進化的計算を用いたモンテカルロガイスターのプレイアウトの改善

令和 3 年度

修士学位論文

進化的計算を用いたモンテカルロガイスターのプレイアウトの改善

Improving Playout of Monte Carlo Geister Using Evolutionary Computation

1245128　　栃川純平

指導教員　　竹内聖悟

2022 年 2 月 28 日

高知工科大学大学院 工学研究科 基盤工学専攻

情報学コース

要 旨

進化的計算を用いたモンテカルロガイスターのプレイアウトの改善

栃川純平

ガイスターは，相手の駒の色がわからない二人不完全情報ゲームである．ガイスターにはモンテカルロベースのプレイヤーがあり，プレイアウトでランダム方策を行っている．プレイアウト中はランダム方策よりも人間の知識を用いたルールベース方策の方が強いとされているが，知識の導入にはゲームに習熟する必要がある．そのため，習熟度によらず方策を作成できると望ましい．Ms.Pac-Man では遺伝的プログラミングを用いてルールベース方策を作成する研究があり，平均スコアが向上したという結果が得られている．しかし，不完全情報ゲームでのプレイアウトでは，相手の状態が正しいとは限らず，どのようなルールが適切か不明である．そのため，適切なルールの特徴を調べる必要がある．

本研究では，モンテカルロガイスターのプレイアウトに用いるルールベース方策を GP で作成を行ったところ，適応度が一致率の場合で性能が良いものが確認できた．作成された方策から，勝敗に直接関係するルールほど重要であり,「青駒を脱出口へ向かわせる」のように自分からみて駒色が正しいルールの方がより重要であることがわかった．また，ルールベース方策の指手選択を確率的にし，GA によってパラメータ調整を行ったところ，調整前に比べ性能の向上はみられなかった．さらに，勝率と適応度の相関係数を調べたところ，不完全情報での一致率を適応度とした場合に正の相関があることがわかった．

キーワード　ガイスター，進化的計算，プレイアウト方策，モンテカルロ木探索

Abstract

Improving Playout of Monte Carlo Geister Using Evolutionary Computation

Jumpei Tochikawa

Geister is a two-player imperfect information game in which the color of the opponent's pieces is unknown. In Geister, the Monte Carlo player use a random policy in playout. During playout, the rule-based policy using human knowledge is considered stronger than the random policy. However, we need to be proficient in the game to introduce human knowledge. Therefore, it is desirable to be able to create policies regardless of proficiency level. There is research on using genetic programming to create rule-based policy in Ms. Pac-Man. In this research, the average score improved. However, the opponent's state is not always correct in playout of imperfect information games, and it is unclear what rules are appropriate. Therefore, it is necessary to investigate the characteristics of appropriate rules.

In this study, we used GP to create rule-based policy for the playout of Monte Carlo Geister, and found that the performance was better when the fitness was the rate of agreement. From policies created, it was found that the rules directly related to win or lose were more important, and the rules in which the piece color was correct from the player's point of view, such as "Let the blue piece go to the escape door," were more important. In addition, when the move selection of the rule-based policy was made probabilistic and the parameters were adjusted by GA, there was no improvement in performance compared to before the adjustment. Furthermore, we examined the

correlation coefficient between the win rate and the fitness, and it was found that there was a positive correlation when the rate of agreement with imperfect information was used as the fitness.

key words geister, evolutionary computation, playout policy, Monte Carlo tree search

目次

図目次

表目次

第 1 章

はじめに

ゲームは，ルールが明確で評価がしやすく，また興味を持ちやすいという特徴があることから情報処理の例題として用いられてきた．チェス，将棋や囲碁のような二人完全情報ゲームやバックギャモンやポーカーのような不完全情報ゲームというように，ゲームには情報処理的に様々な性質を持つものが多く存在している．ゲーム AI を強くするために，これまで様々な研究が行われてきており，完全情報ゲームであるチェスにおいて人間のトッププレイヤーの実力を上回るだけでなく，チェスよりも問題空間が大きい将棋，囲碁においてもすでに人間のトッププレイヤーの実力を上回っている．一方で，不完全情報ゲームにおいてはバックギャモン，ポーカーでトッププレイヤーに相当する実力を持つ AI が研究されている [1] ものの，まだ弱いものが多い状態である．

不完全情報ゲームは，盤面に見えない情報があるゲームのことをいう．このようなゲームでは，不確定な情報があり正確な探索が困難なことから，プレイアウトを用いるモンテカルロベースの探索を使用することが多い．プレイアウトとは，ゲームを終局までシミュレーションすることである．また，不確定な情報を推定することで有効な意思決定を可能にするとされている．不完全情報ゲームでは，現在の盤面を可能とする局面の中からランダムに状態を仮定し，その局面に対してプレイアウトを行う．しかし，局面を仮定しているということから，自分の駒や手札などの情報が正確である一方で，相手の駒や手札などの情報は正確であるとは限らず，プレイヤーごとに情報の正確性に差が生じてしまうという問題点がある．不完全情報ゲームのプレイアウトベースのプレイヤは，指手をランダムに選択するランダム方策が多い．ランダム方策では，実際のゲームでは選ばれない指手が選ばれやすくなるとされている．囲碁や General Game Playing などでは人間の知識を使用したルールベー

ス方策にすることで，上記のような指手の数を減らすということを行っている [2][3]．そして，ルールベース方策はプレイアウトを決定的にすることでランダム性が失われ弱くなるとされており，ランダム性を入れるために確率の導入などがされている．しかし，不完全情報ゲームはプレイアウト時相手の状態が正確とは限らないため，どのようなルールが適切か不明である．そのため，適切なルールの特徴を調べる必要がある．

人間の知識を導入するためには，人間がゲームに習熟している必要がある．しかし，ゲームに習熟するには多くの時間がかかってしまう．そこで，人間の知識によらずルールベース方策を作成することができると望ましい．Ms.Pac-Man では遺伝的プログラミング (Genetic Programming，以降では GP) を用いてルールベース方策を作成する研究 [4] がされている．この研究では，作成した方策がランダム方策のものに比べて，平均スコアが 18% 向上したという結果が得られている．

ガイスターは，相手駒が見えない二人不完全情報ゲームである．ガイスターには，機械学習を用いないプレイヤー [5] やミニマックス法を用いたプレイヤー [6]，モンテカルロベースのプレイヤー [7][8] など様々なものがある．しかし，ガイスターにおけるモンテカルロベースのプレイヤーは．ランダム方策を行っている．また，簡単なルールベース方策で対戦を行ったところ弱くなってしまった．そこで本研究では，二人不完全情報ゲームであるガイスターを対象とし，モンテカルロベースのプレイヤーの方策をルールベースにしプレイアウトの改善を目指し，性能の違いによるルールの違いによる特徴を調査する．ルールベース方策の作成には，進化的計算を使用する．進化的計算には適切な適応度を使用する必要があるため，用いる適応度についても調査する．

第 2 章

ガイスター

ガイスターは，Alex Randolph が作成した駒移動型の二人不完全情報ゲームである．このゲームは，1982 年にドイツ年間ボードゲーム大賞にノミネートされている．

ガイスターは，互いのプレイヤーが青駒と赤駒を各 4 個ずつ計 8 個の駒を用いて行うゲームである．特徴は，各プレイヤーは自身の駒の色は確認することが可能であるが相手の駒の色を確認できないことである．ゲームは，6×6 の盤面で行う．ゲーム開始前に各プレイヤーは決められた場所に 8 個の駒を自由に配置し，交互に駒を動かしてゲームを進める．各プレイヤーは自分の手番に自身の駒 1 つを上下左右に 1 マス動かすことができる．移動先のマスが自身の駒があるマスもしくは盤外であった場合その方向へ移動することはできないが，相手の駒があるマスであった場合その駒を取ることができる．このとき，取った相手の駒色を確認することができる．ゲームを進めていきどちらかのプレイヤーが勝利条件のいずれかを満たしたらゲーム終了である．勝利条件は以下の 3 つである．

- 自分の青駒を脱出させる
- 相手の青駒を全て取る
- 自分の赤駒を全て取らせる

脱出させるとは，青駒を脱出口まで進んでから次の自分の番に脱出を宣言することである．脱出口は，盤面の四隅のマスであり，図 2.1 では下部分を自陣としたとき，脱出口は上部分の隅のマスのことを表している．

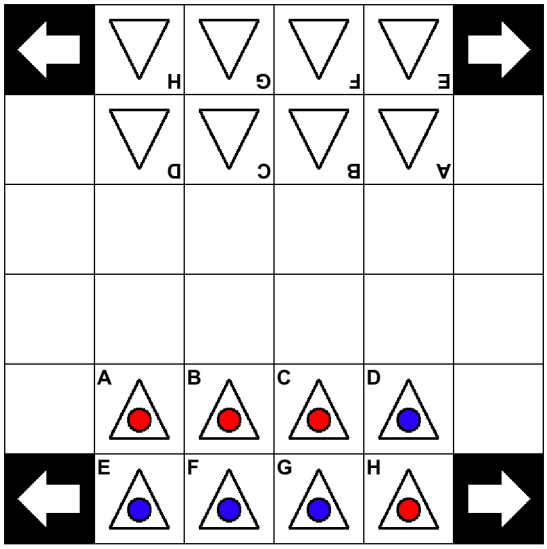

図 2.1　ガイスターの盤面

第 3 章

関連研究

3.1 モンテカルロ木探索 (MCTS)

モンテカルロ木探索 (以降では MCTS) は，2000 年後半に登場した手法である．これは，終局までプレイアウトを行い最も勝率の良いものを選ぶモンテカルロ法と木探索を組み合わせた手法である．単純なモンテカルロ法には，悪い手であってもランダムシミュレーションが均等に行われることと平均勝率を使用するためランダムシミュレーション内の最善手と最悪手の影響をうまく反映できないという問題点がある [9]．一方，MCTS は木探索と組み合わせることにより可能性の高い着手に計算資源をより多く割り振ることができるようになっている．MCTS の基本的な動きを図 3.1 を用いて以下で説明する．

1. 根節点から着手される可能性の高い節点を選び，末端まで辿って行く．節点を選ぶとき，自分のときは最も良い節点を選択し，相手のときは最も悪い節点を選択する
2. 末端の節点のプレイアウト回数が閾値を超えていれば，節点を展開しさらに木を下りる
3. 下りてきた節点でプレイアウトを行う
4. プレイアウト結果に応じて辿ってきた経路上の節点を更新する
5. 1〜4 を決められた時間または回数行う

木を下りて行くときの節点の選択の指標として様々なものがあるが，よく使われるのは UCB1 値と呼ばれるものである．UCB1 値は，以下の式で求められる．

$$UCB1 = (\text{その手の勝率}) + C\sqrt{\frac{2\log(\text{シミュレーションの総数})}{(\text{その手のシミュレーション回数})}}$$

C は定数であり，問題に合わせて調整するものである．ゲーム研究においては基本的に報酬が勝ちか負けで範囲が [0,1] の範囲であるため理論的には $C = 1$ でよい [**?**]．UCB1 値を用いたアルゴリズムが UCT (UCB applied to Tree) と呼ばれている．

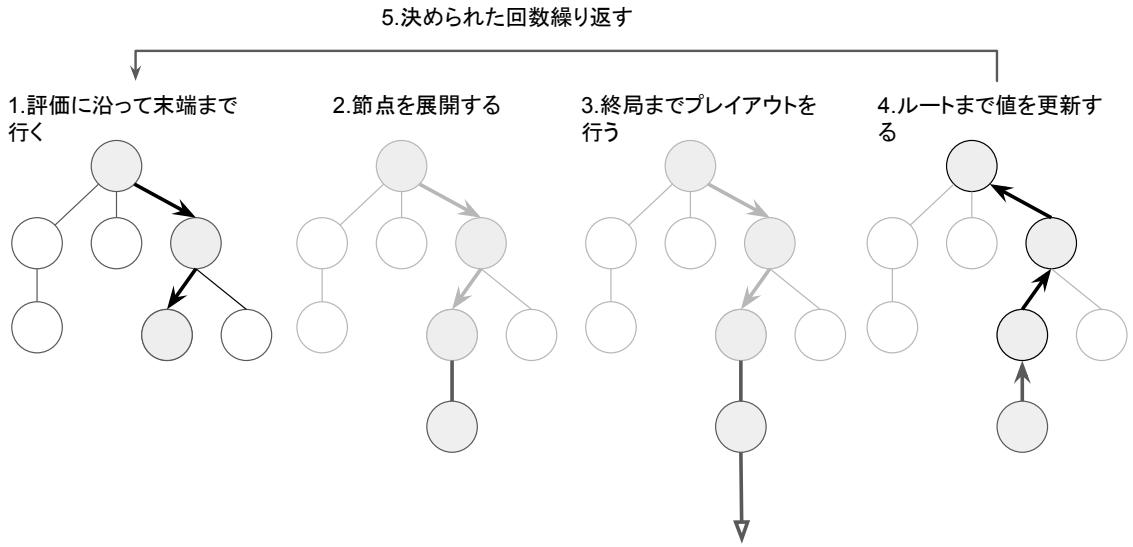

図 3.1　MCTS の動き

3.2　不完全情報ゲームにおける MCTS

不完全情報ゲームにおける MCTS は，見えない情報があり試合終了の判定ができないため，そのまま行うことは難しい．そのため，不完全情報ゲームで行う場合，現在の局面から実現可能な局面集合から局面を仮定して MCTS を行う．単純な方法として図 3.2 のようにプレイアウトを行うたびに局面を仮定し直す方法がある．図 3.2 では，まず局面 $s1$ に仮定してプレイアウトを行い，別の局面 $s5$ に仮定して次のプレイアウトを行っている．これは，同じ探索木に対して仮定を行ってプレイアウトを行っているため，仮定した局面によってプレイアウト結果が変わってしまい探索に影響を与える可能性がある．本研究では，この単純な方法を用いて MCTS を行う．

別の方法の中で，図 3.3 のようにいくつかの仮定した局面を用意し，その局面に対して独

立した探索木を生成するという方法である [10]．この方法では，それぞれの局面でそれぞれプレイアウトを行い，それぞれの結果を合わせて指手の決定を行う．この方法は，それぞれ独立した木でプレイアウトを行っているため，図 3.2 の方法とは違い局面の仮定による探索の影響が抑えられる．

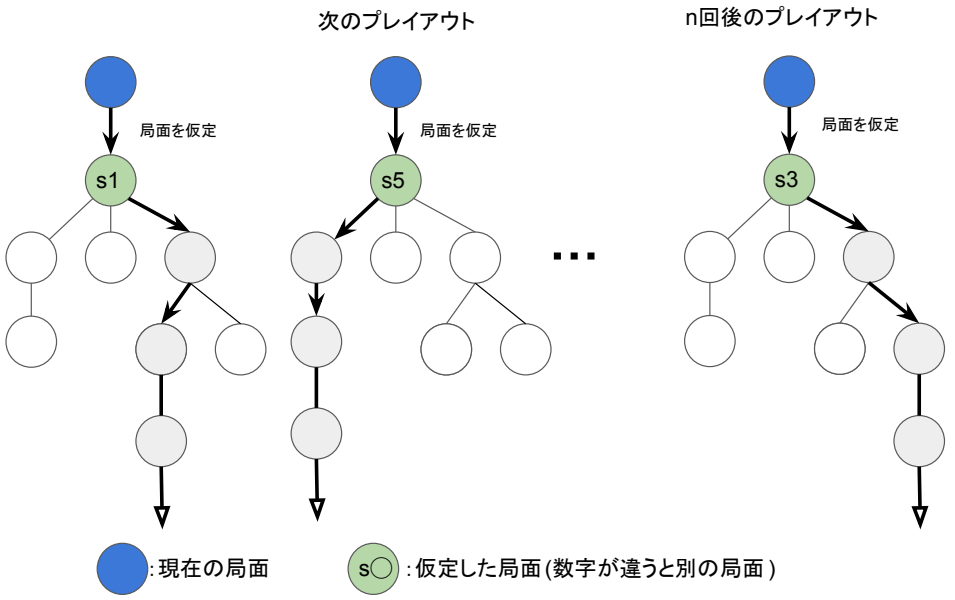

図 3.2　不完全情報ゲームにおける MCTS：毎回仮定し直す場合

別の方法として，Information Set MCTS(以降では ISMCTS) というものがある [10]．Information Set とは，特定のプレイヤーが持っている知識から考えられるゲームの状態の集合のことを表している．これは前述した，現在の局面から実現可能な局面集合と同義である．ゲームによっては，仮定した局面によって指せる指手の数の変化，相手の指手が見えないといった様々な要素があり，それらに対応したアルゴリズムが提案されている．

3.3　モンテカルロベースのガイスタープレイヤー

モンテカルロベースのガイスタープレイヤーとして三塩らのプレイヤー [7] や鴛淵らのプレイヤー [8] がある．これらのプレイヤーは，どちらも推定を行っており，推定した内容は相手駒の配置にのみ使用されている．プレイアウトはランダムプレイアウトを行っている．

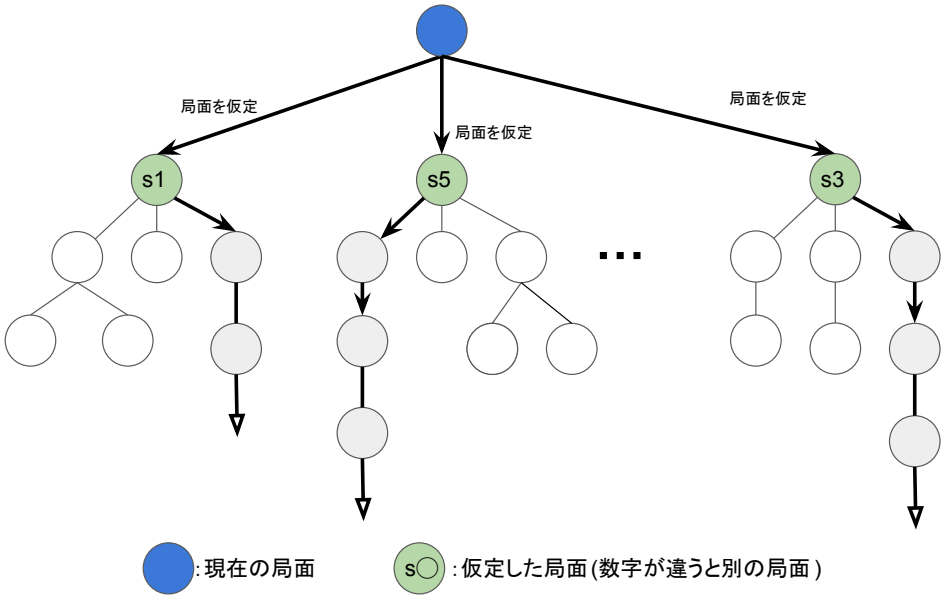

図 3.3　不完全情報ゲームにおける MCTS：独立した木で行う場合

ランダムプレイアウトでは，指手がランダムに選択されるため実際には指さないような手を指すことが多くなってしまう．

3.4　進化的計算

進化的計算とは，生物の進化メカニズムに基づきデータ構造の変形，合成，選択を行う手法である．代表的なものとして，遺伝的アルゴリズム (Genetic Algorithms，以降では GA)，進化的戦略 (Evolution Strategy)，進化的プログラミング (Evolutionary Programming)，遺伝的プログラミング (GP) がある．

3.4.1　遺伝的アルゴリズム (GA)

GA は，生物の遺伝と進化のモデルを利用したアルゴリズムであり，最適化問題などに使われている [11]．GA は，基本的に選択，交叉，突然変異の 3 種類の遺伝子操作を使用する．GA の処理手順を以下に示す．

1. 初期個体の集合を生成
2. 各個体の適応度を評価
3. 適応度に応じて個体を親として選択
4. 突然変異や交叉などを交えながら，次の世代を生成
5. (1)〜(4) を世代数繰り返す

初期個体の集団は，一般的にランダムに生成する．親の選択方法は，基本的に適応度が高いものを選択するが，期待値戦略，ランク戦略，エリート保存戦略など様々な選択方法がある [11]．GA では突然変異を行うことで，集団中の遺伝子に多様性を持たせることでできるだけ広い空間を探索できるようにしている．

GA を実装する上では以下を設計する必要がある．

- 交叉や突然変異を行う部分
- 選択方法
- 適応度の計算

3.4.2　遺伝的プログラミング (GP)

GP は，GA を拡張したものの 1 つである．GP とは，関数の集合を用意しておき，その関数の組み合わせにより与えられた現象を説明できるプログラムを求めることである [12]．GP では遺伝子を木構造で表現しており，ランダムに個体の集合を生成し評価を行い，その中で最もよい個体を親として次の世代を作成していく．本研究では，プレイアウトの方策を個体としている．次の世代を生成するときに，突然変異や交叉など遺伝的操作を適応することでプログラムを進化させていく．

進化の過程は GA と同じ順序で行う．GP を行う際は，GA で設計するものに加え，関数集合を定義しておく必要がある．本研究で作成する木構造の簡単な if 文の例を図 3.4 に示す．GP では，図 3.4 のような木構造で個体を表現している．この木構造を表現するにあたり，3 つの引数を取る関数 if を定義し，第一引数に図 3.4 の左のノード (条件式)，第二引数

に図 3.4 の真ん中のノード (true のときの行動)，第三引数に図 3.4 の右のノード (false のときの行動) を取るように設計する．

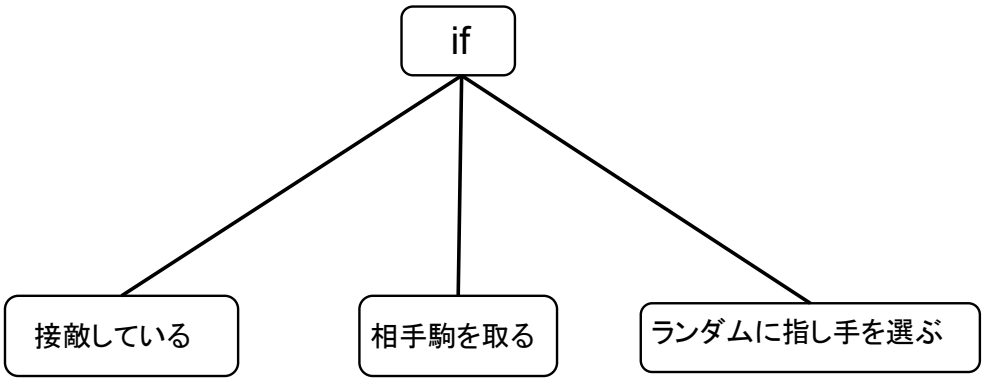

図 3.4　作成する木構造の例

3.5　GP を用いたルールベース方策の作成の研究

GP を用いプレイアウトで使用するルールベースの方策を作成する研究として，Ms.Pac-Man を題材にしたもの [4] がある．この研究では，GP を用いてルールベースの方策を作成し，モンテカルロ木探索のプレイアウトで使用した結果，平均スコアがランダム方策のときよりも 18%向上したという結果が得られている．ここでは，一定の手数動かした後のスコアの平均を適応度としていた．

第 4 章

提案内容

本研究では，ガイスターにおいて進化的計算を用いてルールベースのプレイアウト方策を作成することで，モンテカルロベースプレイヤーの性能向上を目指し，性能の違いによるルールの特徴を調査する．プレイアウト方策は，事前にガイスターのルールを基に関数を定義し，それを用いて GP によって作成する．不完全情報ゲームはプレイアウト時に現在の局面を可能な状態から仮定し完全情報状態にする．そのため，関数で色情報を区別する場合と区別しない場合で差が現れると考えた．そこで，GP は色情報を区別しない・区別するときとすべて使用するときで行う．また，GP では一致率と平均悪手を適応度として使用し，教師データは不完全情報と完全情報の場合を使用する．木村らの深層学習を用いたガイスター AI の研究 [13] では，完全情報化した状態であった場合学習が上手くいったという結果が得られている．そのため，本研究でも不完全情報の状態と完全情報の状態の両方で学習を行い，差が現れるか調査する．作成した方策とランダム方策のモンテカルロ木探索プレイヤーで対戦実験を行い，性能を評価する．

また，ルールベース方策はプレイアウトを決定的にすることでランダム性が失われ弱くなるとされており，ランダム性を入れるために確率の導入などがされている．よって，作成した方策を確率的に指手を選択できるようにし，GA を用いてパラメータの調整を行う．使用する方策は GP の実験で勝率がよかったものを使用する．GA 内で使用する適応度は，GP で使用した不完全情報の一致率と平均悪手を使用する．対戦は，パラメータを調整する前のものとの対戦を行い，性能を評価する．

4.1　GP で用いた関数セット

本研究では以下に述べる関数を実装し，GP で使用した．下記に示したもの以外にも，if_then_else，比較演算子，論理演算子，0 以上 6 未満のランダムな整数値を使用している．

4.1.1　必ず使用する関数

- **自身の青駒の残数を取得 (getNumOfMyBlue)**

  整数値を返す，if 文の条件式に用いる関数である．ガイスターの敗北条件に「自身の青駒を全て取られる」があるため，自身が負けそうかの指標にできると考えた．自身が負けそうかということは行動選択のために必要な要素だと考えたため実装した．

- **自身の赤駒の残数を取得 (getNumOfMyRed)**

  整数値を返す，if 文の条件式に用いる関数である．ガイスターの勝利条件に「自身の赤駒を全て取らせる」があるため，自身が勝てそうかの指標にできると考えた．自身が勝てそうかということも行動選択のために必要な要素だと考えたため実装した．

- **相手の青駒の残数を取得 (getNumOfOppBlue)**

  整数値を返す，if 文の条件式に用いる関数である．ガイスターの勝利条件に「相手の青駒を全て取る」があるため，自身が勝てそうかの指標にできると考えた.「自身の赤駒の残数を取得」と同様の理由より実装した．

- **相手の赤駒の残数を取得 (getNumOfOppRed)**

  整数値を返す，if 文の条件式に用いる関数である．ガイスターの敗北条件に「相手の赤駒を全て取る」があるため，自身が負けそうかの指標にできると考えた.「自身の青駒の残数を取得」と同様の理由より実装した．

- **自陣脱出口付近に自駒がある (isHavingKeeper)**

  真理値を返す，if 文の条件式に用いる関数である.「自陣脱出口付近に相手駒がある」と同じ理由から実装した．調べている範囲も同じである．

- **ランダムに手を指す (doRandomMove)**

指すことが出来る手からランダムに選択して指す関数である．指して選択の基本的なものとして用意した．

4.1.2 色情報を区別しない関数

- **脱出口までの最短の距離を取得 (getMinimumDistanceToGoal)**

  整数値を返す，if 文の条件式に用いる関数である．ガイスターの勝利条件に「自身の青駒を脱出させる」があり，脱出口に駒を進めることが重要であると考えられた．脱出口までの最短の距離が，脱出口に駒を進めるなどの行動選択の閾値になると考えたため実装した．

- **自陣脱出口までの最短の距離を取得 (getMinimumDistanceToMyGoal)**

  整数値を返す，if 文の条件式に用いる関数である．ガイスターの敗北条件に「相手に脱出される」があり，伊藤らの研究 [14] において，キーパー戦略が一定の有効性があることが示されている．そのため，自陣に戻りやすいように，自陣脱出口までの距離が必要だと考えた．

- **相手駒と隣接しているか (isNextToOpponents)**

  真理値を返す，if 文の条件式に用いる関数である．ガイスターのルール上駒取りがあり，勝利条件にも関わってくるため必要な要素と考え，実装した．

- **自陣脱出口付近に相手駒がある (isOppNearMyGoal)**

  真理値を返す，if 文の条件式に用いる関数である．ガイスターの敗北条件に「相手の青駒に脱出される」があり，伊藤らの研究 [14] において，キーパー戦略が一定の有効性があることが示されている．そのため，脱出口周辺の状況が重要であると考えた．今回は，脱出口と脱出口に隣接しているマスと脱出口から斜め上のマスの 4 マスについて調べている．

- **隣接している相手駒を取る (doCaptureMove)**

  隣接している相手駒があれば，その駒を取る手を選択し指す関数である．指手が複数存在している場合，その中からランダムに選ばれる．勝利条件に「相手の青駒を全て取

る」があり，優先的に相手駒を取るものが必要だと考えた．

- **最も脱出口に近い自駒を脱出口に近づける (doMoveCloserToGoal)**

  脱出口に最も近い自駒を脱出口に近づくように指手を選択し指す関数である．盤面を左右の半分に分け，脱出口方向と左側のとき左，右側のとき右を選択するようにしている．勝利条件に「自分の青駒を脱出させる」があるため，脱出口に向かわせる行動は重要だと考えた．

- **隣接している相手駒から逃げる (doEscapeMoveFromOpponents)**

  隣接している相手駒から離れるような指手を選択し指す関数である．隣接している相手駒がある場合，その駒から離れるような指手のうち，移動先も相手駒と隣接していない指手を選択する．逃げる指手であるにも関わらず，指した後すぐに取られてしまってはいけないと考え，このような選択にした．敗北条件に「自身の青駒を全て取られる」があるため，相手から逃げることも必要だと考えた．

- **脱出口から遠ざける (doMoveLeavingFromGoal)**

  脱出口から遠ざけるような指手を選択する関数である．盤面を左右の半分に分け，脱出口逆方向と中央方向を選択するようにしている．勝利条件に「自分の青駒を脱出させる」があるため，脱出口に向かわせることは重要であると考えられる．しかし，脱出口に向かわせると駒を取られるリスクが大きくなるため，脱出口から遠ざける指手も必要になると考え，実装した．

- **最も自陣脱出口に近い自駒を自陣脱出口近づける (doMoveCloserToMyGoal)**

  最も自陣脱出口に近い自駒を自陣脱出口に近づける指手を選択し指す関数である．盤面を左右の半分に分け，自陣脱出口方向と左側のとき左，右側のとき右を選択するようにしている．伊藤らの研究 [14] において，キーパー戦略が一定の有効性があることが示されている．そのため，自陣脱出口付近に自駒があることが重要になってくると考え実装した．

指手を指す関数では，該当する手が存在しない場合，ランダムな手を指すようにしている．

4.1.3　色情報を区別する関数

色情報を区別する関数は，色情報を区別しない関数を青駒と赤駒それぞれで行った場合のものとなっている．実装した関数は，以下の 18 個である．

- **脱出口までの青駒の最短距離を取得 (getMinimumDistanceOfBlueToGoal)**
- **脱出口までの赤駒の最短距離を取得 (getMinimumDistanceOfRedToGoal)**
- **自陣脱出口までの青駒の最短距離を取得 (getMinimumDistanceOfBlueToMyGoal)**
- **自陣脱出口までの赤駒の最短距離を取得 (getMinimumDistanceOfRedToMyGoal)**
- **相手青駒と隣接しているか (isNextToBlueOpponents)**
- **相手赤駒と隣接しているか (isNextToRedOpponents)**
- **自陣脱出口付近に相手青駒がある (isBlueOppNearMyGoal)**
- **自陣脱出口付近に相手赤駒がある (isRedOppNearMyGoal)**
- **隣接している駒が青駒なら取る (doCaptureBlueOpponentMove)**
- **隣接している駒が赤駒なら取る (doCaptureRedOpponentMove)**
- **最も脱出口に近い自青駒を脱出口に近づける (doMoveBlueCloserToGoal)**
- **最も脱出口に近い自赤駒を脱出口に近づける (doMoveRedCloserToGoal)**
- **隣接している駒が赤駒なら逃げる (doEscapeMoveFromBlueOpponents)**
- **隣接している駒が赤駒なら逃げる (doEscapeMoveFromRedOpponent)**
- **脱出口から青駒を遠ざける (doMoveBlueLeavingFromGoal)**
- **脱出口から赤駒を遠ざける (doMoveRedLeavingFromGoal)**
- **最も自陣脱出口に近い自青駒を自陣脱出口近づける (doMoveBlueCloserToMyGoal)**
- **最も自陣脱出口に近い自赤駒を自陣脱出口近づける (doMoveRedCloserToMyGoal)**

4.2　GP・GA の適応度

本研究は，AI プレイヤーを強くすることが目的であるため，適応度は強さと関係するものが必要である．また，適応度は計算する回数が多いため，計算コストは少ない方が良い．本研究では，強いプレイヤーの指手との一致率と平均悪手の 2 種類を適応度として用る．

チェスで GA により評価関数と探索のパラメータを調整した研究 [15] では，強い人間の指手との一致率を適応度として採用した．この研究において，一致率を適応度に用いることでパラメータの調整に成功している．また，一致率は次の一手との比較であるため，計算コストも低い．しかし，ガイスターにはプロのような強い人間プレイヤーが明確に存在しているわけではない．そのため，本研究では強い AI プレイヤーとの一致率を用いる．

平均悪手は，プレイヤーが悪手をどれだけ指しているかの指標である．悪手とは，教師プレイヤーと別の手を指して，評価値が下がった手のことをいう．このとき，評価値の差の値を悪手の値としたとき，その合計の平均が平均悪手である．平均悪手を求める式は以下の式である．

$$\frac{1}{\text{局面数}} \sum \big((\text{プレイヤーが指した手の評価値}) - (\text{最善手の評価値})\big)$$

この式の最善手は，教師プレイヤーが指した手のことを表している．山下らの研究 [16] では，平均悪手でレーティング推定ができることを明らかにした．よって，平均悪手が強さと関連していることから適応度とする．また，平均悪手も次の一手までしか行わないので計算コストも低い．

Ms.Pac-Man で GP を用いてルールベース方策を作成する研究 [4] では実際にゲームをプレイし，スコアを適応度として用いた．ガイスターにおいても実際に対戦を行い，勝率を適応度とすることもできるが，時間が掛かることから現実的ではないと考え，採用しない．

第 5 章

実験

5.1 実験方法

5.1.1 GP の実験

この実験では，GP を用いてプレイアウト方策の作成を行った．GP は個体数 100，世代数 50 で行い，変異率 0.2，交叉率 0.5 とし，5 回行った．今回 GP を行った関数セットの組み合わせを以下のように定義する．

- 色情報なし：必ず使用する関数 + 色情報を区別しない関数
- 色情報あり：必ず使用する関数 + 色情報を区別する関数
- すべて：すべての関数

適応度は一致率と平均悪手を使用し，教師データは不完全情報と完全情報の 2 つ用意した．教師として，GAT2020 のガイスター AI 大会で優勝した Naotti-2020[*1] を用いた．

対戦実験では，GP で作成されたプログラムを使用した MCTS と他のプレイヤーを対戦させ，性能評価を行った．対戦プレイヤーはランダム方策の MCTS プレイヤーを用いた．対戦回数は，先手・後手 500 試合ずつの 1000 試合行った．対戦実験は，プレイアウト回数の変化による変化を見るためにプレイアウト回数を 5,000，10,000，20,000 回で対戦を行った．対戦プレイヤーのプレイアウト回数は 10,000 回である．初期配置による勝率のばらつきを避けるために，初期配置を固定した．初期配置は，全ての初期配置から反転や回転など

[*1] https://github.com/kawakami-naoto/Naotti-2020

して同じ配置になったものを除いた配置からランダムに 500 個選び使用した．

実験で使用するプレイヤーのプレイアウト回数は基本的に 10,000 回とする．また，ルート局面で実現可能な相手駒色の配置で決定化を行い，完全情報化を行っている．これは，1 プレイアウトごとに行っている．

5.1.2　GA の実験

この実験では，作成した方策を指手選択に確率を導入し，GA によってパラメータの調整を行った．GA は GP と同様に個体数 100，世代数 50 で行い，変異率 0.2，交叉率 0.5 とし，5 回行った．使用する方策は，GP の実験で勝率が高かったものを使用した．適応度は一致率と平均悪手を使用し，教師データは不完全情報を用いた．教師は，GP と同じく Naotti-GAT2020 を用いた．

対戦実験は，調整後と調整前で対戦を行った．勝率が高かった方策であるためランダム方策の MCTS で評価を行うと差が確認しにくいと考え，調整前との対戦を行った．対戦回数は，先手・後手 500 試合ずつの 1000 試合行う．初期配置による勝率のばらつきを避けるために，初期配置を固定する．初期配置は，全ての初期配置から反転や回転などして同じ配置になったものを除いた配置からランダムに 500 個選び使用する．

実験で使用するプレイヤーのプレイアウト回数は 10,000 回とする．また，ルート局面で実現可能な相手駒色の配置で決定化を行い，完全情報化を行っている．これは，1 プレイアウトごとに行っている．

5.2　実験結果

5.2.1　GP の結果

図 5.1 は GP の一致率の推移，図 5.2 は GP の平均悪手の推移を表している．グラフは，5 回試行した平均一致率・平均悪手の推移を表している．横軸は世代数を表し，縦軸は図 5.1 が一致率を表しており，図 5.2 が平均悪手を表している．グラフの青が色情報なし，緑が色情報あり，赤が全ての関数を使用した場合を表しており，実線が完全情報，破線が不完全情報のときのグラフとなっている．図 5.1，図 5.2 どちらのときでも世代が進むにつれ，適応度が上昇しており学習はうまくいっていると考えられる．平均悪手では，不完全情報のときは負の値になっていたが，完全情報のときは正の値になるものもあった．これは，平均悪手の計算は山下らの研究 [16] では正の値を除外しているが，今回の実装では正の値も加えてしまっていることが原因であると考えられる．

表 5.1 はランダム方策との対戦結果である．勝率は，5 回試行した平均・最大勝率を表している．対戦の結果，一致率を適応度とした場合は全てで平均勝率は 50%を越え，最大勝率は 82.5%となるものもあった．平均勝率は，色情報ありが最も高くなり，完全情報の 30 世代のときが 73.14%と平均勝率で最も高かった．一方，平均悪手を適応度とした場合，不完全情報での色情報なしが平均勝率 50.32%で最も高かった．最大勝率の中には 69.60%と高いものもあったが，一致率を適応度とした場合に比べて全体的に低くなっていた．また，全ての場合において，世代の変化による勝率の変化は見られなかった．

表 5.3〜5.6 は，それぞれの場合でプレイアウト回数を変えた場合の対戦結果である．表 5.3〜5.6 の平均勝率と最大勝率は前述したものと同じである．最大勝率の「*」は，選ばれた個体がプレイアウト回数 10,000 のときと違うときを表している．表 5.2 は，ランダム方策を用いたプレイヤーの結果であり，これと比較する．表 5.3 と表 5.4 より，一致率でプレイアウト回数を 10,000 回から 5,000 回に減らした場合，平均勝率は約 3〜5%減少しており，ランダム方策のときの変化とほとんど変わらなかった．一方で，平均悪手では表 5.5 と表 5.6 からプレイアウト回数を減らした場合，平均勝率は約 1〜3%の減少となっており，ランダム

方策より勝率の減少が少なかった．プレイアウト回数を 10,000 回から 20,000 回に増やした場合，多くの場合で約 2%〜4%の増加とランダム方策のときより増加量が少なくなった．

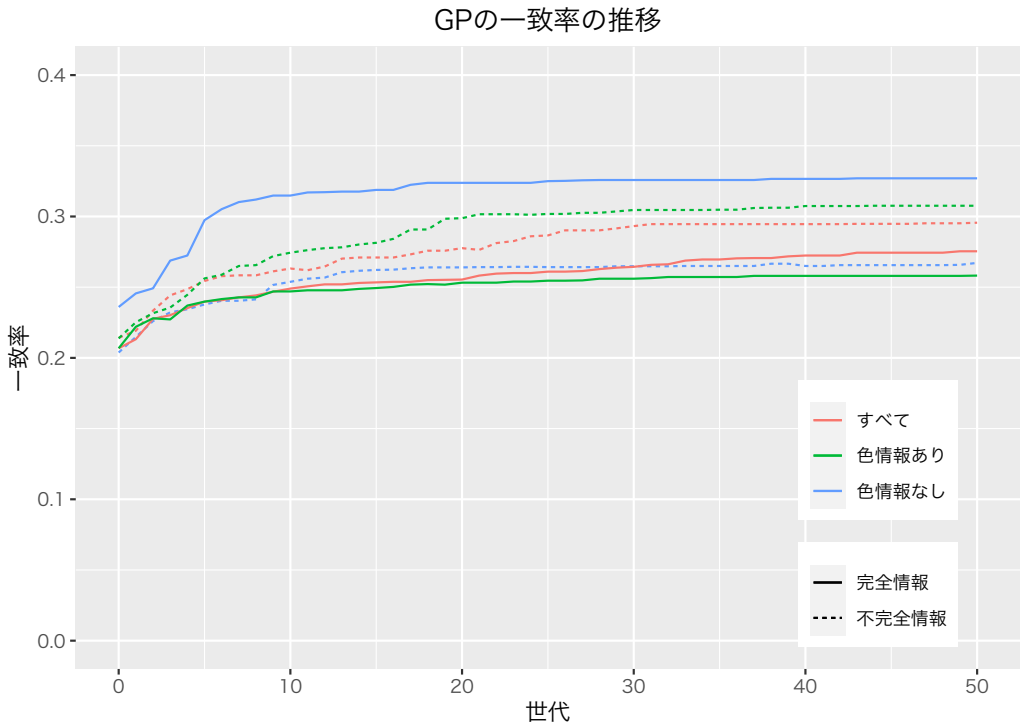

図 5.1　GP の一致率の推移

5.2　実験結果

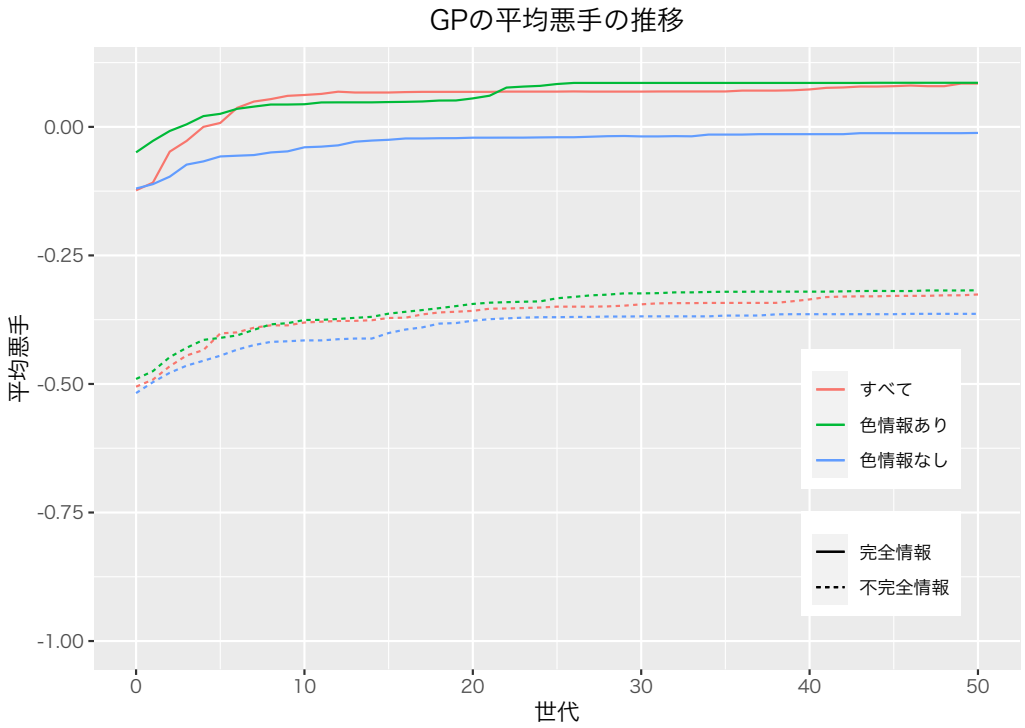

図 5.2　GP の平均悪手の推移

表 5.1　GP で作成した方策を用いたプレイヤーの勝率 (%)：対ランダム

		一致率				平均悪手			
世代		30		50		30		50	
		平均	最大	平均	最大	平均	最大	平均	最大
不完全情報	色情報なし	61.08	67.50	61.34	68.95	50.32	57.00	50.30	55.00
	色情報あり	63.72	76.10	62.82	76.00	42.38	50.20	42.60	52.20
	すべて	61.92	82.5	61.32	79.45	43.03	51.35	45.17	48.6
完全情報	色情報なし	56.12	66.20	56.97	71.15	45.08	69.65	45.33	67.55
	色情報あり	73.14	79.60	72.21	80.45	39.25	45.25	40.25	45.40
	すべて	56.25	70.25	56.60	70.40	37.10	38.75	37.69	40.45

表 5.2　ランダム方策 MCTS プレイヤーのプレイアウト数毎の勝率 (%):対ランダム方策 (プレイアウト数=10000)

プレイアウト数	5,000	10,000	20,000
勝率	46.05	50.15	55.65

表 5.3　不完全情報で一致率の場合のプレイアウト数毎の勝率 (%)：対ランダム方策 (プレイアウト数=10000)

		平均勝率			最大勝率		
プレイアウト数		5,000	10,000	20,000	5,000	10,000	20,000
30 世代	色情報なし	55.29	61.08	64.70	61.40	67.50	71.90
	色情報あり	59.93	63.72	67.25	71.90	76.10	83.50
	すべて	56.91	61.92	63.01	76.05	82.50	82.85
50 世代	色情報なし	59.63	61.34	65.63	64.90	68.95	71.70
	色情報あり	58.88	62.82	65.86	73.35	76.00	82.85
	すべて	57.66	61.32	58.67	75.20	79.45	71.05*

表 5.4 完全情報で一致率の場合のプレイアウト数毎の勝率 (%)：対ランダム方策 (プレイアウト数=10000)

		平均勝率			最大勝率		
プレイアウト数		5,000	10,000	20,000	5,000	10,000	20,000
30 世代	色情報なし	51.92	56.12	59.85	61.70*	66.20	70.45*
	色情報あり	69.13	73.14	75.90	76.70*	79.60	83.50*
	すべて	50.80	56.25	56.50	67.70*	70.25	75.30
50 世代	色情報なし	57.47	56.97	62.36	65.50	71.15	70.45
	色情報あり	68.60	72.21	74.71	75.15*	80.45	82.30*
	すべて	52.78	56.60	57.40	63.40*	70.40	75.50

表 5.5 不完全情報で平均悪手の場合のプレイアウト数毎の勝率 (%)：対ランダム方策 (プレイアウト数=10000)

		平均勝率			最大勝率		
プレイアウト数		5,000	10,000	20,000	5,000	10,000	20,000
30 世代	色情報なし	46.48	50.32	56.00	51.90*	57.00	62.30*
	色情報あり	41.02	42.38	46.31	47.85	50.20	55.40
	すべて	42.74	43.03	44.68	52.55	51.35	53.75
50 世代	色情報なし	45.36	50.30	55.08	51.70*	55.00	63.55
	色情報あり	41.25	42.60	44.28	49.25	52.20	52.60
	すべて	43.49	45.17	48.45	47.30*	48.60	57.20*

表 5.6 完全情報で平均悪手の場合のプレイアウト数毎の勝率 (%)：対ランダム方策 (プレイアウト数=10000)

		平均勝率			最大勝率		
プレイアウト数		5,000	10,000	20,000	5,000	10,000	20,000
30 世代	色情報なし	42.93	45.08	46.08	60.40	69.65	64.50
	色情報あり	37.29	39.25	39.68	44.35	45.25	43.55
	すべて	37.48	37.10	38.46	40.00*	38.75	42.25*
50 世代	色情報なし	47.17	45.33	51.37	63.50	67.55	72.70
	色情報あり	39.36	40.25	39.44	46.80*	45.40	45.70
	すべて	39.62	37.69	37.75	43.80	40.45	41.75

勝率のよかった方策をアルゴリズム 1 とアルゴリズム 2 に示す．しかし，示した 2 つの方策のように簡潔なものもあるが，得られた方策は実行されることのない不要な部分があるなど解釈することが困難なものが多かった．そのため，人の手で不要な部分の削除を行わなければいけなかった．

Algorithm 1 不完全情報の一致率を適応度としたすべての場合で勝率が最も高かった方策

```
if getMinimumDistanceToGoal == isHavingKeeper then
    doMoveCloserToMyGoal
else
    doMoveBlueCloserToGoal
end if
```

Algorithm 2 完全情報の一致率を適応度とした色情報ありの場合で勝率が高かった方策

```
if isBlueOppNearMyGoal then
    doMoveBlueCloserToGoal
else
    doCaptureRedOpponentMove
end if
```

5.2.2　GA の結果

図 5.3 は GA の一致率の推移，図 5.4 は GP の平均悪手の推移を表している．グラフは，5 回試行した平均一致率・平均悪手の推移を表している．横軸は世代数，縦軸は図 5.3 が一致率を表し，図 5.4 が平均悪手を表している．図 5.3，図 5.4 どちらのときでも世代が進むにつれ，適応度が上昇していたが，GP のときに比べ大きく変化しなかった．

表 5.7 は，調整前の方策との対戦結果である．勝率は，5 回試行した平均・最大勝率を表している．対戦の結果，一致率を適応度とした場合は 30 世代の平均勝率が 50.84%となった．1 個体だけ勝率が有意に高かったが残りの個体では性能の向上は見られなかった．平均悪手を適応度とした場合は，30，50 世代どちらでも平均勝率約 20%となり，大きく性能が下がった．

表 5.7　調整後の勝率 (%)：対調整前

世代	30		50	
	平均	最大	平均	最大
一致率	50.84	54.25	49.61	50.6
平均悪手	20.55	21.7	19.52	21.95

5.2　実験結果

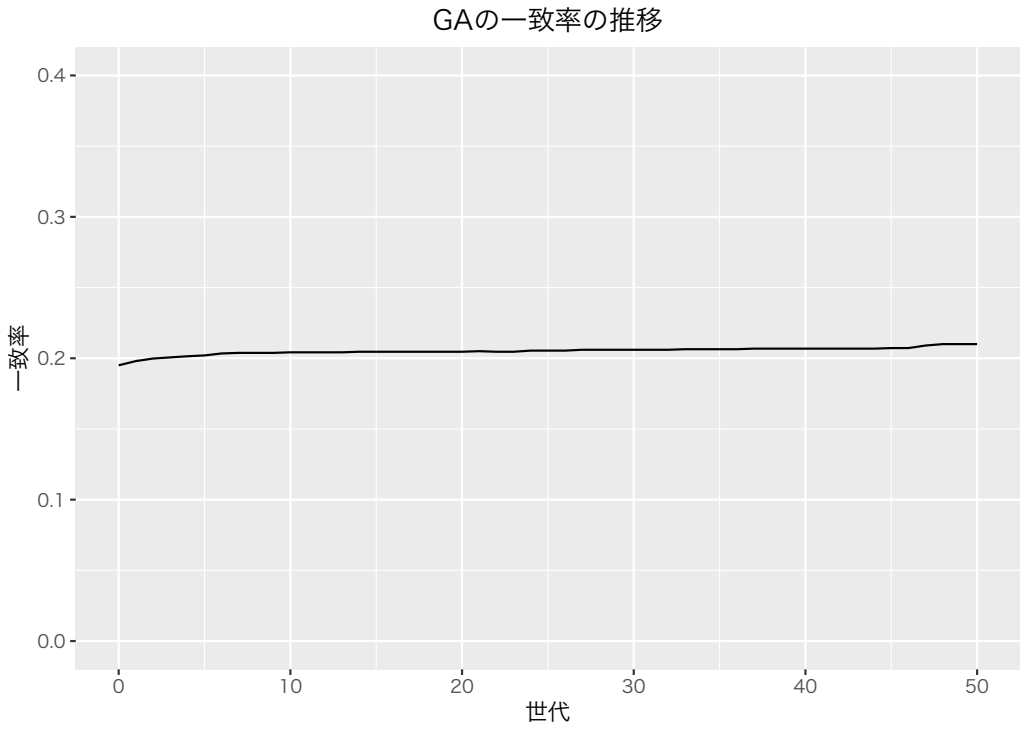

図 5.3　GA の一致率の推移

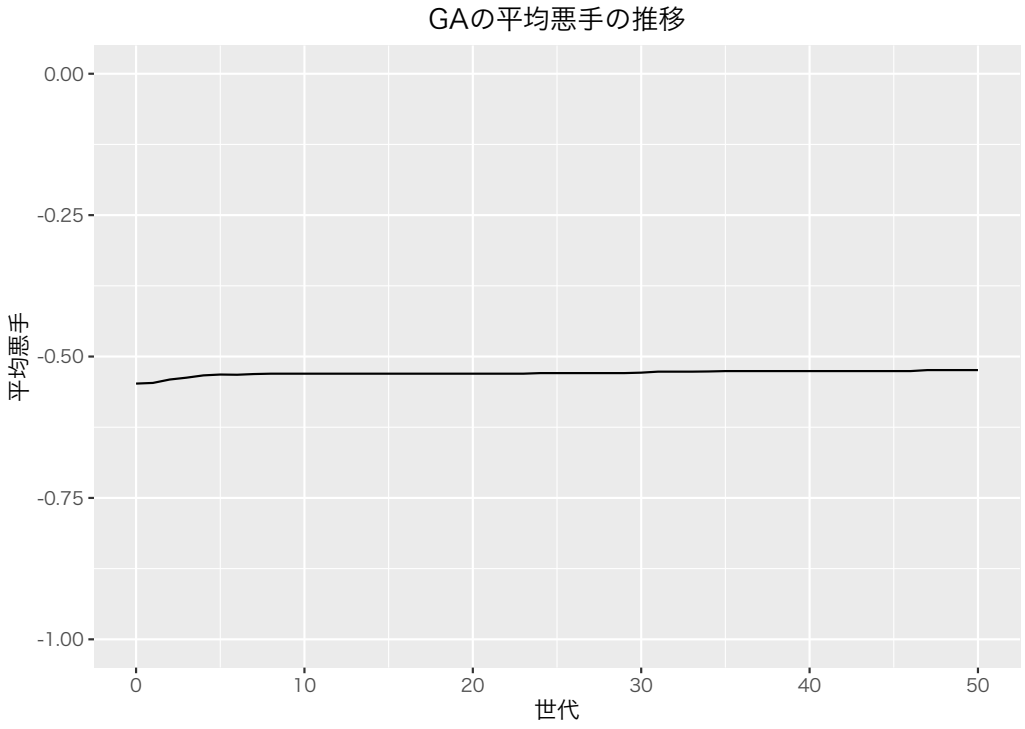

図 5.4　GA の平均悪手の推移

第 6 章

考察

6.1 GP について

実験の結果から一致率を適応度とした場合で性能の良いものが作成できていることが確認できた．関数セットは，表 5.1 の対戦結果から一致率を適応度とした場合にどちらでも勝率が高かった色情報ありが良いと考えられる．すべての関数を使用した場合は，関数数も増えてしまうため一つ一つの関数が選択される確率が減ってしまい，良い方策を見逃してしまう可能性がある．また，色情報なしの場合は，色情報ありと違い青駒と赤駒を区別しておらず，より的確な方策の作成が困難であると考えられる．色情報ありの場合でも，自分の駒色を参照する場合と相手の駒色を参照する場合で関数を分けることでより性能の向上も期待できる．表 5.5 と表 5.6 の対戦結果から平均悪手の場合，プレイアウト回数を減らしたときに勝率の減少を少なくできたことからプレイアウトの質が向上したと考えられる．しかし，平均悪手の場合，勝率が高くなかったことから作成された方策が適切でなかったと考えられる．一方，プレイアウト回数を増やした場合，ランダム方策のときよりも勝率の増加が少なくなっていることから方策をルールベースにすることでプレイアウトの内容が偏ってしまっていると考えられる．また，完全情報での色情報ありを適応度とした場合のような勝率が高いところでは，勝率の増加が難しかったと考えられる．

6.2　方策について

作成された方策から強いプレイヤーにはどのようなルールが使われているか調べた．調べたところ,「脱出口に向かわせる指手 (赤駒のみを除く)」が行われることが多い方策のプレイヤーが勝率が高くなっていた.「相手駒を取る」,「自駒を自陣脱出口に向かわせる指手」が多く行われる方策のプレイヤーの場合は，強いものもあったが弱いものもあった．また,「相手青駒から逃げる」,「脱出口から遠ざかる」が中心の方策のプレイヤーは勝率が低くなっていた．以上のことから，方策で重要となる指手を表 6.1 にまとめた．表 6.1 より，勝敗に直接関係があるものほど重要で勝敗に関係ないものほど重要ではないと考えられる．相手駒を取るは，勝利条件にある通り勝敗に直接関係があるが弱くなるものもあったため，重要度は青駒または自駒を脱出口に向かわせる指手ほど重要でないと考えられる.「脱出口に向かわせる」は，自分から見て動かす駒の色は絶対に正しい．一方で,「相手駒を取る」は，相手から見て取る駒の色が絶対に正しい．よって，自分から見て駒色が正しいルールと相手から見て駒色が正しいルールとの間に，相手の駒色が正確とは限らないことによるプレイアウト結果への影響に差が出ると考えられる．そのため，方策を手番ごとに別々の方策にすることで更なる性能の向上が見込めると考えられる．

本研究の実装上，指すことができない場合ランダムに指すとなっている．GA を行ったところ性能が向上しなかったことから GP で作成された方策の時点で強い方策は十分ランダム性を持っていたと考えられる．また，局面の仮定時にもランダムに駒色の配置を行っており，この部分でもランダム性を持たせられていると考えられる．しかし，本研究では作成したすべての方策で GA を行ったわけではないため，指手選択に確率を導入することが有効でないとは言えない．明らかにランダム性を保持していない方策では，確率的に指手を選択することで強くなることは十分に考えられる．

表 6.1　重要となる指手

重要度	指手
高い	青駒を脱出口へ向かわせる，自駒を脱出口へ向かわせる
	相手駒を取る系，自駒を自陣脱出口へ向かわせる
	赤駒を脱出口へ向かわせる
低い	相手駒から逃げる，脱出口から遠ざかる

6.3　適応度について

適応度が強さに繋がったかを調べるために勝率と適応度との相関係数を調べた．勝率と適応度との相関係数を調べたところ，不完全情報で一致率を適応度とした場合に相関係数が 0.473 と正の相関があることがわかった．現段階では，不完全情報で一致率を適応度として用いる方が最も良いと考えられる．不完全情報での平均悪手を適応度とした場合では相関係数 0.282 と弱い正の相関が見られたが，勝率が高くなかったことから適していないと言える．Naotti-2020 の評価関数は,「1000 × (青駒の数) − (駒の脱出口までの距離の合計)」と簡単なものであったため，評価値に差が出にくかったと考えられる．そのため，Naotti-2020 の評価関数は平均悪手の計算に不向きであったと考えられ，平均悪手の計算に向いた評価関数を利用することで改善する可能性がある．完全情報の相関係数は，どちらとも負の相関が見られた．しかし，一致率の色情報ありのときに勝率が高かったため，適していないとは言い難い．そのため，完全情報はまだ調査が必要であると考えられる．よって，教師データが不完全情報と完全情報かは，どちらが良いかは現在の段階では結論を出すことはできない．

第 7 章

おわりに

本研究では，二人不完全情報ゲームであるガイスターを対象とし，モンテカルロベースのプレイヤーの方策をルールベースにしプレイアウトの改善を目指し，性能の違いによるルールの特徴の調査を行った．方策の作成には GP を用い，確率的に指手選択を行えるようにした方策のパラメータの調整を GA で行った．そのため，本研究では，GP に用いる適応度についても調査を行った．

GP を行った結果，適応度を一致率とした場合で最大勝率 82.5%と性能の良いものが作成できていることが確認できた．しかし，簡潔な方策もあるが，得られた方策が最適化されておらず解釈することが困難であり，人の手で最適化を行わなければいけないものが多かった．また，30 と 50 世代の変化による勝率の変化は見られなかった．作成された方策を調べたところ,「青駒を脱出口に向かわせる」や「相手駒を取る」など勝敗に直接関係があるものほど重要であり,「相手駒から逃げる」や「赤駒を脱出口に向かわせる」などの勝敗に直接関係のないのものほど重要でないと考えられる．しかし,「相手駒を取る」場合は弱くなることもあり，自分から見て駒色が正しいルールと相手から見て駒色が正しいルールで，相手の駒色が正確とは限らないことによるプレイアウト結果への影響に差が出ると考えられる．

GA を行った結果，パラメータを調整する前に比べ性能が向上しなかった．本研究の実装上，指すことのできない場合はランダムに指すことになっており，GP で作成された方策の時点で十分にランダム性を持っていたと考えられる．また，プレイアウト時にランダムに局面を仮定している部分でも，ランダム性を持つことができていたと考えられる．しかし，すべての方策で GA を行ったわけではないため，指手選択を確率的にすることが有効でないとは言えず，方策によっては強くなることは十分に期待できる．

勝率と適応度の相関係数を調べたところ，不完全情報での一致率に正の相関があることがわかった．現段階では，不完全情報での一致率を適応度として用いることが良い．しかし，完全情報の相関係数は負の相関であり，更なる調査が必要であると考えられる．また，不完全情報での平均悪手は弱い正の相関があったが，勝率がよくなかった．これは，Naotti-2020の評価関数が不適切であった可能性がある．そのため，教師データが不完全情報と完全情報かは，どちらが良いかは現在の段階では結論を出すことはできない．

自分から見て駒色が正しいルールと相手から見て駒色が正しいルールで，相手の駒色が正確とは限らないことによるプレイアウト結果への影響に差が出ると考えられるため，自分と相手では重要となるルールが違うことは十分に考えられる．今後の展望としては，手番ごとに別々の方策を用いることが考えられる．また，Naotti-2020 の評価関数が平均悪手の計算に不向きであったため，うまくいかなかったと考えられる．そのため，平均悪手の計算に向いた評価関数で評価し，平均悪手が適応度として適切か調べる必要もあると考えられる．今回は，完全情報の場合に負の相関があったが，一致率の場合は性能が向上していた．そのため，教師データが完全情報の場合に適応度が高いほど強くなるか更なる調査が必要だと考えられる．

謝辞

本研究を進めるにあたり，指導教員である高知工科大学情報学群の竹内聖悟先生には研究室配属から 4 年間に渡り，研究活動や論文執筆，プレゼンテーションなどにおいて多くのご指導をいただきました．論文執筆に当たっては，私の拙い日本語に対しても丁寧なご指導をしていただきました．心より感謝いたします．並びに，本論文の副査を引き受けていただいた松崎公紀教授，高田喜朗准教授に深く感謝いたします．

参考文献

[1] 伊藤毅志, 保木邦仁, 三宅陽一郎. ゲーム情報学概論-ゲームを切り開く人工知能-. コロナ社, 2018.

[2] Sylvain Gelly—Yizao Wang—RémiMunos, Olivier Teytaud. Modification of uct with patterns in monte-carlo go. *Technical Report RR-6062*, Vol. 32, pp. 30–56, 2006.

[3] Maciej Świechowski and Jacek Mańdziuk. Self-adaptation of playing strategies in general game playing. *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 6, No. 4, pp. 367–381, 2013.

[4] Atif M Alhejali and Simon M Lucas. Using genetic programming to evolve heuristics for a monte carlo tree search ms pac-man agent. In *2013 IEEE Conference on Computational Intelligence in Games (CIG)*, pp. 1–8. IEEE, 2013.

[5] 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, 第 2018 巻, pp. 13–16, feb 2018.

[6] 川上直人, 橋本剛. ガイスター ai の研究. 情報処理学会研究会報告, 第 2018-GI-39 巻, pp. 1–6, 2018.

[7] 三塩武徳, 小谷善行. ゲームの不完全情報推定アルゴリズム upp とそのガイスターへの応用. 情報処理学会研究会報告, 第 2014-GI-31 巻, pp. 1–6, 2014.

[8] 鴛淵隆斗, 佐藤直之. ガイスターゲームにおけるモンテカルロ法を利用した駒推定およびブラフ手の生成可能性の検証. 情報処理学会研究会報告, 第 2020-GI-43 巻, pp. 1–7, 2020.

[9] 小谷善行, 岸本章宏, 紫原一友, 鈴木豪. ゲーム計算メカニズムー将棋・囲碁・オセロ・チェスのプログラムはどう動くー. コロナ社, 2010.

[10] Peter I Cowling, Edward J Powley, and 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.

[11] 北野宏明. 遺伝的アルゴリズム. 人工知能, Vol. 7, No. 1, pp. 26–37, 1992.

[12] 平野廣美. 遺伝的アルゴリズムと遺伝的プログラミング-オブジェクト指向フレームワークによる構成と応用. パーソナルメディア株式会社, 2000.

[13] 木村勇太, 伊藤毅志. 深層強化学習を用いたガイスター ai の構築. ゲームプログラミングワークショップ 2019 論文集, 第 2019 巻, pp. 130–135, nov 2019.

[14] 伊藤雅士, 大久保壮浩, 木谷裕紀, 小野廣隆. ガイスター ai のキーパー戦略の有効性. 情報処理学会研究会報告, 第 2019-GI-42 巻, pp. 1–7, 2019.

[15] Omid E David, H Jaap van den Herik, Moshe Koppel, and Nathan S Netanyahu. Genetic algorithms for evolving computer chess programs. *IEEE transactions on evolutionary computation*, Vol. 18, No. 5, pp. 779–789, 2013.

[16] 山下宏. 将棋名人のレーティングと棋譜分析. ゲームプログラミングワークショップ 2014 論文集, 第 2014 巻, pp. 9–16, oct 2014.