UCT 探索による不完全情報下の行動決定

三木理斗[†1]　三輪　誠[†2]　近山　隆[†1]

麻雀や各種カードゲームのような多人数不完全情報ゲームは相手の戦略や局面状態が不確定であるため，静的評価関数による Minimax 探索アルゴリズムを適用することが難しい．本研究ではこのように知識ベースの手法の利用が難しいゲームで良い手を求めるために，モンテカルロシミュレーションをベースにした UCT 探索を麻雀に対して適用した．UCT はランダムに確定させた局面についてのシミュレーションを繰り返すことで期待値の高い手を求めることができ，さらに鳴きによる手番の入れ替わりなどが生じる場合でも対応できると考えられる．実験で UCT プレイヤは牌譜を用いて学習した線形 SVM プレイヤと同等以上の性能を発揮した．

Decision making with imperfect information using UCT search

Ayato MIKI,[†1] Makoto MIWA[†2] and Takashi CHIKAYAMA [†1]

Minimax search for game trees in multi-player games with imperfect information, such as mahjong or some of card games, becomes quite costly, because strategies of other players can be more complex and precise information of game positions is hidden. This paper proposes a method to apply a UCT search algorithm for multi-player games with imperfect information. In UCT, strategies and imperfect information can be handled by simulations with different random situations. Although our experimental mahjong player does not use any human knowledge on play strategies, it showed better performance than other computer players, including a player with knowledge learned from game records of expert players.

1. はじめに

多人数ゲームや不完全情報ゲームでは，二人完全情報ゲームで一般的に用いられる Minimax 探索の適用が難しい．まず多人数ゲームは，二人ゲームと違って各プレイヤの戦略モデルが単純でないという問題がある．二人ゲームではそれぞれのプレイヤは各々の獲得報酬を最大化し，それは相手の獲得報酬を最小化することと等価である．しかしそのような前提は多人数ゲームでは必ずしも成り立たない．例えば，報酬が最大となる手が複数存在する場合に相手プレイヤがどのようにして手を決定するかは単純には決まらない．したがって，多人数ゲームにおいて適切な探索を行うためには，何らかの方法で相手プレイヤの戦略モデルを仮定しなければならない．また，麻雀の鳴きのように不規則に手番が入れ替わるようなゲームにも Minimax 探索は適用しづらい．次に不完全情報ゲームでは，確定できない情報が存在するために探索空間が非常に大きくなるという問題がある．バックギャモンなどの不確定要素のあるゲームに対して Minimax 探索を応用した例はあるが[1)2)]，それに加えて相手の手札などが不明である不完全情報ゲームでは隠されている情報を何らかの方法で推定するか，可能性のある状態を全て探索対象にしなければならないため困難である．Minimax 探索を利用する場合にはさらに探索打ち切りの局面を評価するための静的評価関数が必要となる点も難しい課題として挙げられる．

これらの問題を解決する方法の一つとして，囲碁などで有効であるとして注目を集めている UCT (UCB applied to Trees)[3)] を用いることが考えられる．UCT とはモンテカルロシミュレーションをベースにした木探索手法の一つで，ランダムシミュレーションによる平均報酬とノードの探索回数を考慮して見込みのある手に対してより多くの探索を行う手法である．実際に UCT を多人数ゲームや不完全情報ゲームに応用した研究の例が存在し[4)5)]，両方の性質を併せ持った多人数不完全情報ゲームに対してもその有効性が期待できると考えられる．

本稿では麻雀における手の探索を目的として実験を

†1 東京大学大学院工学系研究科
Graduate School of Engineering, The University of Tokyo
†2 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo

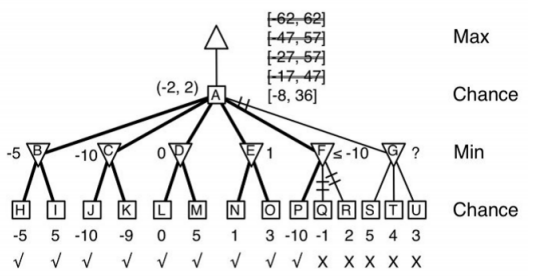

図 1 *-Minimax tree search.

行った．麻雀は多人数ゲームであるため相手プレイヤの戦略が複雑になるが，相手プレイヤの手の選択はUCT アルゴリズムをそのまま利用することである程度多様な戦略モデルに対応した混合戦略を得ることができると考えられる．また麻雀は不完全情報ゲームであるため考えうる全ての場合について探索を行うことが困難であるが，ランダムに生成した局面のシミュレーションを繰り返すことによってそれを補うことができると考えられる．さらに麻雀は静的評価関数を作成することが難しいが，UCT では評価値は終局の結果をそのまま用いることができるため評価関数は必要ない．このように UCT を用いることで，高度な戦略モデルや評価関数を作成することなく，様々なケースに対応した手を選択することができると考えられる．

以降，2 章で関連研究について紹介し，3 章で本研究の手法を示す．そして 4 章で実験結果について示し，最後に 5 章でまとめと今後の課題を述べる．

2. 関 連 研 究

2.1 Minimax 探索の拡張

ここでは Minimax 探索を不確定ゲームに拡張した手法である*-Minimax 探索と，多人数ゲームに拡張した手法である Max^n 探索について紹介する．

2.1.1 *-Minimax 探索

ノード間の遷移が確率的であるような木で Minimax 探索を行う手法として，Ballard によって提案された*-Minimax 探索がある[1)2)]．Minimax 木での Min ノードと Max ノードに対して，子ノードへの遷移が確率的であるようなノードを Chance ノードと呼ぶ．Ballard は Chance ノードを含むようなゲーム木で Alpha-Beta 探索のような枝刈り探索手法を実現するアルゴリズムを提案した．近年になってバックギャモンのプレイヤにそのアルゴリズムを適用する研究が行われ[6)]，枝刈りを行わない場合に対して最大で 95%の探索時間削減を実現したと報告されている．

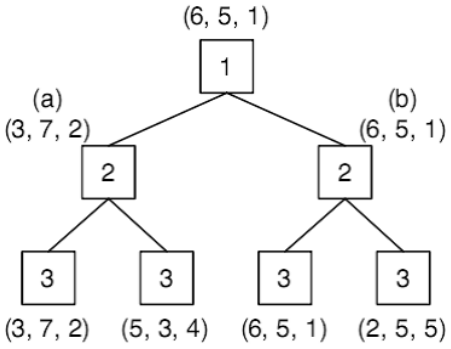

図 2 A sample max^n Tree.

2.1.2 Max^n 探索

Minimax 探索を多人数ゲームに応用した手法として Luckhardt らによる Max^n アルゴリズムがある[7)8)]．図 2 に 3 プレイヤゲームの Max^n 探索木の一例を示す．Max^n 探索各ノードの手番のプレイヤが自身の評価値が最大の手を選ぶという前提で最善手を行う．Luckhardt らはさらに枝刈りによって多人数ゲームでも効率的な探索が可能であることを示した．

2.2 UCT

UCT (UCB applied to Trees) は，多腕バンディット問題における効率的な試行方策である UCB1 を木探索に応用したもので，2006 年に Kocsis らによって提案された[9)3)]．

UCT 探索のアルゴリズムについて簡単に説明する．UCT は図 3 に示すように 4 つのステージからなる．

- 子ノードの選択
  まず未探索の子ノードがある場合にはそれを優先的に選択する．子ノードが全て探索済みの場合，UCT では (1) 式で求められる UCB (Upper Confidence Bounds) 値が最大となる子ノードを探索する．

$$\overline{X}_i + C\sqrt{\frac{\ln T}{T_i}} \tag{1}$$

  $\overline{X}_i$ は子ノード i の平均報酬，T_i は子ノード i の探索回数，T は親ノードの探索回数，C はバランスパラメータである．つまり，基本的には第 1 項の平均報酬の高い手が探索されるが，探索回数が少なくてまだ高い報酬を得る可能性のある手も第 2 項が大きくなるため探索されやすいというアルゴリズムである．
- 子ノードの生成
  子ノードが存在しない場合には合法手を生成して，それに伴って遷移する局面を生成し，新たな子ノードとして追加する．

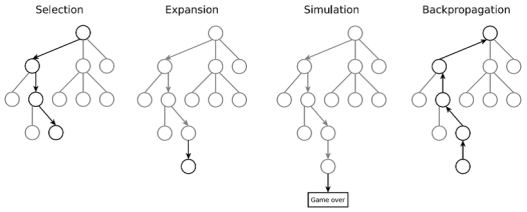

図 3 UCT tree search.

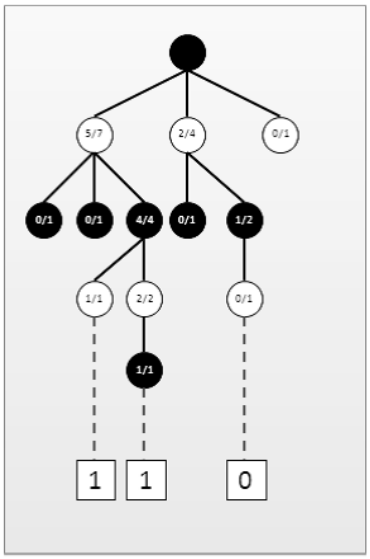

図 4 UCT Tree. The numbers in the nodes are the average payoff (the total payoff / the number of search).

- プレイアウト
  未探索ノードに到達した場合には終局までゲームのシミュレーションを行う．シミュレーションはランダムに行うことも可能だが，ヒューリスティックや評価関数を用いてシミュレーションの質を向上させることもできる[10]．このシミュレーションのことをプレイアウトと呼ぶ．
- 更新
  プレイアウトで得られたゲームの結果そのものを報酬として探索経路上のノードを更新する．

以上の操作を終了条件 (時間や探索回数など) を満たすまで繰り返し，最終的に平均報酬の最も高い子ノードを次の手として選択する．

UCT はシミュレーションを利用したアルゴリズムであるため，精度の高い静的評価関数を使う必要がないのも特徴である．

2.3 多人数ゲームにおける UCT

Sturtevant は 3 プレイヤ以上の多人数ゲームにおける UCT の性能解析の研究を行った[4]．図 2 の (b) のノードに注目すると，このノードの手番であるプレイヤ 2 にとって二つの子ノードの評価値は等しい．この例では単純に最も左の手を選択しているが，このような局面で相手プレイヤがどちらの手を選択するかを判断するためには相手プレイヤの戦略モデルが必要であり，適切な戦略モデルを作成することは一般に難しい．

同様のケースに対して UCT を適用すると，(b) のノードではいずれの子ノードからも同じ報酬が返ってくるため UCB 値による手の選択は探索回数によって決定される．結果的に両方の子ノードは同回数の探索が行われ，親の (b) のノードの評価値は二つの子ノードの平均となる．このようにして，多人数ゲームに UCT を用いるとシミュレーションに基づいた探索が行われるため，相手プレイヤについての混合戦略を得ることができる．

Chinese Checkers (ダイヤモンドゲーム) での実験では，UCT は評価関数を用いた Max^n 探索を上回る性能を発揮したと報告されている．

2.4 不完全情報ゲームにおける UCT

Schäfer は不完全情報ゲームである Skat における UCT の性能解析の研究を行った[5]．Skat は 3 人のプレイヤが手札を 1 枚ずつ出して，最も強いカードを出したプレイヤが得点を取っていくトランプゲームである．3 人の中でソロイストと呼ばれるプレイヤが 1 人存在し，ソロイストは最初に宣言した得点を取ることを目指し，他のプレイヤはそれを阻止することを目的とする．特別な場合を除いて各プレイヤの手札は伏せられているので，Skat は不完全情報ゲームである．不完全情報ゲームに対して UCT を適用するために，Schäfer は UCT の繰り返しの度に相手の手札をランダムに生成して局面を仮定するという操作を行った．仮定とシミュレーションを何度も繰り返すことで，それぞれの手を打った場合の報酬の期待値を得ることができると考えられる．また相手の手札の生成を完全にランダムにするのではなく，これまでのプレーから持っているスートを限定することでより良い性能を出したとも報告されている．実験では Skat のコンピュータプレイヤとして一般的な DDSS (Double Dummy Skat Solver) という手法に対して，互角の性能を出すことができたと報告されている．

2.5 麻雀に関する研究

北川らは将棋における Bonanza の手法に基づいて牌譜の打ち手と評価関数による最善手の一致度の誤差を最小化することを目指し，3 層ニューラルネットワークを用いて打ち手の評価関数を学習する研究を行った[11][12]．ニューラルネットワークの入力層には

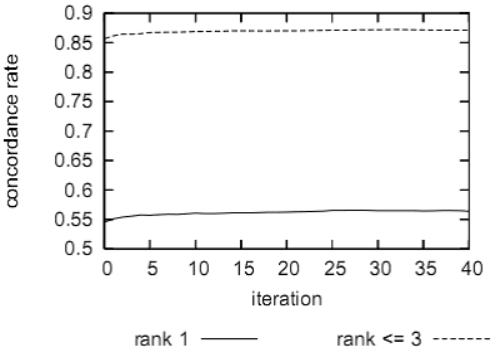

図 5 ツモ局面での一致率 [Kitagawa et al., 2007]

人手で作成した 1,532 の boolean の特徴要素 (表 1) を使用し，6,079 試合分の牌譜からツモ局面については 575,906 局面を学習データとして用い，学習の繰り返し数を 40 回まで増やしながら実験を行っている．結果は図 5 のようになり，最大で 56 %の一致率に到達した．

3. 多人数不完全情報ゲームにおける UCT

この章では本研究の具体的な手法について述べる．Minimax 探索や学習を用いる方法はいずれも評価関数や牌譜など，ゲームに関する知識を必要とするものである．そのため麻雀のように研究が進んでおらず，体系だったコンピュータプレイヤの知識が集積していないゲームにおいては適用が難しい．本研究では多人数不完全情報ゲームである麻雀に対して UCT 探索を適用した．多人数不完全情報ゲームに対して UCT を用いれば，シミュレーションによって複雑な戦略や可能性のある膨大な局面の探索を補うことができるため，多様なケースに対応した手の探索を行うことができると考えられる．また UCT は Minimax 探索と違って不規則な手番の入れ替わりにも対応しやすいため，鳴きがある麻雀にも適していると考えられる．

表 1 麻雀の特徴要素 [Kitagawa et al., 2007]

自分の状態	面前の持ち牌
	面前の持ち牌 2 枚の組み合わせ
	面前の持ち牌 3 枚の組み合わせ
	鳴いた牌の構成と状態
	面子数
	両面搭子数
	カンチャンとペンチャンの和
	対子数
	テンパイしているかどうか
	ドラの枚数
	面前であるかどうか
	親であるかどうか
	リーチしているかどうか
	自分が捨てたことのある牌
他プレイヤの状態	鳴いた牌の構成と状態
	鳴いた回数
	鳴いた牌のドラの枚数
	親であるかどうか
	リーチしているかどうか
	そのプレイヤに対する完全安牌
	筋や壁などで安全度が高い牌
	自分との点差
場の状態	オーラスかどうか
	見えていない牌の残り枚数

手法の全体的な流れは以下のようになる．

(1) 局面中の不完全情報を仮定する
(2) UCT アルゴリズムのループを 1 回行う
(3) 1，2 を繰り返す
(4) 最も平均報酬の高い手を選択する

不完全情報ゲームで探索を行うため，最初に局面の仮定を行う．見えていない牌を数え上げ，それをランダムに振り分けて相手の手牌とツモ山，王牌を生成する．この局面の仮定操作は UCT アルゴリズムの 1 回のループの前に毎回行う．局面を仮定したらルートノードから探索を開始する．ここで，麻雀において手を決定する 2 種類の局面のうち，手牌が全部で 14 枚あってどれか 1 枚を捨てる (あるいは暗カンや加カンをする) 局面を 14 牌ノード，手牌が全部で 13 枚あって相手の捨て牌で鳴くか鳴かないかを選択する局面を 13 牌ノードと呼ぶことにする．それを踏まえて麻雀における UCT 探索のアルゴリズムを Algorithm 1 に示す．

麻雀のゲームの特性上，一般的な UCT とは異なっている個所について説明する．

- 手の生成
  不完全情報局面の仮定によって相手の手牌やツモ山が毎回変わるため，すでに子ノードが展開されているノードでも訪問するたびに手の生成を行う必要がある．そこで新しい手が存在すれば，それに対応する新しいノードを生成して追加する．また，ルートノードでの他プレイヤの鳴きの手は最終的な選択候補に入らないため生成しない．
- 手の選択
  手を選択するときにも手牌やツモ山が変わっているためにすでに展開されたノードの中には打てない手が含まれているので，今回生成した手の中から選ぶようにする必要がある．14 牌ノードにおける手の選択は，その手番のプレイヤが完全に決定権を持つため UCB 値の最大のノードを選択する．一方で 13 牌ノードにおける手の選択は，手番でないプレイヤによる鳴きが入るかどうかで決定権が変わるため，決定権の高いプレイヤの順 (ロン，

明カンまたはポン，チーの順) に UCB 値が鳴かない場合より高いかどうかを調べ，高い場合はその手を選択する．

- プレイアウトと報酬
  プレイアウトはいずれかのプレイヤがあがるか流局になるまでゲームを進め，4 人のプレイヤそれぞれの報酬を観測する．

Algorithm 1 麻雀における UCT 探索アルゴリズム

```
while 終了条件を満たしていない do
  局面を仮定する
  i ← 0
  node[i] ← ルートノード
  while node[i] の探索回数 > 0 do
    if node[i] が終局 then
      break
    end if
    手を生成
    if 新しい手がある then
      新しい子ノードを生成
    end if
    if 生成された手の中に未探索の子ノードがある then
      node[i + 1] ← 未探索の子ノード
    else
      if node[i] が 14 牌ノード then
        node[i + 1] ← 生成された手の中で UCB 値が最大の子ノード
      else if node[i] が 13 牌ノード then
        node[i + 1] ← 生成された手の中で鳴きの決定権と UCB 値によって選択した子ノード
      end if
    end if
    i ← i + 1
  end while
  プレイアウトを行って報酬を観測する
  while i ≥ 0 do
    node[i] の報酬と探索回数を更新
    i ← i − 1
  end while
end while
move ← 平均報酬が最大の子ノードへの手
```

4. 実　　験

本章では，実際に麻雀において UCT 探索を行うプレイヤを実装して行った実験の方法と結果について述べる．

4.1 実 験 環 境

実験は CPU Core2 Duo 3GHz ，メモリ 2GB のマシン環境で行った．実装は C++言語を用いた．

4.2 牌譜との一致率

UCT 探索によって選ばれた手と上級者の牌譜の手との一致率を評価する実験を行った．牌譜データとして，システマティック麻雀研究所[13] で公開されている東風荘の牌譜からランダムに抽出した 1,000 局面を用いた．実験方法として以下の 3 種類のパラメータを変化させたときの一致率の変化を測定した．

- 報酬
- UCB 値のバランスパラメータ C
- UCT アルゴリズムのループ回数

4.2.1 報　　酬

プレイアウトの終局で観測する報酬の種類を以下の 4 つに設定して 14 牌ノード局面での一致率を測定した．

- 得点収支
  あがりの点数による損益をそのまま報酬とした．バランスパラメータ C は得点のスケールに合わせて 1000 に設定した．
- 順位点
  順位点とは，清算後の持ち点から原点（東風荘では 30000 点）を引いて零和化し，1000 で割って順位ごとの賞与点を加えた最終点数である．この点数が試合の結果として各プレイヤの成績に反映される．バランスパラメータ C は順位点のスケールに合わせて 10 に設定した．
- 得点収支を正規化したもの
  (2) 式のようなシグモイド関数を用いて正規化を行った．

$$f(x) = \frac{1}{1 + e^{-kx}} \tag{2}$$

  バランスパラメータ C は $\sqrt{2}$ に設定した．k は 0.0003 に設定した．
- 順位点を正規化したもの
  バランスパラメータ C は $\sqrt{2}$ に設定した．シグモイド関数の k は 0.1 に設定した．
- あがりの二値
  あがったプレイヤの報酬を 1，振り込んだプレイヤもしくはツモられた場合の他 3 人の報酬を 0，

収支のなかったプレイヤの報酬を 0.5 とした．バランスパラメータ C は 1 に設定した．

結果を表 2, 表 3 に示す．UCT のループ回数は表 2 では 100,000 回，表 3 では 10,000 回に設定して実験を行った．結果として，いずれも得点収支を報酬として用いたときが最大の一致率を示した．

4.2.2 UCB 値のバランスパラメータ C

(1) 式のパラメータ C を 100，1000，10000 の 3 種類に設定して 14 牌ノード局面での一致率を測定した．報酬は得点収支，UCT のループ回数は 10,000 回に設定した．結果を表 4 に示す．$C = 1000$ のときが一致率最大となった．これは得点収支のスケールとほぼ一致しているためと考えられる．

4.2.3 アルゴリズムのループ回数

UCT アルゴリズムのループ回数を 100 回から 500,000 回まで変化させて 14 牌ノード局面での一致率を測定した．また，ループ回数を 100 回から 50,000 回まで変化させて 13 牌ノード局面での一致率を測定した．報酬は得点収支を用いた．(1) 式のバランスパラメータ C は 1000 に設定した．

結果を図 6, 図 7 に示す．凡例の Random は完全にランダムに手を選択する場合の一致率，SVM は線形カーネルの SVM-rank[14)15)] で学習した分類器によって得られた最善手との一致率である．SVM の学習にはシステマティック麻雀研究所で公開されている東風荘の牌譜からランダムに抽出した 75 試合分の牌譜を使用し，表 5 に示す約 60,000 の特徴要素を用いて行った．

14 ノード局面においては，ループ回数が 100,000 回のときに UCT の一致率は最高値である 50.0%を記録し，SVM を上回る結果となった．この結果は北川らの研究の結果や SVM の学習に木カーネルを用いた場合には若干及ばないものの[12)16)]，知識を必要としない手法も有効であるということを示していると言える．

表 2 報酬による一致率変化 (100,000 ループ)

報酬	得点収支	順位点	正規化得点	正規化順位点
一致率 [%]	50.0	34.6	32.7	26.6

表 3 報酬による一致率変化 (10,000 ループ)

報酬	得点収支	あがり二値
一致率 [%]	39.7	29.8

表 4 UCB 値の C による一致率変化 (10,000 ループ)

C	100	1000	10000
一致率 [%]	24.3	39.7	33.6

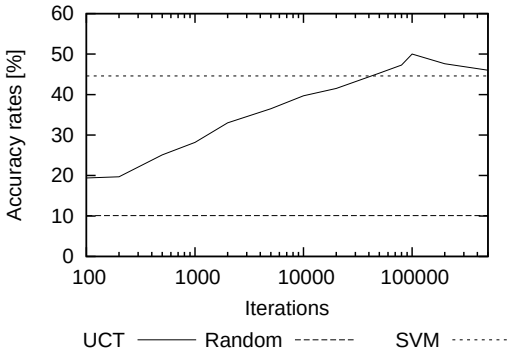

図 6 14 牌ノードでの一致率

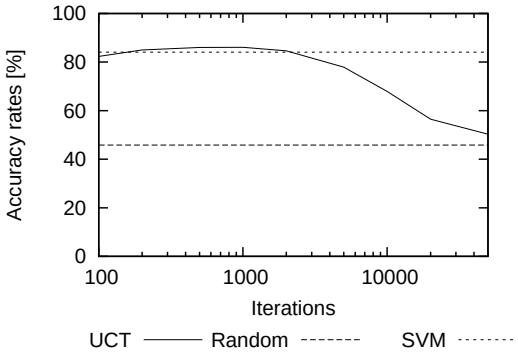

図 7 13 牌ノードでの一致率

一方で 13 ノード局面においては，ループ回数が 10,000 回以上になると一致率が大きく低下するという現象が見られた．13 ノード局面はそのうち 9 割ほどが牌譜では鳴かないという偏りがあるため，図 8, 9 に牌譜で鳴いた局面のみの場合と鳴かなかった局面のみの場合を示した．これによると，ループ回数が増えるにつれて鳴く手が選択されやすくなっていることがわかる．ランダムプレイアウトではほとんどあがることができないため (10,000 回のプレイアウトをテストしたところ，あがりによって終局したのはわずか 38 回であった)，有効な報酬の大半は流局のノーテン罰符によるものとなる．したがって，あがることよりもテンパイすることを目指すような探索になってしまい，シャンテン数を減らせる鳴きの手が選択されやすくなっていると考えられる．プレイアウトに評価関数などを用いて終局であがれる割合を増やせば，この傾向は変わっていくと考えられる．

4.3 コンピュータプレイヤとの対戦

グリーディプレイヤと SVM プレイヤを相手に 2 対 2 の対戦を行った．グリーディプレイヤは全ての合法

手の中で打った後のシャンテン数が最も小さい手の集合を生成し，その中からランダムに手を選択するプレイヤである．対戦は連荘なしの東風戦で行い，ゲームの不確定性による成績のばらつきを抑えるために 1 試合中の 4 局では全く同じ配牌とツモ山になるようにした．席の配置は同種のプレイヤが常に対面の状態で行った．UCT のシミュレーションは 1 手あたり 5 秒に設定し，報酬は得点収支を用いた．(1) 式のバランスパラメータ C は 1000 に設定した．対戦の結果を表 6,7 に示す．グリーディプレイヤとの対戦では UCT プレイヤが和了率，平均収支ともに若干ながら上回っている．同一の配牌，ツモ山に対して，UCT プレイヤの方がより良い手を選択できている可能性があると考えられる．SVM プレイヤとの対戦でもほぼ互角の性能を発揮しており，UCT を用いることで知識なしでもそれなりに良い手を求めることができていると言える．

表 5 SVM features.

手牌	各牌の所持数
	面子の所持数
	含まれる面子の総数
	面子候補の所持数
	含まれる対子の総数
	含まれる両面の総数
	含まれるカンチャンの総数
	含まれるペンチャンの総数
	フーロ数
	面前かどうか
	ホンイツにするための不要牌の数
	チンイツにするための不要牌の数
自分の状態	シャンテン数
	ドラの所持数
	ドラそばの所持数
	親かどうか
	上がり牌の残り枚数
	確定している役数
相手の鳴き牌	フーロ牌
	確定役数
	見えているのが一色かどうか
	フーロ数
場	供託棒の点数
	自分以外のリーチ者の数
	親がリーチしているか
	局番
	本場
	オーラスかどうか
	自分の順位
	トップとの点差
	一つ上位との点差
	一つ下位との点差
	ラスとの点差
相手の捨て牌	リーチ順目
	一発があるか
	現物
	スジ
牌の組み合わせ	1 牌の所持 boolean
	2 牌の組み合わせ
	3 牌の組み合わせ

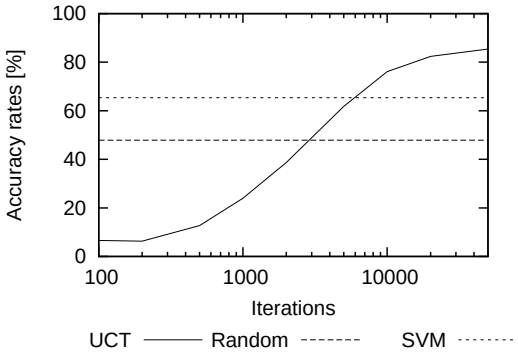

図 8 牌譜で鳴いた局面の一致率

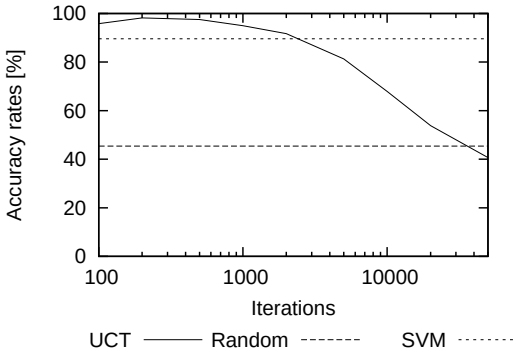

図 9 牌譜で鳴かなかった局面での一致率

5. おわりに

本研究では UCT 探索を用いて多人数不完全情報ゲームにおける手の探索を行う手法について示した．麻雀で行った実験の結果では，UCT プレイヤはグリーディプレイヤや SVM プレイヤに対して互角の性能を発揮することができた．知識の利用が難しい多人数不

表 6 グリーディ対 UCT (62 試合 248 局)

プレイヤ	Greedy A	UCT A	Greedy B	UCT B
和了率 [%]	11.69	18.55	14.92	16.13
平均収支	761	787	716	783

表 7 SVM 対 UCT (100 試合 400 局)

プレイヤ	SVM A	UCT A	SVM B	UCT B
和了率 [%]	19.25	17.75	16.75	19.25
平均収支	71.25	85.75	29	84

完全情報ゲームである麻雀では，モンテカルロシミュレーションをベースにしたUCT探索が有効である可能性があることがわかった．さらにUCTは不完全情報下でシミュレーションによって手を求めることができるだけでなく，鳴きによる不規則な手番の入れ替わりによって探索木が複雑になるゲームにも対応できることがわかった．

今後の課題としてはアルゴリズムの改良などが挙げられる．一つの方向性としてはプレイアウトの質を向上させることが考えられる．現在は不完全情報を仮定するときに相手の手牌をランダムに振り分けており，終盤でも相手の手があがりから程遠いという不自然な状態になってしまっている．そこで相手の手牌を確率的に推定するなどして，より現実的な局面を生成してプレイアウトの質を高めることが考えられる．また，プレイアウトの部分にパターンなどの知識に基づいた着手や評価関数を組み入れてシミュレーションの質を向上させるという方法もある[10]．その際には，今回用いたSVMによる手の分類を利用することも考えられる．探索効率を向上させる方法として，囲碁で用いられているUCTの改良手法を応用することも考えられる．例えば異なる順序の着手で到達した等しい状態の局面は同一のノードとして更新を行うRAVE (Rapid Action Value Estimation)と呼ばれる手法などを応用することが考えられる[17]．さらに探索時間の有限性を利用した枝刈りによって探索効率を改善するなど[18]，様々な手法の適用が考えられる．

参 考 文 献

1) BWBallard. *-Minimax search procedure for trees containing chance nodes. *Artificial Intelligence*, 1983.
2) T.Hauk. Search in trees with chance nodes. Master's thesis, University of Alberta, 2004.
3) L. Kocsis and C. Szepesvari. Bandit based monte-carlo planning. *European Conference on Machine Learning*, pp. 282–293, 2006.
4) N.R. Sturtevant. An Analysis of UCT in Multi-player Games. *Proceedings of the 6th international conference on Computers and Games*, pp. 37–49, 2008.
5) J. Schäfer, M. Buro, and K. Hartmann. The UCT Algorithm Applied to Games with Imperfect Information. *Diploma thesis. Otto-von-Guericke-Universität Magdeburg*, 2008.
6) T. Hauk, M. Buro, and J. Schäfer. Minimax performance in backgammon. Vol. 3846, pp. 51–66, 2004.
7) C.Luckhardt and K.Irani. An algorithmic solution of n-person games. *AAAI*, Vol. 1, pp. 158–162, 1986.
8) N. Sturtevant. Last-Branch and Speculative Pruning Algorithms for Maxˆ n. *IJCAI*, Vol. 669–678, , 2003.
9) P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, Vol. 47, No. 2-3, pp. 235–256, 2002.
10) Y. Wang and S. Gelly. Modifications of UCT and sequence-like simulations for Monte-Carlo Go. *IEEE Symposium on Computational Intelligence and Games, 2007. CIG 2007*, pp. 175–182, 2007.
11) 保木邦仁. 局面評価の学習を目指した探索結果の最適制御. *Proceedings of 11th Game Programming Workshop*, pp. 78–83, 2006.
12) 北川竜平, 三輪誠, 近山隆. 麻雀の牌譜からの打ち手評価関数の学習. *Proceedings of 12th Game Programming Workshop*, 2007.
13) とつげき東北. システマティック麻雀研究所. *http://www.interq.or.jp/snake/totugeki/*.
14) T. Joachims. Optimizing search engines using clickthrough data. *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), ACM*, 2002.
15) T. Joachims. SVM-LIGHT: an implementation of Support Vector Machines (SVMs) in C. *http://svmlight.joachims.org/*.
16) 三木理斗, 三輪誠, 近山隆. 木カーネルを用いた麻雀打ち手の順位学習. *Proceedings of 13th Game Programming Workshop*, pp. 60–66, 2008.
17) S. Gelly and D. Silver. Combining online and offline knowledge in UCT. *Proceedings of the 24th international conference on Machine learning*, pp. 273–280, 2007.
18) 北川竜平. 投入計算量の有限性に基づくUCT探索の枝刈り. Master's thesis, 東京大学, 2009.