バックギャモンのエンドゲームデータベース拡張の試み

三納 侑樹[1,a)]　保木 邦仁[1,b)]

概要：本研究では，2 人零和不確定ゲームであるバックギャモンの終盤におけるデータベース（エンドゲームデータベース，EGDB）の拡張を目指す．そして，既存の GNU Backgammon で構築される両側ベアオフデータベース（BODB）では扱えない配置に対しても適切な勝率を求める．そのために，後ろ向き帰納法とグリーディ方策を用いたモンテカルロ法を採用する．後ろ向き帰納法によって，ベアオフされた駒が (黒, 白) = (14, 14)，(13, 14)，(14, 13) の配置の一部に対して，正確な勝率を求めることができた．また，グリーディ方策を用いたモンテカルロ法を，ベアオフされた駒が (黒, 白) = (14, 14) の配置集合に対して実行すると，全ての配置の推定勝率を高い精度で求められることが明らかになった．

キーワード：ボードゲーム，バックギャモン，エンドゲームデータベース，後ろ向き帰納法，グリーディ方策，モンテカルロ法

A Study of Expansion of the Endgame Database of Backgammon

Yuki Sanno[1,a)]　Kunihito Hoki[1,b)]

Abstract: This study aims to extend the endgame database (EGDB) of backgammon, a two-player zero-sum game with uncertainty. Additionally, this study seeks to compute accurate winning probabilities for positions that cannot be handled by the existing Bearoff Database (BODB) constructed by GNU Backgammon. To achieve this, we employ backward induction and the Monte-Carlo method with greedy policies. Through backward induction, we were able to compute accurate winning probabilities for certain positions where the number of bearing off checkers are (black, white) = (14, 14), (13, 14), and (14, 13). Furthermore, by applying the Monte-Carlo method with greedy policies to the set of positions where the bearing off checkers are (black, white) = (14, 14), we found that estimated winning probabilities for all positions could be obtained with high accuracy.

Keywords: board game, backgammon, endgame database, backward induction, greedy policy, Monte-Carlo method

1. 研究の背景

2 人零和ゲームの人工知能は，人間よりも強くなってきた．囲碁では，2016 年当時，世界最強の棋士と称されていた韓国のイ・セドル選手に，人工知能 AlphaGo が勝利している[*1]．また，スタークラフト 2 というリアルタイム戦略ゲームでは，2019 年に当時最強プレイヤーであった Mana 選手に勝利したことが報告されている[*2]．

2 人零和ゲームではまた，近年，互いに最善を尽くした場合の勝敗（ゲーム値）を求めるという試みが注目を集めている．チェッカーは，配置数が約 5×10^{20} 通りあり，初期配置の勝敗を求めることが困難とされていた．しかし，Schaeffer によって，数十台のコンピュータで並列に計算をすることで，引分となることが明らかにされた [5]．オセロでは，2000 年代までは 4×4 や 6×6 に縮小されたオセロについて後手勝ちであることが報告されていたが [11]，最近，Takizawa により，初期配置の勝敗が引分であることが報告され話題になった [7]．不完全情報ゲームでも，ポー

[1] 電気通信大学
The University of Electro-Communications
[a)] y.sannou21@gmail.com
[b)] k.hoki@uec.ac.jp

[*1] 圧勝「囲碁 AI」が露呈した人工知能の弱点，日本経済新聞電子版，https://www.nikkei.com/news/print-article/?R_FLG=0&bf=0&ng=DGXMZO98496540W6A310C1000000&uah=DF080320166916 （最終アクセス 2024）

[*2] AlphaStar：多様性のある学習環境で高度なスキルを獲得，日経 XTECH，https://xtech.nikkei.com/atcl/nxt/mag/rob/18/00007/00012/ （最終アクセス 2024）

カーの一種であるテキサスホールデムのナッシュ均衡の高精度な近似が Bowling らによって得られ，ディーラー（先手）が有利であることが報告された [1].

さらにゲーム開始時だけではなく，ゲーム中の様々な状況に対して網羅的に，ゲーム値を付けていくような研究もなされている．チェッカーやチェス，Awari では，ゲーム終盤のゲーム値を求める，エンドゲームデータベース（EGDB）に関する研究が盛んである．チェッカーについては，Schaeffer らが，盤上の駒の数が 9 ピース以下の配置すべてと 10 ピースの途中までの配置，合わせて約 13 兆配置の EGDB を，148GB で構築した [4]．チェスについては，2018 年に，盤面に 7 ピース以下の EGDB が 18.4TB で構築されており，現在 8 ピースの場合について計算中である[*3]．Awari では，1990 年代から EGDB を構築し，その範囲を拡張していく研究がなされていた [2], [9]．配置数が非常に多く，すべての配置のゲーム値を求めることはメモリサイズの観点から難しいとされていたが，2003 年に Romein らが配置を分割し，メモリに保持する配置数を減らすことによって，すべての配置にゲーム値をつけたデータベースを構築した [3].

不確定ゲームのバックギャモンの人工知能は，研究が進められ，人間と同等かそれ以上の性能を獲得するに至った．例として，TD-Gammon ではニューラルネットワークや強化学習，ヒューリスティック探索といった手法が用いられる [8]．その一方で，互いに最善を尽くした場合の勝率を求めることは，現状では非常に困難だと考えられる．バックギャモンは，2 人のプレイヤが自駒を全てベアオフ（ゴール）する早さを競う，双六のようなゲームである．バックギャモンの人工知能でも，EGDB が活用されている．一般公開されているバックギャモンをプレイする人工知能「GNU Backgammon」ではベアオフ DB（BODB）が構築されている[*4]．また，バックギャモンの人工知能として，eXtreme Gammon が市販されている．この人工知能は GNU Backgammon と同様 BODB を構築する機能を搭載しているが，その技術的詳細は公開されていない[*5]．

本研究では，GNU Backgammon に搭載されている BODB を拡張した EGDB を構築することを目指す．後ろ向き帰納法や開始点探査を伴うモンテカルロ法（MCES 法）を用いることで，ゲーム終盤の各配置の勝率を推定する．

2. バックギャモンのルール

三角形のポイントと呼ばれるマスを 24 個持つプレイ盤

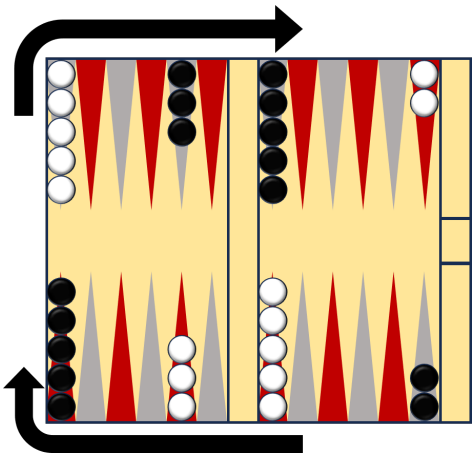

図 1 初期配置と黒駒が進む方向（白駒は反対方向）

Fig. 1 The initial position and the direction in which the black pieces move (the white pieces move in the opposite direction).

と，黒白それぞれ 15 個の駒とサイコロを 4 つ使用する．一方のプレイヤが 15 個の黒駒を，他方のプレイヤが 15 個の白駒をプレイ盤に配置してゲームを開始する．交互にサイコロを 2 つずつ振り，出目の数に応じて駒を進める．このとき，駒は交差するように進める．**図 1** に初期配置と黒駒を進める方向を示す．白駒は黒駒とは真逆の方向に進める．バックギャモンはこのように駒を進めて，相手プレイヤより先に全ての駒をゴールすることを目指すゲームである．駒をゴールさせることをベアオフという．

以下に簡単にゲームの進行手順を示す．

1. 先手プレイヤを決め，行動を決定する．
2. 以降各プレイヤは 2 つのサイコロを振り，出目に応じて自身の駒を進める．
3. 勝敗がついていれば終了，そうでなければ手番プレイヤを交換して手順 2 に戻る．

以降では，各手順について詳しい説明を行う．プレイヤは P1（白），P2（黒）とする．ルールは望月らの書籍に準拠する [10].

手順 1 では，先手後手の決定と先手の行動がなされる．サイコロを各プレイヤが 1 つずつ振ることで先手後手を決める．より大きい出目を出したプレイヤが先手プレイヤとなり，そのとき出ている出目を使ってゲームを開始する．数字が同じときはやり直す．例として，P1 が出目 5 を，P2 が出目 3 を出したとき，P1 が先手プレイヤとなり，出目 5 と 3 を使用して駒を進める．

手順 2 は，プレイヤの行動についてである．手番プレイヤは 2 つのサイコロを振り，出目 1 つにつき自身の駒を出目の数だけ進める．つまり，駒を 2 回進めることができる．ゾロ目の場合は 4 回進めることができる．ただし，相手の駒が 2 個以上あるポイントに駒を進めることはできない．

[*3] Endgame tablebase, https://en.wikipedia.org/wiki/Endgame_tablebase （最終アクセス 2024）
[*4] GNU Backgammon, http://gnu.ist.utl.pt/manual/gnubg/html_node/index.html（最終アクセス 2024）
[*5] eXtreme Gammon, https://www.extremegammon.com/Majorfeatures.aspx （最終アクセス 2024）

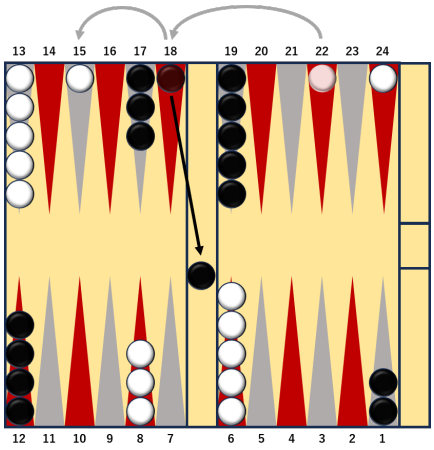

図 2 P1（白）が出目 4 と 3 で黒駒をヒットした後の配置
Fig. 2 The position after P1 (white) hits a black piece with a roll of 4 and 3.

相手の駒が 1 つだけあるとき，ヒットが起きる．これは，移動先の相手の駒をバーと呼ばれるプレイ盤中央へ移動することである．バーは振り出しであり，バーに置かれた駒はゴールから最も離れたポイントから進めることになる．

例を**図 2** に示す．P1 の手番で出目 4 と 3 がでたとき，1 回目に出目 4 を使ってポイント 18 に進める．ポイント 18 には黒駒が 1 つだけあるため，ヒットとなり，黒駒をバーに移動させる．2 回目に出目 3 を使って，ポイント 15 に進め，手番を終える．このとき，1 回目に出目 3 をポイント 22 の駒に使用することはできない．なぜなら，出目 3 で進めるポイント 19 には黒駒が 2 つ以上あるためである．バーに移動した駒は，白だとポイント 24 から，黒だとポイント 1 から進める．

バーに置かれた駒は最優先で盤上に出さなければならず，バーに自駒があるときに他の自駒を進めることはできない．また，ヒットが起こりうる配置，すなわち白と黒の駒が交差しきっていない配置をコンタクト（図 2），交差しきった配置をノーコンタクト（**図 3**）という．

次に，ベアオフの仕方についての説明を行う．インナーボードと呼ばれるポイントに自駒すべてを入れることで，ベアオフを行うことができる．インナーボードとは各プレイヤのゴール直前の 6 ポイントのことである．図 2 では，P1 がポイント 1 ～ 6，P2 が 19 ～ 24 である．全ての駒をインナーボードに入れると，ゴールまでのポイント数が出目と等しいポイントにある駒をベアオフすることができる．出目と等しいポイントとそれより大きいポイントに駒がなければ，ゴールから最も遠い駒をベアオフすることができる．図 3 で，P1 の手番で出目 6 と 5 がでたとする．全ての白駒がインナーボード内にあるため，まず，ポイント 6 の駒をベアオフする．次にポイント 5 には駒がなく，それ以上のポイント 6 にも駒がないため，ポイント 4 の駒をベアオフする．

手順 3 では勝敗がついていればゲームが終了する．勝利

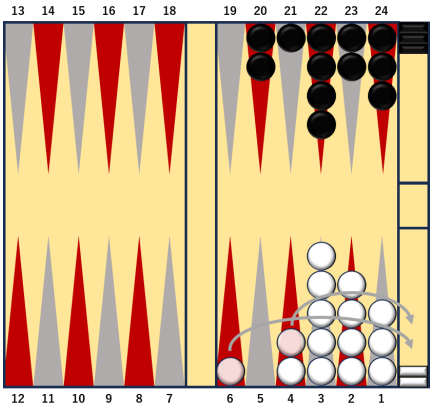

図 3 P1（白）が出目 6 と 5 でベアオフした後の配置
Fig. 3 The position after P1 (white) bears off with a roll of 6 and 5.

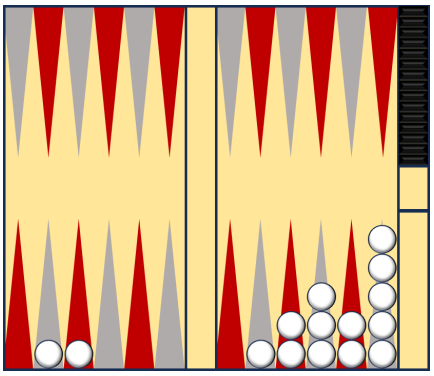

図 4 P2（黒）が行動後にギャモン勝ちした配置
Fig. 4 The position of P2 (black) winning by gammon after P2's move.

条件は，自身の駒 15 個すべてをベアオフすることである．勝ち点の基準は 1 点である．ベアオフしたときの相手プレイヤの駒の配置に応じて勝利点が異なる．それを以下に示す．ここでは，P1 が全ての駒をベアオフしたとする．P2 が全ての駒をベアオフしたときも同様である．

バックギャモン勝ち P2 のベアオフした駒が 0 個，かつ，P1 のインナーボードまたはバーに P2 の駒が 1 つ以上残っている．勝ち点 3 倍．

ギャモン勝ち バックギャモン勝ちではないが，P2 のベアオフした駒が 0 個である．勝ち点 2 倍．

シングル勝ち バックギャモン勝ちでも，ギャモン勝ちでもない．勝ち点 1 倍．

図 4 に P2 が行動後にギャモン勝ちした例を示す．白がベアオフした駒が 0 個であるが，黒のインナーボードまたは，バーに白駒は存在しない．そのため，P2 はギャモン勝ちとなる．

最後にその他の細かなルールやダブリングキューブ，対戦方法の説明を行う．P1 と P2 の手番ではサイコロの出目 2 つ（ゾロ目の場合は 4 つ）を使って駒を動かすが，配置

によっては全ての出目を使用できない場合がある．そのような場合に備えて次のルールが設けられている．両方の出目が使用できる行動が存在するならば，必ずそれを行う．また，どちらの出目も使用できるが，どちらか一方しか使用できなら，大きい方の出目を使わなければならない．つまり，可能な限り多くの出目を使用し，大きい出目を使用しなければならない．

ダブリングキューブは，最初は両プレイヤが使用でき，勝ち点を 2 倍していくルールである．例えば，P1 がダブリングキューブの使用を宣言すると，P2 はこの宣言を受けるか，降りるかを選択する．受けるとゲームの両プレイヤの勝ち点が 2 倍になる．降りるとそのときの倍率 ×1 点の勝ち点を宣言側（P1）が得る．以降のダブリングキューブの使用権は被宣言者が持つことになる．ダブリングキューブ使用の宣言は何度でも行うことができる．

ポイントマッチは，一定の目標点数を定め，先に勝ち点が目標点数を超えた方の勝ちとするルールである．これに対して，アンリミテッドマッチは目標点数を定めず，各 1 試合でポイント獲得数を決めるルールである．

3. GNU Backgammon のベアオフデータベース

GNU Backgammon はインターネット上で一般公開されているバックギャモンの人工知能であり，両側 BODB と片側 BODB の 2 種類の BODB を構築する機能を搭載している．

両側 BODB には，それぞれのプレイヤのインナーボードとゴールに駒すべてが在る配置の，正確な勝率が登録されている．例として図 3 の配置は，全ての黒駒は黒のインナーボード（ポイント 19 ∼ 24）とゴールに，白駒も白のインナーボード（ポイント 1 ∼ 6）とゴールにある．このような配置を両側 BODB では扱う．それぞれのインナーボードに各プレイヤの駒が 15 個ずつ全てある配置で，約 2GB で構築可能である．

片側 BODB では，一方のプレイヤの駒だけに着目して，後何回サイコロを振れば全ての駒をベアオフできるかの確率分布を求め，ノーコンタクトの両側配置の勝率を推定する．

4. 研究目的

本研究では，GNU Backgammon の BODB の両側データベースを拡張することを目指す．インナーボードの外のポイントに駒がある配置に加えて，コンタクトの配置も扱う両側 EGDB を構築する．研究の第一歩として，まずは以下の制限を設ける．

- ダブリングキューブを使用しない．
- アンリミテッドマッチを想定する．

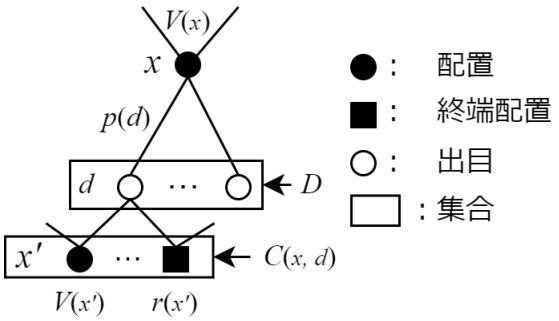

図 5 配置の 1 ステップダイナミクス
Fig. 5 The one-step dynamics of the positions.

- 勝敗はシングル勝ち/負けのみを扱う．利得は 1，または −1.

制限の 3 つ目については，本研究ではゲーム終盤として白黒の駒がそれぞれ必ず 1 つはベアオフしている配置を扱う．そのため，バックギャモン勝ちやギャモン勝ちの条件は満たされない．

5. 実験方法

X を駒の配置すべてからなる集合とする．全ての $x \in X$ において，これからプレイヤ i が行動する配置 x の手番で，プレイヤ i から見た勝率 $V_i(x)$ を求める．ここで，i は白か黒であり，全ての $x \in X$ で $V(x) = V_{白}(x) = V_{黒}(I(x))$ である．ここで，$I(x)$ は配置 x を反転した配置を得る関数である．反転とは，駒の色を入れ替えた後（白なら黒，黒なら白），ポイント $k = 1, \cdots, 24$ にある駒を $25 - k$ の位置に移動させる操作である．また，x が終端しているときのプレイヤ i が得る利得を $r_i(x)$ とし，全ての $x \in X$ で $r(x) = r_{白}(x) = r_{黒}(I(x))$ である．

2 人のプレイヤが勝率に最大値を与える行動を選択するとき，$V_i(x)$ は以下の式を満たす．

$$V_{白}(x) = \begin{cases} r_{白}(x) & (x \text{ が終端}) \\ \displaystyle\sum_{d \in D} p(d) \cdot \max_{x' \in C(x,d)} -V_{白}(I(x')) & (\text{その他}) \end{cases}$$

ただし，D はサイコロ 2 つの出目の集合，$p(d)$ は出目が d になる確率，$C(x, d)$ は配置 x で出目が d のときに手番プレイヤが選択できる配置の集合である．**図 5** にこれらの値の関係を模式的に示す．

5.1 後ろ向き帰納法による計算

初めに，後ろ向き帰納法を用いていくつかの配置 x の勝率 $V(x)$ を求める．後ろ向き帰納法は，図 5 のような部分を持つ有向グラフにおいて，勝率が既に決定している終端配置から遡って，配置 x の勝率 $V(x)$ を求める手法である．反復処理を行うことで，できる限り多くの配置 x の勝率を求めることを目指す．有向グラフに閉路が存在するとき，

Algorithm 1 後ろ向き帰納法

Input: 配置集合 $X_{b,w}$
Output: 表 V

$X_{\text{uk}} \leftarrow X_{b,w}$
X_{new} を空集合に初期化
V の各 x の勝率を不明に初期化（$V(x) \leftarrow$ 不明）
loop
  for all $x \in X_{\text{uk}}$ **do**
    for all $d \in D$ **do**
      if $\max_{x' \in C(x,d)} -V(I(x'))$ が求まる **then**
        $e(d) \leftarrow \max_{x' \in C(x,d)} -V(I(x'))$
      else
        $e(d) \leftarrow$ 不明
    if $\forall d \in D, e(d) \neq$ 不明 **then**
      $V(x) \leftarrow \sum_{d \in D} p(d) \cdot e(d)$
      x を X_{uk} から取り除き，X_{new} に追加
  if X_{new} が空集合 **then return** V
  X_{new} を空集合にする

勝率を決定できない場合がある．このような配置は，十分な反復処理がなされても，勝率が不明のままである．

本研究で用いた後ろ向き帰納法のアルゴリズムを Algorithm 1 に示す．入力は配置集合 $X_{b,w}$ である．$X_{b,w}$ はベアオフされた駒が，黒が b 個，白が w 個のときの配置集合である．出力 V は配置 x と勝率 $V(x)$ のタプルをあらわす表とみなす．X_{uk} は勝率が不明な配置の集合，X_{new} は 1 回の反復内で新たに勝率が求まった配置の集合である．X_{new} が空集合になると反復処理を終える（18 行目）．11 行目の最大値が求まらない場合とは，勝率が不明な x' が $C(x,d)$ にあり，勝率 1 の x' が $C(x,d)$ にないときである．

5.2 グリーディ方策を用いた開始点探査を伴うモンテカルロ法

5.1 節で用いた後ろ向き帰納法による計算では，全ての配置 x の勝率を求められない場合がある．そのため，Algorithm 1 によって得られた表 V にグリーディ方策を用いた MCES 法を用いることで，後ろ向き帰納法によって勝率が求まらなかった配置 x の勝率を推定する [6]．

本研究で用いた MCES 法のアルゴリズムを Algorithm 2 に示す．入力は Algorithm 1 によって得られた表 V，配置集合 $X_{b,w}$，反復回数 N である．出力は，配置と新たに得られる推定勝率のタプルの表 V_{new} である．$Returns$ はゲームプレイによって得られた開始点 x_0 の報酬を格納するリストである．ゲームプレイの生成では，ある配置 x についてサイコロの出目 d を一様ランダムに決定する．配置 x と出目 d によって得られる配置の集合 $C(x,d)$ から，勝率が最大の手を選択するグリーディ方策を取る．そして，ゲームが終端したとき，その利得 $r(x)$ を $Returns(x_0)$ に追加する．このようにして，様々な配置でゲームプレイを

Algorithm 2 グリーディ方策を用いた MCES 法

Input: 表 V, 配置集合 $X_{b,w}$, 反復回数 N
Output: 表 V_{new}

$Returns \leftarrow$ サイズ $|X_{b,w}|$ のリストを 0 で初期化
for all $x \in X_{b,w}$ **do**
  if $V(x)$ が不明 **then** $V_{\text{new}}(x) \leftarrow 0$
  else $V_{\text{new}}(x) \leftarrow V(x)$
for $i \leftarrow 1$ to N **do**
  for all $x_0 \in X_{b,w}$ **do**
    for all $(x,d) \in X_{b,w} \times D$ **do**
      $\pi(x,d) \leftarrow \text{argmax}_{x' \in C(x,d)} - V_{\text{new}}(I(x'))$
    π を用いて，x_0 から始まるゲームプレイを生成
    $Retruns(x_0)$ に $r(x_0)$ を追加
    $V_{\text{new}}(x_0) \leftarrow \text{average}(Returns(x_0))$
return V_{new}

表 1 後ろ向き帰納法の結果
Table 1 The result of backward induction.

ベアオフされた駒 (黒, 白)	総配置数	値がついた配置数
(14, 14)	601	282
(13, 14)	7525	2458
(14, 13)	7525	2302

いくつも生成し，それで得られた利得の平均を，配置 x の推定勝率とする．

5.3 その他のテクニック

配置 x を表 V, V_{new} に登録するにあたり，配置と 1 対 1 に対応づけられた整数値があると都合が良い．これは，湊によって 1993 年に考案・命名された ZDD（zero-suppressed binary decision diagram：ゼロサプレス型二分決定グラフ）を用い，配置集合を実装して行う [12]．

また，バックギャモンの総配置数は $18,528,584,051,601,162,496 \simeq 10^{19}$ であることが知られている[*6]．この配置数は非常に大きく，計算機のメモリに，配置すべてを網羅する表を持たせることは困難である．そこで，本研究ではバックギャモンの配置を分割する．分割はベアオフしている黒と白の駒の数に応じて行う．例えばゲーム終盤の分割は，$(b,\ w)$= (14, 14)，(13, 14)，(14, 13) というように分割する．

6. 実験結果

ベアオフされている駒が $(b,\ w) = (14,\ 14)$，$(13,\ 14)$，$(14,\ 13)$ の 3 通りについて後ろ向き帰納法を行った結果を**表 1** に示す．

ゲームの性質から，ノーコンタクトの配置は駒をヒットしたりされたりすることがなく，同じ配置が繰り返される

[*6] Backgammon GALORE!, `https://www.bkgm.com/rgb/rgb.cgi?view+371`（最終アクセス 2024）

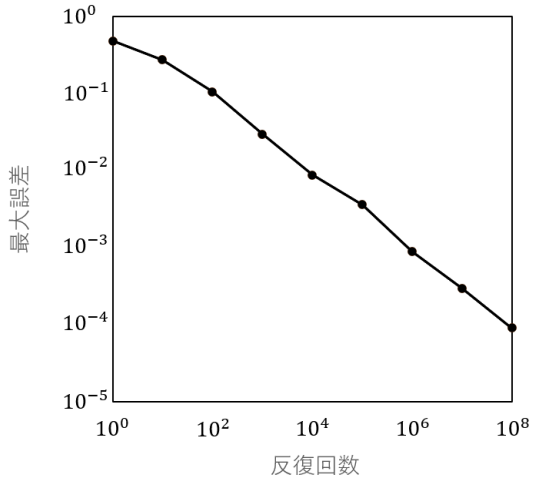

図 6 最大誤差の減少
Fig. 6 Decrease in maximum error.

ことはない．そのため，ノーコンタクトの配置すべてには値がついているはずである．構築したそれぞれの EGDB に対して，ノーコンタクトの配置すべての値が求まっていることを確認した．また，手番が白プレイヤの場合の勝率を登録しているため，ベアオフされた駒が (14, 14) のとき，黒駒の配置に関わらず，白駒がポイント 1 ～ 3 にある配置にも値をつけることができた．これは，いかなる出目であってもベアオフすることができるからである．コンタクトの配置の多くには，値をつけることができなかった．これは，値が不明な配置が $C(x,d)$ にいつまでも存在することになり，最大値が求まらず，勝率 $V(x)$ が求まらないためだと考えらえる．

後ろ向き帰納法で値がつかなかった勝率を推定するために，ベアオフされた駒が $(b, w) = (14, 14)$ のときの配置に MCES 法を適用した．**図 6** に推定勝率の最大誤差の推移を示す．横軸は反復回数，縦軸は最大誤差の両対数グラフとした．最大誤差は，

$$\max_{x \in X_{b,w}} \left| V_{\text{new}}(x) - \sum_{d \in D} p(d) \cdot \max_{x' \in C(x,d)} -V_{\text{new}}(I(x')) \right|$$

で評価した．図 6 より，反復を繰り返せば誤差が減少していくことが確認できた．

7. まとめ

本研究では，後ろ向き帰納法によってベアオフされた駒が (黒, 白) = (14, 14)，(13, 14)，(14, 13) のときの両側 EGDB を構築した．さらに，ベアオフされた駒が (14, 14) の配置集合に対しては，MCES 法を適用して勝率を推定し，網羅的な両側 EGDB を構築した．本研究で扱った駒の配置は盤上に駒が 3 つまでしかなく，極めて限定的と言わざるをえないが，それでも，GNU Backgammon の両側 BODB では扱えないコンタクトの配置の高精度な推定勝率を含む両側 EGDB を構築することができた．また，後ろ向き帰納法では値がつけられなかった配置の勝率は，MCES 法によって高い精度で推定できることが示された．

今後は，ベアオフされた駒の数を減らしていき，より大きな EGDB を構築することを目指す．また，MCES 法よりも効率の良い強化学習法の適用も検討する．

謝辞 本研究は JSPS 科研費 JP24K15244 の助成を受けたものです。

参考文献

[1] Bowling, M., Burch, N., Johanson, M. and Tammelin, O.: Heads-up Limit Hold'em Poker Is Solved, *Science*, Vol. 347, No. 6218, pp. 145–149 (2015).
[2] Romein, J. W. and Bal, H. E.: Awari Is Solved, *ICGA Journal*, Vol. 25, No. 3, pp. 162–165 (2002).
[3] Romein, J. W. and Bal, H. E.: Solving Awari with Parallel Retrograde Analysis, *Computer*, Vol. 36, No. 10, pp. 26–33 (2003).
[4] Schaeffer, J., Björnsson, Y., Burch, N., Lake, R., Lu, P. and Sutphen, S.: Building the Checkers 10-Piece Endgame Databases, *Advances in Computer Games* (van den Herik, H. J., Iida, H. and Heinz, E. A., eds.), Springer, pp. 193–210 (2004).
[5] Schaeffer, J., Burch, N., Bjornsson, Y., Kishimoto, A., Muller, M., Lake, R., Lu, P. and Sutphen, S.: Checkers Is Solved, *Science*, Vol. 317, No. 5844, pp. 1518–1522 (2007).
[6] Sutton, R. S. and Barto, A. G.: *Reinforcement Learning*, MIT Press (1998).
[7] Takizawa, H.: Othello Is Solved, *preprint arXiv:2310.19387* (2023).
[8] Tesauro, G.: Programming Backgammon Using Self-teaching Neural Nets, *Artificial Intelligence*, Vol. 134, No. 1–2, pp. 181–199 (2002).
[9] van der Goot, R.: Awari Retrograde Analysis, *Computers and Games (CG 2000)* (Marsland, T. and Frank, I., eds.), pp. 87–95 (2001).
[10] 望月正行，景山充人，桑門昌太郎：改訂新版 バックギャモン・ブック，河出書房新社 (2017).
[11] 竹下祐輝， 池田諭：縮小盤オセロにおける完全解析，宮崎大学工学部紀要，Vol. 44, pp. 221–227 (2015).
[12] ERATO 湊離散構造処理系プロジェクト：超高速グラフ列挙アルゴリズム〈 フカシギの数え方 〉が拓く、組み合わせ問題への新アプローチ，森北出版 (2015).