Vol.2014-MPS-100 No.23
2014/9/26

不完全情報ゲームにおける適応的モンテカルロ木探索手法の提案

大佐賀　猛[1]　野中　秀俊[1]　吉川　毅[1]　杉本　雅則[1]

概要：モンテカルロ木探索は完全情報ゲームを対象として提案された手法であるが，不完全情報ゲームに対しても有効であることが知られている．これに対し，モンテカルロ木探索を再帰的に繰り返し行う手法や情報集合を用いて木を探索する手法などが提案されている．しかし，ゲームの状態によってはこのような工夫を行うよりも通常のモンテカルロ木探索を行う方が性能がよい場合がある．本報告では，探索の回数や深さを適応的に変化させる適応的モンテカルロ木探索を提案する．本報告における評価の対象問題として，既存の Synthetic Tree のパラメータを再定義し，より多様な不完全情報ゲームに対応できる汎用的なモデルを導入した．このモデルを用いた評価実験により，提案手法は既存手法よりも高い性能を示すことができた．

1. はじめに

モンテカルロ木探索（Monte Carlo Tree Search，以下 MCTS）はモンテカルロ法とゲーム木探索を組み合わせた手法で，完全情報ゲームのうち，特に囲碁などの状態数が多いゲームで有効な手法であるとされている．カードゲーム等の不完全情報ゲームでは，一般に観測できない情報があるため，状態数がさらに多くなる傾向があり，これらに MCTS が適用されるようになってきている．

不完全情報ゲームに初めて MCTS を適用した研究として Perfect Information Monte Carlo(PIMC) 探索 [1] がある．これは観測できない状態の中からランダムにサンプリングを行うことによって状態を定め，ゲーム終了までランダムに着手を決定するシミュレーションを繰り返し行うアルゴリズムである．このアルゴリズムを用いてブリッジやハーツ等で強いプレイヤープログラムが作られた．

Furtak らは PIMC 探索におけるプレイアウト中の方策を様々な方策に変更した手法，Imperfect Information Monte Carlo(IIMC) 探索 [2] を提案している．この手法は，通常の PIMC 探索よりも性能が向上しているが，計算時間が増大していることが問題として挙げられている．また，深く探索する手法は不完全情報が多いゲームをプレイする場合には逆効果になる場合がある [3].

本研究では様々なゲームに対応するために，ゲームの状態に応じてプレイアウトの方策や探索の回数を適応的に変更させる手法，適応的モンテカルロ木探索 (Adaptive Monte Carlo Tree Search, 以下 AMCTS) を提案する．AMCTS はゲームの状態に応じて十分な回数，十分な深さの探索を行うためにプレイアウト中の方策をランダムに選び着手を決定する方法と PIMC 探索によって着手を決定する方法を使い分ける手法である．

本手法の有効性を検証するために Long らが提案した Synthetic Tree[4] においてパラメータの再定義を行い，より一般的な不完全情報ゲームに対応できる汎用的なモデルを構築した．このモデルを利用して評価実験を行った結果，AMCTS は PIMC 探索と比べて約 5%高い性能を示し，かつ IIMC 探索の計算時間と比べて最大約 42%短縮することを示した．

2. 研究背景

2.1 UCB(Upper Confidence Bound)

UCB アルゴリズムは，多腕バンディット (Multi-Armed Bandit) 問題に対して提案された手法で，式 (1) で定義される UCB 値が最も高いスロットマシンが選択される．

$$UCB(s) = \overline{x}_s + \sqrt{\frac{2\ln n}{n_s}} \tag{1}$$

s はスロットマシン，$\overline{x}_s$ はスロットマシン s において観測した報酬の平均値，n は全体の試行回数，n_s は s を選んだ回数である．まだ一度も選ばれていないスロットマシンの UCB 値は最大にする．また UCB1-tuned[5] は UCB の戦略に加え報酬の分散を考慮した戦略である．UCB1-tuned 値は式 (2)(3) で表される．

[1] 北海道大学
Hokkaido University

Algorithm 1 PIMC 探索 (InfoSet I, int N)

```
for i = 0 to M(I) do
    val[i] = 0
end for
for i = 0 to N do
    x =Sample(I)
    for m = 0 to M(I) do
        v =PerfInfoValue(x, m)
        val[m]+ = v
    end for
end for
return arg max_m {val[m]}
```

$$\mathrm{UCB1-tuned}(s) = \overline{x}_s + \sqrt{\frac{\ln n}{n_s}\min\{1/4, V_s(n_s)\}} \quad (2)$$

$$V_s(n_s) = \left(\frac{1}{t}\sum_{i=1}^{t} x_{s,i}^2 - \overline{x}_{s,t}^2 + \sqrt{\frac{2\ln n}{t}}\right) \quad (3)$$

$x_{s,i}$ はマシン s の i 回目の試行で得られた報酬，$\overline{x}_{s,t}$ はマシン s を t 回試行した時の得られた報酬の平均値である．理想的な戦略との報酬差に対する理論的な評価式は与えられていないが，評価実験によって UCB よりも性能が高いことが示されている．本研究の戦略決定のアルゴリズムには，UCB1-tuned を使用した．

2.2 Perfect Information Monte Carlo(PIMC) 探索

Perfect Information Monte Carlo(PIMC) 探索は MCTS を不完全情報ゲームに適用する際に，ゲーム全体の状態を特定できないという問題を解決するために提案された手法である．

これは不確定な状態においてとりうる状態を列挙（サンプリング）する手法である．PIMC 探索の擬似コードを **Algorithm 1** に示す．情報集合 I はゲーム木のノードの集合であり，N はサンプリング回数である．$M(I)$ は I から選択できる行動の数である．Sample(I) は I から取りうる状態からひとつ状態をランダムにサンプリングする関数である．PerfInfoValue(x, m) は状態 x から行動 m を行い状態遷移した後，プレイアウトによって各プレイヤーがランダムに行動し，終端ノードの評価値を得る関数である．

2.3 Imperfect Information Monte Carlo(IIMC) 探索

完全情報ゲームで MCTS の性能を向上させるために MCTS を再帰的に適用する手法が提案されている [6]．これを不完全情報ゲームに一般化した手法が Imperfect Information Monte Carlo(IIMC) 探索 [2] である．IIMC は PIMC 探索の PerfInfoValue を FinishedGameValue に変更している点が異なる．

FinishedGameValue の擬似コードを **Algorithm 2** に示す．このアルゴリズムの中で，ComputeMove はランダムなプレイヤーやルールベースプレイヤー，より洗練されたプレイヤー等のアルゴリズムなどを適用することによって行動選択を行っている．[2] では FinishedGameValue に PIMC 探索を適用したものについて議論している．今後，IIMC 探索とは FinishedGameValue に PIMC 探索を適用したものとする．

Algorithm 2
FinishedGameValue(Node x, Move m,Player P)

```
y =MakeMove(x, m)
while y is not terminal node do
    y =MakeMove(y,ComputeMove(P, I(y)))
end while
return value of y in view of the player to move in x
```

3. 評価モデル

[4] では，不完全情報ゲームが持つ特徴を 3 つにまとめ，様々な特徴において PIMC 探索の性能を評価するために Synthetic Tree というモデルが提案されている．本実験では提案手法を評価するための不完全情報ゲームのモデルとして Synthetic Tree を用いた．

Synthetic Tree は以下の 3 つのパラメータによって特徴付けられる．

- Leaf Correlation(lc) 確率 lc で終端ノードの組が同じ報酬になる．確率 $1-lc$ で終端ノードの組が異なる報酬になり，どちらか片方がランダムに 1 の報酬が与えられ，もう片方に -1 の報酬が与えられる．
- Bias(b) 上記の lc で同じ報酬と判定された終端ノードの組に対して，確率 b で報酬 1，確率 $1-b$ で報酬 -1 が与えられる．
- Disambiguation Factor(df) それぞれのプレイヤーが行動を選択する度に，確率 df でそのプレイヤーが保有する世界の候補の数を半分にする．この操作が起きたとき，再帰的にこの操作を繰り返す．例えば，確率 df を満たし世界の候補の数を半分にした後，また確率 df を満たしたとき，更にそのプレイヤーの世界の候補の数が半分になる．

lc, b は実際のゲームをランダムにプレイして到達した終端ノードの親ノードのすべての報酬の値を記録することで得られる．df は，行動毎に取りうるゲーム全体の状態の数を計算し，行動を選択する度に前の世界の候補の数と比べることにより得られる．これにより，実際のゲームを Synthetic Tree のパラメータで表現することができる．

世界の候補の数が 2，木の深さが 2 の場合の Synthetic Tree の例を**図 1** に示す．丸で示されているルートノードがチャンスノードと呼ばれるノードであり，そこから確率的に子ノードへと遷移する．上向き三角，下向き三角はそれぞれのプレイヤーを示しており，下向き三角についている影の色はそのプレイヤーから見て同じ状態に見える状態

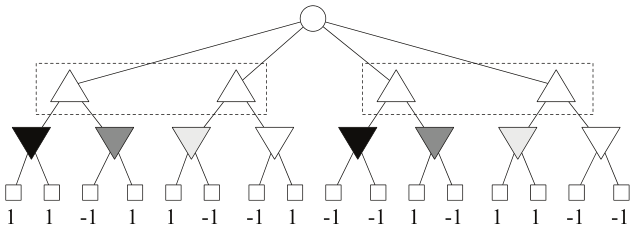

図 1 世界 $W = 2$, 木の深さ $d = 2$ の Synthetic Tree

を示している．

このモデルでは報酬が $1, -1$ の二つしか与えられないため，Bias を以下のように再定義を行った．

Bias の再定義

平均 b, 標準偏差 $1/R$ の正規分布を想定する．R は正の整数とする．この正規分布からサンプル i を一つ取り出す．取り出した i を $0 \leq i \leq 1$ とするために，$i < 0$ のとき $i = 0$,$i > 1$ のとき $i = 1$ とした．取り出した i が以下の式を満たす最小の正の整数 r が終端ノードに報酬として与えられる．

$$i \leq \frac{2(r+1)-1}{2R} \tag{4}$$

これにより，報酬の分布は平均 b，分散 $(1/R)^2$ の正規分布に従う．Long らの方法では $1, -1$ の報酬だったものが，0 から R までの整数の報酬を与えることができ，より一般的なモデルとした．

4. 提案手法

本章では提案手法である，適応的モンテカルロ木探索 (Adaptive Monte Carlo Tree Search, AMCTS) のアルゴリズムについて述べる．まずゲームの状態に応じて適応的にプレイアウト回数を設定する方法と，プレイアウトの方策を設定する方法について述べる．その後，アルゴリズムの全体の流れについて述べる．

4.1 適応的アルゴリズム

AMCTS は IIMC 探索のプレイアウトの方策をゲームの状態に応じて変更するアルゴリズムである．また，プレイアウト回数もゲームの状態に応じて変更する．まず，プレイアウト回数の設定について述べ，その後，プレイアウトの方策の設定について述べる．

4.1.1 プレイアウト回数の設定

現在の手番の状態における世界の候補の数 W，行動の数 A に対して，プレイアウト回数 N は式 (5) で設定する．

$$N = WAC \tag{5}$$

C は定数であり，ゲームに応じて設定する．これにより，世界の候補が少ない時や，行動の選択肢が少ない時に性能を落とすことなくプレイアウト回数を削減することができる．

プレイアウト回数は完全情報ゲームと不完全情報ゲームで意味が異なる．完全情報ゲームの場合は木を構築しながら探索を行うため，プレイアウト回数を無限回行うと最適解に到達するということが示されている [7]．一方，不完全情報ゲームの場合は状態のサンプリングを行いながら探索を行うため木の構築を行わない．そのため現在の手番の状態からの評価しか行わないため，プレイアウト回数を増やしても最適解に到達する保証はない．また，状態は一意に定まらないため，求めた最適解に到達できないことがある．よって PIMC 探索や IIMC 探索のプレイアウト回数はそれぞれの状態における行動を適切に評価するように設定する必要がある．

以上の考察により，必要なプレイアウトの回数 N は W, A に比例すると考えられるため，式 (5) を用いた．

4.1.2 プレイアウトの方策の設定

プレイアウトの方策はまずランダムに一定回数探索を行った後に式 (6) を満たすときにプレイアウトの方策を PIMC 探索として再度探索を行う．

$$v(a^*) - v(a_i) < \frac{1}{\sqrt{W}}(\sigma_a^* + \sigma_i) \tag{6}$$

a^* はランダムプレイアウトで評価値が一番高い行動，σ_i は行動 i のプレイアウトで得られた報酬の標準偏差，$v(a_i)$ は a_i の評価値である．この二つの方策をゲームの状態に応じて使い分けることにより，IIMC 探索と比べて探索回数を減らすことができる．

プレイアウト時に PIMC 探索が必要となるのは，評価値が高い終端ノードが少なくかつその周りの評価値が低い時である．その状態のときに PIMC 探索で探索することによってより高い報酬を獲得できる行動を探索することができ，ノードの評価値が終端ノードの評価値が高いノードを選択した回数が多い評価値となる．このことからプレイアウトに PIMC 探索を使うべき状態は最も評価値が高いノードと他のノードとの評価値の差が僅差，かつ評価値が高いノードと低いノードが混在している，つまりプレイアウトで得られた報酬の標準偏差が大きいノードが存在することである．また，真の状態と異なっている状態が多ければ多いほど，サンプリングを行ったときに真の状態である確率が下がるため，PIMC 探索による再探索の基準を式 (6) で与えた．

プレイアウトの方策として PIMC 探索を適用した際に着手を決定する計算時間は，PIMC 探索中のプレイアウト回数を M，現在の手番の状態からゲーム終了までの行動の回数を d とすると $O(Md)$ である．ランダムプレイアウトは $O(1)$ であるため，ランダムプレイアウトよりも良い着手が見つかる可能性がある時のみ PIMC 探索を行うことにより，すべての状態に対してプレイアウト時に PIMC 探索

を行う IIMC 探索よりも計算量を削減できる．

4.2 AMCTS のアルゴリズム

現在の手番の状態 s において選択できる行動の集合を A_s とする．集合 A_s の要素 $a \in A_s$ の行動の評価値を $v(a)$ とする．以下に AMCTS の詳細なアルゴリズムを示す．

step1 初期化
s から取りうる行動を列挙する．列挙した行動は A_s に追加する．A_s の全ての要素の行動の評価値を $v(a) = 0$ に初期化する．

step2 プレイアウト回数の計算
s や $|A_s|$ に応じたプレイアウト回数 N を式 (5) で求める．

step3 状態サンプリング
s を含む取り得るすべての状態 S を求め，S の中からランダムに一つサンプリングする．

step4 ノード選択
UCB1-tuned にて探索すべき行動 a を選択する．

step5 プレイアウト
step4 で選択した行動 a を行い，状態遷移させたのちランダムプレイアウトを行う．プレイアウトの結果を $v(a)$ に加算する．

step6 繰り返し
step3-5 を N 回繰り返す．

step7 探索終了判定
式 (6) を満たす場合は **step8** を行う．満たさない場合は **step9** を行う．

step8 再探索
A_s の全ての要素に関して $v(a) = 0$ にした後，**step3-5** を N 回繰り返す．ただし，**step5** のプレイアウト中の方策は相手の行動はランダムで選択し，自分の行動の PIMC 探索によって選択する．

step9 探索終了
$\arg\max_a v(a)$ を満たす a を選択する．

5. 評価実験

5.1 実験設定

実験は Bias を再定義した Synthetic Tree を用いて行った．$lc = 0.3, bias = 0.75, W = 8, d = 6, |A| = 2$ とし，報酬の範囲を 0 から 4 までの整数とした．df の値を 0.1 ずつ変更し，Synthetic Tree を作成した．この Synthetic Tree に対して，AMCTS と PIMC 探索，IIMC 探索をそれぞれ先攻後攻を入れ替え，計 3000 回ずつ対戦を行った．IIMC の方策は PIMC 探索を使用した．実験はプレイアウト回数を固定する実験と，一つの着手を決めるために使える計算時間を固定する実験の二種類行った．プレイアウト回数は $N = 400$ とし，プレイアウト中の PIMC 探索のプレイアウト数は $M = 50$ とした．AMCTS のプレイアウト数は式（5）より $W = 8, A = 2$ のとき $N = 400$ となるように $C = 25$ とした．計算時間 t は 0.01,0.001 で固定し，計算時間が t を超えた場合はその時点での評価値を基に着手を決定した．この時の各プレイヤーの一試合あたりの平均獲得報酬と時間を記録した．

5.2 実験結果

プレイアウト回数を固定した実験結果は**図 2**，**図 3**，**図 4**，**図 5** のようになった．図 2, 図 4 は df を横軸にとったときの一試合あたり平均獲獲得報酬を表している．図 3, 図 5 は df を横軸にとったときの一試合あたりの計算時間を表している．各グラフのエラーバーは 95%の信頼区間を表している．図 2 より df が高いとき，つまり不完全情報があまり多くないときに AMCTS が PIMC 探索よりもより多く報酬を獲得している．図 4 より AMCTS が若干性能が下回っているが，df が大きくなるにつれて計算時間が短くなっており，最大で約 42%短くなっている．

計算時間を固定した実験結果は**図 6**, **図 7**，**図 8**, **図 9** のようになった．同じ計算時間で性能を比較した場合，図 6 より，df が大きい値では AMCTS は PIMC 探索よりも高い性能を示した．また図 7 より，df が大きい値では図 4 と比べて性能が向上していることが分かる．これらの結果についての考察は 6.3 節で述べる．

6. 考察

実験結果より AMCTS が PIMC 探索よりも高い性能を示すことができた．その要因は式 (5)(6) である．これらの式についての考察を行う．

6.1 プレイアウト回数の考察

プレイアウト回数は性能を大きく左右する要因となる．プレイアウト回数を増やすことによって，多くの状態における行動の評価を行うことができ，評価値がより信頼できるものになる．既存の手法ではプレイアウト回数を事前に与えていた．しかしのその方法ではゲーム全体で考えた時にすべてのゲームの状態で適切なプレイアウト回数であるとは言えない．その理由は二つ挙げられる．

まず一つ目の理由は状態の候補の数が変化することである．サンプリングする際，候補数に比例してサンプリングを行う必要がある．サンプリングの回数はプレイアウト回数と等しく，候補数 W が小さい場合，プレイアウト回数を減らすことができる．

二つ目の理由は現在の手番から選択できる行動の数がゲームの状態によって変化することである．行動の数に比例したプレイアウト回数が必要である．ゲーム終了までに遷移しうる状態においても行動の数が変化するが，どの状態に遷移するかは不明であるため，考慮してない．

今回の実験では $A = 2$ で固定であったが，df が大きい

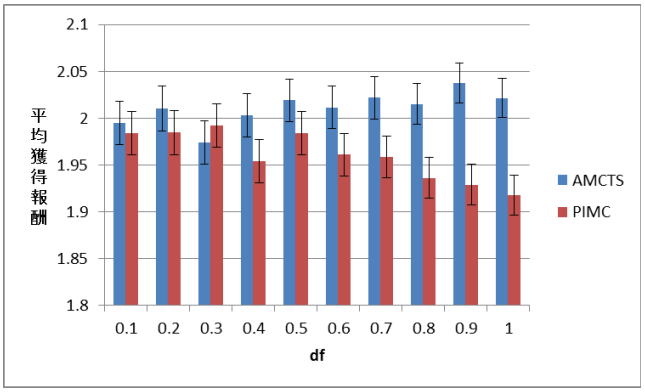

図 2　N 固定獲得報酬（AMCTS 対 PIMC 探索）

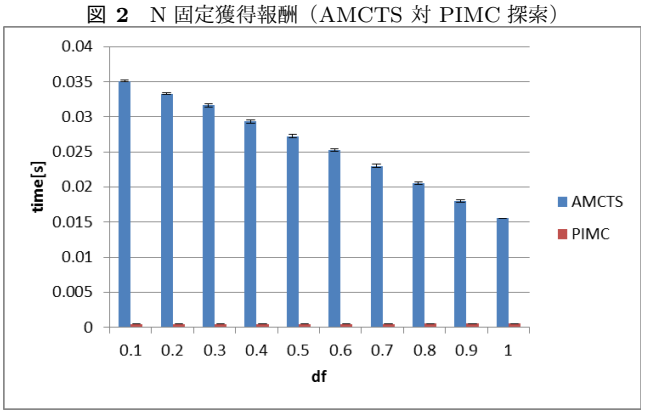

図 3　N 固定計算時間（AMCTS 対 PIMC 探索）

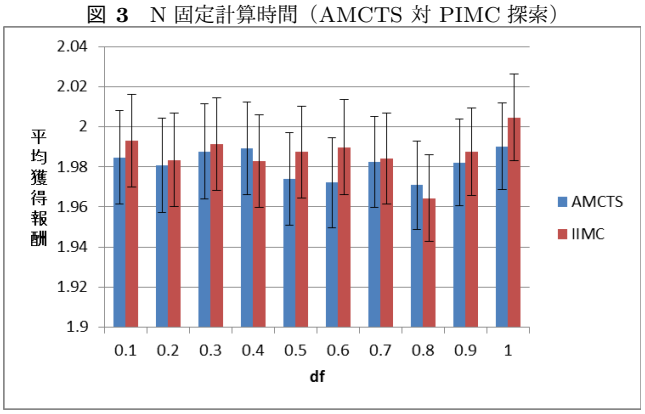

図 4　N 固定獲得報酬（AMCTS 対 IIMC 探索）

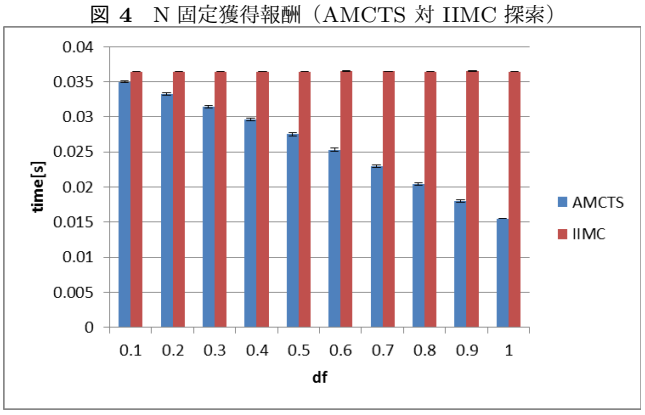

図 5　N 固定計算時間（AMCTS 対 IIMC 探索）

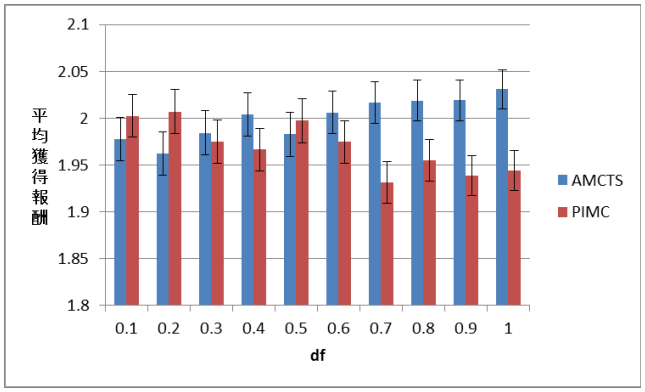

図 6　t=0.01 獲得報酬（AMCTS 対 PIMC 探索）

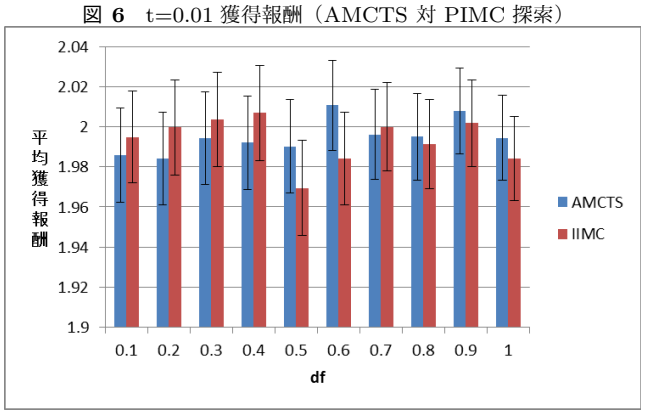

図 7　t=0.01 獲得報酬（AMCTS 対 IIMC 探索）

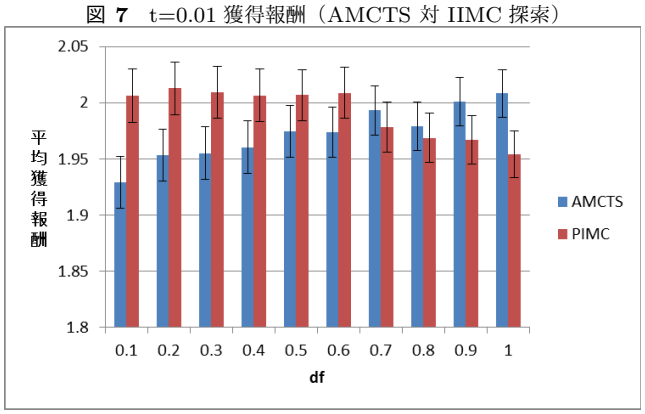

図 8　t=0.001 獲得報酬（AMCTS 対 PIMC 探索）

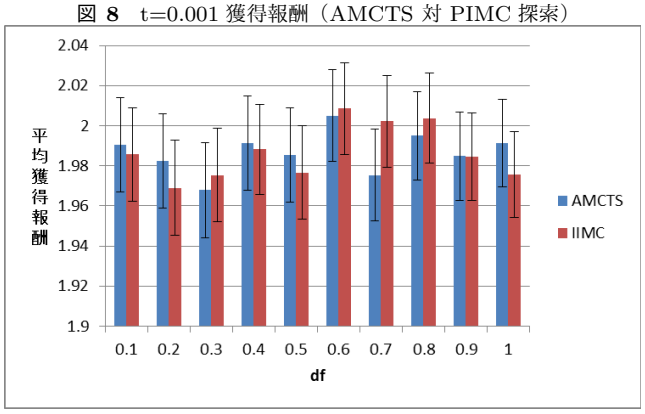

図 9　t=0.001 獲得報酬（AMCTS 対 IIMC 探索）

とき，つまり W が小さくなりやすいとき，計算時間が短縮されているのが分かる．

6.2　プレイアウトの方策についての考察

プレイアウトの方策をランダムとすると探索一回あたりにかかる時間が短いが，最適解を見つけにくいという特徴

がある．しかし，不完全情報が多い状態のときにはすべての状態を網羅的に探索するのは効率が良くない場合がある．そのような場合に対してはランダムプレイアウトは有効に機能する．一方でプレイアウトに PIMC 探索を用いた場合はランダムの方策よりも最適解を見つけやすいが，一回の計算に時間がかかり限られた時間の中で状態を評価することは困難である．よってこれらの探索手法を使い分けることが非常に効果的であると考えられる．例えば，ランダムプレイアウトの最適解を見つけにくい状態とは行動の順番が重要な場合である．例えば囲碁における定石では，石を置く順番が最終的な勝ち負けに大きく影響する．このような状況では，PIMC 探索によるプレイアウトが有効であると考えられる．そのため不完全情報の数，状態の評価値によってプレイアウトの方策を変更する必要がある．図 2 より，AMCTS は式 (6) の方策の切り替えがうまく機能しており，高い性能を示している．また図 4 より，AMCTS はプレイアウトの方策をすべて PIMC 探索を行っている IIMC 探索と比べてほぼ変わらない性能を示している．以上より，プレイアウトの方策をランダムでよいところはランダムで探索を行い，探索回数を増やすべきところでは PIMC 探索を使い分けることにより探索スコアを向上させ，更に計算時間を短縮することに成功したと考えられる．

6.3 実験結果の考察

図 2，図 4 より AMCTS は PIMC 探索よりも高い性能を示しているが，AMCTS と IIMC 探索と比べるとほぼ同じ性能を示した．IIMC 探索の性能を上回るためには式 (6) を改良するか，プレイアウトの方策を新たに追加する必要がある．IIMC 探索は不完全情報が多いときにはうまく性能を発揮できないため，不完全情報に強いアルゴリズムが適していると考える．

計算時間については図 3，図 5 より df に比例して減少しておりかつ，性能も向上していることから式 (5) の有効性が示されたが，AMCTS の計算時間は PIMC 探索の計算時間と比べて多く時間がかかっている．そのため，図 8 より AMCTS に十分な計算時間が与えられなかった場合の性能の比較を行うと，df が小さいときに性能に大きな差があることが分かる．これは AMCTS がプレイアウトの方策を PIMC 探索とする場合，以前の探索した結果を破棄して新たに探索を始めるからであると考えられる．これを解決するためには以前の探索の結果を破棄するのではなく，以前の探索の結果と新たな探索の結果をうまく組み合わせる必要がある．

7. まとめ

本報告では不完全情報ゲームをプレイするためのアルゴリズムである AMCTS を提案した．この手法は IIMC 探索を応用したアルゴリズムであり，プレイアウト中の方策をゲームの状態に応じて変更することによって適切な探索を行うアルゴリズムである．またプレイアウト回数をゲームの状態に応じて変更し，計算時間の削減を行った．実験より，提案手法が IIMC 探索や PIMC 探索と比べて性能を向上させることに成功し，計算時間の短縮に成功した．しかし，十分な計算時間が与えられなかった場合は PIMC 探索に劣ることがあり，また IIMC 探索と比べて計算時間は改善されているが，性能を上回ることができなかったことが今後の課題である．

本報告の評価実験では人工的な不完全情報ゲームのモデルである Synthetic Tree で提案手法の評価を行った．今後は木の深さや，枝の数からも状態を判断しハーツや大貧民といった実際のゲームに対して評価実験を行いたい．その際に Synthetic Tree で用いた各パラメータが実際のゲームの特徴を適切に表現しているかどうかを確認する必要がある．

参考文献

[1] M.Ginsberg, GIB: imperfect information in a computationally challenging game, Journal of Artificial Intelligence Research 14 pp.303 - 358 2001.
[2] T. Furtak, M. Buro, Recursive Monte Carlo Search for Imperfect Information Games, Computational Intelligence in Games (CIG).
[3] Peter I. Cowling, Edward J. Powley, Daniel Whitehouse, Information Set Monte Carlo Tree Search IEEE Transactions on Computational Intelligence and AI in Games VOL.3, NO.2, JUNE 2012.
[4] J. R. Long, N. R. Sturtevant, M. Buro and T. Furtak, Understanding the Success of Perfect Information Monte Carlo Sampling in Game Tree Search, Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence 2010.
[5] P. Auer, N. Cesa-Bianchi and P. FISCHER, Finite-time Analysis of the Multiarmed Bandit Problem, Kluwer Academic Publishers. Manufactured in The Netherlands 2002.
[6] T. Cazenave, Nested Monte-Carlo Search, Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence 2009.
[7] L. Kocsis, C. Szepesvari, Bandit Based Monte-Carlo Planning, Machine Learning: ECML 2006.