# NOCCA × NOCCAの強解決

山本敦也[1,a)]　保木邦仁[1,b)]

**概要**：NOCCA × NOCCA は「ンダノガ」が 2017 年に販売したボードゲームであり、二人完全情報零和ゲームである。盤上のコマを移動させるという点は将棋をはじめとする多くのゲームと共通しているが、盤を立体的に用いてコマの上にコマを乗せることができるという特徴を持つ。本研究では、後退解析を用いて擬到達可能な配置すべての勝敗を計算し、初期配置は先手必勝であり手数を伸ばそうとする後手に最短で 41 手で勝つという結果が得られた。また、この最短手数の最大値は 69 手という結果も得られ、この最大手数は到達可能な配置で実現されることも分かった。

# Strongly Solving NOCCA × NOCCA

Atsuya Yamamoto[1,a)]　Kunihito Hoki[1,b)]

***Abstract:*** NOCCA × NOCCA is a board game released by "Undanoga" in 2017 and is a two-person perfect-information zero-sum game. The game is similar to Shogi and many other games in that the player moves the pieces on the board, but it is unique in a sence that the board can be used three-dimensionally to place the pieces on top of each other. In this study, wins and loses were calculated for all pseudo-reachable positions using retrograde analysis. As a result, the initial position is a first win game, and first player wins against second player who tried to increase the number of moves in the shortest 41 moves. The results also showed that the maximum number of these shortest moves is 69, and the it was also found that this maximum number of moves is achieved with a reachable position.

## 1. はじめに

二人有限完全情報零和ゲームは理論上、全ての手番に対してゲーム値 (例えば、勝ち・負け・引分) を決めることができる。これまで様々なゲームにおいてゲーム値を求めるための研究が行われてきた。ゲームの解決は以下の 3 つに分類される [1]。

**強解決**　すべての手番に対してゲーム値が判明している

**弱解決**　開始点のゲーム値と選択すべき行動列が判明している

**超弱解決**　開始点のゲーム値は判明しているが行動列は判明していない

本研究では、二人完全情報零和ゲーム「NOCCA × NOCCA」を強解決する。このゲームには千日手のようなルールは存在しなくて、一つのゲームプレイに同一のコマの配置を 2 回以上繰り返すことがある。このようにして、いつまでも続くゲームプレイの結果も本研究では形式的に引分と書く。このゲームではコマの数の増減がないため、配置の集合をコマの数により分割することができない。

## 2. 関連研究

これまで様々なゲームにおいて強解決や弱解決を目指す研究が行われてきた。これまでに強解決された二人完全情報零和ゲームの例とコマなどの配置数を表 1 に示す。

諏訪は、3 × 3 NOCCA × NOCCA の強解決を行い、初期配置が先手必勝であることや 3 分の 1 程度のコマの同形 (手番プレイヤの対称性や盤の対称性を考慮して同じとなる配置の集合) でツークツワンクになることを示した [4]。また、同形の数は 38 554 であると報告した。

シンペイは互いに 4 つずつ持つ駒を置いていき、使い

[1] 電気通信大学情報理工学研究科
Graduate School of Informatics and Engineering, The University of Electro-Communications
[a)] y2231160@edu.cc.uec.ac.jp
[b)] k.hoki@uec.ac.jp

切ったら駒を移動させ「上の世界」または「下の世界」で駒を 3 つ並べることを目標とするゲームである。田中はこれの強解決を行い、初期配置が後手必勝であることやそれに要する手数が 21 であることなどを示した [9]。同一配置を繰り返すゲームプレイが存在するがその場合引分とした。この研究も諏訪と同様に同形を数え、到達不可能な配置も含む。

どうぶつしょうぎは将棋に類似したゲームであるが、将棋と比べて簡潔なルールとなっている。同じ配置に 3 回到達すると引分となるため、有限ゲームである。田中はこれの強解決を行い、初期配置が後手必勝であることやそれに要する手数が 78 であることなどを示した [7]。この研究も同様に同形を数えているが、ゲーム木の深さ優先探索を行い到達不可能な配置を排除している点が $3 \times 3$ NOCCA $\times$ NOCCA やシンペイの研究とは異なる。

十六むさしは日本の古いボードゲームである。田中はこれの強解決を行い、初期配置で先手必勝であることを示した [8]。ここで、千日手は黒勝ちとして、引分にはならない有限ゲームとした。同形を数えて、到達不可能な配置も含む。十六むさしはボード上の駒の数が広義単調減少するゲームであるため、駒の数により配置の集合を分割し、計算中のメモリ使用量を減らすという方法がある。しかし、すべての配置をメモリ上にのせることができるため、このような分割は行っていない。後退解析の結果は、1 エントリ 8 ビットの表として得た。

Quixo は、一直線に自身のマークを並べることを目標とするゲームである。田中らはこれの強解決を行い、初期配置が引分であることを示した [6]。配置数は、手番の対称性による同型のみを考慮し、到達不可能な配置も含む。Quixo は○や×の数が広義単調増加していくゲームで、それらの数のよる配置の集合の分割を行い、計算中のメモリ使用量を削減している。メモリに保存する配置の集合は最大 4 つで、大幅な使用メモリ領域の削減に成功している。

Awari はアフリカの古いゲームである。Romein らはこれを強解決し、初期配置で引分であることを示した [2]。同形を数えて、どれも到達可能である。Awari はボード上の石が広義単調減少するゲームであり、石の数による配置の集合の分割を行い、できるだけディスクではなくメモリを使って表に効率よくアクセスするための工夫をした。

## 3. NOCCA × NOCCA

$5 \times 6$ の盤と、黒白 5 個ずつのコマを使用する。一方のプレイヤは黒コマ 5 個、他方は白コマ 5 個を持つ。図 1 のように各プレイヤはそれぞれのプレイヤから見て手前 5 マスにこれらのコマを配置する。

まず、先手と後手を何かしらの方法で決める。そして先後 2 人が交互にコマをひとつずつ動かすことでゲームが進行する。コマは前後左右斜めへ、1 マス移動できる。移動

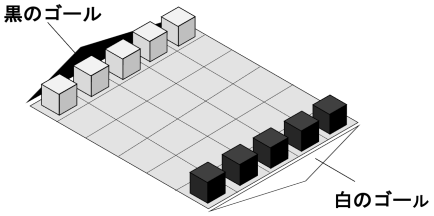



図 1: 初期配置

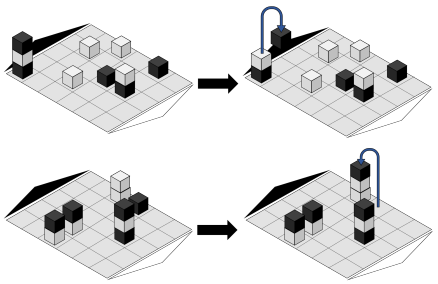


図 2: 黒が勝利する着手の例

先のマスにコマがあるとき、そのコマの上に移動する。これを「乗っかる」と呼ぶ。乗っかられているコマを動かすことはできないが、乗っかっているコマは通常通りに移動させることができる。乗っかっているコマにさらに乗っかることもできる。しかし、コマの重なりは 3 段までで 4 段目に乗っかるようには移動できない。

一方のプレイヤが勝利条件のいずれかを満たしたらゲームは終了する。各プレイヤの勝利条件は、

- そのプレイヤのコマが相手陣のゴールに侵入する
- 相手のコマ全てに乗っかるなどして、動けるコマをなくす

ことである。相手陣のゴールには、相手の初期位置である 5 マスすべてから 1 手で行くことができる。

## 4. 組合せ集合

コマの配置は、組合せ集合として表現することができる。組合せ集合とは一般に、アイテムの組合せを要素とする集合である。例えば、アイテムの全体集合が $\{a, b, c, d\}$ ならば、アイテムの組合せ集合は $2^{2^4} = 65536$ 通りつくることができる。{ac,bc,d},{λ,acd,ab,bd,d},{c},∅ はそれぞれ組合せ集合の一例である。ここで、λ は大きさ 0 の組合せを表し、∅ は空集合を表す。

### 4.1 場合分け二分木

組合せ集合を表すデータ構造の一つに場合分け二分木がある。場合分け二分木は、根から葉に向かう有向枝からなる二分木である。節点にはアイテムが割り当てられており、2 本の枝に分岐している。深さが同じ節点には同じアイテムが割り当てられる。2 本の枝の一方には 0、他方には 1 のラベルが付与されており、以降では 0 のラベルをもつ枝

表 1: コマや石などの配置数と解決状況

| ボードゲーム | 配置数 | 初期配置 | 初期配置手数 | 最大手数 | 最大分割 [1] |
|---|---|---|---|---|---|
| 3 × 3 NOCCA × NOCCA [4] | 38 554 | 先手勝ち | 11 | 不明 | - |
| シンペイ [9] | 12 102 165 | 後手勝ち | 21 | 49 | - |
| どうぶつしょうぎ [7] | 99 485 568 | 後手勝ち | 78 | 173 | - |
| 十六むさし [8] | 141 737 590 784 | 先手 (茶) 勝ち | 35 | 57 | -[2] |
| NOCCA × NOCCA (本研究) | 147 969 899 280 | 先手勝ち | 41 | 69 | -[3] |
| Quixo [6] | 847 288 609 443 | 引分 | - | 不明 | 26 293 088 250 |
| Awari [2] | 889 063 398 406 | 引分 | - | 不明 | 203 648 015 936 |

[1] コマの数が単調減少 (増加) するゲームにおいて、コマの数による分割の最大配置数

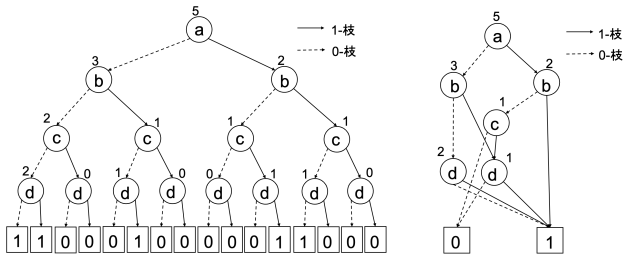
[2] 十六むさしは分割する方法はあるが、行っていない。

[3] 段数の数の組合せで分割する方法があるが、行っていない。

を 0-枝、1 のラベルをもつ枝を 1-枝と呼ぶ。そして葉には 0 または 1 のラベルが付与されている。以降では 0 のラベルをもつ葉を 0-葉、1 のラベルをもつ葉を 1-葉と呼ぶ。

組合せ二分木において、1-枝と 0-枝はその節点のアイテムが組合せに属するかどうかの場合分けを表す。すなわち 1-枝はその節点のアイテムが組合せに属することを、0-枝はその節点のアイテムが組合せに属さないことを表す。また、1-葉と 0-葉は根節点からその葉までの経路に対応する組合せが集合に属するかどうかを表している。すなわち 1-葉はその葉に対応する組合せが集合に属することを、0-葉はその葉に対応する組合せが集合に属さないことを表す。

先に挙げた組合せ集合の例 {λ,acd,ab,bd,d} を場合分け二分木で表現すると図 3a のようになる。図 3a の葉以外の各節点には、その節点を始点として 1-葉にいたる経路数を記録している。以降では節点 $n$ を始点として 1-葉にいたる経路数を節点 $n$ の値を呼ぶ。葉の数は 2 のアイテム数乗になり、アイテム数が多いと場合分け二分木で表す方法は現実的ではない。




(a) 場合分け二分木　　(b) ZDD

図 3: 組合せ集合 {λ,acd,ab,bd,d} を表現するデータ構造

### 4.2 ZDD

ZDD (zero-suppressed binary decision diagram: ゼロサプレス型二分決定グラフ) は、場合分け二分木と同様、組合せ集合を表すデータ構造である。これは非巡回有向グラフによる表現で、湊が 1993 年に考案・命名した [10]。前節で述べた場合分け二分木をある規則に従って圧縮することによって得られる。

場合分け二分木から ZDD を得るための圧縮規則を説明する。次の二つの圧縮規則を可能な限り適用する。

- 冗長節点の削除: 1-枝が 0-葉を指すとき、この節点を取り除き、親節点すべてを 0-枝に直結させる。
- 等価節点の共有: 等しい部分木を共有する。

これらの圧縮規則を適用して節点を削除していくと、最終的にこれ以上小さくならない既約な形が得られる。この既約形は、圧縮規則をどの節点から順に適用するかに関係なく、同じ組合せ集合であれば必ず同じ形になることが知られている。図 3a に示した場合分け二分木に上記の圧縮規則を適用して得られる既約な ZDD を図 3b に示す。

場合分け二分木と同様に節点の値を記録しているが、これは葉に近い分岐節点から根の方向に各節点の値を求めていくことにより、節点数に比例する回数の足し算で節点の値すべてを求めることができる。根節点の値は、組合せ集合の要素数に対応する。このような経路の数え上げができるという性質を用いると、ある組合せが集合に含まれるかどうかだけではなく、すべての組合せを辞書順に並べたとき、その組合せが何番目であるかということを効率よく知ることができる。すなわち、$x$ 個の組合せに対して、0 から $x-1$ までの整数値を 1 対 1 で効率よく対応させることができる。

ある組合せが与えられたとき、それに対応する整数値を ZDD を使い求めるアルゴリズムを表 2 に示す。またある整数値が与えられたとき、それに対応する組合せを ZDD を使い求めるアルゴリズムを表 3 に示す。これらのアルゴリズムは、それぞれ組合せの最小完全ハッシュ関数とその逆関数になっている。計算時間のオーダーはどちらも ZDD の最大経路長である。

## 5. 後退解析

後退解析は、終端節点に勝敗が与えられている有向グラフにおいて、各内部節点に勝敗を決定していく手法である。反復処理により、終端節点に近い節点から勝敗を決定して

表 2: 組合せ (コマの配置) に対応する整数値を求めるアルゴリズム。アイテムはマスとマス状態の対。

| | |
|---|---|
| Input | 組合せ集合を表す ZDD と組合せ $a$ |
| Output | 組合せ $a$ に対応する整数値 $k$ |
| 1 | $k \leftarrow 0, n \leftarrow$ 根節点 |
| 2 | 以降繰り返し: |
| 3 | if $n$ が 0-葉 then return 0 |
| 4 | if $n$ が 1-葉 then return $k$ |
| 5 | if $n$ のアイテムが $a$ に属す then |
| 6 | $k \leftarrow k +$ ($n$ の 0-枝の子節点の値) |
| 7 | $n \leftarrow n$ の 1-枝の子節点 |
| 8 | else |
| 9 | $n \leftarrow n$ の 0-枝の子節点 |

表 3: 整数値に対応する組合せ (コマの配置) を求めるアルゴリズム。アイテムはマスとマス状態の対。

| | |
|---|---|
| Input | 組合せ集合を表す ZDD と整数値 $k$ ($0 \leq k <$ 根節点の値) |
| Output | 整数値 $k$ に対応する組合せ $a$ |
| 1 | $a \leftarrow \emptyset$, $n \leftarrow$ 根節点 |
| 2 | 以降繰り返し: |
| 3 | if $n$ が 1-葉 then return $a$ |
| 4 | if $n$ の 0-枝の子節点 $\leq k$ then |
| 5 | $k \leftarrow k -$ ($n$ の 0-枝の子節点の値) |
| 6 | $a \leftarrow a \cup$ ($n$ のアイテム) |
| 7 | $n \leftarrow n$ の 1-枝の子節点 |
| 8 | else |
| 9 | $n \leftarrow n$ の 0-枝の子節点 |

いく。この手法は「どうぶつしょうぎ」や「チェッカー」などに用いられている [4] [9] [7] [8] [6] [2] [3]。

解析する有向グラフを組 $(V, E)$ によって定義する。$V$ は節点の集合、$E$ は枝の集合とする。各節点のゲーム値は、手番プレイヤからみたものとする。

後退解析では通常、コマの配置と手番プレイヤから導くことのできる勝敗のみを扱うこととなる。この場合節点は、手番プレイヤと配置との組と 1 対 1 に対応することとなる。そして例えば、繰り返しが勝敗に影響を与えるゲームはあらわに扱うことができない。ただし、配置を繰り返して引分になる場合は、ゲーム値が不明な節点が最後まで残るという性質を利用して、引分を間接的に扱うことができる。

ゲームのルールが持つ対称性を利用して、一方の手番プレイヤに現れる配置のみを考えたアルゴリズムを表 4 に示す。この場合、節点は配置に対応する。このアルゴリズムは、値が不明の節点集合 $V_{\mathrm{uk}}$ に対して反復処理を行い、値が勝ちの節点集合 $V_{\mathrm{w}}$、値が負けの節点集合 $V_{\mathrm{l}}$ が得られる。ここで $V_{\mathrm{nw}}$ は値が新たに勝ちであると判明した節点集合、$V_{\mathrm{nl}}$ は値が新たに負けであると判明した節点集合であり、これらが空集合になったら反復処理を終了する。$V_{\mathrm{w}}$ や $V_{\mathrm{l}}$ に属さない節点は値が決定されなった節点である。

表 4: 後退解析のアルゴリズム

| | |
|---|---|
| Input | 有向グラフ $(V, E)$ |
| Output | 節点集合 $V_{\mathrm{w}}, V_{\mathrm{l}}, V_{\mathrm{uk}}$ |
| 1 | $V_{\mathrm{uk}} \leftarrow V$ |
| 2 | $V_{\mathrm{w}}, V_{\mathrm{l}}, V_{\mathrm{nw}}, V_{\mathrm{nl}}$ を空集合に初期化 |
| 3 | 以降繰り返し: |
| 4 | for each $x \in V_{\mathrm{uk}}$ do |
| 5 | if 節点 $x$ が勝ち then |
| 6 | $x$ を $V_{\mathrm{uk}}$ から取り除き、$V_{\mathrm{nw}}$ に追加 |
| 7 | else if 節点 $x$ が負け then |
| 8 | $x$ を $V_{\mathrm{uk}}$ から取り除き、$V_{\mathrm{nl}}$ に追加 |
| 9 | else |
| 10 | $x$ の子節点集合を $V_{\mathrm{c}}$ とする |
| 11 | if $V_{\mathrm{c}}$ に $V_{\mathrm{l}}$ の要素がある then |
| 12 | $x$ を $V_{\mathrm{uk}}$ から取り除き、$V_{\mathrm{nw}}$ に追加 |
| 13 | else if $V_{\mathrm{c}}$ が $V_{\mathrm{w}}$ の部分集合 then |
| 14 | $x$ を $V_{\mathrm{uk}}$ から取り除き、$V_{\mathrm{nl}}$ に追加 |
| 15 | if $V_{\mathrm{nw}} \cup V_{\mathrm{nl}} = \emptyset$ then return $V_{\mathrm{w}}, V_{\mathrm{l}}, V_{\mathrm{uk}}$ |
| 16 | $V_{\mathrm{w}} \leftarrow V_{\mathrm{w}} \cup V_{\mathrm{nw}}$ |
| 17 | $V_{\mathrm{l}} \leftarrow V_{\mathrm{l}} \cup V_{\mathrm{nl}}$ |
| 18 | $V_{\mathrm{nl}}$, $V_{\mathrm{nl}}$ を空集合に初期化 |

反復処理 $i$ 回で値がつかない節点は、その配置から互いに $i$ 手で負けないような、グラフの各節点に対する行動計画があるような配置である。また、反復処理が終了しても値がつかない節点は、その配置から互いのプレイヤが負けないような、グラフの各節点に対する行動計画があるような配置である。

## 6. NOCCA × NOCCA の配置数の見積り

本章では、NOCCA × NOCCA の配置数を見積もる方法と、その計算結果を示す。

### 6.1 方法

本研究では、以降の条件すべてを満たす配置を擬到達可能な配置とかく。

- $5 \times 6$ の盤上に白コマと黒コマがそれぞれ 5 個ずつ
- コマの重なりは 3 段まで
- 同じマスの同じ段に複数のコマが存在しない。
- 真上から見たときに、白コマと黒コマがどちらも 1 個以上見えている

ゲームのルールが持つ対称性を利用して、本研究では黒の手番のみを考える。ゲーム開始点から到達可能で、勝利条件を満たさない配置はすべて擬到達可能な配置である。

### 6.2 結果

擬到達可能な配置数をコマの重なりの段数に着目し、場合分けをして計算した結果、147 969 899 280 となった (表 5)。また、左右反転を考慮した同形を数えた結果、73 986 754 080 となった。なお、擬到達可能な配置のう

表 5: 段数の数の組合せで場合分けして数えた擬到達可能な配置数 (左右の対称性を考慮して削除していない)

| 3 段 | 2 段 | 1 段 | 擬到達可能な配置数 |
|---|---|---|---|
| 3 | 0 | 1 | 26 308 800 |
| 2 | 2 | 0 | 39 463 200 |
| 2 | 1 | 2 | 1 068 795 000 |
| 2 | 0 | 4 | 2 244 469 500 |
| 1 | 3 | 1 | 712 530 000 |
| 1 | 2 | 3 | 8 977 878 000 |
| 1 | 1 | 5 | 21 546 907 200 |
| 1 | 0 | 7 | 11 799 496 800 |
| 0 | 5 | 0 | 35 626 500 |
| 0 | 4 | 2 | 2 244 469 500 |
| 0 | 3 | 4 | 17 955 756 000 |
| 0 | 2 | 6 | 41 298 238 800 |
| 0 | 1 | 8 | 32 448 616 200 |
| 0 | 0 | 10 | 77 571 343 780 |
| 計 | | | 147 969 899 280 |

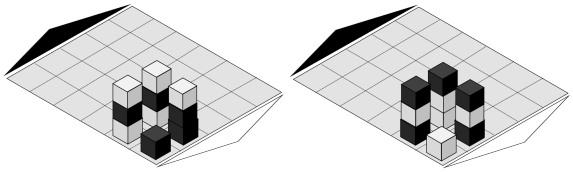


図 4: (左) 黒可能着手なしで負けになる終端配置と (右) 白着手直後に実現されることのない到達不可能な配置の例

ち、終端配置を 30 個、到達不可能な配置を 60 個みつけた (図 4)。8 章の結果より、終端配置は 31 個以上ないことが示された。また、到達不可能な配置は 61 個以上ある可能性はあるが、著者らは 61 個以上ないと予想した。

NOCCA × NOCCA は先行研究と同様にコマの数で配置の集合を分割することはできないが、各段数の数で分割する方法が考えられる。しかし、その最大配置数が全体の半数を超えており、メモリ使用量の削減の面でさほど有効ではない。

## 7. 最小完全ハッシュ関数の構築

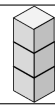
本章では、NOCCA × NOCCA の擬到達可能な配置の最小完全ハッシュ関数とその逆関数を構築する方法とその結果を示す。なお、左右の対称性を考慮して配置数を半分程度にするようなことは、ここでは考えない。

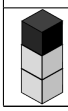
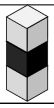
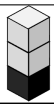
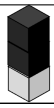
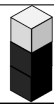
これら 2 つの関数の構成は、武田らがしたようにフロンティア法 (ZDD を効率よく得る手法の一つ、文献 [10] 参照) のような方法を用いて行う [5]。なお、本章の「節点」は場合分け二分木や ZDD の節点の意である。

### 7.1 方法

各マスのマス状態は表 6 の 15 通りである。本実験では、表 6 の 15 通りのマス状態から 1 つを盤上の各マスに割り

表 6: マス状態とその番号

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| |  |  |  |  |  |  |  |

| 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |

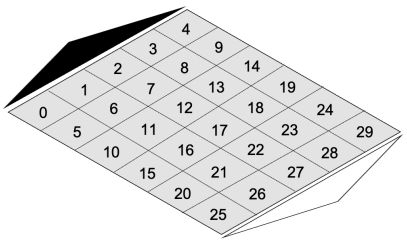



図 5: マスのラベル

当てていくことにより配置を考える。

表 7 はフロンティア法のような方法を NOCCA × NOCCA で行う疑似コードであり、擬到達可能な配置の最小完全ハッシュ関数を実現する ZDD を構築する。この ZDD の各節点には、「盤上の特定のマスは特定の状態である」というアイテムが割り当てられる。例えば、根節点には「マス 0 は状態 0 である」というアイテムを割り当てる。あるマスに対して、マス状態 0 からマス状態 13 であるかどうかを表す節点を生成し、これらすべてで 0-枝を選ぶときにそのマスに状態 14 を割り当てる。ここでマスには図 5 のように 0 から 29 までのラベルをつけている。

表 7 では、節点 $n$ はメンバとして以下を持つ。

- $n$.nw: $n$ を訪問するまでに置いた白コマの数
- $n$.nb: $n$ を訪問するまでに置いた黒コマの数
- $n$.f: $n$ に対応するマスにオブジェクトが既にあるかどうかを表すフラグ
- $n$.topw: 盤を真上から見たときに見える白コマの数
- $n$.topb: 盤を真上から見たときに見える黒コマの数

節点 $n_0$ と $n_1$ の「メンバ nw、nb、f の値」が等しく、「topw が 0 であるか否か」、「topb が 0 であるか否か」が等しく、「割り当てられているアイテム」も等しいとき、これらの節点は等価 (二分木の部分木が同等) であると判断される。

$N_d$ は組合せ二分木において深さ $d$ の節点集合、すなわち同じアイテムが割り当てられた節点集合である。State はマス状態 0 から 13 までの集合で、Square はマス 0 から 29 までの集合である。

マス状態 $s$ が 13 のとき 18 行目でフラグ $n'$.f を立てないのは、節点 $n$ と $n'$ に割り当てられたアイテムのマスが異なるためである。また、25 行目で $n'$.nb に 3 を足すのは、

表 7: ZDD を構築するアルゴリズム

```
State ← (0, ..., 13), Square ← (0, ..., 29)
N_d ← ∅ for d = 0,...,419
根節点のメンバをすべて 0 に初期化
d ← −1, N_0 ← { 根節点 }
for each i in Square do
  for each s in State do
    d ← d + 1
    for n ∈ N_d do
      for b = 0 to 1 do
        if n から b-枝へ進んだとき
            擬到達可能な配置にならないことが確定 then
          n の b-枝の行き先を 0-葉にする
        else if d = 419 then
          n の b-枝の行き先を 1-葉にする
        else
          n′ ← 新たな節点を生成
          n′.f ← 0
          if s ≠ 13 and (n.f = 1 or b = 1) then
            n′.f ← 1
          n′.nw ← n.nw
          n′.nb ← n.nb
          if b = 1 then
            n′.nw ← n′.nw + s の白コマの数
            n′.nb ← n′.nb + s の黒コマの数
          else if s = 13 and n.f = 0 then
            n′.nb ← n′.nb + 3
          n′.topw ← n.topw
          n′.topb ← n.topb
          if b = 1 and s の一番上のコマが白コマ then
            n′.topw ← n′.topw + 1
          else if (b = 1 and s の一番上のコマが黒コマ)
              or (b = 0 and s = 13 and n.f = 0) then
            n′.topb ← n′.topb + 1
          if N_{d+1} に n′ と等価な n″ がある then
            n の b-枝の行き先を n″ にする
          else
            N_{d+1} に n′ を追加
            n の b-枝の行き先を n′ にする
```

$n'$ がマス状態 0 から 13 のどれでもなかったからである。表 7 により構築される ZDD は既約ではないため、ここからさらに圧縮規則を適用して圧縮する。等価節点の共有はすでに済んでいるので冗長節点の削除のみを適用する。

### 7.2 結果

表 7 と冗長節点の削除により既約な ZDD を構築した。メモリ約 400KB で ZDD を保持することができ、合計の節点数は葉節点 2 つを含めて 17 512 であった。また根節点の値は 147 969 899 280 であり、6 章で計算した擬到達可能な配置数と一致した。さらに、左右の対称性を考慮した同形の数は 73 986 754 080 であり、これも 6 章で計算した値と一致した。

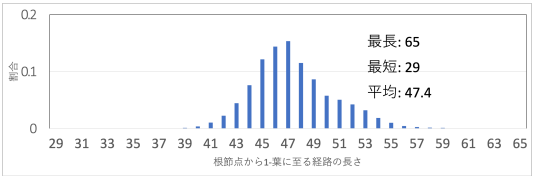



図 6: 擬到達可能なコマの配置に対応する、ZDD の経路長の分布

構築した ZDD を用いて擬到達可能な配置とそのハッシュ値の対を列挙するのに要した時間は、Intel Core 3.50 GHz のコア 1 つを使った逐次計算で約 21 時間であった。図 6 に ZDD の根節点から 1-葉までの全 147 969 899 280 の経路に関する統計情報を示す。

ZDD を構築した結果、経路長は平均 47.4 になった。これは、表 2 と表 3 のように実装されたハッシュ関数とその逆関数 (これらの計算は後退解析のボトルネックになり得る) が 47 回程度の繰り返し処理で終わることを意味する。

## 8. 後退解析の実行

本章では、後退解析のプログラムの概要と結果について述べる。なお、本章の「節点」は、コマの配置空間の節点の意である。

### 8.1 方法

本研究では、表 4 の $V_{\mathrm{w}}$、$V_{\mathrm{l}}$、$V_{\mathrm{uk}}$ を次のように実装し、ディスクに保存した。まず、擬到達可能な配置すべてを $V_{\mathrm{uk}}$ に登録する。ここで、$V_{\mathrm{uk}}$ の実装は、先行研究 [8] と同様に 1 エントリ 8 ビットの表で行い、勝敗が決定した時の反復回数をエントリの値とした。エントリの値が 0 か否かで $V_{\mathrm{uk}}$ への帰属を表現する。次に、黒プレイヤがあと 1 手で勝てる擬到達可能な配置にはこの表のエントリに 1 の値を登録した。エントリの値が奇数ならば $V_{\mathrm{w}}$ に属すと考える。偶数ならば $V_{\mathrm{l}}$ に属すと考える。なお、表のエントリの番地は、ZDD により得られる整数値により配置と 1 対 1 対応する。

計算中のメモリには、$V_{\mathrm{uk}}$、$V_{\mathrm{w}}$、$V_{\mathrm{l}}$、$V_{\mathrm{n}}$ を 1 エントリあたり 2 ビットの表で保持する。反復奇数回目で新たに値のついた節点は勝ち、偶数回目で新たに値のついた節点は負けと見なすことができ、表 4 の $V_{\mathrm{nw}}$ と $V_{\mathrm{nl}}$ を $V_{\mathrm{n}}$ にまとめることができるため、1 エントリあたり 2 ビットの表が実現する。

並列化を実装するにあたってボス・ワーカーの 2 種類のスレッドを用意した。ボスは主に表の操作を行い、ワーカーは子節点の列挙を主なタスクとする。ワーカーを複数用意することによって計算時間の短縮を試みた。ボスとワーカーの具体的なタスク内容は以下の通りである。

ボス:

- 各整数値に対して表を参照して、対応する配置の勝

表 8: 主要な結果

| | |
|---|---|
| 手番プレイヤが勝つ配置 | 106 144 078 911 |
| 手番プレイヤが負ける配置 | 41 129 930 509 |
| 引分になる配置 | 695 889 860 |
| 擬到達可能な配置 | 147 969 899 280 |
| 初期配置 | 41 手で手番プレイヤ勝ち |
| 最大手数 | 69 手 (30 配置) |
| 平均合法手数 | 23.4 |

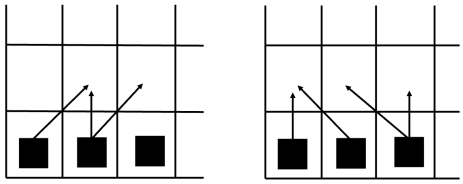


図 7: (左) 41 手で勝てる着手と (右) 43 手で勝てる着手

敗が不明であるときにワーカーに整数値を渡す。

- ワーカーから整数値と子節点集合の対を受け取り、勝敗を求めて表に書き込む。

ワーカー:

- ボスから整数値を受け取り、子節点の整数値を列挙してボスに渡す。

### 8.2 結果

表 4 に基づいて後退解析を実装した。ワーカーを 7 つ用意し、AMD EPYC 7452 32-Core Processor で約 420 時間で終了した。後退解析による主な計算結果を表 8 に示す。また、より詳細な結果を表 A·1 に示す。初期配置である図 1 は先手勝ちで、手数を伸ばそうとする後手に最短で 41 手かかることが分かった。

初期配置の先手の合法手は 21 個あり、そのうち 41 手で勝てる着手は 6 つであった (図 7)。駒を横に移動する 8 つの着手はすべて先手負けになる着手であり、それ以外の 7 つの着手は手数が伸びるものの先手の勝ちであった。また、勝ちに要する最大手数は 69 手であるという結果も得た。勝ちまでに 69 手を要する配置の例を図 8 に示す。内部節点の平均子節点数は約 23.4 であった。これは擬到達可能な配置のうち、あと 1 手で勝てる配置を除いたものの平均合法手数である。また、合法手数が 0 である配置は 30 個あり、6 章で発見した終端配置の数と一致した。

## 9. まとめ

後退解析を用いて NOCCA × NOCCA の擬到達可能な配置すべての勝敗を計算し、初期配置は 41 手で先手勝ちであることなどの結果を得た。先手有利という考えがプレイヤ上級者間に共通の認識としてあり、本研究の結果はこれを裏付けるようなものであった。現在では後手に 4 段コマ (4 段目に乗っかることができる) を 1 つ与えたルールや、盤の大きさを 6 × 7 に拡張したルールで楽しまれている。これらのルールでも同様の解決をすることができるかが今後の課題である。

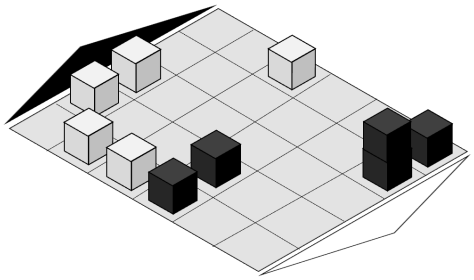


図 8: 黒番ならば、勝ちまでに 69 手を要する配置の例

### 参考文献


[1] L. V. Allis. Searching for solutions in games and artificial intelligence. *Ph.D. thesis, University of Limburg*, 1994.

[2] J. Romein. Solving the game of awari using parallel retrograde analysis. *IEEE Computer*, Vol. 36, No. 10, pp. 26–33, 2003.

[3] J. Schaeffer. Checker is solved. *Science*, Vol. 317, No. 5844, pp. 1518–1522, 2007.

[4] 諏訪壮紀. ボードゲーム「ノッカノッカ」の解析. 法政大学情報科学部コンピュータ科学科卒業論文, 2021.

[5] 武田弾. N メンズモリスのコマの配置数の分析. 研究報告ゲーム情報学 (GI), Vol. 2020-GI-43, No. 8, pp. 1–8, 2020.

[6] 田中智, ボネフランスワ, ティクシリューセバスチャン, 田村康将. Quixo の強解決. ゲームプログラミングワークショップ 2020 論文集, Vol. 2020, pp. 181–188, 11 2020.

[7] 田中哲朗. 「どうぶつしょうぎ」の完全解析. 情報処理学会研究報告, Vol. 22, pp. C1–C8, 2009.

[8] 田中哲朗. 十六むさしの強解決. ゲームプログラミングワークショップ 2020 論文集, Vol. 2020, pp. 194–201, 11 2020.

[9] 田中哲朗. ボードゲーム「シンペイ」の完全解析. 情報処理学会論文誌, Vol. 48, No. 11, pp. 3470–3476, 2007.

[10] 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.

## 付　　録

後退解析で、繰り返し処理をするたびに勝敗が決定した擬到達可能な配置数を記す。

表 A·1: 黒番で勝敗がつく手数の分布 (負ける方はできるだけ手数を伸ばす) の分布。白番でも同じ。

| 手数 | 配置数 | 手数 | 配置数 |
|---|---|---|---|
| 1 手勝ち | 77 645 562 828 | 35 手勝ち | 26 077 221 |
| 2 手負け | 22 410 730 165 | 36 手負け | 18 510 133 |
| 3 手勝ち | 15 142 536 934 | 37 手勝ち | 12 725 495 |
| 4 手負け | 7 707 358 885 | 38 手負け | 9 461 541 |
| 5 手勝ち | 2 996 378 622 | 39 手勝ち | 6 281 721 |
| 6 手負け | 2 530 587 844 | 40 手負け | 5 122 307 |
| 7 手勝ち | 1 793 971 952 | 41 手勝ち | 3 427 186 |
| 8 手負け | 1 708 892 403 | 42 手負け | 3 122 345 |
| 9 手勝ち | 1 509 313 136 | 43 手勝ち | 2 027 453 |
| 10 手負け | 1 085 543 075 | 44 手負け | 2 001 316 |
| 11 手勝ち | 1 080 382 276 | 45 手勝ち | 1 256 933 |
| 12 手負け | 855 684 894 | 46 手負け | 1 327 058 |
| 13 手勝ち | 992 892 748 | 47 手勝ち | 815 882 |
| 14 手負け | 829 104 200 | 48 手負け | 879 28 |
| 15 手勝ち | 952 307 416 | 49 手勝ち | 516 770 |
| 16 手負け | 782 437 940 | 50 手負け | 535 828 |
| 17 手勝ち | 878 640 363 | 51 手勝ち | 313 835 |
| 18 手負け | 718 653 702 | 52 手負け | 305 484 |
| 19 手勝ち | 796 900 982 | 53 手勝ち | 194 950 |
| 20 手負け | 656 328 294 | 54 手負け | 176 195 |
| 21 手勝ち | 694 050 711 | 55 手勝ち | 123 798 |
| 22 手負け | 570 638 553 | 56 手負け | 111 626 |
| 23 手勝ち | 568 580 405 | 57 手勝ち | 77 589 |
| 24 手負け | 455 202 391 | 58 手負け | 65 019 |
| 25 手勝ち | 424 155 415 | 59 手勝ち | 36 591 |
| 26 手負け | 327 678 503 | 60 手負け | 29 351 |
| 27 手勝ち | 287 199 523 | 61 手勝ち | 14 906 |
| 28 手負け | 215 552 713 | 62 手負け | 8 074 |
| 29 手勝ち | 176 144 437 | 63 手勝ち | 5 305 |
| 30 手負け | 128 061 051 | 64 手負け | 2 244 |
| 31 手勝ち | 99 613 880 | 65 手勝ち | 1 704 |
| 32 手負け | 69 681 294 | 66 手負け | 582 |
| 33 手勝ち | 51 549 796 | 67 手勝ち | 118 |
| 34 手負け | 36 136 187 | 68 手負け | 28 |
| | | 69 手勝ち | 30 |