期待最終順位に基づくコンピュータ麻雀プレイヤの構築

水上 直紀[1,a)]　鶴岡 慶雅[1,b)]

概要：長期的な戦略に基づく手の決定は繰り返しゲームにおいて重要である．本論文では麻雀の繰り返しゲームの性質に着目して最終順位を考慮したコンピュータ麻雀プレイヤの構築法について述べる．牌譜中に現れた点数状況から最終順位を予測するモデルの学習を行う．モンテカルロ法のシミュレーションでの報酬を予測モデルの結果を用いることで最終順位に基づく手をプログラムは選択する．オンライン麻雀サイト「天鳳」で作成されたプログラムの実力を評価した結果，レーティングとして，中級者を超える 1844 点が得られた．

Building Computer Mahjong Players Based on Expected Final Ranks

Naoki Mizukami[1,a)]　Yoshimasa Tsuruoka[1,b)]

Abstract: Deciding moves based on a long-term strategy is important in repeated games. This paper describes a method for building a mahjong program that considers expected final ranks by taking the repeated-game property of mahjong into account. We train prediction models using information about game states and final ranks in game records. Our program decides moves based on the final ranks obtained from Monte Carlo simulation using the prediction models. We have evaluated the playing strength of the resulting program on a large online mahjong site "Tenhou". The program has achieved a rating of 1844, which is considerably higher than that of the intermediate human player.

1. はじめに

ゲームの中には「繰り返しゲーム」と呼ばれるゲームが存在する．これは同一の部分ゲームを繰り返し行うことで，最終的な勝者が決まるようなゲームである．具体例としてゲームの分野ではトーナメントポーカー，バックギャモンなどが繰り返しゲームであり，スポーツの分野でも野球やカーリングが例として挙げられる．このように，繰り返しゲームはボードゲームやスポーツなどにも幅広く存在し，エキスパートプレイヤは，部分ゲームの勝率を最大化するのではなく，長期的な戦略を持ってこれらのゲームをプレイしている．そのような長期的な戦略をコンピュータで実現することは，人工知能の研究においても有意義だと考えられる．

繰り返しゲームにおける戦略決定に関する研究として，Miltersen らはトーナメントポーカーにおいてプレイヤの取り得る行動を削減し，チップの状態数をまとめることで近似的な最適戦略を求めた [1]．Papahristou らはバックギャモンの自己対戦の結果からゲーム開始時の各点数状況における全体の勝率表を作成した [2]．これによりマッチプレイにおいて，部分ゲームの期待値を最大化するプレイヤに対して勝率を向上させた．

麻雀は 1 局ごとに役に応じた点数を獲得し，全部で 4 または 8 局行う．全ての局が終了した時点で最終的に最も多くの点を持っているプレイヤが麻雀の勝者である．この性質から麻雀は繰り返しゲームであるといえる．繰り返しゲームとして麻雀を対象にした研究は我々の知る限り存在しない．

麻雀を対象にした研究として，水上らは相手のモデル化とシミュレーションによる麻雀プレイヤを構築した [3]．モデル化では、相手の手牌を完全に予測しようとするのではなく，ゲームに大きく影響する部分のみを考慮している．具体的には (a) 聴牌である確率，(b) 特定の牌が相手への

[1] 東京大学
The University of Tokyo
[a)] mizukami@logos.t.u-tokyo.ac.jp
[b)] tsuruoka@logos.t.u-tokyo.ac.jp

当たり牌になる確率，(c) 予想される相手の和了点数の 3 つを考え，これらの予測モデルを牌譜から学習している．学習された相手モデルを用いて現在の局面から 1 局終了までをシミュレーションすることで局単位での最適な手を決定している．その結果，局単位の収支を増加させることに成功し，それに伴い人間との対戦成績も向上した．しかしながらこのプレイヤが明らかに悪い手を選択する状況も存在する．具体例として 2 点挙げる．ひとつは，現在の順位と残りの局数に応じた押し引きができないことである．たとえば、現在プレイヤが 1 位であり，2 位以下を大きく引き離している場合であっても，放銃する危険を冒して和了を目指してしまう．もうひとつは，最終局において適切な点数で和了できないことである．すなわち．最終局において，和了しても最下位が確定するような得点の低い手を作ってしまう．この 2 つの問題点の原因は，最終順位を考慮せずに手を選択していることにある．

この問題を解決するひとつの方法はゲーム全体を通した最適戦略を計算することである．しかし，麻雀は情報集合の要素数が非常に多く，局ごとの最適戦略を計算することすら計算量的に困難である．そこで本研究では，局終了時の各プレイヤの得点状況から最終的な順位を予測し，1 局において最終順位を考慮した手を選択するプレイヤを構築する手法を提案する．

本論文は以下の構成になっている．初めに 2 章で麻雀のルールと用語，3 章で関連研究を述べる．提案手法として，4 章で現在の点数状況からの期待最終順位予測，5 章で予測モデルとモンテカルロ法を用いた手の決定，6 章で提案手法の対戦結果について述べる．最後に 7 章で本研究の結論について述べる．

2. 麻雀のルールと用語

この章では麻雀の得点に関するルールについて解説する．麻雀は自分の牌を組み合わせて役（特定の構成）を作り，**和了（ホーラ）**し，役に応じた点数を得るゲームである．これを 1 局として定められた回数の局数を行う（通常は 4 または 8 回）．最初の持ち点は 25,000 から開始し，最終局（**オーラス**）を終了した時の得点の多さに応じて順位が決まる．同点の場合は，ゲーム開始時の親に近いほうが優先される．途中で誰かが持ち点が 0 未満になっても試合は終了する．オーラスが終了しても誰も 30,000 点を超えていない場合は**サドンデス**と呼ばれ，誰かが 30,000 点を超えるまでゲームが続行される．

和了に関する用語としては，ツモして和了することをツモ和了，あるいは略してツモと呼び，相手の捨てた牌で和了することを**ロン**と呼ぶ．またロンされることを**放銃**と呼ぶ．ツモの場合，点数を残りの 3 人が和了点数を分割して支払う．またロンの場合，放銃したプレイヤが全ての和了点数を支払う．

プレイヤは 1 局ごとに親または子のいずれかの役割が当てられ，和了した時の点数が異なる．基本的には，同じ役であっても，親の和了した時の点数は，子の点数のおよそ 1.5 倍である．ツモの際の点数は，親がツモの場合，それぞれの子が支払う点数は和了点数の 3 分の 1 である．子がツモの場合，親が支払う点数は子の 2 倍である．

以下に一般的な麻雀用語を説明する．

リーチ　鳴きを一度も行わず，あと 1 枚で和了できる手牌になった時に宣言出来る行為である．リーチ宣言の後は手牌を変更することはできない．すなわち，リーチ後にツモで入手した牌は，和了できる場合を除きすべてそのまま捨てなくてはならない．

ダマ　リーチを出来る状態であっても，リーチしないこと

聴牌（テンパイ）　和了に必要な牌の枚数が 1 枚の状態

降り　自分の和了を諦め，相手の和了牌を捨てないようにする行為

回し打ち　和了を目指した最善手を選ぶのではなく，放銃を避けつつ和了や聴牌を狙う行為

流局　誰も和了することなく，ツモれる牌がなくなること

現物　相手がすでに捨てた牌．ルール上，現物をロンされることはない

リーチを行うには 1,000 点を支払う必要がある．この 1,000 点を**リーチ棒**と呼ぶ．このリーチ棒は和了したプレイヤの和了点に追加される．流局した場合はこのリーチ棒は次の局に持ち越される．

点数を決める追加要素に**本場**がある．最初は 0 本場から始まり，親が和了するまたは流局した際に，本場が増える．誰かが和了した時に，本場の数 ×300 点が和了点数に追加される．

流局時には聴牌していたプレイヤは点数を得て，聴牌していないプレイヤは点数を払う．その点数は，聴牌していたプレイヤの数によって決まる．

3. 関連研究

コンピュータ麻雀プレイヤに関して，水上ら [3] は相手プレイヤを麻雀における重要な 3 つの要素（聴牌率，待ち牌率，和了点）に分離してモデル化し，それぞれの予測モデルを教師付き機械学習によって構築した．得られた予測モデルを用いて，各合法手によって得られるであろう得点をシミュレーションにより推定し，自プレイヤの手を決定する手法を提案している．シミュレーション中に自分の手番から次の自分の手番までの 1 手探索を入れることで，既存の最強プログラム「まったり麻雀」 [4] に対して統計的に有意な差ではないものの勝ち越すことに成功した．

とつげき東北は，現在の得点状況，和了率，放銃率等のパラメータからシミュ―レーションを用いて最終的な順位分布を予測する MJSIM0 を開発した [5]．このシミュレーションでは，プレイヤは現在の得点に依存しない確率に基

づき，和了や放銃を行う．そのため予想される成績は，実際のものとはずれたものとなる．またシミュレーションに時間がかかるため既存の麻雀プログラムに利用するのは困難である．実際，最終順位を考慮した麻雀プログラムは筆者の知る限り存在しない．

ポーカーを対象にした研究として Miltersen らはトーナメントポーカーにおいて近似の最適な戦略を求めた [1]．トーナメントポーカーとは相手のチップの数が 0 になるまでゲームを行うポーカーの一種である．そのため 1 ゲーム毎の期待値を最大化する「キャッシュゲーム」と異なり，トーナメントポーカーでは，現在のチップの数に応じた戦略が必要になる．この研究では，プレイヤの行動を削減するため Jam/Fold と呼ばれるプレイヤの行動をオールインするか降りるかどちらかしか行わないプレイヤを考え，チップの数を 50 刻みで抽象化することでチップの状態の削減も行った．この抽象化したゲームを解くことで，ゲーム全体の近似最適戦略を求めることに成功した．

Papahristou らはバックギャモンのマッチプレイにおける全体の勝率表を作成した [2]．バックギャモンは勝利した条件によって 1 点または 2 点が与えられる．マッチプレイとはあらかじめ全体の勝利までの点数を設定し，1 ゲームを繰り返すことで先にその点数を満たしたプレイヤが最終的な勝者となる遊び方である．この研究では 1 ゲームを繰り返す強化学習によりバックギャモンプレイヤを構築した．そのプレイヤの自己対戦結果からゲームの初期局面の 1, 2 点勝ちの割合から現在の点数状況からの全体の勝率を求めた．これによりマッチプレイにおいて，1 ゲームの期待値を最大化するプレイヤに対してマッチプレイの勝率を向上させた．

麻雀ではポーカーと異なり手を効率的に抽象化する方法は提案されておらず，抽象化したゲームとして解くことは現在のところ不可能である．また，点数状況がバックギャモンに比べ膨大であるため，あらかじめ勝率表を持つことも不可能である．そこで本研究では現在の得点状況から最終順位の予測を行うモデルを牌譜から機械学習する．そしてその予測モデルを用いて最終順位を考慮したプレイヤを構築する．

4. 順位予測

この章では現在の点数状況から最終順位を予測するモデルを学習する．この予測モデルを用いて手を決定するため，高い予測精度が求められる．

4.1 順位予測モデルの学習

麻雀の勝負において重要なのは最終得点ではなく最終順位である．最終順位を予測する方法としては，回帰で直接推定する方法と，各順位をとる確率を一旦求め，その確率を用いて期待順位を求める方法が考えらえる．今後の研究の展開として各順位の報酬が線形でない場合も想定しているため，本研究では後者の方法をとる．具体的には，最終順位を予測するモデルとして多クラスロジスティック回帰モデルを使用することで 1 位から 4 位を予測する多クラス問題として捉える．出力としてソフトマックス関数を使用することで，1 位から 4 位のそれぞれの確率を出力し，現在の得点状況から期待される最終順位を推定する．ソフトマックス関数は次の式（1）で表される．

$$P(rank = r) = \frac{\exp\left(\mathbf{w_r}^{\mathrm{T}}\mathbf{x}\right)}{\sum_{i=1}^{4}\exp\left(\mathbf{w_i}^{\mathrm{T}}\mathbf{x}\right)}, \qquad (1)$$

ここで $\mathbf{x}$ は現在の点数状況を表す特徴ベクトルである．$\mathbf{w_r}$ は順位 r の特徴量に対しての重みベクトルである．この式を利用して期待最終順位（Expected Final Rank，EFR）は次の式（2）で表される．

$$EFR = \sum_{r=1}^{4} r \times P(rank = r) \qquad (2)$$

学習に使用する特徴量を **表 1** に示す．表中の自分の順位，相手の順位，残り局数，東風か半荘と点差に関して以下に述べる．前半部分はこの特徴量の次元を表し，点差とは特定の相手との点数を差を全てのプレイヤの持ち点の合計である 100,000 で割って正規化した値である．これにより，点差を 0 以上 1 以下の実数で表現でき，その値を特徴量の値とする．

このように相手との点差をそのまま特徴量として使用することも当然可能であるが，麻雀では和了点数が固定されているため，点差と期待順位に単純な線形相関はない．そこで意味のある点差を考慮する．意味のある点差とは逆転に必要な和了の点数を和了の種類によって点差を抽象化することである．ここでは和了の種類を出和了，ツモ和了，直撃，他者のツモ和了，流局の 5 種類とする．出和了とは特定の相手以外からのロン和了である．直撃とは特定の相手からのロン和了である．麻雀には子のロン和了でも 30 種類ほど和了点が存在するが，ここでは頻出の 10 種類ほどを用いている．例えば，相手と 9,700 点差と 9,900 点差と 10,100 点差は満貫を出和了しても逆転しない点差であるが，満貫をツモると 9,700 点差と 9,900 点差だけ逆転する．意味のある点差を考えることで 9,700 点差と 9,900 点差を同等に扱うことが出来る．**表 2** に抽象化した点差を示す．

本場が 1 つ増えると出和了の場合は，300 点分追加で差が縮み，ツモの場合は，400 点分，追加で差が縮む．このように単純に一つの基準で意味のある点差を考えるとリーチ棒や本場に対応できない．しかしながら和了の種類を分けることでリーチ棒や本場の影響も考慮した点差を考えることが出来る．これにより，リーチ棒や本場といった要素を考慮した特徴量が生成可能になり，学習局面の少ないであろうリーチ棒や本場が多い局面であっても，単純にこれ

らを特徴量に加えるよりも精度の高い予測が可能になると考えられる．特徴ベクトルの次元は 8,161 になった．

予測モデルの学習は，以下の式（3）で表現される目的関数を最小化することにより行った．

$$L(\mathbf{w}) = -\sum_{i=1}^{N}\sum_{r=1}^{4} c_{i,r} P(\mathbf{X_i}) + \frac{\lambda|\mathbf{w}|^2}{N}, \qquad (3)$$

ここで N はトレーニングデータのサンプル数, $\mathbf{X_i}$ は i 番目のトレーニングデータ, c_i は 4 次元のベクトルで，各順位に対応する 2 値（1 または 0）のデータである．λ はトレーニングデータに過学習することを防ぐ正則化項である．ここでは正則化項 λ を 0.01 とする．

重みベクトルの学習に FOBOS [6] を使用する．また重みベクトルの更新は Adagrad [7] を使用する．更新式は次で表現される．

$$w_{t+1,i} = w_{t,i} - \frac{\eta g_{t,i}}{\sqrt{1+\sum_{k=1}^{t} g_{k,i}^2}}, \qquad (4)$$

$w_{t,i}$ は重みベクトルの i 番目の要素である． η は学習率，t は更新回数，g は目的関数の確率的勾配である．学習率 η を 0.01 とする．

4.2 学習に用いる牌譜

上記の聴牌予測には多くの教師データが必要になる．教師データとしてはインターネット麻雀サイト天鳳 [8] の鳳凰卓の牌譜を用いた [*1]．鳳凰卓でプレイできるのは全プレイヤの中でも上位 0.1% 程度であり牌譜の質は高いと考えられる．以下，牌譜と示した場合はこの鳳凰卓の牌譜のことを指す．

牌譜中の局開始時の点数状況から特徴量を抽出し最終順位を組み合わせて教師データとした．局面数はおよそ 2.5×10^7 である．

4.3 聴牌予測の精度

学習が上手くできているかを調べるため，学習によって得られた順位予測モデルのの精度を調べた．評価は平均絶対誤差を用いる．テストデータは 95,856 局面を用いた．

テストデータに対しての結果を **表 3** に示す．ベースラインは現在の順位を予想順位として返す．予測モデルはベースラインに比べ最終順位の予測性能が高い．結果の 3 行目以降は予測モデルから特定の特徴量を用いない場合の結果である．最も重要と思われる純粋な点差を除いたしても結果は大きく悪くならないことがわかる．どの特徴量も除くと結果は悪くなるため，すべての特徴量が有効であったといえる．

5. モンテカルロ法を用いた手の決定

この章では，プログラムが順位予測モデルを使ってどのように手を決定するかを述べる．この研究で用いるシミュレーションは，水上らが用いたシミュレーション [3] とほぼ同様であるが，相違点について以下について述べる．シミュレーションの報酬が先行研究ではゲーム上の得点であったが，今回はその得点から得られる次の局開始時の点数状況からの期待順位になるのが主な変更点である．シミュレーションによって決まる手のことを**モンテカルロの手**と呼ぶ

モンテカルロの手を用いることは序盤であっても可能であるが，和了まで時間がかかるため，精度の高い手を選択することは難しい．そのため序盤は一人麻雀の手を用いる．一人麻雀の手からモンテカルロの手に切り替えには以下の条件のいずれかを満たしたときである．

- 誰かがリーチをかけた時
- ツモが可能な牌の数が 16 枚以下
- $\frac{\sum_{p\in opponents} EL(p,Tile)}{fold\,value} \leqq 0.2$

3 つ目の条件は先行研究で使われた条件と同じである．$EL(p,Tile)$ は p に対して牌 $Tile$ を切った時の期待失点であり，$fold\,value$ は最初から降りた時の期待値である．

モンテカルロの手は次の式によって求められる．

$$\begin{aligned} &Score(Tile) \\ &= Sim(Tile) \times \prod_p (1 - LP(p,Tile)) + \sum_p LP(p,Tile) \\ &\times Max(EFR(x_0 - HS(p,tile), x_p + HS(p,tile)), EFR()), \end{aligned} \qquad (5)$$

ここで $Sim(Tile)$ はシミュレーションの結果であり，ここでは EFR で求められる期待最終順位である．$LP(p,Tile)$ は相手 p に対して牌 $Tile$ を切った時に放銃する確率，$HS(p,tile)$ は相手の予測点数である．x_0 を自分の点数，x_p を相手 p の点数である．$EFR()$ は現在の期待最終順位であり，式の後半は振り込んだ後の点数状況が現在の期待最終順位よりも良くなることないという制約である．

この式はある牌を切った時の期待順位を表している．したがって $Score(Tile)$ は 1 以上 4 以下の値である．この計算をプレイヤの持っているすべての牌に対して計算し，最も小さい値すなわち期待順位が最も良い牌が切られる．

5.1 シミュレーションの中の挙動

$Sim(Tile)$ の中身について自分の手番での動作を**図 1** に示す．先行研究と異なる点はシミュレーション中のリーフ（和了，降り，放銃）での報酬がゲーム上の値でなく，期待最終順位に変更される点である．$ODEV(One\text{-}Depth\,Expected\,Value)$ とは自分の手番から次の手番までの一手探索した結果である．この結果に基づいて降るかどうか判定する．当然この $ODEV$ の判定も期待最終順位に基づいて降りかどうかを決定する．

5.2 降り成功率

先行研究 [3] ではシミュレーション中に降りる場合，降

[*1] 2009 年 2 月 20 日から 2013 年 12 月 31 日までに行われた対局

表 1 順位予測の特徴量

特徴量	次元数
自分の順位，相手の順位，残り局数，東風か半荘と点差	$4 \times 4 \times 8 \times 2 = 256$
自分の順位，残り局数，残りの親の回数，サドンデスか，東風か半荘	$4 \times 8 \times 3 \times 2 \times 2 = 384$
自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），出和了の点差	$4 \times 4 \times 5 \times 11 = 880$
自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），ツモ和了の点差	$4 \times 4 \times 5 \times 11 = 880$
自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），直撃の点差	$4 \times 4 \times 5 \times 11 = 880$
自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），他者によるツモ和了の点差	$4 \times 4 \times 5 \times 12 = 960$
自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），流局の点差	$4 \times 4 \times 5 \times 5 = 400$
自分の順位，相手 2 人の順位，残り局数（0，1，2，3，4 以上），相手 2 人の出和了の点差	$4 \times 4 \times 4 \times 5 \times 11 = 3520$
バイアス項	1

t

表 2 点差表

出和了	子	0, 1000, 1300, 2000, 2600, 3900, 5200, 7700, 8000, 12000, それ以上
	親	0, 1500, 2000, 2900, 3900, 5800, 7700, 11600, 12000, 18000, それ以上
ツモ和了	子対子	0, 1400, 1900, 2500, 3400, 5000, 6500, 9900, 10000, 15000, それ以上
	相手が親	0, 1600, 2200, 3000, 4000, 6000, 7800, 11900, 12000, 18000, それ以上
	自分が親	0, 2000, 2800, 4000, 5200, 8000, 10800, 15600, 16000, 24000, それ以上
直撃	子	0, 2000, 2600, 4000, 5200, 7800, 10400, 15400, 16000, 24000, それ以上
	親	0, 3000, 4000, 5800, 7800, 11600, 15400, 23200, 24000, 36000, それ以上
他者のツモ和了	子対子	0
	子対親	200, 300, 500, 600, 1000, 1300, 1900, 2000, 3000, それ以上
流局		0, 3000, 4000, それ以上

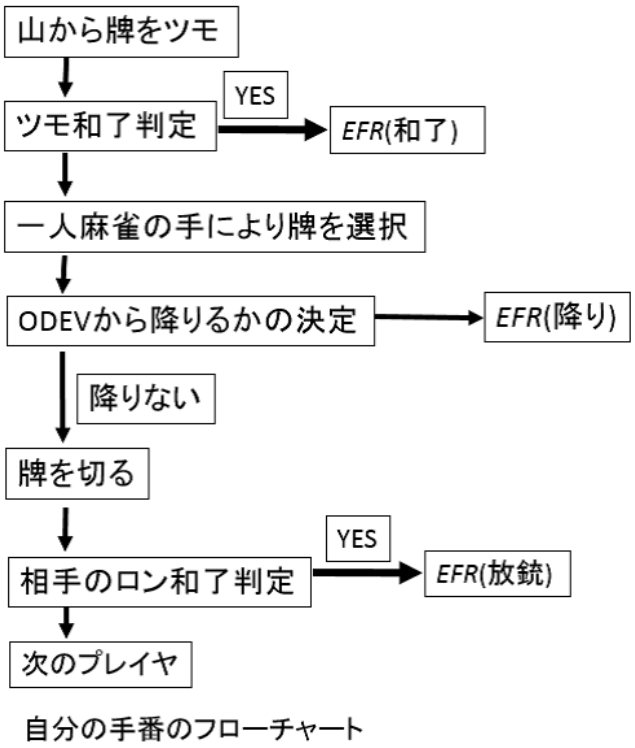

図 1 フローチャート

りはどんな手牌であっても必ず成功するとしていた．そのため鳴きを多用し流局時の聴牌料を狙う挙動がみられた．実際には鳴いた後に放銃することが多く，鳴いた手は悪手となることが多い．悪手となる鳴きを防ぐため，今回はシミュレーション中に降りる場合には，降りの成功率を考慮する．具体的には現在の相手の待ち牌率を利用して近似の降り成功率を計算する．

降り成功率を求めるアルゴリズムを Alg 1 に示す．基本的に，最初にプログラムの手牌に流局までの手番の数だけ牌をツモると考える．プログラムから見えていない牌を数え，各牌のツモる確率を計算する．その確率に手番数を掛け合わせた牌を仮想的にツモ牌としてプログラムの手牌に加える．したがって手牌の数は整数でなく実数になる．次に相手の待ち牌率の予測結果に基づき安全な順から牌を切る．相手は確率に基づきツモやほかのプレイヤに放銃する．最終的にプログラムが放銃しない確率を降り成功率とする．

この降り成功率は，図 1 のリーフの降りの値を変更することで利用する．先行研究 [3] ではこの値はプログラムがツモの後は何も切らないという仮定を置いた時，すなわち相手のツモと流局時の失点の平均値を使用していた．今回は相手のツモ，流局，相手の放銃，自分の放銃それぞれの局面の期待最終順位をもとめ，確率を掛け合わせた値とする．

6. 結果

提案手法によって得られた麻雀プログラムと牌譜との一致率を調べた．またインターネット上の麻雀対戦サイトである天鳳 [8] で対戦させた．比較となるプレイヤは水上 [3] と，このプログラムに降り成功率を導入した「降り成功率導入」と，さらに期待最終順位に基づく「提案手法」の 3 つである．実験で使用した CPU は Core i7 3632 QM でクロック数は 2.2GHz，8 コアを使用してシミュレーション部

表 3　順位予測に関する評価

プレイヤ	平均絶対誤差
ベースライン	0.809
予測モデル	0.763
-自分の順位，相手の順位，残り局数，東風か半荘と点差	0.764
-自分の順位，残り局数，残りの親の回数，サドンデスか，東風か半荘	0.764
-自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），和了の点差	0.765
-自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），ツモ和了の点差	0.764
-自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），直撃の点差	0.764
-自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），他者によるツモ和了の点差	0.764
-自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），流局の点差	0.764
-自分の順位，相手 2 人の順位，残り局数（0，1，2，3，4 以上），相手 2 人の出和了の点差	0.764

表 4　一致率

	Rank1	Rank2	Rank3
提案手法	0.619	0.827	0.908
降り成功率導入	0.621	0.829	0.909
水上 [3]	0.620	0.834	0.915

分を並列に計算した．提案手法のプログラムは一手 1.5 秒で行った．

6.1　牌譜との一致率

実力を測定するため牌譜との一致率を調べた．テストとして同じ牌譜中のツモ局面を 10,000 局面を使用した．結果を**表 4** に示す．$Rank\,n$ は第 n 候補に牌譜での打牌が入っている割合とする．結果として牌譜一致率の向上は見られなかった．

6.2　天鳳での対戦における評価設定

実力を測定するため，インターネット上の麻雀対戦サイトである天鳳 [8] で対戦を行った．ルールは東風戦，赤あり，持ち時間は一手 3 秒に考慮時間が 5 秒である．鳴きについても同様の持ち時間があり，鳴ける局面では次のプレイヤは勝手に牌を引くことができない．天鳳 [8] では成績に応じて対戦できる卓が異なる．卓は 4 種類あり，上から鳳凰卓，特上卓，上卓，一般卓がある．卓の種類は上の卓を選択可能な成績であっても上級卓のみに限定した．プログラムをサイト上で対戦させるための入出力インターフェースは自作した．評価としては平均順位（レーティング）を用いる．レーティング（R）とは平均順位と負の相関を持つ基準である．具体的には（6）で計算される．

$$R' = R + (50 - Rank \times 20 + \frac{AveR - R}{40}) \times 0.2 \quad (6)$$

$Rank$ は前回のゲームでの順位である．$AveR$ は卓の平均の R である．初期 R は 1500 であり，およそ平均順位が 0.1 下がるごとにレーティングは 100 点ほど上昇する．

レーティングを用いた評価方法としては安定レーティングと保証安定レーティング [9] の二つが提案されている．安定レーティングは今までの順位分布をとり続けた場合のレーティングである．保証安定レーティングは連続する試合の結果を恣意的に切り出した時の安定レーティングを用いて，安定レーティングの現実的な範囲を保証する安定レーティングのことである．これは異なる試合数の恣意的に良かった結果を取り出しても比較可能な評価方法である．

6.3　天鳳での対戦における結果

提案手法と降り成功率導入と先行研究である水上らの手法 [3] の 3 つの比較を行った．対戦結果を**表 5** に示す．Bootstrap Resampling [10] を用いて平均順位の信頼区間（p-value ≤ 0.05）を求めた．信頼区間はパーセンタイル法によって決定した．ブートストラップ回数は 10,000 回とした．

水上 [3] と降り成功率導入の平均順位の差はウェルチの t 検定（p-value $= 9.3 \times 10^{-5}$）により統計的に有意な差が見られた．降り成功率導入と提案手法の平均順位の差はウェルチの t 検定（p-value $= 0.26$）により統計的に有意な差が見られなかった．

降り成功率導入と提案手法の保証安定レーティングについては降り成功率導入のほうが高かった．この値は実力が安定しない人間同士が成績を比較するときに有用であり性質上，試合数が多いほうが高い値を出しやすい．実際，降り成功率導入の保証安定レーティングを求めるときに切り出した試合数は 4,300 ほど使っており，提案手法の 1,500 試合程度の良かった部分を切り出してもあまり成績は伸びない．

和了・放銃・副露時放銃率と局収支は**表 6** に示す．これらは 1 局のプレイヤの強さを測定するためによく用いられている．先行研究に比べ降り成功率を導入することで，無駄な鳴きを減らし副露時放銃率を下げることに成功した．提案手法と降り成功率導入では和了率，放銃率とも少し改善した．これはこのプログラム自体が上級卓平均に比べ強く終盤を点数状況的で優位に立つ局面が多いため，オーラス等で無理して攻める機会が減り，降りていればよい局面が増えたため放銃率の改善したと思われる．またオーラス 4 位などは攻めて和了するケースが増え，和了率が改善し

表 5　順位分布

	1 位率	2 位率	3 位率	4 位率	平均順位	試合数	安定レーティング	保証安定レーティング
提案手法	0.294	0.264	0.242	0.200	2.34 ±0.05	1508	1844	1752
降り成功率導入	0.277	0.273	0.251	0.198	2.37 ±0.03	4755	1821	1781
水上 [3]	0.241	0.281	0.248	0.230	2.46 ±0.04	2634	1718	1690

表 6　和了・放銃率と局収支

	和了率	放銃率	副露時放銃率	局収支
提案手法	0.253	0.113	0.114	582
降り成功率導入	0.247	0.116	0.114	619
水上 [3]	0.245	0.131	0.127	412

表 7　終盤に関するデータ

	トップ死守率	トップまくり率	ラス回避率
提案手法	0.807	0.104	0.237
降り成功率導入	0.781	0.106	0.227
水上 [3]	0.750	0.086	0.241

たと思われる．

局収支とは全局で得られた得点を全局数で割った値である．基本的にはこの値が高いほど強いプレイヤとされる．降り成功率導入はこの値を最大化する目的で手を選択していた．提案手法は局収支が降り成功率導入より低い．しかしながら平均順位は降り成功率導入よりも向上しているため，この指標からこのプログラムが局収支で無く最終順位を考慮したプレイヤであることがわかる．

点数状況判断の影響が最も出やすい終盤や順位に関してのデータを **表 7** に示す．表中のトップ死守率はオーラスでトップであるときに，最終結果がトップであった割合である．トップまくり率はオーラスにトップでない時に，最終結果がトップになった割合，ラス回避率はオーラスで 4 位の時に，最終結果が 4 位以外である割合である．提案手法は降り成功率導入に比べ，トップ死守率，ラス回避率の成績が向上した．この指標から，提案手法が点数状況の判断が向上したといえる．

6.4　考察

図 2 に得点を考慮した手の例を示す．局面はオーラス，点数状況はプログラムが 35,700 点で 1 位であり，2 位の親が 35,400 点，あとの二人とは点数的には離れている．今，4 位からリーチがかかり，同順にプログラムが聴牌したところである．

この局面で大事なことはリーチを打って順位を悪化させないことと振り込む危険のある牌を切らないことである．人間であれば，このリーチに振り込みさえしなければトップであり，リーチを打つことで順位が悪くなるため，リーチ者に対して現物かつ聴牌も取れる 6 ピンを切ってダマにする．これにより次の順目でツモの可能性が残り，危険牌を引いた時には降りに回ることもできる．点数状況を考慮に入れない先行研究のプログラムであれば，6 ピンを切ってリーチという選択であったが，これは点数状況的に悪手

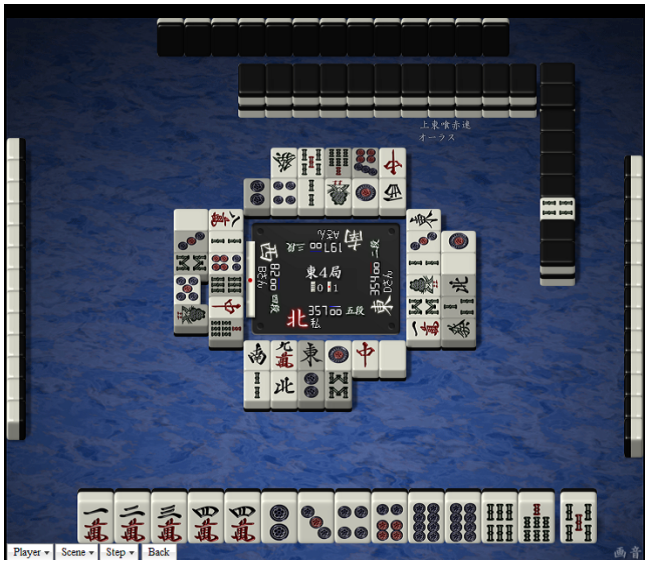

図 2　得点を考慮した例

である．提案手法は 3 ピンを切って降りる選択をした．これは 6 ピンを切ってリーチするより良い手である．最終的にはこの局は流局して，1 位を守った．

現在のプログラムが最善手である 6 ピン切ってダマが出来ない理由は，このプログラムがリーチが打てる時にはすべて打つというルールを用いているためである．得点状況を考えてリーチを打つ判断は強くなるうえで必須である．

提案手法の挙動を見る限り，現在の順位が 1 位の時は引き，4 位の時は押す傾向が確認された．相手の得点予測を平均値で推定しているため，相手との点差が平均値の 2 倍離れている場合は，放銃しても逆転されることはないとプログラムは誤って考えてしまう．実際には裏ドラが乗れば逆転することが頻出する．そこで振り込んだ場合の期待順位は現在の期待順位を上回ることはないとしている．そのため局を進めるために安い手に対してあえて放銃するという判断ができない．これを解決するためには相手の得点予測を平均ではなく分布で予測する必要がある．これによって逆転の起きる確率を考慮することが可能になり，意味のある得点差を考慮した特徴量もより効果的に働くと期待される．

7.　おわりに

本研究では現在の点数状況から最終順位を予測するモデルを構築し，シミュレーションの報酬として予測順位を使用することで，局収支ではなく最終的な順位を考慮したコンピュータ麻雀プレイヤの手を決定するアルゴリズムを提

Algorithm 1 降り成功率

function 降り成功率
  $turn$ ▷ 残り順目
  $tile$ ▷ 自分から見える牌
  $hand$ ▷ 自分の手牌
  $winning\_tile$ ▷ 待ち牌率
  $sum = 0$ ▷ 牌の合計
  $win = 1$ ▷ 特定牌の当たらない確率
  $win\_tsumo = 0$ ▷ 相手がツモ和了する確率
  $discard\_turn = 1$ ▷ 各順で切る牌が当たらない確率
  foreach $i \in all\,kinds\,of\,tiles$ **do**
    $sum \mathrel{+}= tile_i$
  end for
  foreach $i \in all\,kinds\,of\,tiles$ **do**
    $hand_i \mathrel{+}= (4 - tile_i)/sum \times turn$ ▷ 仮想的に実数枚の牌を引く
    foreach $j \in opponents$ **do**
      if $Player_j.waiting = TRUE$ **then**
        $win_i \times= 1 - winning\_tile_{ij}$
      end if
    end for
  end for
  foreach $i \in all\,kinds\,of\,tiles$ **do**
    $win\_opponent \mathrel{+}= (4 - tile_i)/sum \times (1 - win_i)$
  end for
  $tmp\_turn = 0$
  foreach $i \in all\,kinds\,of\,tiles$ **do** ▷ win_i の高い順にソート
    while $tmp\_turn < turn$ **do**
      if $tmp\_turn + hand_i < tmp\_turn + 1$ **then**
        $discard\_turn_{int\,tmp\_turn}$
        $\times= pow(win_i, hand_i)$ ▷ 実数枚の牌を切って当たらない確率
        $tmp\_turn \mathrel{+}= hand_i$
      else
        $discard\_turn_{int\,tmp\_turn}$
        $\times= pow(win_i, int(tmp\_turn{+}1) - tmp\_turn)$
        $hand_i \mathrel{-}= int(tmp\_turn + 1) - tmp\_turn$
        $tmp\_turn = int(tmp\_turn + 1)$
      end if
    end while
    $probability = 1$ ▷ 降り成功率
  end for
  for $i = 0$ to $turn - 1$ **do**
    $probability \times= discard\_turn_i$
    for $j = 1$ to $sum(Player.waiting = TURE)$ **do** ▷ 自分を除く聴牌しているプレイヤの数
      $probability \times= (1 - win\_tsumo) \times power(1.0 - win\_tsumo, sum(Player.waiting = TURE) - 1)$
    end for
  end for
  return $probability$
end function

案した．予測順位自体の精度も高く，既存の研究よりも高速に行うことが可能になった．実際の手を決定する局面においても 1 位なら引き，4 位なら押し気味の手を選択することが可能になり，人間のような長期的な視点による手の選択が可能になった．結果として天鳳 [8] において 1,508 試合を戦わせた結果，安定レーティングが 1844 点となり，上級卓の平均を超え，その上の特上卓でプレイ可能な実力にまでなった．

今後の課題のひとつは点数状況に応じた 1 局の序盤の手作るモデルの構築である．今のプログラムは序盤は一人麻雀の手を選択するだけであり，どのような点数状況であっても同じ手を指す．そのためオーラスで 4 位という状況にあっても，和了したとしても順位の変わらない手を選択してしまう．当然，上級者は今の点数状況に応じて，何点の手を作るべきかを考慮しており，その判断は最終的な順位に大きく影響するため実力向上に必須である．またオーラスの 4 位を確定する和了はほかのプレイヤから見ても興ざめであり，プログラムの打ち手の完成度という点でも重要である．

参考文献

[1] Miltersen, P. B. and Sørensen, T. B.: A near-optimal strategy for a heads-up no-limit Texas Hold'em poker tournament, *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pp. 191–198 (2007).

[2] Papahristou, N. and Refanidis, I.: Opening Statistics and Match Play for Backgammon Games, *Artificial Intelligence: Methods and Applications*, pp. 569–582 (2014).

[3] Mizukami, N. and Tsuruoka, Y.: Building a Computer Mahjong Player Based on Monte Carlo Simulation and Opponent Models, *Computational Intelligence and Games* (2015).

[4] kmo2: まったり麻雀，`http://homepage2.nifty.com/kmo2/` (2014).

[5] とつげき東北：MJSIM0, `http://totutohoku.b23.coreserver.jp/hp/MJSIM0.htm` (2003).

[6] Duchi, J. and Singer, Y.: Efficient online and batch learning using forward backward splitting, *The Journal of Machine Learning Research*, Vol. 10, pp. 2899–2934 (2009).

[7] Duchi, J., Hazan, E. and Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization, *The Journal of Machine Learning Research*, Vol. 12, pp. 2121–2159 (2011).

[8] 角田真吾：天鳳，`http://tenhou.net/` (2014).

[9] とつげき東北：保障安定レーティング，`http://totutohoku.b23.coreserver.jp/hp/SLtotu14.htm` (2001).

[10] Noreen, E. W.: Computer-intensive methods for testing hypotheses: an introduction, *Computer* (1989).