけ重点的にシミュレーションを行うべきかについて議論し，6，7 章では，アームを引く（節点を訪問する）回数が確率変数でないという仮定のもとでの，優れた期待値の最大値の推定値について議論した．加えて，本研究を MCTS の要素技術の観点でまとめると，シミュレーションの利得の与え方について 3 章で，シミュレーションを開始する節点の選び方について 4，5 章で，推定値の算出の仕方について 6，7 章で主に議論した．

本研究の意義の 1 つは上記のように，様々な仮定のもとでどのような MCTS が良いのかの観点や，MCTS の要素技術について 3 つの観点等，多角的に MCTS についての研究を行い，MCTS に対するより深い理解に繋がることにある．そして，その知見が，様々な特徴を持つ局面で，万遍なく優れた性能を発揮出来る MCTS を考案するための，手がかりになると期待される．以下で各章で行った研究について簡単に振り返る．

3 章では，局面の特徴として，次善手の偏りを定義し，分析した．この特徴は最善手を選べなかった時の損失が片方のプレイヤーの方が多く，形勢が大きく偏るが，最善手を選び続ければ，形勢は互角という特徴であり，囲碁での攻め合いで負けると多くの石を取られる（取れる）が，互いに最善を尽くしても形勢に大差ないという状況に対応する．そして，この次善手の偏りが大きいほど MCTS の性能が下がることを示した．また，ゲーム上のスコアを利用して，シミュレーション結果を補正することで性能の低下

を緩和できることを示した．しかしながら依然として，性能の低下はあるという意味で，MCTS が性能を発揮することが難しい局面であることを示した．

4 章では，多腕バンディット問題において，最善手を見分けるためのアルゴリズムと最善手をより多く選ぶためのアルゴリズムが異なり，かつ両立は出来ないという知見から，木探索においても，アルゴリズムを使い分けることで改善を目指した．提案手法は，MCTS でのプレイアウトをモデル化して，節点での利得の理論値の主観確率分布に基づく，探索を続けることの価値というメタ情報を使い，シミュレーションをする葉を決めるという初めての探索手法である．提案手法の効果を実験を通じて調査した結果，終局の深さが一様なゲーム木では悪化したが，終局の深さがばらついているゲーム木では改善した．この結果は，終局での勝ち負け情報が手に入らない場合に，提案手法の方法では期待されるほど効果的にどこからシミュレーションするかを決められていないということである．その一方で，勝ち負け情報が手に入る場合には，効果的にシミュレーション開始節点を決められているということである．囲碁や，オセロ等のゲーム木は大きく，ゲームの始めの方の局面では，終局は近くには無いという観点では，一様な深さの木の方が現実的な想定だが，十分に終局に近づいた場合，深さが一様でないゲーム木の方が一様な場合よりも現実的な想定である．また，チェスのようなゲームでは，開始局面から数手で終局し得るため，深さが一様でないゲーム木に近いと思われる．尚，ミニマックス探索と比べて，MCTS ではチェスでの性能が低いと経験的に知られているが，確かな評価関数が手に入らないような場合，例えば General Game Playing のような設定では，MCTS も比較的有効であると考える．

5 章では，終局に着目し，利得の理論値が判明したら，それに応じて今までのシミュレーション結果を修正するという手法を提案した．4 章の手法では終局の深さ一様な木では無い場合に改善し，一様な木では改善しなかった．一様な木では無い場合，ある手を選ぶとまだゲームが続くが，他の手では終局するという局面がある．この手法は終局が近くにある場合にその情報を上手く使うことを意図した手法であり，既存手法 MCTS-Solver の改善を試みた，初めての研究である．提案手法は 4 章の手法と比べて，分布を使わず，main tree の節点全てを辿らない済む分，計算が高速である．実験の結果，既存手法よりも優れた結果となった．この手法は終局が近くに無い場合は通常の UCT と変わらないため，適用しても悪影響は無いという長所がある．

6 章，7 章では，5 章での知見を一般化し，より一般的な状況での改善を目指し，終局に限らず，新たに得た情報に基づき，今までのシミュレーション結果を解釈し直すという方向で研究を行った．

6 章では，最善手は期待利得が最大の手という事実に基づき，最大値の推定量について，多腕バンディット問題（MAB）を対象に議論した．そして，各アームの推定値の重み付き平均を取ることで，期待値の最大値の推定する新たな推定量を提案し，分析した．提案手法の重みは最善のアームである確率の上限に基づいた重みである．理論的な分析を通じて，アームを引く回数に対する推定値のバイアスの収束性，及び，推定値の分散についても上界を示した．加えて，実験を行った結果，提案手法は，設定に対す

る推定値の安定性の面で優れた性能を示した．また，バイアスと分散のバランスの観点から，実験結果を解釈すると，UCT での推定値の算出方法である，平均による推定は，分散を減らすことを重視し過ぎていると明らかになった．提案手法は，直接的には sponsored search auctions のような，多腕バンディット問題の設定下で，期待値の最大値を推定する問題に適用出来る．応用としては，最善である確率の上界をどのように与えるかは議論の余地があるものの，MCTS 以外にも，強化学習に適用可能である．

7 章では，6 章で提案した推定量を，UCT に実際に応用し，二人零和完全情報ゲームを対象に，提案手法を分析した．最善である確率の上界が推定量の重みを計算するために必要となるが，7 章では上界を UCT の収束性の仮定に基づき定めた．実験では，終端節点で確率的な利得が得られる場合と確定的な利得が得られる場合の大きく分けて 2 種類の木を用いた．ゲームでの序中盤では，終端節点まで探索木が成長すること稀で，前者に近いと考えられ，逆に，終盤では，後者に近いと考えられる．また，前者の方が MAB に近い木である．実験の結果，確率的な利得が得られる木では，プレイアウト数が大きい時，提案手法は既存手法より，最善手を選べるかの観点で，有効であった．確定的な利得が得られる木での実験において，広く探索するような設定では，UCT よりも提案手法の方法の方が，良い結果となった．しかしながら，より狭く探索するような設定では，UCT の方が良い結果となった．また，実験では，推定値についても分析を行った．その結果，確定的な利得が得られる木で，狭い探索をした際に，改善が限定的であった原因の一つは，推定の標準偏差の高さにあることを示唆する結果となった．本研究は，推定値のバイアスと分散のどちらをどれだけ優先すべきかという観点で，6 章で提案した推定量を MCTS に応用した初めて研究である．

最後に，本研究に関連して，より引いた視点から，MCTS について今後研究すべき課題について記す．まず，応用に近い課題として，3 章に関連して，探索するドメインの特徴による，探索アルゴリズムの使い分け，調整が挙げられる．これはつまり，ドメインの特徴を捉える方法と，それに適したアルゴリズムを選択，あるいはアルゴリズムを調整する方法を用いることで，全体として性能を上げるということである．例えば， 2.2.3 節で紹介した General Video Game Playing（GVGP）で，好成績を残しているエージェントの多くは，複数の探索アルゴリズムをゲームの特徴に合わせて使い分けており，典型的には A*と MCTS を使い分けている．例えば，迷路のような問題では A*が役立つ．GVGP のゲーム中のオブジェクトは数種類であることが多いので，A*でいくつかのオブジェクトへの経路探索を行うことで出口に行ける．MCTS においては，ナイーブにランダムシミュレーションをすると，迷路の壁にぶつかってばかりで，出口に進まず，出口に達することは難しい．その一方で射撃等，玉を撃って敵に当てると利得が得られるようなゲームで，玉が出てから敵に当たるまで時間がかかる場合，時間の分だけ深く読まない限り，利得が得られない．その一方で，一旦敵に向かって玉を撃ってしまえば，その後は何をやっても，ランダムシミュレーションでも，玉が敵に当たり利得が得られる．その場合，MCTS では，玉を敵に向かって打つということを出来る一方で，事前知識なしでの A*では，深く読めず難しい．どのように特徴を捉える

か，それに対してどの探索アルゴリズムを使うか，調整するかについてはまだ研究の途上である．

基礎に近い課題としては，4 章に関連して，最善手を見つけるためのシミュレーションの割り振り方法や 7 章に関連して，手の利得の推定方法についての研究も重要である．これらは相互に関連するため，統合的な視点からさらに研究を発展させる必要があると考える．

謝辞

金子知適准教授には研究に関わる全般的な指導をして頂きました．田中哲朗教授には game programing seminar や勉強会でお世話になりました．山口和紀教授には KY ゼミででお世話になりました．その他 GRACO ゼミ，game programing seminar，KY ゼミの参加者の皆様，金子研の皆様には議論に参加して頂き，有用なコメントを多数頂きました．日本学術振興会には，特別研究員として採用して頂き，研究の励みになっただけでなく，研究費を特別研究員奨励費 16J07455 として支援して頂きました．

心より御礼申し上げます．

参考文献

[1] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Proceedings of the 25th Annual Conference on Learning Theory (COLT)*, June 2012.

[2] Jean-Yves Audibert and Sébastien Bubeck. Best arm identification in multi-armed bandits. In *COLT-23th Conference on Learning Theory-2010*, pp. 13–p, 2010.

[3] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, Vol. 47, No. 2-3, pp. 235–256, 2002.

[4] Terje Aven. Upper (lower) bounds on the mean of the maximum (minimum) of a number of random variables. *Journal of Applied Probability*, Vol. 22, No. 3, pp. 723–728, 1985.

[5] P. Baudiš. Balancing mcts by dynamically adjusting the komi value. *ICGA Journal-International Computer Games Association*, Vol. 34, No. 3, p. 131, 2011.

[6] Petr Baudiš and Jean loup Gailly. PACHI: State of the art open source go program. In *Advances in Computer Games*, Vol. 7168, pp. 24–38. Springer, 2012.

[7] Eric B. Baum and Warren D. Smith. A bayesian approach to relevance in game playing. *Artificial Intelligence*, Vol. 97, No. 1, pp. 195–242, 1997.

[8] Yngvi Bjornsson and Hilmar Finnsson. Cadiaplayer: A simulation-based general game player. *Computational Intelligence and AI in Games, IEEE Transactions on*, Vol. 1, No. 1, pp. 4–15, 2009.

[9] Zahy Bnaya, Alon Palombo, Rami Puzis, and Ariel Felner. Confidence backup updates for aggregating mdp state values in monte-carlo tree search. In *Eighth Annual Symposium on Combinatorial Search*, 2015.

[10] Cameron Browne, Edward J. Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Trans. Comput. Intellig. and AI in Games*, Vol. 4, No. 1, pp. 1–43, 2012.

[11] Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in finitely-armed and continuous-armed bandits. *Theoretical Computer Science*, Vol. 412, No. 19, pp. 1832 – 1852, 2011. Algorithmic Learning Theory (ALT 2009).

[12] Murray Campbell, A. Joseph Hoane, and Feng-hsiung Hsu. Deep blue. *Artificial intelligence*, Vol. 134, No. 1-2, pp. 57–83, 2002.

[13] Olivier Cappé, Aurélien Garivier, Odalric-Ambrym Maillard, Rémi Munos, and Gilles Stoltz. Kullback-leibler upper confidence bounds for optimal sequential allocation. *Annals of Statistics*, Vol. 41, No. 3, pp. 1516–1541, Jun. 2013.

[14] T. Cazenave. Sequential halving applied to trees. *IEEE Trans. Comput. Intellig. and AI in Games*, Vol. 7, No. 1, 2015.

[15] Benjamin E Childs, James H Brodeur, and Levente Kocsis. Transpositions and move groups in monte carlo tree search. In *Computational Intelligence and Games, 2008. CIG'08. IEEE Symposium On*, pp. 389–395. IEEE, 2008.

[16] P.-A. Coquelin and R. Munos. Bandit algorithms for tree search. In *Uncertainty in Artificial Intelligence*, 2007.

[17] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Computers and games*, pp. 72–83. Springer, 2007.

[18] Carlo D'Eramo, Marcello Restelli, and Alessandro Nuara. Estimating maximum expected value through gaussian approximation. In *International Conference on Machine Learning*, pp. 1032–1040, 2016.

[19] Markus Enzenberger, Martin Muller, Broderick Arneson, and Richard Segal. Fuego–an open-source framework for board games and go engine based on monte carlo tree search. *Computational Intelligence and AI in Games, IEEE Transactions on*, Vol. 2, No. 4, pp. 259–270, 2010.

[20] Jelle Van Eyck, Jan Ramon, Fabian Guiza, Geert Meyfroidt, Maurice Bruynooghe, and Greet Van den Berghe. Guided monte carlo tree search for planning in learned environments. In *Asian Conference on Machine Learning*, pp. 33–47, 2013.

[21] Zohar Feldman and Carmel Domshlak. Simple regret optimization in online planning for markov decision processes. *J. Artif. Intell. Res.*, Vol. 51, pp. 165–205, 2014.

[22] Hilmar Finnsson and Yngvi Björnsson. Game-tree properties and mcts performance. In *Proceedings of 2nd International General Game Playing Workshop (GIGA2011)*, pp. 23–30, 2011.

[23] Timothy Furtak and Michael Buro. Minimum proof graphs and fastest-cut-first search heuristics. In *IJCAI*, pp. 492–498, 2009.

[24] Raluca D. Gaina, Diego Pérez-Liébana, and Simon M. Lucas. General video game for 2 players: framework and competition. In *Computer Science and Electronic Engineering (CEEC), 2016 8th*, pp. 186–191. IEEE, 2016.

[25] Aurélien Garivier, Emilie Kaufmann, and Wouter M. Koolen. Maximin Action Identification: A New Bandit Framework for Games. In *29th Annual Conference on Learning Theory (COLT)*, Vol. 49 of *JMLR Workshop and Conference Proceedings*, New-York, United States, June 2016.

[26] Sylvain Gelly, Levente Kocsis, Marc Schoenauer, Michèle Sebag, David Silver, Csaba Szepesvári, and Olivier Teytaud. The grand challenge of computer go: Monte carlo tree search and extensions. *Commun. ACM*, Vol. 55, No. 3, pp. 106–113, March 2012.

[27] Tobias Graf and Marco Platzner. Adaptive playouts for online learning of policies during monte carlo tree search. *Theoretical Computer Science*, Vol. 644, pp. 53–62, 2016.

[28] Tobias Graf, Lars Schaefers, and Marco Platzner. On semeai detection in monte-carlo go. In *Computers and Games*, pp. 14–25. Springer, 2014.

[29] Junichi Hashimoto, Akihiro Kishimoto, Kazuki Yoshizoe, and Kokolo Ikeda. Accelerated uct and its application to two-player games. In *Advances in Computer Games*, pp. 1–12. Springer, 2011.

[30] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, Vol. 58, No. 301, pp. 13–30, 1963.

[31] Hiroyuki Iida, Makoto Sakuta, and Jeff Rollason. Computer shogi. *Artificial Intelligence*, Vol. 134, No. 1, pp. 121–144, 2002.

[32] N. Ikehata and T. Ito. Monte-carlo tree search in ms. pac-man. In *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*, pp. 39–46, Aug 2011.

[33] Takahisa Imagawa and Tomoyuki Kaneko. Enhancements in monte carlo tree search algorithms for biased game trees. In *Computational Intelligence and Games (CIG), 2015 IEEE Conference on*, pp. 43–50. IEEE, 2015.

[34] Takahisa Imagawa and Tomoyuki Kaneko. Monte carlo tree search with robust exploration. In *International Conference on Computers and Games*, pp. 34–46. Springer, 2016.

[35] Takahisa Imagawa and Tomoyuki Kaneko. Estimating the maximum expected value through upper confidence bound of likelihood. In *2017 Conference on Technologies and Applications of Artificial Intelligence*, to appear.

[36] Emil Juul Jacobsen, Rasmus Greve, and Julian Togelius. Monte mario: platforming with mcts. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pp. 293–300. ACM, 2014.

[37] Nicolas Jouandeau and Tristan Cazenave. Monte-carlo tree reductions for stochastic games. In Shin-Ming Cheng and Min-Yuh Day, editors, *Technologies and Applications of Artificial Intelligence, 19th International Conference, TAAI 2014, Taipei, Taiwan, November 21-23, 2014. Proceedings*, Vol. 8916 of *Lecture Notes in Computer Science*, pp. 228–238. Springer, 2014.

[38] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, Vol. 4, pp. 237–285, 1996.

[39] Emilie Kaufmann and Wouter Koolen. Monte-carlo tree search by best arm identification. *arXiv preprint arXiv:1706.02986*, 2017.

[40] Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In NaderH. Bshouty, Gilles Stoltz, Nicolas Vayatis, and Thomas Zeugmann, editors, *Algorithmic Learning Theory*, Vol. 7568 of *Lecture Notes in Computer Science*, pp. 199–213. Springer Berlin Heidelberg, 2012.

[41] Michael Kearns, Yishay Mansour, and Andrew Y. Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine learning*, Vol. 49, No. 2, pp. 193–208, 2002.

[42] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pp. 282–293. Springer, 2006.

[43] R. E. Korf and D. M. Chickering. Best-first minimax search. *Artificial Intelligence*, Vol. 84, pp. 299–337, 1996.

[44] T. L. Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, Vol. 6, No. 1, pp. 4–22, 1985.

[45] Marc Lanctot, Mark H. M. Winands, Tom Pepels, and Nathan R. Sturtevant. Monte carlo tree search with heuristic evaluations using implicit minimax backups. In *2014 IEEE Conference on Computational Intelligence and Games, CIG 2014, Dortmund, Germany, August 26-29, 2014*, pp. 1–8. IEEE, 2014.

[46] Yun-Ching Liu and Y. Tsuruoka. Regulation of exploration for simple regret minimization in monte-carlo tree search. In *Comput. Intellig. and Games (CIG), 2015 IEEE Conference on*, pp. 35–42, Aug 2015.

[47] Jeffrey Richard Long, Nathan R. Sturtevant, Michael Buro, and Timothy Furtak. Understanding the success of perfect information monte carlo sampling in game tree search. In Maria Fox and David Poole, editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. AAAI Press, 2010.

[48] Leandro Soriano Marcolino, Albert Xin Jiang, and Milind Tambe. Diversity beats strength?-towards forming a powerful team. In *15th International Workshop on Coordination, Organisations, Institutions and Norms (COIN 2013)*, 2013.

[49] Dana S. Nau. An investigation of the causes of pathology in games. *Artificial Intelligence*, Vol. 19, No. 3, pp. 257–278, 1982.

[50] Dana S. Nau. Pathology on game trees revisited, and an alternative to minimaxing. *Artificial intelligence*, Vol. 21, No. 1-2, pp. 221–244, 1983.

[51] Takuya Obata, Takuya Sugiyama, Kunihito Hoki, and Takeshi Ito. Consultation algorithm for computer shogi: Move decisions by majority. In *Computers and Games*, pp. 156–165. Springer, 2011.

[52] Judea Pearl. Asymptotic properties of minimax trees and game-searching procedures. *Artificial Intelligence*, Vol. 14, No. 2, pp. 113–138, 1980.

[53] Tom Pepels, Tristan Cazenave, Mark H. M. Winands, and Marc Lanctot. Minimizing simple and cumulative regret in monte-carlo tree search. In *Computer Games*, pp. 1–15. Springer, 2014.

[54] Tom Pepels, Mandy J. W. Tak, Marc Lanctot, and Mark H. M. Winands. Quality-based rewards for monte-carlo tree search simulations. In Torsten Schaub, Gerhard Friedrich, and Barry O'Sullivan, editors, *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, Vol. 263 of *Frontiers in Artificial Intelligence and Applications*, pp. 705–710. IOS Press, 2014.

[55] Diego Perez-Liebana, Spyridon Samothrakis, Julian Togelius, Tom Schaul, Simon M Lucas, Adrien Couëtoux, Jerry Lee, Chong-U Lim, and Tommy Thompson. The 2014 general video game playing competition. *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 8, No. 3, pp. 229–243, 2016.

[56] P. Perick, D.L. St-Pierre, F. Maes, and D. Ernst. Comparison of different selection strategies in monte-carlo tree search for the game of tron. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pp. 242–249, Sept 2012.

[57] Aske Plaat, Jonathan Schaeffer, Wim Pijls, and Arie de Bruin. Best-first fixed depth minimax algorithms. *Artificial Intelligence*, Vol. 87, pp. 255–293, 1996.

[58] Raghuram Ramanujan, Ashish Sabharwal, and Bart Selman. On adversarial search spaces and sampling-based planning. In *ICAPS*, Vol. 10, pp. 242–245, 2010.

[59] Raghuram Ramanujan, Ashish Sabharwal, and Bart Selman. On the behavior of uct in synthetic search spaces. In *Proc. 21st Int. Conf. Automat. Plan. Sched., Freiburg, Germany*, 2011.

[60] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, Vol. 3, No. 3, pp. 210–229, 1959.

[61] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, Vol. 529, No. 7587, pp. 484–489, 2016.

[62] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[63] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, Vol. 550, No. 7676, pp. 354–359, 2017.

[64] David Silver, Richard S. Sutton, and Martin Müller. Sample-based learning and search with permanent and transient memories. In *Proceedings of the 25th international conference on Machine learning*, pp. 968–975. ACM, 2008.

[65] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in Neural Information Processing Systems*, pp. 2164–2172, 2010.

[66] Stephen J.J. Smith and Dana S. Nau. An analysis of forward pruning. In *AAAI*, pp. 1386–1391, 1994.

[67] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*, Vol. 1. MIT press Cambridge, 1998.

[68] Gerald Tesauro, V. T. Rajan, and Richard Segal. Bayesian inference in monte-carlo tree search. In *the 26th Conference on Uncertainty in Artificial Intelligence (UAI 2010)*, 2010.

[69] William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pp. 285–294, 1933.

[70] David Tolpin and Solomon Eyal Shimony. MCTS based on simple regret. In *AAAI*, 2012.

[71] Hado van Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, pp. 2613–2621, 2010.

[72] Hado van Hasselt. Estimating the maximum expected value: an analysis of (nested) cross validation and the maximum sample average. *arXiv preprint arXiv:1302.7175*, 2013.

[73] Mark H. M. Winands, Yngvi Bjornsson, and Jahn-Takeshi Saito. Monte carlo tree search in lines of action. *Computational Intelligence and AI in Games, IEEE Transactions on*, Vol. 2, No. 4, pp. 239–250, 2010.

[74] Mark H. Winands, Yngvi Bjornsson, and Jahn-Takeshi Saito. Monte-carlo tree search solver. In *Proceedings of the 6th international conference on Computers and Games*, CG '08, pp. 25–36, Berlin, Heidelberg, 2008. Springer-Verlag.

[75] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, Vol. 1, No. 1, pp. 67–82, 1997.

[76] David H. Wolpert and William G. Macready. Coevolutionary free lunches. *IEEE Transactions on evolutionary computation*, Vol. 9, No. 6, pp. 721–735, 2005.

[77] Min Xu, Tao Qin, and Tie-Yan Liu. Estimation bias in multi-armed bandit algorithms for search advertising. In *Advances in Neural Information Processing Systems*, pp. 2400–2408, 2013.

[78] Daisaku Yokoyama and Masaru Kitsuregawa. A randomized game-tree search algorithm for shogi based on bayesian approach. In *PRICAI*, pp. 937–944. Springer, 2014.

[79] 今川孝久, 金子知適. 難しさが手番で異なる局面でのモンテカルロ木探索の性能の改善. ゲームプログラミングワークショップ 2013 論文集, pp. 162–169, nov 2013.

[80] 今川孝久, 金子知適. モンテカルロ木探索における子孫の勝敗確定時のプレイアウト結果の修正. ゲームプログラミングワークショップ 2016 論文集, Vol. 2016, pp. 13–20, 2016.

[81] 今川孝久, 金子知適. モンテカルロ木探索における状態価値の推定方法の改善. ゲームプログラミングワークショップ 2017 論文集, Vol. 2017, pp. 34–41, 2017.

[82] 伊藤毅志, 佐藤慎太郎. モンテカルロ木探索における確定勝利優先アルゴリズム. ゲームプログラミングワークショップ 2008 論文集, Vol. 2008, pp. 140–143, oct 2008.