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

今川　孝久[1,a)]　金子　知適[1]

概要：モンテカルロ木探索 (MCTS) は，囲碁などのゲームで使われている優れた探索手法である．ゲームにおいて終局は，勝ち負けが確定し，終局の前の局面で最善を尽くした場合の勝ち負けも手番次第では確定する重要な局面である．しかし，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．本研究では MCTS での終局の扱いについて議論し，勝ち負けの確定の計算とともに，勝ち負けが確定した手の評価の調整を行う手法を 3 つ提案した．実験の結果，そのうち 1 つ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．

Modification of Playout Results in Monte Carlo Tree Search When Offsprings' True Outcomes are Identified

Takahisa Imagawa[1,a)]　Tomoyuki Kaneko[1]

Abstract: Monte Carlo Tree Search (MCTS) is an effective search algorithm used in games e.g. the game of Go. Terminal positions in games are important because they prove the game-theoretical values of the positions, and also those of their previous positions in some cases. However, it is still not obvious how MCTS should treats terminal positions. In this paper, we discuss treatment of terminal positions in MCTS and propose three ways to modify evaluation of moves when their game-theoretical values are proved. In our experiment, the one of the proposed methods is better than existing methods. We also analyze the methods by investigating the evaluation of the best move.

1. 導入

UCT [4] に代表される，モンテカルロ木探索 (MCTS) は，囲碁や General Game Playing などで効果的に動作すると知られ，特に 2015 年囲碁のヨーロッパ王者に勝ち，更に 2016 年にトッププロに勝ったプログラム AlphaGo [5] にも使われている優れた探索手法である．MCTS は 1 人ゲーム等でも用いられているが，本研究では特に，二人零和有限確定完全情報ゲームを対象とする．

UCT は探索木中の手の評価を，プレイアウトと呼ばれるランダムな着手で行うアルゴリズムで，プレイアウト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 [4].

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，最善手を選べる必要がある．その観点から考えると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，親が最善で勝ちか負けかが確定し，最善でない手が確定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 [7] や，Monte-Carlo Tree Search Solver [6] (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

1　東京大学
The University of Tokyo
a)　imagawa@graco.c.u-tokyo.ac.jp

© 2016 Information Processing Society of Japan

で勝ち，もしくは負けと確定したとしても，それまでのプレイアウト結果が 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プレイアウト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．

本研究では，以上の観点から，MCTS-Solver とプレイアウト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．

2. 関連研究

2.1 UCT

UCT は，モンテカルロ木探索（MCTS）の一種であり，探索木と呼ばれる，各ゲームの局面を節点に，その局面からの合法手を辺に対応させた木を作り，プレイアウトと呼ばれる，ランダムに終局まで手を選ぶことを行い，その結果をもとに探索木の有望そうな節点（局面）を優先的に展開していくアルゴリズムである．有望さはプレイアウトで得られた利得をもとに判断する．利得は勝敗に応じて与えるのが一般的で，本研究では，負けを -1 勝ちを 1 とした．UCT では，探索木の根節点から葉節点まで，アルゴリズム UCB1 [1] に従い，UCB 値を計算し，その値が最大の手を選び，子節点を訪問するということを繰り返す．具体的には，t をプレイアウト回数，$t_j(t)$ を t 回プレイアウト後の節点 j の訪問数，$r_{j,t_j(t)}$ を節点 j の利得（j の親節点の手番から見た利得）の見積もり値，c を第二項の調整のためのパラメータとすると， $t+1$ 回目では

$$\mathrm{UCB} = r_{j,t_j(t)} + c\sqrt{2\ln t_i(t)/t_j(t)} \tag{1}$$

が最大の子節点を選ぶ．葉に到達したらプレイアウトを行う．葉が訪問されたら探索木を成長させる．探索木の成長は葉の子節点を全て探索木に含み，それらの子節点を新たに探索木の葉節点とすることで行う．プレイアウトを行ったら，その結果（勝ち・負け）を葉から根まで伝えて行き，各節点での見積もり値と訪問数を更新する．UCT の場合は節点 j の利得の見積もり値として節点 j とその子孫のプレイアウトの平均利得を使う．更新は節点 j が選ばれた場合

$$t_j(t+1) \leftarrow t_j(t) + 1 \tag{2}$$

$$r_{j,t_j(t+1)} \leftarrow \frac{r_{j,t_j(t)}t_j(t) + r}{t_j(t+1)} \tag{3}$$

で行われる．これらを繰り返すことで各手の見積もり値の精度を高めていく．UCT は，十分に大きな探索木で十分にプレイアウトを行う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 [4]．尚，UCB 値のパラメータ c は 2 とした．

UCT における，葉でない節点 j の値の更新は，C_j を節点 j の子，r_j と t_j を節点 j が葉であった時に行われたプレイアウト結果の平均利得とその回数とすると，節点 j の子 C_j の訪問数と見積もり値を用いて，

$$t_j(t+1) \leftarrow \sum_{k\in C_j} t_k(t+1) + t_j \tag{4}$$

$$r_{j,t_j(t+1)} \leftarrow -\frac{\sum_{k\in C_j} r_{k,t_k(t+1)}t_k(t+1) + r_j t_j}{t_j(t+1)} \tag{5}$$

と表すこともできる．

2.2 MCTS-Solver

MCTS-Solver は Lines of Action というゲームのために考案された手法 [6] で，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．プレイアウト結果の更新は通常の UCT と同様に行うが，その他に最善で勝ち・負け・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，利得の見積もり値が徐々に真の値に近づくことを期待できる．尚，文献 [6] 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 (1) 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 (5) 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のが理想的で，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・祖先の評価に使われているため，ここに改善の余地があると考えられる．

2.3 Accerated UCT

本研究と同様に古いプレイアウト結果を減らすという形で調整を行い，MCTS の改善を目指したものとして Accerated UCT [3] が挙げられる．Accerated UCT では UCT のプレイアウト結果の更新の部分を変更し，親の平均利得を算出する際に子の重み付き平均利得をとる．$t+1$ 回目のプレイアウト終了後，まず葉の値は，式 (2) (3) で更新する．そして節点 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 $v_{j,t+1}$ は $v_{j,t+1} \leftarrow v_{j,t}\lambda + \mathbb{I}_{t+1}\{\text{j is chosen}\}$ という形で更新される．また，Accrerated UCT では，$t+1$ 回目の

© 2016 Information Processing Society of Japan

プレイアウトでの探索木の葉 i の展開の際，子の他に仮想的な子を作り，その葉から行われた今までのプレイアウト結果 r_i を仮想的な子 c_i に入れて $r_{c_i,t+1} = r_i$ とし，その子の重みを $v_{c_i,t+1} = t_i$ にしておく．子の重み $v_{j,t+1}$ を使い親 i の利得は

$$r_{i,t_i(t+1)} \leftarrow -\frac{\sum_{j \in C_i \cup \{c_i\}} v_{j,t+1} r_{j,t_j(t)}}{\sum_{j \in C_i \cup \{c_i\}} v_{j,t+1}} \qquad (6)$$

という形で更新される．探索中に c_i は訪問されないので，徐々に仮想的な子の重みが減り，子孫の結果が優先されるという効果がある．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 値第二項は調整されない．

文献 [3] 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)　(3)　で更新した値を用いた．

3. 提案手法

本項では，MCTS-Solver の拡張として，3 つの提案方法を提案する．3 つの手法では，プレイアウト終了後，通常の UCT と同様に利得の見積もり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，訪問数及び利得の見積もり値を修正する．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．その際の修正では，負けが確定した手がある場合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，$t+1$ 回目のプレイアウト後，葉節点では，まず，式 (2)　(3)　で値を更新する．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，利得の見積もり値を修正後，葉節点以外の更新は式 (4)　(5)　で行う．

3.1 利得の見積もり値の修正

勝ち負けが確定した場合の単純な修正方法として，今までのプレイアウト結果を最善を尽くした場合の結果に置き換えるということが挙げられる．この手法を Replacement MCTS-Solver (Replacement) と呼ぶ．Replacement はもしある手 j が負けと確定した場合，$r_{j,t_j(t+1)} \leftarrow -1$ とする．Replacement では訪問数は UCT と同様に増える．

3.2 訪問数の修正

前述の手法では，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，プレイアウトが十分行われた後で負けが確定した場合は大きく見積もり値が変化し得る．その結果，ある手（まだ勝ち負けが確定していない）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考えられる．また，上記と関連して，負け確定した手は（全ての手が負けの場合を除き）最善手ではないにもかかわらず，その手の見積もり値を親の見積もり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見積もり値は過度に高くなる．これもまた望ましくないと考えられる．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．こうすることで，UCB 値の第 2 項を調整し，負けが確定した影響で，その先祖の見積もり値が大きく下がり，しばらく訪問されないという状況を避けられると期待する．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，$t_j(t+1) \leftarrow \varepsilon$, $r_{j,t_j(t+1)} \leftarrow -1$ とする．

3.3 利得の見積もり値の変化に応じた訪問数の修正

前述の手法 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プレイアウト結果の違いに大きく影響されてしまうことも予想される．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．見積もり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，今までの見積もり値が負けの利得 -1 から離れているほど，節点 i とその先祖の見積もり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見積もり値の修正による変化が大きいほど訪問数を減らし，見積もり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．具体的には，IRemoval は $t+1$ 回目のプレイアウトである手 j が負けと確定した場合，まず，式 (7)　で t 回目のプレイアウト結果の更新終了時の訪問数と見積もり値を使い，$t+1$ 回目

[*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 で手番のプレイヤにとっての勝ち負けを意味する．

© 2016 Information Processing Society of Japan

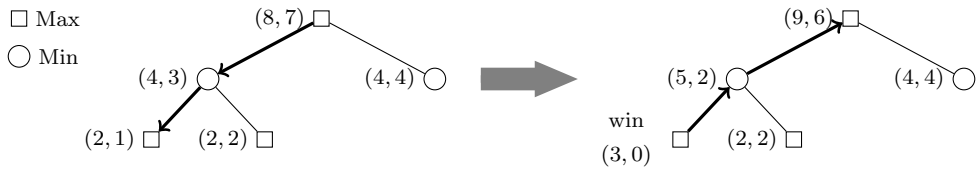


図 1 左下の節点の子孫でのプレイアウト結果が Max プレイヤの勝ちとなり，かつ左下の節点が Max プレイヤの勝ちと確定した時の提案手法の更新の例．各節点の (a, b) は節点の Max プレイヤの勝ち数，負け数がそれぞれ a, b を意味する．

のプレイアウトの利得 r が得られた時の見積もり値を求める．そして，式 (8) でその値が正しい値である -1 から離れているほど j の訪問数を減らす．更に，見積もり値を正しい値である -1 にする．

$$r_{j,t_j(t+1)} \leftarrow \frac{r_{j,t_j(t)} t_j(t) + r}{t_j(t) + 1} \tag{7}$$

$$t_j(t+1) \leftarrow \frac{(1 - r_{j,t_j(t+1)})}{2}(t_j(t) + 1) \tag{8}$$

$$r_{j,t_j(t+1)} \leftarrow -1 \tag{9}$$

プレイアウト結果の保持の仕方として，プレイアウトの（勝数，負数）と（勝率，訪問数）は一対一に対応する．IRemoval で行う修正は，勝数，負数を保持していると考えると，手 j で保持されている今までのプレイアウト結果の内，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プレイアウト結果も取り除くことに対応する．例を**図 1** に示す．例では簡単のため手番ではなく Max プレイヤ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，探索初期での見積もり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大きな変化と節点 j の親 i の見積もり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．節点 i の値を算出するのに負け確定の手 j の見積もり値を使っているという意味では Replacement と同じではあるが，その訪問数は Replacement に比べて少ないと期待されるためである．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，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，手 j の節点 i の見積もり値への影響は，少ないと期待される．

4. 実験

4.1 実験設定

実験で用いたゲーム木について説明する．本研究では文献 [2] のゲーム木を基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．大雑把に述べると文献 [2] のゲーム木は文献 [4] の P-game の変種である．文献 [2] では，ゲーム木の各手 (辺) には値を割り当てる．初期局面 (根節点) は Max プレイヤの手番である．ある整数値 a を決め，Max(Min) プレイヤの手には，一つは 0（これが最善手となる），他の手には等確率に $\{-a, \ldots, -1\}$ ($\{1, \ldots, a\}$) の値を割り当てる．終局 (終端節点) のスコアは初期局面からその局面に至るまでに選んだ手の値の総和となる．その局面は総和が正だと Max プレイヤの勝ち，負だと負け，0 だと引き分けとなる．本研究では，総和が 0 の場合も Max プレイヤの勝ちとした．このゲーム木では，互いに最善を尽くすと手の値の総和が 0 になるので，Max プレイヤの勝ちとなる．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，根節点は最善で Max プレイヤの勝ちであることは変わらない．このゲーム木では探索する前から最善手が分かっているという特長がある．

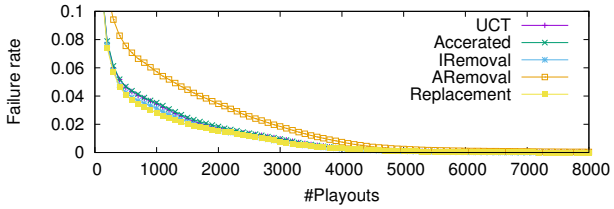
4.2 実験結果

4.2.1 誤答率の変化

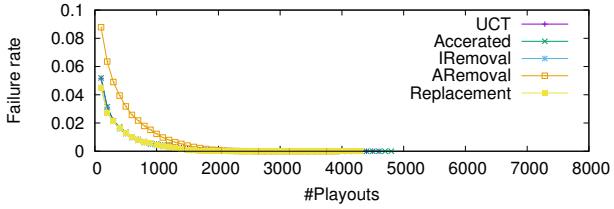
実験では，幅，最大深さ $(4, 12), (8, 8), (16, 16)$ のゲーム木に対し，$P = 0.1, 0.2$ の 2 種類の終端確率を設定し，それぞれの設定に対し，各アルゴリズム，UCT + MCTS-Solver (UCT)，Accerated UCT + MCTS-Solver(Accerated)，UCT + Replacement，UCT + ARemoval，UCT + IRemoval で探索し，利得の見積もり値が最も高い手を選ばせて，最善手以外を選んだ比率（誤答率）を計測した．Accerated の λ は $0.9, 0.99, 0.999$ の内，予備実験で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 2**）は ARemoval が誤答率が UCT の誤答率と比べて有意に高くなった（有意水準 0.05 のウェルチの t 検定で，両側検定を行った）．特に，**図 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，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

© 2016 Information Processing Society of Japan

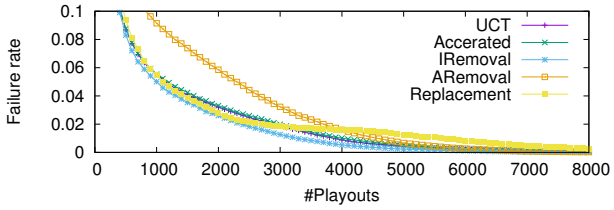


(a) 終端確率 0.1

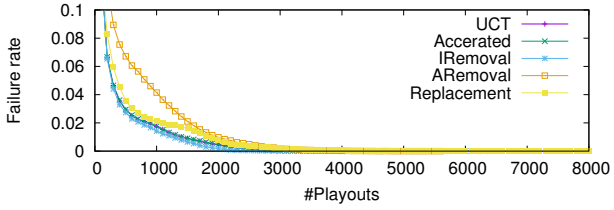


(b) 終端確率 0.2

図 2 （分枝数，最大深さ）= (4, 12) での誤答率の推移



(a) 終端確率 0.1

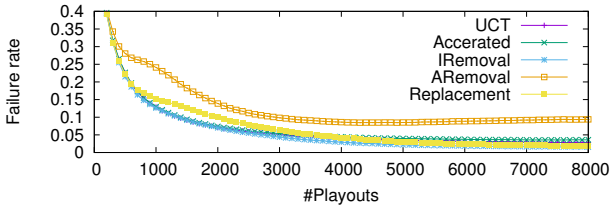


(b) 終端確率 0.2

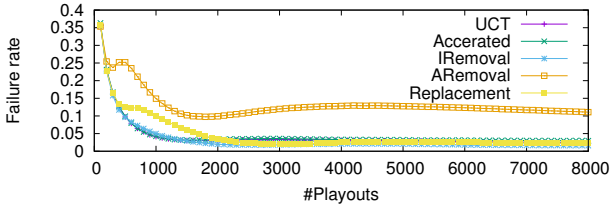
図 3 （分枝数，最大深さ）= (8, 8) での誤答率の推移

るためである．

分枝数 8，最大深さ 8 （**図 3**），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，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．



(a) 終端確率 0.1



(b) 終端確率 0.2

図 4 （分枝数，最大深さ）= (16, 16) での誤答率の推移

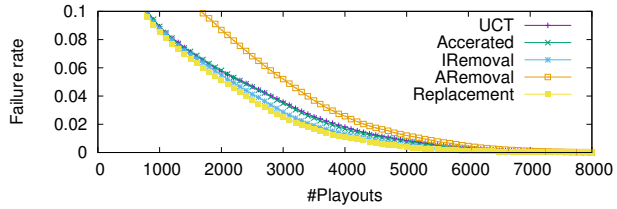
より大きな木である分枝数 16，最大深さ 16 の時の誤答率（**図 4**）は，終端確率 0.1，0.2 の時も同様の傾向で，プレイアウト数 200 辺りから ARemoval は有意に高くなった．また，Replacement は始め UCT より誤答率が有意に高いものの，プレイアウト数 4000 ほどから低くなり，IRemoval も UCT より有意に高くなることがあるものの，プレイアウトを重ねるとより低くなった．

今度はより難しいと考えられる木を対象として，誤答率を計測した．このような木では，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．そこで，今度のゲーム木では，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．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 $a = 10$ とした時の結果を**図 5**，**図 6**，**図 7** 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 の時と同様であった．

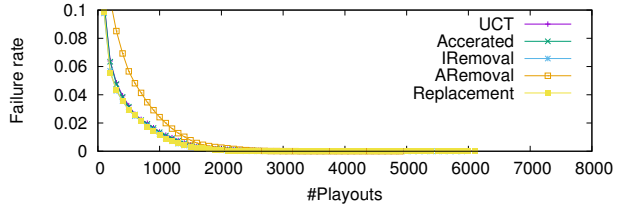
4.2.2 最善手の見積もり値の分析

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見積もり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

© 2016 Information Processing Society of Japan

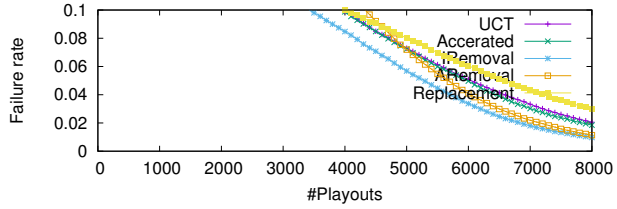


(a) 終端確率 0.1

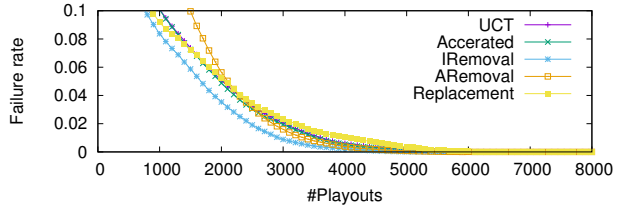


(b) 終端確率 0.2

図 5 手の値の比率 2,（分枝数，最大深さ）= (4, 12) での誤答率の推移

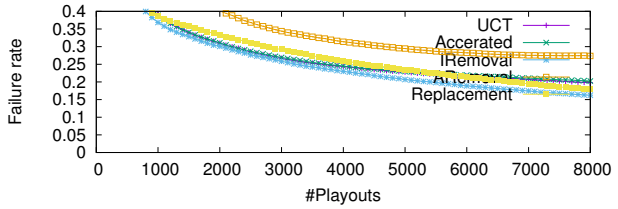


(a) 終端確率 0.1

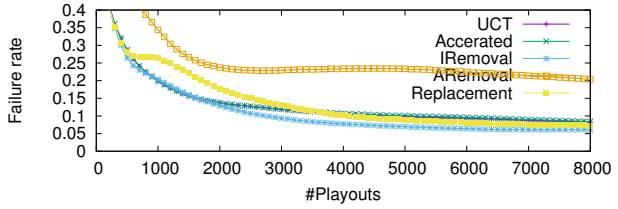


(b) 終端確率 0.2

図 6 手の値の比率 2,（分枝数，最大深さ）= (8, 8) での誤答率の推移

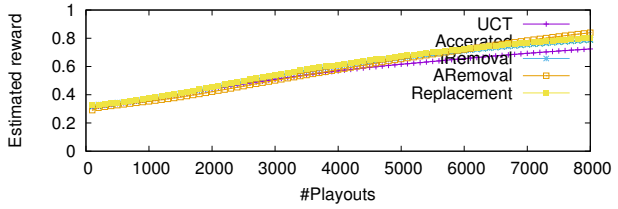


(a) 終端確率 0.1

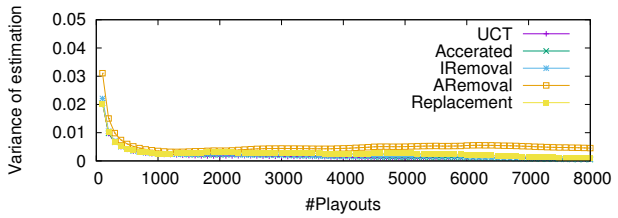


(b) 終端確率 0.2

図 7 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

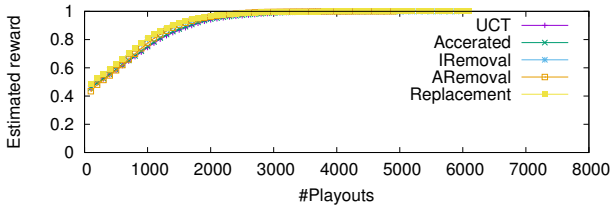
図 8 終端確率 0.1，手の値の比率 2,（分枝数，最大深さ）= (4, 12) での最善手の見積もり値の平均と分散の推移

本のゲーム木での値の平均をとった．

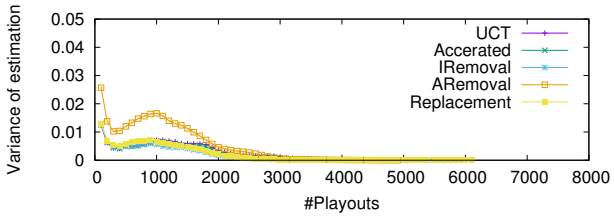
このゲーム木では，互いに最善を尽くすと，Max プレイヤ（根節点は Max プレイヤの手番）の勝ちであり，互いに最善を尽くした場合の利得は 1 であるので，利得の見積もり値が 1 に近いほど，正確であることを意味する．そして分散が小さいほど，各探索による違い（乱数が異なることから，プレイアウト結果が異なり，その結果としてプレイアウトをする節点の優先順位が異なる可能性がある．）によらず，安定して見積もり値を算出しているということを意味する．尚，根節点の勝ちが確定すると見積もり値は 1 になる．以下，実験結果について提案手法毎に記す．

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 8**）の時の見積もり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(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

© 2016 Information Processing Society of Japan

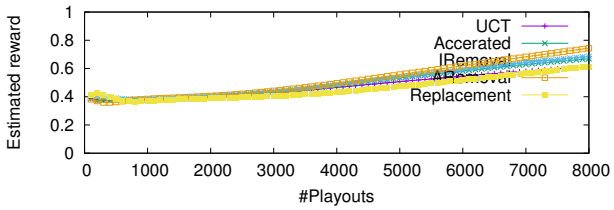


(a) 平均

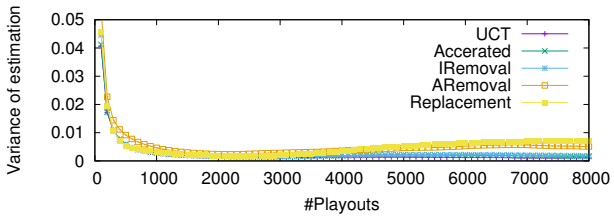


(b) 分散

図 9 終端確率 0.2，手の値の比率 2，（分枝数，最大深さ）= (4, 12) での最善手の見積もり値の平均と分散の推移

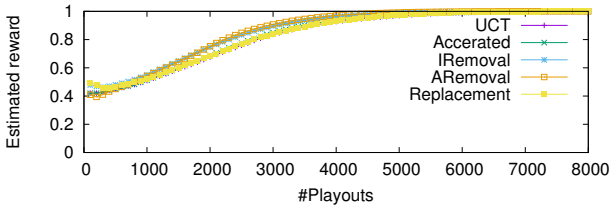


(a) 平均

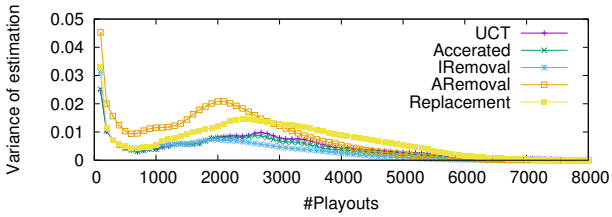


(b) 分散

図 10 終端確率 0.1，手の値の比率 2，（分枝数，最大深さ）= (8, 8) での見積もり値の平均と分散の推移

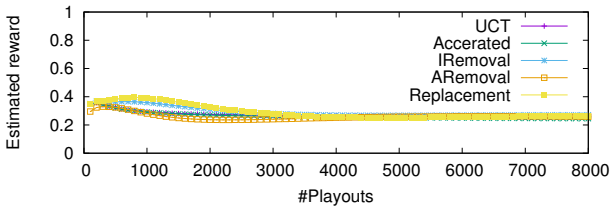


(a) 平均

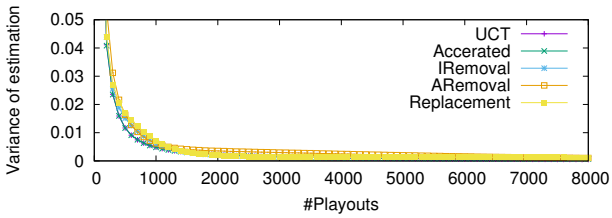


(b) 分散

図 11 終端確率 0.2，手の値の比率 2，（分枝数，最大深さ）= (8, 8) での見積もり値の平均と分散の推移



(a) 平均

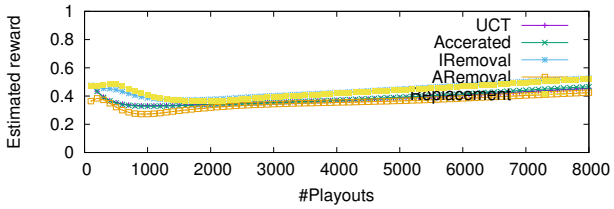


(b) 分散

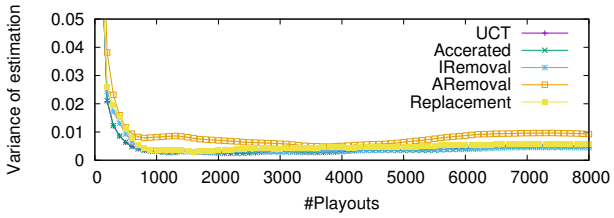
図 12 終端確率 0.1, 手の値の比率 2, (分枝数, 最大深さ) = (16, 16) での最善手の見積もり値の平均と分散の推移

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，見積もり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 9**）の時の見積もり値も，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，勝ち負けが確定し，勝ちが確定した場合でも結果としてその親の手が負けと確定し，訪問数が減るため，見積もり値が安定しないということと予想され，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 10** **図 11**），も同様の傾向で，UCT と比べて見積もり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

© 2016 Information Processing Society of Japan



(a) 平均



(b) 分散

図 13 終端確率 0.2, 手の値の比率 2, (分枝数, 最大深さ) = (16, 16) での最善手の見積もり値の平均と分散の推移

さ 16，終端確率 0.1 の時（図 8），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 9）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 と比べて見積もり値が正確であったことは，UCT の見積もり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 8，図 9）の時の Replacement の見積もり値の平均は 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 10 図 11）は，Replacement の見積もり値は 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，見積もり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見積もり値を算出するため，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，見積もり値の分散の結果はこれを支持している．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 のように見積もり値を修正し，その値が親の見積もり値を計算するのに使われる手法である．最善手の見積もり値の平均と分散の分析を通じて，IRemoval の見積もり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，見積もり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. まとめ

本研究では，探索中にある手の勝ち負けが確定するような状況で，古いプレイアウト結果をどう扱うかについて議論し，3 つ手法を提案し，その性能について，ゲーム木を作って，実験を行い，特に Inconsistency Removal MCTS-Solver (IRemoval) は既存手法よりも優れた性能を示した．また，性能について最善手の見積もり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．

参考文献

[1] Auer, P. et al.: Finite-time Analysis of the Multiarmed Bandit Problem, *Machine Learning*, Vol. 47, No. 2-3, pp. 235–256 (2002).

[2] Furtak, T. and Buro, M.: Minimum Proof Graphs and Fastest-Cut-First Search Heuristics., *IJCAI*, pp. 492–498 (2009).

[3] Hashimoto, J. et al.: Accelerated UCT and its application to two-player games, *ACG*, Springer, pp. 1–12 (2012).

[4] Kocsis, L. and Szepesvári, C.: Bandit based monte-carlo planning, *ECML*, Springer, pp. 282–293 (2006).

[5] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. et al.: Mastering the game of Go with deep neural networks and tree search, *Nature*, Vol. 529, No. 7587, pp. 484–489 (2016).

[6] Winands, M. H. et al.: Monte-Carlo Tree Search Solver, *CG*, Springer-Verlag, pp. 25–36 (2008).

[7] 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, No. 11, pp. 140–143 (2008).

© 2016 Information Processing Society of Japan