UCTを用いたガイスターAIの研究

錦織 光司[1,a)]　青木 蓮樹[1]　橋本 剛[1]

概要：近年, 麻雀やポーカーといった不完全情報ゲームではAIが人間を超えたが，ガイスターや軍人将棋のようなチェス型の不完全情報ゲームでは強いAIが実現されていない．将棋，囲碁，チェスといった完全情報ゲームでは，人間よりも強いAIが開発されているが，完全情報ゲームの探索方法を不完全情報ゲームに用いると戦略融合の影響を受ける．本研究では，ガイスターを題材にUCTを用いた不完全情報ゲームAIの実現について議論する．既存手法では先にゲームの状態を決めて探索している．この方法をガイスターに用いると，先に相手駒の種類を決めてから探索することとなり，戦略融合の悪影響を受けやすいが，他の方法はこれまで検討されていない．そこで，先に駒種を決めて探索する方法以外に，勝敗判定に必要な最小限の駒種を決めて探索する方法と最後まで駒種を決めないで探索する方法を検討し，対戦実験により，性能を比較した．

Research of Geister AI using UCT

Nishikori Koji[1,a)]　Aoki Renju[1]　Hashimoto Tsuyoshi[1]

Abstract: In recent years, AI has surpassed humans in imperfect information games such as mahjong and poker, but strong AI has not been achieved in chess-type imperfect information games such as Geister and military chess. Although AI has been developed that is stronger than humans in complete information games such as Shogi, Igo, and chess, the use of search methods from complete information games for incomplete information games is affected by strategy fusion. In this study, we discuss the realization of AI for incomplete information games using UCT, taking Geister as a case study. Existing methods search the game by determining the game state first. However, this method is susceptible to the negative effects of strategy fusion, because the search is performed after determining the types of opponent's pieces. Therefore, we have examined the method to search for the minimum number of piece types necessary to determine the winner and the method to search without determining the piece types until the end, in addition to the method to search by determining the piece types first, and compared the performance of these methods in a game experiment.

1. はじめに

将棋，囲碁，チェスといった完全情報ゲームでは，人間よりも強いAIが開発された．不完全情報ゲームにおいてもポーカーや麻雀では人間以上の強さを持つAIが報告されている．しかし，ガイスターや軍人将棋などチェス型の不完全情報ゲームでは人間を超えるAIは報告されていない．本研究では，2017年よりAI大会が開催されており，研究が盛んに行われているガイスターのAIを題材とする．

[1] 松江工業高等専門学校
National Institute of Technology, Matsue College, Matsue, Shimane 690–8518, Japan
[a)] s2215@matsue-ct.ac.jp

ガイスターはチェス型の不完全情報ゲームである．赤駒と青駒の2種の駒を用い，相手駒の種類が観測できないという不完全情報性を持つ．勝利条件は，自分の赤駒をすべて取らせる，相手の青駒をすべて取る，自分の青駒を脱出マスから脱出させるの3種である．

ゲームAIを作成するにあたって重要な要素として，探索と評価関数がある．評価関数では，方策勾配法を用いたAI[1]がGAT2021ガイスターAI大会で優勝し，注目されている．探索では，MinMax探索を用いた方法[2]が試された．この方法は，盤上の相手駒を「紫駒」という「青駒のように脱出できるが取ると赤駒になる駒」と定義し，MinMax探索を行う．最悪のケースを想定しており，詰み

を発見するのが得意である．

ゲーム探索に用いられる探索方法として，モンテカルロ木探索 (Monte Carlo Tree Search : MCTS)[3] がある．MCTS は囲碁 AI 向けに開発された探索方法であり，ほとんどの囲碁 AI に使われている．MoGo[4] は MCTS の一種である UCT (UCB applied to trees)[5] を用いて，人間相当の強さを示した．それ以降は囲碁で MCTS が用いられる場合は UCT が用いられるようになった．AlphaGo[6] は UCT を用いて，囲碁のトッププレイヤに勝利し，人間を超える強さを示した．そして，AlphaZero[7] は，チェス，将棋，囲碁の完全情報ゲームにおいて人間を超える強さを持つ AI をさらに超える強さを示した．このように UCT は完全情報ゲームの探索において成果を上げている．一方，不完全情報ゲームでは試された例は少なく，有効性は明らかになっていない．ガイスターでは AlphaZero を用いた AI が大会参加したが結果は振るわなかった．しかし，平均的に勝ちやすい手を選択する特徴を持つ UCT がガイスターにおいて有効な可能性はある．

完全情報ゲームの探索方法を不完全情報ゲームで用いるときに起こる問題に戦略融合 (Strategy fusion)[8] がある．戦略融合は，ゲームの状態ごとの探索結果を組み合わせて戦略を求めようとしたときに，分からないはずの情報を分かっていると誤認してしまうことが戦略に影響してしまうことである．ガイスターでは MinMax 探索を行ったとき，戦略融合の影響で簡単な詰みが原理的に求まらないことがあり，詰みを求めるために紫駒探索が提案された．UCT を不完全情報ゲームで用いるとき，戦略融合の影響を緩和するため，探索木のノードをゲームの状態ではなく情報集合とした ISMCTS(Information Set Monte Carlo Tree Search)[9] が提案され，Load of the Rings，the Phantom (4, 4, 4) game，Dou Di Zhu といったゲームで試された．文献 [9] では，ISMCTS を用いるとき，情報集合の中からランダムにゲームの状態を決めて探索する方法が紹介されている．この方法をガイスターに用いると，先に駒配置を決めてから探索することとなり，戦略融合の悪影響を受けやすいが，他の方法はこれまで検討されていない．本研究では，不完全情報ゲームとして探索するために，先に駒配置を決めない方法を検討する．

2 章ではガイスターについて述べる．3 章では関連研究について述べる．4 章では戦略融合について述べる．5 章では UCT をガイスター AI に適用する際の工夫について述べる．6 章では実験について述べる．7 章では実験結果から考察を行う．8 章では GAT2022 ガイスター AI 大会での結果について述べる．9 章ではまとめを行う．

2. ガイスター

ガイスターはドイツ発祥のチェスに似た不完全情報ゲームである．ルールは簡単だが，不完全情報要素により，強い AI を作るのは難しい．2017 年から AI 大会が開催されており，2018 年からは GAT (Game AI Tournament) にて AI 大会が開催されている．大会参加者は年々増加しており，不完全情報ゲームの研究題材として注目されている．

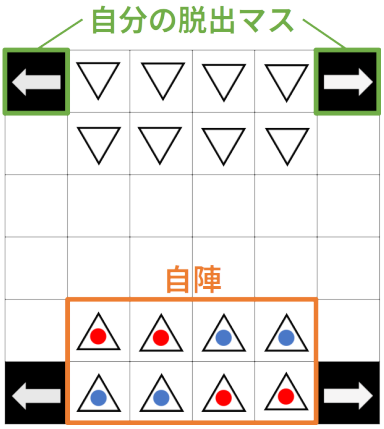

図 1 ガイスターの初期配置の一例
Fig. 1 Example of Geister’s initial placement.

2.1 ルール

ガイスターでは，各プレイヤは背中に青い印がついた良いオバケ (青駒) と，背中に赤い印がついた悪いオバケ (赤駒) の 2 種の駒を用いる．ガイスターの初期配置の一例を**図 1** に示す．はじめに，各プレイヤは赤駒，青駒をそれぞれ 4 個ずつ自陣 8 マスに自由に配置する．このとき，相手の駒の種類は観測できない．手番プレイヤは自分の駒を 1 個選択し，動かす．駒は上下左右 1 マス動け，自分の駒があるマスには移動できず，相手の駒があるマスに自分の駒を移動すると，相手の駒を取ることができる．相手の駒を取ったとき，取った駒の種類を知ることができる．勝利条件は，相手の青駒をすべて取る，自分の赤駒をすべて取らせる，自分の青駒を脱出マスから脱出させるの 3 種である．

3. 関連研究

3.1 不完全情報ゲームにおける UCT

UCT が不完全情報ゲームで用いられた例は少ない．Chinese Dark Chess で用いられている [10] が，固有のヒューリスティクスが多分に用いられており，参考にするのは難しい．多人数ゲームでは木探索をするのは難しいので木探索ではないモンテカルロ法が用いられている．須藤らは不完全情報ゲームである大貧民において，モンテカルロ法を用いた AI である「snowl」を開発し，第 4 回，第 5 回 UEC コンピュータ大貧民大会で優勝した [11][12]．このとき，モンテカルロ法を不完全情報ゲームに適用するにあたって，具体的なカードの種類をランダムに決めてから探索する方法が取られた．この方法は，大貧民 AI の共通手法となり [13]，UNO など他の不完全情報ゲームにモンテカルロ法を適用する際にも用いられている [14]．UCT をガイスター

に適用する際にも snowl と同様に行うことが考えられる．

3.2 ガイスターに関する研究

ガイスターの AI は，相手駒の色を推測する AI[15]，紫駒と MinMax 探索を用いた AI[2]，方策勾配法を用いた AI[1] などがある．相手駒の色を推測する AI は，相手駒の動きや位置から青駒らしさを評価し，指し手を決める．GAT2018 ガイスター AI 大会で優勝した．ブラフをかけられると推測をするのは難しいが有望な方法である．紫駒と MinMax 探索を用いた AI は，相手駒を紫駒という「取ったら赤，脱出時は青に変化する駒」と定義し，MinMax 探索を行う．常に最悪のケースを考えるため，必勝手順を見つけることが得意である．また，悲観的な動きをすることが特徴で，あまり相手の駒を取らない．Naotti-2020 という名前で GAT2020 ガイスター AI 大会に参加し，優勝した．方策勾配法を用いた AI は，早指し Agent という名前で GAT2021 ガイスター AI 大会に参加し，探索を行わない評価関数のみの AI にも関わらず優勝した．探索と組み合わせることでさらに強くなることが期待される．

4. 戦略融合

不完全情報ゲームに完全情報ゲームの探索方法を用いるとき，ゲームの状態ごとに探索を行い，探索結果を合計する方法が考えられる．探索結果を合計するとき，分からないはずの情報を分かっていると誤認してしまうことが探索結果に影響してしまうことがある．これを戦略融合 (Strategy fusion)[8] という．

Cowling ら [9] の示した簡単なゲームを用いて戦略融合の一例を説明する．**図 2** にゲームの状態ごとに構築した探索木を示す．探索木のノードはゲームの状態を表し，△はゲームの決定状態，○は環境状態，□は報酬である．このゲームは一人のプレイヤで行われ，ゲーム開始時にゲームの状態が x と y のどちらか一方に 1/2 の確率で決まる．プレイヤはゲームの状態がどちらか知ることはできない．プレイヤはゲーム開始時に a1 か a2 のどちらかの手を選択する．a2 を選択したとき，+0.5 の報酬を得てゲームが終了する．a1 を選択したとき，a3 か a4 を選択する．a3 を選択したとき，ゲームの状態が x なら -1，y なら +1 の報酬を得て終了する．a4 を選択したとき，ゲームの状態が x なら +1，y なら -1 の報酬を得て終了する．

図 2 の木で探索した場合，ゲームの状態が x のときは a4 を，y のときは a3 を選択すると，どちらのゲームの状態でも+1 の報酬を得られるため a1 を選択する．しかし，実際はゲームの状態が分からないため，確実に +1 を得ることはできない．このように戦略融合の影響を受ける．

図 3 はゲームの状態が分からないことを考慮し，木のノードをゲームの状態ではなく情報集合とした探索木である．図 3 の探索木では△のノードは情報集合を表す．a3 を選択すると 1/2 の確率で +1 の報酬を得て，1/2 の確率で -1 の報酬を得るため，期待報酬は 0 となる．a4 を選択した場合も同様であり，a1 を選択したときの期待報酬は 0 となる．よって，確実に +0.5 の報酬を得られる a2 を選択する．このようにゲームの状態ではなく情報集合を用いた探索木を用いることで戦略融合の効果を緩和することができる．

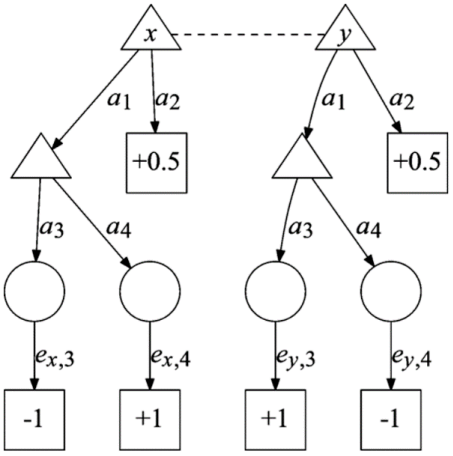

図 2 ノードにゲームの状態を用いた探索木 [9]
Fig. 2 Search tree using game state for nodes.[9]

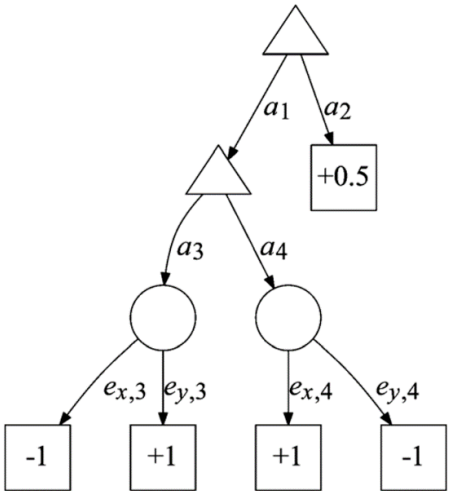

図 3 ノードに情報集合を用いた探索木 [9]
Fig. 3 Search tree using information set for nodes.[9]

戦略融合の影響を緩和するため，探索木のノードをゲームの状態ではなく，情報集合とした MCTS を ISMCTS (Information Set Monte Carlo Tree Search)[9] という．

4.1 駒配置全列挙

ガイスターでの戦略融合の例を示す．川上らは**図 4** のように，ガイスターにおいてあり得る駒配置をすべて列挙し，MinMax 探索を行う「駒配置全列挙 (Enumerating All Color Placements: EACP)」を行った [2]．すべての駒配置を考慮した探索ができており，最善の手を選択できるよう

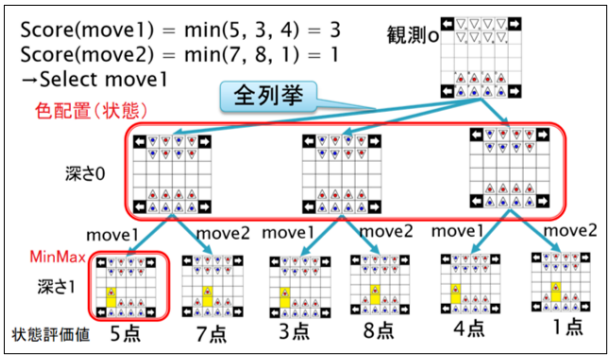

図 4 色配置全列挙の例 [2]
Fig. 4 Example of the EACP.[2]

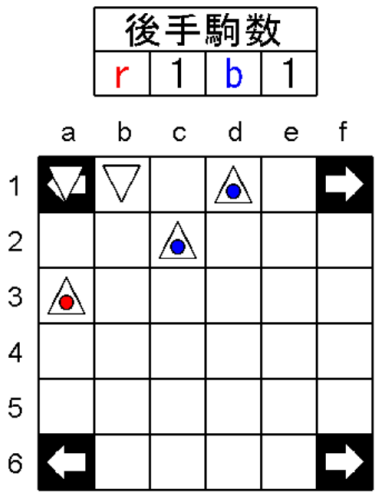

図 5 色配置全列挙がうまくいかない局面 [2]
Fig. 5 Position where the EACP does not work.[2]

に思える．しかし，駒配置全列挙では原理的に必勝手を正しく求められないことがある．具体例を**図 5** に示す．先手が a3 ↑と a2 に駒を移動したときの場合を考えると，相手の b1 の駒が赤駒のときは a1 の青駒を取れば先手勝ち，相手の b1 の駒が青駒のときは b1 の駒を取れば先手勝ちであり，3 手で先手勝ちとし，a3 ↑を最善手と選択する．しかし，実際は相手の青駒が a1，b1 のどちらにあるか不明なため，3 手で勝つことはできない．実際は d1 →と動かし青駒を脱出させて 5 手で先手勝ちである．このように MinMax 探索を単純にガイスターに適用すると簡単な必勝手すら発見できないことがある．

5. UCT のガイスターへの適用

完全情報ゲームの探索方法である UCT を不完全情報ゲームであるガイスターに適用するとき，ISMCTS のようにノードを情報集合とすることが考えられる．文献 [9] では snowl で用いられた方法と同様に，ゲームの状態をランダムに決定してから探索する方法が紹介されている．ガイスターに適用すると，ランダムに駒配置を決定してから探索する方法 (A) となる．A の方法を**図 6** に示す．A の方法はランダムに決めた駒配置から完全情報ゲームとして探索したこととなる．この方法では戦略融合の悪影響を受ける可能性がある．この理由から不完全情報ゲームとして探索するために，先に駒配置を決めずに最低限駒の種類を決める方法 (B) と最後まで駒の種類を決めない方法 (C) を提案する．

ガイスターは駒の種類によって動きが変わらないため，駒の種類を後から決めても問題なく探索できる．B の方法を**図 7** に示す．B の方法は相手の駒の種類を決めずに探索を開始し，相手の駒を取ったときや，相手の駒が脱出マスに乗ったときに，その駒の種類のみを確率的に決定する．例えば**図 8** のような局面がプレイアウト中で現れたとき，取る駒は $\frac{2}{5}$ の確率で赤駒，$\frac{3}{5}$ の確率で青駒とする．勝敗判定に必要な駒以外決めずに評価できる．しかし，B の方法は駒の種類を一部決めるため，完全に不完全情報ゲームとして探索した事にはならない．

一切駒の種類を決めなくても局面を評価することが可能である．C の方法を**図 9** に示す．C の方法は相手駒の種類を決めずに探索を開始し，自分目線で勝敗が決まった時に評価をする．自分の赤駒をすべて取らせたとき，自分の青駒が脱出したときなど，自分目線で勝利したとき，探索の過程で種類の分からない相手駒が脱出マスに乗ったり，種類の分からない相手駒を取ったりするため，すでに負けている可能性がある．これを考慮するため，勝敗ではなく自分が勝っている確率 (自分勝利確率) を用いて評価する．自分勝利確率は式 (1) で計算する．

$$D = E \times (1 - F) \tag{1}$$

ここで，D は自分勝利確率，E は相手駒が脱出マスに乗った時に赤駒である確率を累積したもの，F は相手の赤駒をすべて取っている確率である．

F は取った相手駒の数が 4 個未満の時は 0，4 個以上の時は以下の式 (2) で計算する．

$$F = \frac{{}_{Ub}C_{Lb}}{{}_{Uk}C_{Gk}} \tag{2}$$

ここで，Ub は駒の種類が決まっていない相手青駒の数，Lb は赤駒を 4 個取ったときの取った青駒の数，Uk は駒の種類が決まっていない相手駒の数，Gk は取った相手駒の数である．

例えば図 9 のシミュレーション結果の局面で，プレイアウト中に相手駒が脱出マスに乗り，その駒が $\frac{4}{8}$ の確率で赤駒なら，式 (3) のように計算する．

$$\begin{aligned} F &= \frac{4}{8} \times (1 - \frac{{}_4C_1}{{}_8C_5}) \\ &= \frac{13}{28} \end{aligned} \tag{3}$$

駒の種類を決めずに評価できるため，不完全情報ゲームとして探索したこととなる．

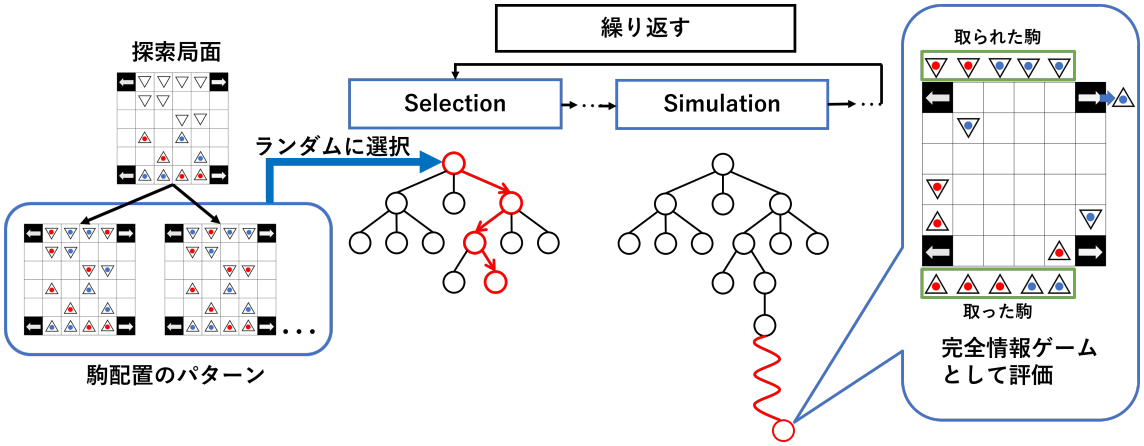

図 6 A の方法
Fig. 6 Method A.

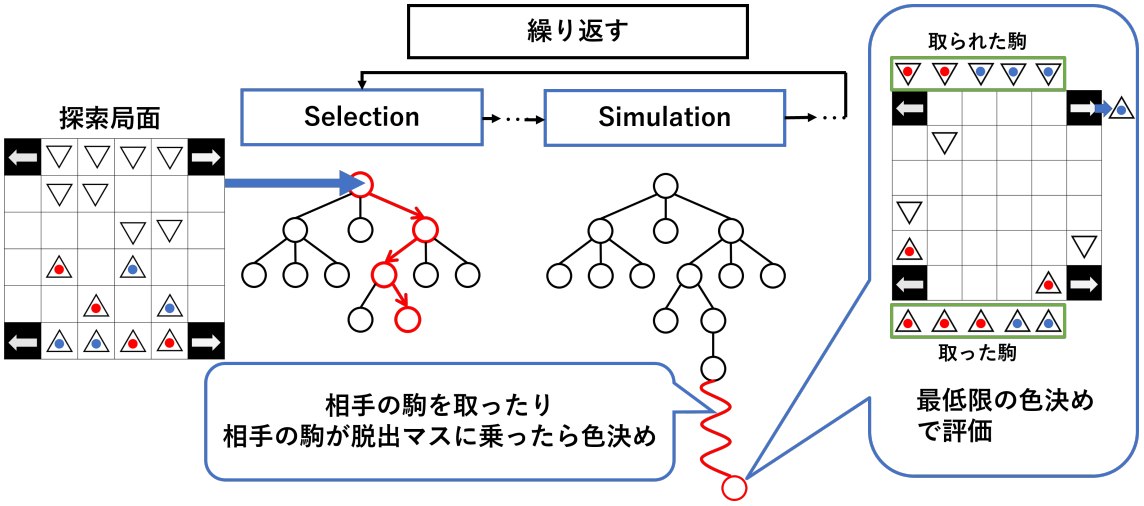

図 7 B の方法
Fig. 7 Method B.

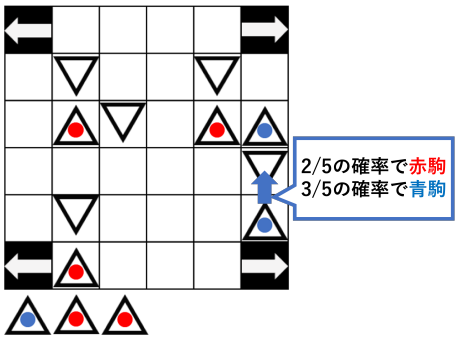

図 8 B の具体例
Fig. 8 Example of method B.

6. 実験

6.1 実装

実験に用いるガイスター AI は，UCT 部分，ガイスターのゲーム部分，ガイスターサーバとの通信部分の 3 部分で構成し，それぞれ実装を行った．UCT 部分に関しては，青木が AI(A) を実装した．AI(A) を元に AI(B)，AI(C) を実装した．ガイスターのゲーム部分と，ガイスターサーバとの通信部分は Naotti-2020 を参考に実装した．

6.2 実験方法

A, B, C の方法を用いた UCT のガイスター AI を用いて，総当たり戦を先手後手それぞれ 500 戦ずつ行った．また，GAT2020 ガイスター AI 大会優勝 AI である Naotti-2020 との対戦と GAT2021 ガイスター AI 大会優勝 AI である早指し Agent との対戦を先手後手それぞれ 50 戦ずつ行った．ここで，UCT のプログラムは 30000 回のプレイアウトで探索を打ち切るようにし，Naotti-2020 は探索深さを 6，早指し Agent は 500 万回学習したものを利用した．そして，それぞれの対戦棋譜から UCT の AI の特徴を調べた．

6.3 実験結果

AI(A)，AI(B)，AI(C) の総当たり戦の結果を**表 1** に示す．AI(C) は AI(A)，AI(B) に負け越した．AI(A)，AI(B) は互角だった．

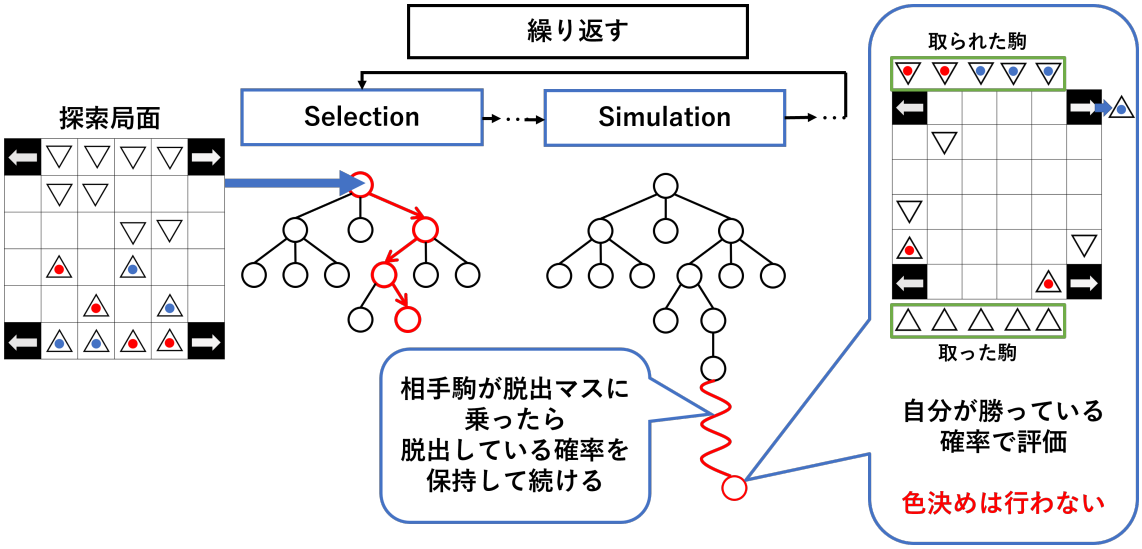

図 9 C の方法
Fig. 9 Method C.

表 1 AI(A)，AI(B)，AI(C) の総当たり戦の結果
Table 1 AI(A), AI(B), AI(C) round-robin tournament results.

先手 \ 後手	A	B	C
A	—	252 勝 248 敗	325 勝 175 敗
B	249 勝 250 敗 1 分	—	319 勝 179 敗
C	169 勝 331 敗	173 勝 327 敗	—

表 2 UCT と Naotti-2020 の対戦結果
Table 2 Results of UCT vs. Naotti-2020.

UCT \ 手番	先手	後手
A	10 勝 39 敗 1 分	11 勝 39 敗
B	9 勝 41 敗	7 勝 43 敗
C	2 勝 48 敗	3 勝 47 敗

表 3 UCT と早指し Agent の対戦結果
Table 3 Results of UCT vs. HayazashiAgent.

UCT \ 手番	先手	後手
A	22 勝 28 敗	19 勝 31 敗
B	15 勝 35 敗	18 勝 31 敗 1 分
C	5 勝 45 敗	13 勝 36 敗 1 分

UCT の AI と Naotti-2020 との対戦結果を**表 2** に示す．AI(A)，AI(B)，AI(C) すべて大きく負け越したが全く勝てないわけではない．UCT の AI が勝つときは青駒が脱出して勝利することがほとんどだった．

UCT の AI と早指し Agent との対戦結果を**表 3** に示す．AI(A)，AI(B) は負け越したものの 3 ～ 4 割勝った．AI(C) は大きく負け越した．

図 10 は AI(A) と早指し Agent との対戦で現れた局面である．AI(A) が赤駒を青矢印のように動かし，3 個の相手駒に隣接させ，相手に取らせることで勝利した．

図 11 は AI(B) と Naotti-2020 との対戦で現れた局面である．AI(B) が青駒を青矢印のように動かし，青駒を取られないように逃げつつ，脱出マスに近づけた．

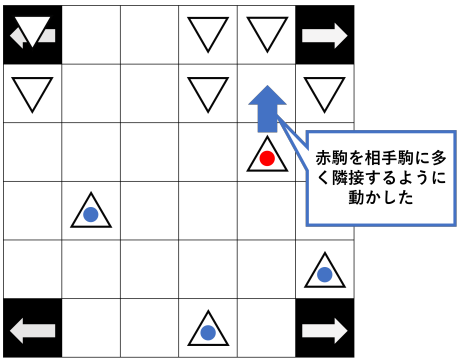

図 10 AI(A) が赤駒を相手駒にぶつけた局面
Fig. 10 A position that AI(A) hit a red piece against an opponent's piece.

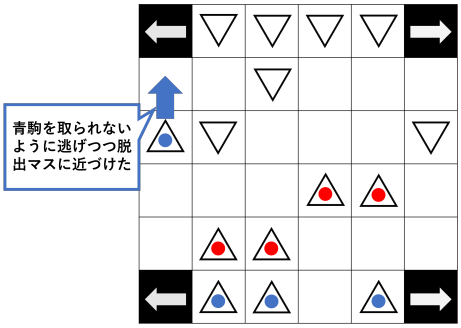

図 11 AI(B) が青駒を逃げた局面
Fig. 11 A position that AI(B) escaping a blue piece.

図 12 は AI(B) と AI(C) の対戦で現れた局面である．AI(B) が緑矢印のように駒を取れば 5 手で脱出して勝ちである．この手を逃すと黄矢印のように相手に駒を取られて阻止される．実際は脱出による勝ちを逃し，赤矢印のように青駒を動かした．

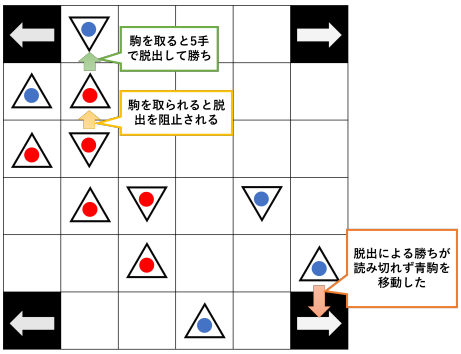

図 12 AI(C) が詰みを逃した局面
Fig. 12 A position that AI(C) missed a win.

7. 考察

AI(A)，AI(B)，AI(C) を比較すると動きの特徴に違いは発見できなかったが，総当たり戦の結果から AI(A) と AI(B) は互角，AI(C) が弱い印象だ．UCT の平均的に勝ちやすい手を選択する特徴が，相手の駒に隣接するように赤駒をぶつける動きや，青駒を相手に取られないように逃げる動きにつながった．赤駒をぶつける動きは早指し Agent に通用し，勝利することがあった．しかし，Naotti-2020 には通用せず，赤駒を取らせて勝つことはできなかった．UCT の AI は Naotti-2020 の相手駒をあまり取らない特徴と相性が悪いと考えられる．特に，AI(C) は勝率が低く，AI(A)，AI(B) に比べて脱出することを重視していない可能性がある．また，5 手程度の簡単な詰みが読めないことがあり，必勝の局面から負けてしまうことがあった．Naotti-2020 のような詰み探索の強い AI と組み合わせることで強くなると考えられる．詰み探索が弱い状態で前大会優勝 AI である早指し Agent に善戦できたことから，UCT はガイスターにおける有効な探索方法の一つであると考える．

8. GAT2022 ガイスター AI 大会

AI(B) に紫駒を組み合わせた AI「Nissy」でガイスター AI 大会に参加した．Nissy は紫駒に詰み探索を任せ，詰み以外のときの手選択は UCT が行うことで，UCT の詰みが読めない弱点を解決したものである．結果は 9 チーム中 3 位と成績は良かったが，大会 1 位と 2 位の AI には一度も勝てず，相性が悪いと考えられる．特に大会 1 位の Naotti-2022 は赤駒，青駒をどちらもゴールに進めていく戦略をとるが，UCT はゴール付近の駒を取りやすいので勝つのが難しかった．

9. まとめ

本研究では，不完全情報ゲームであるガイスターに関して，完全情報ゲームで成果を挙げている UCT を適用した．UCT を用いた AI は，GAT2021 ガイスター AI 大会で優勝した最強の AI の一つである早指し Agent に負け越したものの 3 ～ 4 割勝利することができた．ただし，Naotti-2020 のように相性不利な相手がいることが課題である．また，本研究の AI は，自駒の初期配置をランダムに決めたが，勝ちやすい初期配置がある可能性があり，考慮したほうがよいだろう．

謝辞

本研究で対戦させていただいた Naotti-2020 を開発した川上直人さん，早指し Agent を開発した志賀薫さん，ガイスターサーバを開発した三好健文さんに感謝いたします．

参考文献

[1] 志賀薫, 伊藤毅志: 自己対戦を用いたガイスター AI における行動優先度の学習, 情報処理学会研究報告, Vol.2021-GI-46, No.16, pp.1-7, 2021.
[2] 川上 直人, 橋本 剛: 『紫駒』を用いた MinMax 探索によるガイスター AI の研究, 情報処理学会論文誌, Vol.62, No.10, pp.1706-1723 (2021).
[3] Rémi Coulom: Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search, In 5th International Conference on Computers and Games, pp.72-83 (2006).
[4] Sylvain Gelly, Yizao Wang, Rémi Munos, Olivier Teytaud: Modification of UCT with Patterns in Monte-Carlo Go. [Research Report] RR-6062, INRIA. 2006 ffinria-00117266v3f (2006).
[5] Levente Kocsis, Csaba Szepesvari: Bandit Based Monte-Carlo Planning, In Proceedings of the 15th European Conference on Machine Learning 2006, Lecture Notes in Computer Science, Vol.4212, pp.282-293 (2006).
[6] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, Demis Hassabis: Mastering the Game of Go without Human Knowledge, Nature, Vol.550, pp.354–359 (2017).
[7] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, Demis Hassabis: A general reinforcement learning algorithm that masters chess, shogi and Go through self-play, Science, Vol.362(6419), pp.1140–1144 (2018).
[8] Ian Frank , David Basin: Search in games with incomplete information: A case study using Bridge card play, Artificial Intelligence, Vol.100, No.1-2, pp.87–123, (1998).
[9] Peter I. Cowling, Edward J. Powley, Daniel Whitehouse: Information set Monte Carlo tree search, IEEE Transactions on Computational Intelligence and AI in Games, Vol.4, No.2, pp.120-143 (2012).
[10] Shi-Jim Yen, Cheng-Wei Chou, Jr-Chang Chen, I-Chen Wu, Kuo-Yuan Kao: Design and Implementation of Chinese Dark Chess Programs, IEEE Trans. Computational Intelligence and AI in Games, Vol.7, No.1, pp.66–74 (2015).
[11] 須藤 郁弥, 篠原 歩: モンテカルロ法を用いたコンピュータ大貧民の思考ルーチン設計, 第 1 回 UEC コンピュータ大貧民シンポジウム (2009).

[12] 須藤 郁弥, 成澤 和志, 篠原 歩: UEC コンピュータ大貧民大会向けクライアント「snowl」の開発, 第 2 回 UEC コンピュータ大貧民シンポジウム (2010).
[13] 大渡 勝己, 田中 哲郎: 方策勾配を用いた教師有り学習によるコンピュータ大貧民の方策関数の学習とモンテカルロシミュレーションへの利用, 情報処理学会研究報告, Vol.2016-GI-35, No.10, pp.1-8, 2016.
[14] 松岡 確, 堀内 研, 中山 泰一: 不完全情報ゲーム UNO のモンテカルロ法による解法, 情報処理学会第 76 回全国大会公演論文集, pp.605-606 (2014).
[15] 末續鴻輝, 織田祐輔: 機械学習を用いないガイスターの行動アルゴリズム開発, GAT2018 論文集, pp.13 − 16 (2018).

