Vol.2012-MPS-90 No.11
2012/9/19

ゲーム木のモンテカルロ探索と詰め探索の併用について

但馬 康宏[1,a)]

概要：現在のゲーム木探索に関する研究は，大きく分けて，(1) 評価関数を用いた評価値の探索，(2) モンテカルロ法による探索 (モンテカルロ木探索)，(3)AND-OR 木に対する探索 (詰め探索)，の 3 つの課題に分類できる．それぞれにおいて，代表的な効率の良いアルゴリズムが知られており，評価値の探索においてはα β法が，モンテカルロ法においては UCT 法が，詰め探索においては df-pn 法が有名である．本研究では，ペンタゴと呼ばれるゲームにおいて，UCT アルゴリズムによるモンテカルロ木探索と df-pn アルゴリズムによる AND-OR 木探索を併用し強化がなされることを実験により確認した．その結果，探索時間が短く，モンテカルロ法において十分ではない場合に詰め探索による補助が有効であることを確認した．

キーワード：ゲーム木探索，モンテカルロ木探索，詰め探索，UCT，df-pn

Combining UCT and df-pn algorithm in the game tree search

Yasuhiro Tajima[1,a)]

Abstract: Game tree search is a study field to select the best move for a specific game. There are three major approaches for this problem. (1) Evaluation function and searching its min-max value. alpha-beta algorithm is the most famous searching algorithm for a game tree. (2) Monte-Carlo tree search : UCT is an application of k-armed bandit problem to the game tree search. (3) AND-OR tree search : the proof number search is an algorithm to find the best move which leads to a winning node, and the df-pn is its effective algorithm. In this paper, we combine UCT and df-pn algorithms in the game named “pentago”. From the evaluation, we confirmed the combined algorithm is more efficient than the normal UCT. Especially, this combination is powerful if search time is not enough.

Keywords: game tree search, Monte-Carlo tree search, AND-OR tree, UCT, df-pn

1. はじめに

ゲームにおいて強い手を選択するアルゴリズムは，ゲーム木の探索によって行われることが一般的である．特にゲーム木が決定性計算により作成できる二人完全情報ゲームにおいては，ゲーム木を展開し，途中局面を評価する評価関数の min-max 値を求めることにより強い手を選択する．このとき，評価関数の作成が重要となり，ゲームの強さそのものを左右する．

近年，モンテカルロ法によるゲーム木探索も注目を集めている [2]．これは，ゲーム木をランダムに手を選択しながら末端まで探索 (ランダムシミュレーション) し，その勝敗を統計情報として保存する．ランダムシミュレーションの数を増やすことにより，より強い手を選択する手法である．この方法の特徴として，評価関数が作りにくいゲームにおいても勝敗の決定ができればよい手を見つけられる点が挙げられる．

ゲームの完全解析においては，ゲーム木を AND-OR 木として探索することにより注目節点の必勝，必敗を判定するアルゴリズムが盛んに研究されている．この手法の代表的なアルゴリズムとして，証明数探索 (proof number search) [3] が挙げられる．これは，探索中の節点について，必勝必敗を決定するために必要な残り節点数の見積もりを

[1] 岡山県立大学 情報システム工学科
Department of Systems Engineering, Okayama Prefectural University, 111 Kuboki, Soja, Okayama 719-1197, Japan
[a)] tajima@cse.oka-pu.ac.jp

基準に最良節点を決め，探索を行う手法である．局所的な情報から深さ優先で証明数探索を行う df-pn アルゴリズム [4] が高性能な探索法として知られている．

本研究では，ペンタゴとよばれる盤面が回転する五目並べについて，モンテカルロ法と証明数探索を併用した場合の強さについて評価実験を行った．ペンタゴは，評価関数を作成するための効果的な評価要素が知られておらず，また，盤面上の石の数が単調増加であるなど，モンテカルロ法による実装に適したゲームである．モンテカルロ法に UCT アルゴリズムを用い，詰め探索に df-pn を用いた．併用は，まず始めに詰め探索により一定時間探索を行い，必勝手が見つからなかった場合は，残りの時間を UCT 探索によりよい手を見つけることとした．評価実験として，詰め探索を用いない UCT との対戦を行い，その勝敗を調べた．その結果，探索時間が短く，UCT アルゴリズムのみでは強い手を見つけにくい状況では，詰め探索を併用した本手法の優位が認められた．

2. UCT アルゴリズム

UCT アルゴリズムは，ゲーム木の各節点において，手の選択を k-armed バンデット問題としてモデリングすることにより探索を行うアルゴリズムである．k-armed バンデット問題は以下のように定義される問題である．

- k 個の確率変数 $X_1, X_2, \cdots, X_k$ は $0-1$ の範囲の値をとる．それぞれの期待値はあらかじめ定まっているが未知であり，アルゴリズム実行中に変化しないものとする．これらを arm と呼ぶ．X_i の期待値を μ_i で表す．X_i の値は分布 P_i にしたがうものとする．
- 1 回の試行とは，X_i $(i = 1, 2, \cdots, k)$ を 1 つ選択し，その値を得ることにより終了する．
- n 回の試行を繰り替えした後の総獲得値 $\sum_{j=1}^{n} X_{\Lambda(j)}$ を最大化することが目的である．ここで，$\Lambda(j)$ は，j 回目の試行における選択を表す．すなわち，j 回目の試行で X_i を選択した場合，$\Lambda(j) = i$ である．

μ_i $(i = 1, 2, \cdots, k)$ の中で最大のものを μ^* と表す．同様に $\mu^* = \mu_i$ である i について，X_i を X^* と表す．あるアルゴリズムにおいて, 合計 n 回の試行の中で X_i を試行した回数を $T_i(n)$ と表す．また，$\bar{x_i} = (1/T_i(n)) \sum_{1 \le j \le n, \Lambda(j)=i} X_i$ とする．さらに，$\Delta_i = \mu^* - \mu_i$ とする．

UCB1 アルゴリズムは，以下のとおりである．

(1) すべての X_i $(i = 1, 2, \cdots, k)$ について 1 度づつ試行する．

(2) 以下の値がもっとも大きな j について試行を行う．

$$\bar{x_j} + \sqrt{\frac{2 \ln n}{n_j}}$$

ここで n は現時点での総試行回数であり，n_j は現時点までに X_j を試行した回数である．

(3) 前ステップを繰り返す．

このとき，以下の定理が成り立つ [1].

定理 1 UCB1 アルゴリズムを用いて n 回の試行を行ったときの選択の系列を $\Lambda_u(k)$ $(k = 1, 2, \cdots, n)$ とする．このとき, $n \cdot \mu^* - \sum_{k=1,\cdots,n} X_{\Lambda_u(k)}$ は以下の値で抑えられる．

$$\left(8 \sum_{i:\mu_i < \mu^*} \frac{\ln n}{\Delta_i} \right) + \left(1 + \frac{\pi^2}{3} \right) \left(\sum_{j=1}^{k} \Delta_j \right)$$

最適な arm を知っている場合の獲得量と実際の差 $n \cdot \mu^* - \sum_{k=1,\cdots,n} X_{\Lambda_u(k)}$ を以後，後悔値と呼ぶ．

さらに，任意の $i = 1, 2, \cdots, k$ について，

$$E(T_i(n)) \le \frac{8 \ln n}{\Delta_i^2} + 1 + \frac{\pi^2}{3}$$

が成り立つ．すなわち，ある arm i が試行される回数の期待値は，Δ_i の二乗に反比例する．

囲碁や将棋などの二人確定完全情報零和ゲームの過程は，初期局面を根，可能手を枝に持ち，その可能手により進められた局面を子節点にもつゲーム木で表現することができる．上記 UCB1 アルゴリズムは，以下のようにゲームにおける着手選択に適用される．

- 現在の局面に対応するゲーム木の節点から，直接枝が張られているすべての子節点を k-armed バンデット問題の arm に対応させる．
- UCB1 アルゴリズムにおける試行は，対応する子節点からランダムに着手を選択し対戦結果をシミュレートし，その結果が勝ちならば $X_i = 1$ 負けならば $X_i = 0$ とする．

UCT アルゴリズムは図 1 に示すアルゴリズムで行われるゲーム木探索である．すなわち，ランダムシミュレーションを行うにあたり，部分的なゲーム木を展開し，木に含ま

```
UCT algorithm
初期ゲーム木 t を作成する; (ex. 根 1 段の子節点)
各節点の勝敗数を初期化;
while (制限時間まで繰り返す) begin
    注目節点 := 根;
    while (注目節点が t の葉となるまで) begin
        注目節点から UCB1 アルゴリズムにより子節点 s
        を 1 つ選択し，新たに注目節点とする;
    end
    注目節点からランダムシミュレーションを行う;
    結果を根から注目節点までの各節点に反映させる;
    if (注目節点の訪問回数が展開条件を満たした) then
        注目節点のすべての可能手に対する子節点を t に付け加える;
    fi
end
根の子節点のうち勝率が最大となる手を選択する;
```

図 1 UCT アルゴリズム

れる節点においては UCB1 アルゴリズムにて次の手を選択し，葉に到達したらランダムに次の手を選択することで 1 回のシミュレーションとする手法である．この方法により，1 回のランダムシミュレーションの前半は過去の統計情報に基づいた選択でゲームが進められ，木の葉に到達した時点でランダムに着手が選択されてゲームが進められる．過去の統計情報に基づくことにより，木の探索においては min-max 探索と同じ効果をもたらしている．

UCT アルゴリズムの特徴として，ランダムシミュレーションの回数が無限大となっても，ゲーム木の根における各可能手の勝率は，各子節点より下の節点の後悔値分だけ 1 より小さな値を上界値として持つ．したがって，無限にシミュレーションを繰り返しても，必勝の手を見つけるとは限らない．同様に必敗の手に関しても，相手の後悔値の分だけ勝率が得られるため，0 より大きな値を下界値として持つ．したがって，この場合も必敗の手を排除できるとは限らない．

3. df-pn アルゴリズム

ゲーム木をその必勝，必敗のみに注目すると，AND-OR 木として表すことができる．すなわち，自分の手番においては，いずれかひとつの可能手が必勝であればそれを選択するため OR ノードとなり，相手の手番に対応する節点では，すべてが必勝 (相手の必敗) となればその節点が必勝 (相手の必敗) となるので AND ノードとなる．この AND-OR 木において，葉が真 (勝ち) となる節点にたどり着ける手を見つけるアルゴリズムが詰め探索である．一般的な評価値の min-max 値を求める手法と同じように探索することもできるが，中間節点の値が定められていないため，葉からの根に向けての値の再評価以外に本質的な計算方法が存在しない．これを効率的に行うために，証明数探索が提案されている [3].

AND-OR 木の各節点 v において，証明数 pn_v および反証数 dn_v を以下のように定義する．

- 先手勝ち節点: $pn_v = 0, dn_v = \infty$
- 後手勝ち節点: $pn_v = \infty, dn_v = 0$
- 中間節点:

$$pn_v = \min_{child\ u} pn_u, \quad dn_v = \sum_{child\ u} dn_u$$

- 未展開の葉: $pn_v = 1, dn_v = 1$

この値を OR ノードでは pn が最小，AND ノードでは dn が最小の子節点を選択することにより，次に展開すべき未展開の節点 (現在は葉) を見つける．必勝手が見つかれば，根 r において $pn_r = 0$ となり，根が必敗局面ならば $dn_r = 0$ となる．図 2 のアルゴリズムで順次木を展開して探索を行う．

証明数探索は，根から現在展開中の最良の葉まで逐次選択を行いながら，ゲーム木の展開を行う．証明数が少ない

```
PN algorithm
根 r のみの木 t を作る;
注目節点 c := r とする;
pn_c := 1, dn_c := 1;
while (pn_r ≠ 0 かつ dn_r ≠ 0)
    for (t の葉にたどり着くまで)
        c が OR 節点ならば pn 最小,
        c が AND 節点ならば dn 最小となる
        子節点 d について c := d とする;
    end
    if (c は先手勝ち節点)
        pn_c = 0, dn_c = ∞;
    else if (c は後手勝ち節点)
        pn_c = ∞, dn_c = 0;
    else
        c を展開し，すべての子節点 e で
        pn_e := 1, dn_e := 1;
    fi
    while (c ≠ r)
        pn_c = min_{child u} pn_u,
        dn_c = Σ_{child u} dn_u;
        c := c の親節点;
    end
    pn_c = min_{child u} pn_u,
    dn_c = Σ_{child u} dn_u;
end
```

図 2 証明数探索

節点は，勝敗を調べるべき残りの葉が少ないと見込まれ優先的にチェックが行われる．証明数を評価値の min-max 探索のように，その節点より深く探索すべきか否かを判断する材料として用い，深さ優先探索で証明数探索と同じ探索結果を出力する探索法が df-pn[4] アルゴリズムである．pn, dn の他に閾値 thp, thd を用いて，注目節点の証明数 (反証数) が閾値よりも大きくなれば，そこでの探索を打ち切り親節点に向かい，閾値よりも小さければ，適切な子節点を探して探索を続ける．すなわち，注目節点 c において以下の操作を行う．

(1) 証明数，反証数を求め，与えられた閾値より大きければ注目節点を親に移す．

(2) 子節点のうち，証明数 (子節点の反証数) が最も小さいもの e を選び，注目節点をその子節点に移す．このとき，子節点の閾値を

- 証明数の閾値 thp_e は，thd_c から e の兄弟節点の証明数を減じたものとし，
- 反証数の閾値 thd_e は，thp_c と e の次に証明数が小さい e の兄弟節点の証明数に 1 を加えたもの小さい方とする．

この閾値を超える値を持つ節点は，未検証の節点を閾値以上に持っていることとなり，そのまま探索を続けることは効率的ではないので，注目節点を親節点に

移す.

図 3 にアルゴリズムを示す．証明数探索のアルゴリズムに比べ，証明数反証数を AND-OR ノード区別なく扱えるように反転させている点に注意.

dfpn algorithm
根 r のみの木 t を作成する;
注目節点 $c := r$ とする;
$thp_c := \infty, thd_c := \infty$;
$pn_c := 1, dn_c := 1$;
while ($pn_r \neq 0$ かつ $dn_r \neq 0$)
  if (c は手番の勝ち節点)
    $pn_c = 0, dn_c = \infty$;
    $c := c$ の親節点;
    continue;
  else if (c は手番の負け節点)
    $pn_c = \infty, dn_c = 0$;
    $c := c$ の親節点;
    continue;
  else
    c が未展開ならば展開し，すべての子節点 e で
    $pn_e := 1, dn_e := 1$;
  fi
  $pn_c = \min_{child\ u} dn_u$,
  $dn_c = \sum_{child\ u} pn_u$;
  if ($thp_c < pn_c$ or $thd_c < dn_c$)
    $c := c$ の親節点;
    continue;
  fi
  c の子節点 f のうち，dn_f が最小のものを d とする;
  c の子節点 f のうち，dn_f が 2 番めに小さいものを e とする;
  $thp_d := thd_c + pn_d - dn_c$;
  $thd_d := \min(thp_c, dn_e + 1)$;
  $c := d$;
end

図 3 df-pn アルゴリズム

本研究における評価対象のゲームでは，引き分けが存在するため，図 3 のアルゴリズムにその処置が必要となる．具体的には引き分けの葉 c において $pn_c = 0$, $dn_c = 0$ とし，必勝必敗局面とみなされないよう処置を行った.

4. 評価実験

前章までのアルゴリズムを用いて，ゲーム木探索アルゴリズムを実装した．評価の対象とするゲームはペンタゴとよばれる二人完全情報ゲームである.

4.1 ペンタゴ

ペンタゴは，盤面の回転する五目並べである.

- 盤面は，6x6 の 36 マス
- 盤面の縦横中央にそれぞれ分割線があり，3x3 の部分盤面 4 つが並んで，全体の盤面となる.
- それぞれの部分盤面は，3x3 の中央のマスを中心に，左右に回転させることができる.

ゲームは以下の操作を繰り返すことにより進められる.

(1) 手番のプレイヤーは，任意の空きマスに自石をひとつ置く.

(2) 3x3 の部分盤面のひとつを右か左に 90 度回転させる.

以上を互いに繰り返し，先に五目並べた方が勝ちである．具体的な勝利条件を以下の通りとした.

- プレイヤーが石を置いた直後，および回転を行った直後にそれぞれのプレイヤーごとに以下のチェックを行う.
  (1) 六目が直線に並んでいる本数 n を数える.
  (2) 上記の六目以外に五目が直線に並んでいる本数 m を数える.
- $n + m$ が大きいプレイヤーを勝ちとする.
- $n + m$ が同数ならば，ゲーム続行とする.

このゲームは，五目並べと同じ勝利条件やルールであるにもかかわらず，盤面が回転することで可能手数が多くなり，ゲームの完全解析はなされていない．また，五目並べの戦術が利用可能かどうかもはっきりせず，モンテカルロ法による着手選択が有効なゲームであると思われる.

4.2 実験内容と結果

対戦実験として，以下の 2 つのアルゴリズムを対戦させた.

- 探索時間の前半 1/n を df-pn で探索し，必勝手が見つからなかった場合は，残り時間を UCT で探索する (提案アルゴリズム).
- 探索時間のすべてを UCT で探索する (比較元アルゴリズム).

時間配分は $n = 2, 3$ の 2 種類とし，探索時間は，1 手 30 秒，40 秒，60 秒の 3 種類で実験を行った．表 1 に対戦結果を示す.

詰め探索の時間は，全体の 1/3 とすると少なすぎ，勝率は悪くなっている．先手後手合計の勝率が最も高かったのは，全体の探索時間を 30 秒とし，詰め探索に利用する時間を全体の 1/2 とする場合だった．これは，全体の探索時間が短いほど，UCT のみでは強い手を選択できず，詰め探索の効果があらわれた結果とみることができる．探索時間 60 秒で詰め探索を 1/3 とした場合，先手の勝率が実験結果の中で最も高く，75% の勝率となった．これは，先手は攻撃に注力したほうが強い手となり，後手は相手の手に対応する手を探した方が強い手となる特徴を表していると言える．すなわち，先手の場合は詰めを見つけることが勝率のアップに直結するのに対して，後手の場合は，詰め探索に時間を割いてよい手を見つけられずに，少ない時間で UCT 探索を行っていることを表している.

表 1 提案アルゴリズムの勝率

		探索時間 (s)		
		30	40	60
$n = 2$	先手	13.5 / 20 = 0.675	13.5 / 20 = 0.675	10.5 / 20 = 0.525
	後手	11 / 20 = 0.550	8 / 20 = 0.400	6 / 20 = 0.300
	合計	24.5 / 40 = 0.613	21.5 / 40 = 0.538	16.5 / 40 = 0.413
$n = 3$	先手	7 / 20 = 0.350	9.5 / 20 = 0.475	15 / 20 = 0.750
	後手	9 / 20 = 0.450	10 / 20 = 0.500	7 / 20 = 0.350
	合計	16 / 40 = 0.400	19.5 / 40 = 0.488	22 / 40 = 0.550
		提案アルゴリズムについて：(勝利数) / (ゲーム数) = (勝率)		

後手の勝率が先手を上回ったのは，探索時間が 30 秒と 40 秒でいずれも詰め探索は全体の 1/3 とした場合である．これは，詰め探索に割く時間が最も少ない場合の 2 通りであり，詰め探索が先手における探索に有利に働いていることを裏付けている．また，全体の探索時間が短めの場合に，詰め探索がより有効に働いている．

5. おわりに

可能手の数が多く，盤面の状況が大きく変化するゲームにおいて，UCT と df-pn を組み合わせたアルゴリズムを実装し，対戦実験を行った．その結果，詰め探索は，全体の探索時間が短い場合により有効に働き，先手において詰めを発見する場合に効果があることが確かめられた．現在は，UCT と df-pn でゲーム木を共有するのみで，お互いの計算結果を利用していないが，今後の課題として相互の計算結果の逐次的な利用法が考えられる．

参考文献

[1] P. Auer, N. Cesa-Bianchi, P. Fischer, Finite-time analysis of the multiarmed bandit problem, Machine Learning, 47(2,3), pp.235–256, 2002.
[2] L. Kocsis, C. Szepesvari, Bandit based Monte-Carlo planning, LNCS 4212, pp.282–293, 2006.
[3] L.V. Allis, M. van der Meulen, H.J. van den Herik, Proof-number search, Artificial Intelligence, 66(1), pp.91–123, 1994.
[4] A. Nagai, H. Imai, Proof for the equivalence between some best-first algorithms and depth-first algorithms for AND/OR trees, IEICE Trans. Inf. & Sys., E85-D(10), pp.1645–1653, 2002.