令和 5 年度

修士学位論文

# 推定情報の活用を目指した遺伝的プログラミングによる人狼知能プレイヤの生成

## Generation of AIWolf Utilizing Estimated Information Through Genetic Programming

1265105　　田頭裕

指導教員　　竹内聖悟

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

## 推定情報の活用を目指した遺伝的プログラミングによる人狼知能プレイヤの生成

田頭裕

人狼ゲームは不完全情報ゲームであり，相手の役職が明確に把握できない特性を有している．そのため，人狼ゲームでは役職推定の研究が盛んであり，サポートベクタマシンや深層ニューラルネットワークなど多くの方法で役職推定がなされている．しかし，その役職推定情報をどのように扱うのが強いのかについては明らかになっていない．ガイスターでは遺伝的プログラミングを用いて性能の違いによるルールの特徴の調査を行い，高い性能のルールを確認している．このように不完全情報ゲームにおいて遺伝的プログラミング（GP）を用いてルールを調査することは可能であり，人狼知能でも同様にルールを調べることは可能であると考えた．

本研究では，人狼知能において GP を用いて役職推定情報の有効な活用方法について調査し，意思決定の改善を行うことで人狼知能エージェントの性能向上を確認できた．特に，性能向上が見られたルールにおいては，残り人数が半分以下になった際により追放するべき役職に重点的に投票を行うことが効果的であることが示唆された．

さらに，役職推定情報だけでなく，その他の得られる情報との組み合わせにより，性能が一層向上することも確認された．この研究は，人狼知能において役職推定情報を最適に活用する手法の一環として，GP が有益であることを示唆している．

**キーワード**　人狼知能，推定情報，遺伝的プログラミング

# Abstract

## Generation of AIWolf Utilizing Estimated Information Through Genetic Programming

Yutaka Tagashira

Werewolf is an incomplete information game, in which the role of the opponent is not clearly known. For this reason, role estimation has been studied extensively in wargames, and many methods such as Support Vector Machine and Deep Neural Network have been used for role estimation. However, it is not clear how to handle the information of role estimation. Geister uses Genetic Programming to investigate the characteristics of rules with different performance, and confirms the rules with high performance. Thus, it is possible to investigate the rules using genetic programming (GP) in incomplete information games, and we hypothesized it would be possible to investigate the rules in the same way in AIWolf.

In this study, we investigated the effective use of role estimation information using GP in AIWolf, and confirmed the performance improvement of the AIWolf agent by improving its decision making. In particular, the information suggests that, when the number of remaining players is less than half, it is more effective to focus the vote on the roles that should be ousted.

Furthermore, it was confirmed that the performance was further improved not only by the role estimation information, but also by the combination with other obtained information. This study suggests that GP may be beneficial as part of a method to optimally utilize role estimation information in AIWolf.

***key words*** AIWolf, Estimated Information, Genetic Programming

# 目次

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なものが多く，勝ち負けなどで評価がしやすいことから AI の研究で長く扱われてきた．ゲーム情報学において，ゲームは将棋やチェスなどの二人完全情報ゲームやガイスターやポーカーなどの不完全情報ゲームというようにそれぞれのゲームの持つ性質によって分類される．将棋やチェスなどの完全情報ゲームにおいては長年研究がなされており，ゲーム AI の強さという分野においては人間のトッププレイヤーを上回る強さの AI が開発されている．一方で，不完全情報ゲームにおいてはバックギャモンやポーカーでトッププレイヤーを上回る強さの AI が開発されているものの，未だ一部のゲームでは弱い AI しか開発されていない．完全情報ゲームと不完全情報ゲームの差は完全情報性であり，これは盤面上にわからない情報があるかどうかということである．完全情報ゲームである将棋やチェスでは盤面のコマの配置や持ち駒といった情報が全て見えているのに対し，ガイスターでは相手の駒，ポーカーでは相手の手札が見えていない．このような不完全情報ゲームにおいてはわからない情報を推定するということが重要であり，強化学習などを用いた様々な推定が行われている．

人狼ゲームは，他者の役職がわからない多人数不完全情報ゲームであり，人狼ゲームの AI は人狼知能と呼ばれる．人狼ゲームにおいて推定は他者の役職に対して行われることが多く，対象がどの程度人狼らしいという様に扱われる．人狼知能では，SVM（Support Vector Machine）を用いた人狼推定 [1]，DNN（Deep Neural Network）を用いた役職推定 [2]，Q 学習と推定情報を用いた人狼知能 [3] など様々なものがある．しかし，これらの研究では得られた役職推定情報をどの様に扱うのが強いのかが明確にされていない．そこで，遺伝的プログラミング（Genetic Programming，以下 GP）に着目した．GP は進化的計算

手法の一種で，関数の集合を用意しておき，それらの関数を組み合わせることで説明可能なプログラムを求める手法である．

本研究では人狼知能において GP を用いて役職推定情報の有効な活用方法について調査し，意思決定の改善を行うことで人狼知能エージェントの性能向上を目指す．また，人狼エージェントの性能向上が見られるルールについて明確化していく．GP によるルールの作成を行うことでルールを人間が理解可能で作成したルールの妥当性について人間が判断することが可能となる．また，人間が実際プレイする際に作成したルールを流用することが可能である．

本論文の構成は次のとおりである．第 2 章では本研究で取り扱う人狼ゲームについての説明を行う．次に，第 3 章では本研究の関連研究や手法について説明する．その後，第 4 章では提案を行う．さらに，第 5 章で今回行う実験について説明し，結果を示したのちに第 6 章にて考察を行う．最後に，第 7 章にて本論文のまとめを行う．

# 第 2 章

# 人狼ゲーム

人狼ゲームは，基本ルールに則って人間同士が対話を行う TRPG（Table Talk Role Playing Game）に分類されるゲームである．人狼ゲームは 1986 年，当時学生だった Dimitry Davidoff 氏によって開発された「mafia」というゲームが発祥とされている.「mafia」は「「情報を得ていない多数派」対「情報を得た少数派」」という基本ルールに則って行われるゲームであり，その後世界中で数多くのルールが加筆修正されたゲームである.「mafia」が開発されて 10 年ほど経った 1997 年頃，アメリカの Andrew Protkin 氏によって「村に紛れ込んでいる人狼」という設定が加えられた．加えて，ルールがよりわかりやすい形に修正され，現在の人狼ゲームが確立された.

人狼ゲームは，村人陣営と人狼陣営に分かれて討論を行うゲームである．このゲームは昼と夜の時間に分かれており，昼の時間では全員で討論を行うことで村から怪しい人物を追放する．この時間では，村人陣営は村に紛れる人狼を話し合いによって特定して追放することを，人狼陣営は追放を免れることを目的として進行していく．夜の時間では会話が制限され，それぞれの役職が持つ固有の能力を使うことができる．一般的には夜の時間からスタートし，夜の時間と昼の時間をどちらかの勝利条件が満たされるまで繰り返される．村人陣営の勝利条件は「村に紛れ込んでいる人狼を全て追放すること」，人狼陣営の勝利条件は「人狼とそれ以外の人数を同数にすること」である.

本研究で扱う人狼は，5 人で行う 5 人人狼と 15 人で行う 15 人人狼である．役職の内訳は 5 人人狼では「村人 2 人，占い師 1 人，人狼 1 人， 狂人 1 人」であり，15 人人狼では「村人 8 人，占い師 1 人，狩人 1 人，霊媒師 1 人，人狼 3 人， 狂人 1 人」である．それぞれの役職の役割，能力は以下の通りである.

- 村人陣営
  - 村人
    - 固有の能力を持たない人で最も人数が多い.
    - 数少ない情報の中から人狼を見つけ出し投票により追放する.
  - 占い師
    - 夜の時間に 1 人を占うことができる.
    - 村人陣営にとって貴重な情報を持つ役職.
    - 人狼に襲撃されることが多い.
  - 狩人（15 人人狼のみ）
    - 夜の時間に 1 人を護衛することができる.
    - 護衛された人は襲撃から守られ，人狼の襲撃対象がその人であればその日に犠牲者は出ない.
  - 霊媒師（15 人人狼のみ）
    - 夜の時間に霊媒結果を得られる.
    - 占い師と同様に，村人陣営にとって貴重な情報を持つ役職.
- 人狼陣営
  - 人狼
    - 夜の時間に 1 人襲撃することができる.
    - 昼の時間は人狼だと疑われないように行動・発言する.
    - 夜の時間では人狼同士のみで会話ができる.
  - 狂人
    - 固有の能力を持たない人.
    - 人狼陣営に所属しており，占い師などの役職を騙る.
    - 場を掻き乱すことで場をかきみだすことで人狼の勝利へと繋げる.

ここで，人狼ゲームには様々な役職があるが，表 2.1 に示すように役職ごとに共通点や相

違点があることに注意する必要がある．

表 2.1　15 人人狼における役職ごとの違い

| 陣営 | 役職 | 勝利条件 | 占い時の判定 |
|---|---|---|---|
| 村人陣営 | 村人 | 人狼を全員追放する | 人狼ではない |
| | 占い師 | 人狼を全員追放する | 人狼ではない |
| | 狩人 | 人狼を全員追放する | 人狼ではない |
| | 霊媒師 | 人狼を全員追放する | 人狼ではない |
| 人狼陣営 | 狂人 | 人狼が半数以上を占める | 人狼ではない |
| | 人狼 | 人狼が半数以上を占める | 人狼 |

近年では人狼知能の強さを競う人狼知能国際大会が開かれており，大会では人狼知能用のプロトコルを用いるプロトコル部門と自然言語を用いる自然言語部門に分かれている．また，前述の不完全情報性に加えて，不確定性や多人数などの多くの性質を持つ．そのため，強い人狼知能の開発，役職の推定 [1][2][4][5][6]，戦略の研究 [7][8] などの様々な研究が行われている．

## 2.1　人狼知能における用語

以下に人狼知能においてよく用いられる用語とその意味について記述する．

- **カミングアウト（CO）**

  自分の役職を公開することであり，基本的に占い師などの固有の能力を持つ役職が行う．また，人狼陣営が他の役職を騙る際にも使われ，時折占い師が 2 人や 3 人カミングアウトすることもあり得る．この時の同じ役職についてカミングアウトしている人を「対抗」と呼ぶ．

- **追放**

  追放の対象となったプレイヤはゲームから除外され，ゲーム終了まで発言や投票，各役

職ごとの固有の行動ができなくなる．対象は後述する「投票」や「襲撃」によって選ばれる．

- **投票**

  昼の時間の最後に行われ，投票で選ばれたプレイヤは追放される．得票数が同数であった場合はもう一度投票を行い，追放者を決定する．これを決選投票と呼ぶが，同票の処理はその際のルールによって異なる．

- **占い**

  占い師の夜の時間の行動である．占い師のプレイヤは 1 人選び，そのプレイヤが人狼か人間であるかを知ることができる．占い師は昼の時間になるとこの情報を村に共有することができ，他のプレイヤはこの情報を誰が人狼かを判断していく．一般的には人狼であれば黒，人間であれば白と報告される．

- **霊媒**

  霊媒師の夜の時間の行動である．霊媒師のプレイヤは，その日の昼の時間に投票によって追放されたプレイヤが人狼か人間かを知ることができる．霊媒師は昼の時間になるとこの情報を村に共有することができ，その他のプレイヤは村に残る人狼の人数を把握することができる．また，追放されたプレイヤが占われていた場合は占いの真偽を判断することができる．

- **護衛**

  狩人の夜の時間の行動である．狩人のプレイヤは 1 人選び，そのプレイヤを人狼の数激から守ることができる．もし人狼が護衛対象を襲撃していた場合，その夜は犠牲者が出ない．ただし，人狼が別のプレイヤを襲撃した場合，通常通り犠牲者が出て犠牲者は追放される．また，狩人は自分を護衛することはできない．

- **襲撃**

  人狼の夜の時間の行動である．人狼は夜の時間に人狼同士で会話し，襲撃先を決定する．襲撃の対象となったプレイヤは護衛されていなければゲームから追放される．また，襲撃によって追放されたプレイヤは，次の昼の時間に通知され，誰が襲撃されたかを知

ることができる.

# 第 3 章

# 関連研究

## 3.1　人狼知能における推定

不完全情報ゲームは，ゲーム内でわからない情報が付随してくる．そのため，わからない情報について仮定や予測をしながらゲームを進めていく必要がある．加えて，人狼ゲームのような多人数ゲームでは，それぞれのプレイヤについて仮定する必要があるため，仮定した情報の正確性がゲームの進行に大きく影響する．そのため，人狼知能では相手の情報を予測する推定の技術に関する研究が盛んである．

梶尾らの研究では，人狼知能大会における統計分析と SVM を用いた人狼推定を行うエージェントの設計に関する研究を行っている [1]．この研究では，SVM を用いた人狼推定を行うエージェントで性能が向上することを示している．彼らはまず，2015 年に行われた人狼知能大会の対戦ログデータを用いて，どのような場合に村人陣営の勝率が上がるかを分析している．その結果，村人側の全ての役職に関して，勝利時の人狼の推定精度が敗北時の人狼の推定精度と比べて有意に高いことを確認している．さらに，SVM によって人狼を推定する人狼知能の作成している．結果，ルールベース AI と比較して人狼の推定精度が 10%近く向上しており，提案手法で人狼を見つけられる可能性が高くなることを示している．また，勝率についてはルールベース AI が 39.22%，SVM による推定 AI が 40.96%，SVM による推定と投票先に関する発話を行う AI が 41.84%であり，SVM を使用することで 1.74%の向上を，投票先に関する発話する場合はさらに 0.88%の向上を確認したと報告されている．

福田らの研究では，15 人人狼における会話情報による役職推定の研究をおこなっている [2]．この研究では，DNN による役職推定による役職の推定率を示している．特徴として，

経過日数，役職推定者の役職，各プレイヤの生死，発言情報のうちの全ての COMINGOUT の発言と他の発言の 100 個を入力した役職推定モデルを提案している．その後，GAT2017 と CEDEC2018 の人狼知能大会のログデータをテストデータとして役職推定情報の正答率を確認している．この時，正答の判定は各日にちが終わったタイミングで行い，3 日目までの正答率を示している．その結果，各人狼知能大会の 1〜3 日目の全てで正答率 98.5%以上を確認したと報告されている．

## 3.2　進化的計算

進化的計算は，生物の遺伝子の複製や突然変異，選択淘汰のメカニズムをもとにした手法である．代表的な進化的計算手法には，遺伝的アルゴリズム（Genetic Algorithms），遺伝的プログラミング（GP），進化的戦略（Evolution Strategy），進化的プログラミング（Evolutionary Programming）がある [9].

### 3.2.1　遺伝的プログラミング（GP）

遺伝的プログラミング（GP）とは，関数の集合を用意しておき，それらの関数を組み合わせることで説明可能なプログラムを求める手法である [10]．GP では図 3.1 のように遺伝子構造を木構造で表しており，各個体が実際のプログラムを指すように構築される．ランダムな個体の集合を初期個体とし，それらの中から選択を繰り返すことで次世代を作成する．通常，次世代の生成においては，適応度の高い個体が選択される傾向がある．この GP の過程で交叉（図 3.2）や突然変異（図 3.3）といった遺伝的操作が発生し，それらを適応させることでプログラムを進化させていく．以下に GP の基本的な過程を簡単に示す.

1. 初期個体集合を生成
2. 各個体の適応度を計測
3. 初期個体の中から適応度を元に選択し，次世代とする
4. （3）について遺伝的操作を適応

5. 適応度の変更が発生したものについて適応度の計測
6. 完成した次世代個体を親個体として次世代のループへ進む準備をする
7.（3）～（7）を世代数繰り返す

図 3.1 に木構造の簡単な例を示す．この例は，”if(第一引数，第二引数，第三引数)”による木構造であり，第一引数には条件式（一番左のノード），第二引数には条件が true の時の行動や意思決定（中央のノード），第三引数には条件が false の時の行動や意思決定（一番右のノード）となるように実装する．

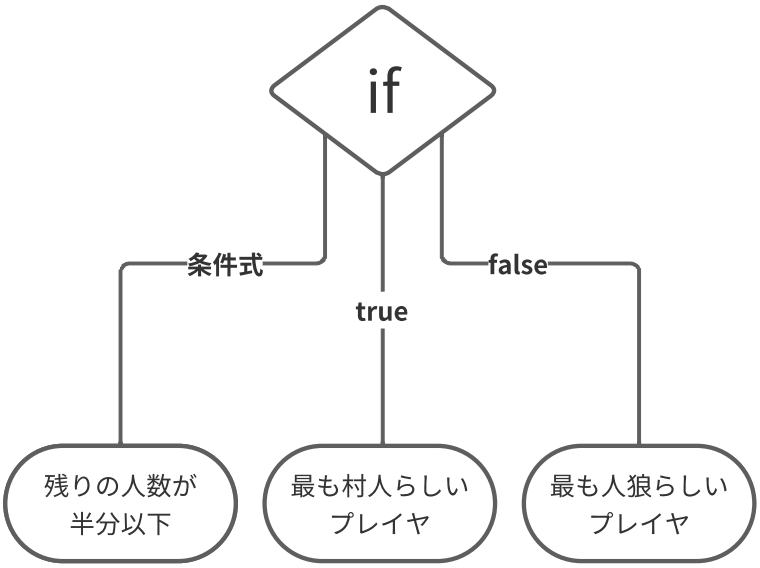


図 3.1　実装する木構造の一例

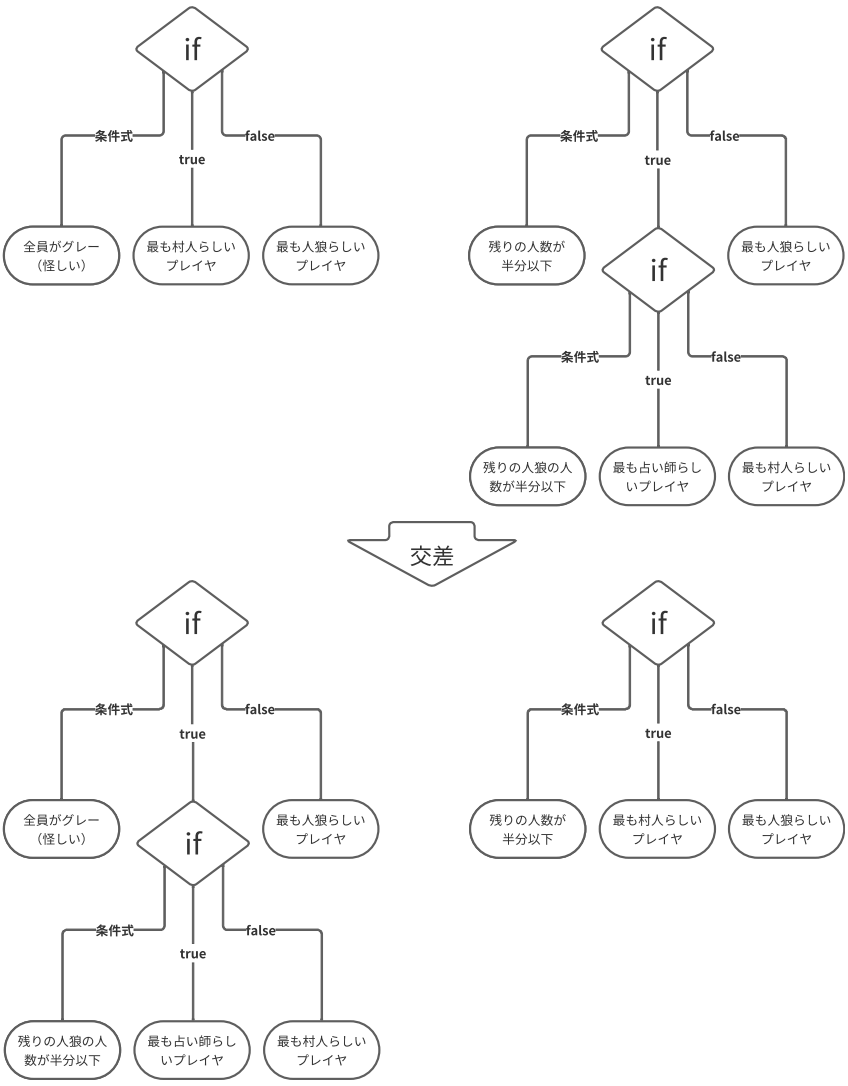


図 3.2　交叉の例

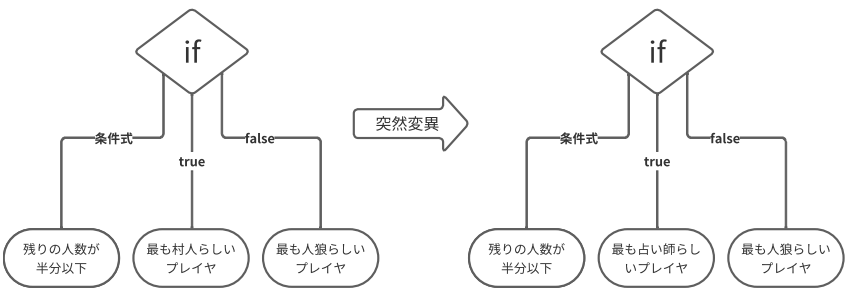


図 3.3　突然変異の例

## 3.3　GP に関する研究

Alhejali らの研究では，完全情報ゲームである Ms Pac-Man において GP を利用し，性能の向上を目指している [11]．この研究では，平均スコアにおいて 19,154.7 から 22,625.1 の約 18%の性能向上が見られたと報告されている．ただし，Pac-Man と人狼ゲームの最大の違いは完全情報性にあり，この違いに対処するため，既存の役職推定手法を活用することを考える．また，Pac-Man の研究では，GP がモンテカルロ木探索に適用されているが，本研究では探索を伴わないルールの構築を目指していく．

栃川らの研究では，二人不完全情報ゲームであるガイスターにおいて，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している [12]．この研究では，GP を用いて方策を作成し，最大勝率 82.5%の性能を確認している．しかし，解釈が難しく，人の手による最適化が必要な方策も存在していたと報告されている．

# 第 4 章

# 提案手法

本研究では人狼知能において GP を用いて役職推定情報の有効な活用方法について調査し，意思決定の改善を行うことで人狼知能エージェントの性能向上を目指す．第 2 章で述べたように，人狼ゲームには主に「投票」「占い」「霊媒」「護衛」「襲撃」5 つの行動があり，これら全てに GP を適用することが可能であると考えられる．ただし，今回の研究では第 5 回人狼知能国際大会において，1 位のエージェント以外で他の役職と比較して勝率が低かった人狼に焦点を当てて実験を行う [13]．さらに，ルールを作成する際には行動ごとに最適なルールを生成するために対象の行動以外は変更を行わずに実験する．また，GP では生成したルールで対戦を行い，その勝率を適応度として使用する．最終的に GP で生成されたルールの人狼知能を用いて対戦実験を行い，性能を評価する．

役職推定情報を用いる例として,「人狼陣営が投票を行う場合」を考える．このとき，人狼らしさが 1 位以外に投票すると疑われてしまうが，人狼らしさ 1 位と 2 位の間に大きな差がなければ 2 位に投票してもあまり疑われない．その結果，仲間を助けることで勝率の向上を目指すといった方法がある．

## 4.1 GP で用いた関数セット

本研究では以下に示す関数セットを使用する．

### 4.1.1　非終端子

以下に非終端子についての説明を記述する．if_then_else は木のベースに用いられ，残りについては if_then_else の条件式として扱われる．また，

- if_then_else

  木のベースには IF_ELSE 文を主に用いる．第 1 引数に条件式，第 2 引数と第 3 引数で条件式の結果に応じて行動を行う対象を決める．
- 比較演算子

  比較演算子として，$>$，$\geq$，$<$，$\leq$，$=$ と $\neq$ を用いる．第 1 引数と第 2 引数に比較する対象を持つ．
- 算術演算子

  算術演算子として，$+$，$-$，$*$ を用いる．第 1 引数と第 2 引数に int 型または float 型を受け取り，対応した計算を行う．ただし，int 型は int 型同士，float 型は float 型同士で計算を行う．また，/ の除算に関しては 0 除算が発生する恐れがあるため除外した．
- 論理演算子

  論理演算子として，&&，|| を用いる．第 1 引数と第 2 引数に対象を持つ．

### 4.1.2　役職推定情報を用いる終端子

これらは役職推定情報を用いる終端子であり，返り値の型は str 型，float 型，bool 型である．以下では，それぞれの返り値の型ごとに説明している．

#### 終端子：対象となるエージェントを返す

これらは主に if_then_else の引数として扱われる．以下に各終端子についての説明を記述する．これらは，投票における投票先というような行動における対象となるものとなる．GP のコードを生成する箇所では str 型で返すが，実際のコードでは人狼知能特有の Agent

クラスの関数である．また，それぞれの関数に関して，味方を含めたもの，味方を含めないもの，味方のみの 3 通りで取得を行う関数を作成した．

- **人狼らしさが最大のエージェントを取得（getMaxWerewolfAgent）**

  str 型の値を返す．本関数は SVM によって得られた人狼らしさが最大のエージェントを返すように実装した．これを return 文で返すために str 型の値で返す．
- **人狼らしさが 2 番目に大きいエージェントを取得（getMax2ndWerewolfAgent）**

  str 型の値を返す．本関数は SVM によって得られた人狼らしさが 2 番目に大きいのエージェントを返すように実装した．これを return 文で返すために str 型の値で返す．
- **人狼らしさが 3 番目に大きいエージェントを取得（getMax3rdWerewolfAgent）**

  str 型の値を返す．本関数は SVM によって得られた人狼らしさが 3 番目に大きいのエージェントを返すように実装した．これを return 文で返すために str 型の値で返す．
- **人狼らしさが最小のエージェントを取得（getMinWerewolfAgent）**

  str 型の値を返す．本関数は SVM によって得られた人狼らしさが最小のエージェントを返すように実装した．これを return 文で返すために str 型の値で返す．
- **人狼らしさが 2 番目に小さいエージェントを取得（getMin2ndWerewolfAgent）**

  str 型の値を返す．本関数は SVM によって得られた人狼らしさが 2 番目に小さいのエージェントを返すように実装した．これを return 文で返すために str 型の値で返す．
- **人狼らしさが 3 番目に小さいエージェントを取得（getMin3rdWerewolfAgent）**

  str 型の値を返す．本関数は SVM によって得られた人狼らしさが 3 番目に小さいのエージェントを返すように実装した．これを return 文で返すために str 型の値で返す．
- **村人らしさが最大のエージェントを取得（getMaxVillagerAgent）**

  str 型の値を返す．本関数は SVM によって得られた村人らしさが最大のエージェントを返すように実装した．これを return 文で返すために str 型の値で返す．
- **村人らしさが最小のエージェントを取得（getMinVillagerAgent）**

  str 型の値を返す．本関数は SVM によって得られた村人らしさが最小のエージェント

を返すように実装した．これを return 文で返すために str 型の値で返す．

- **占い師らしさが最大のエージェントを取得（getMaxSeerAgent）**

  str 型の値を返す．本関数は SVM によって得られた占い師らしさが最大のエージェントを返すように実装した．これを return 文で返すために str 型の値で返す．

- **占い師らしさが最小のエージェントを取得（getMinSeerAgent）**

  str 型の値を返す．本関数は SVM によって得られた占い師らしさが最小のエージェントを返すように実装した．これを return 文で返すために str 型の値で返す．

- **霊媒師らしさが最大のエージェントを取得（getMaxMediumAgent）**

  str 型の値を返す．本関数は SVM によって得られた霊媒師らしさが最大のエージェントを返すように実装した．これを return 文で返すために str 型の値で返す．

- **霊媒師らしさが最小のエージェントを取得（getMinMediumAgent）**

  str 型の値を返す．本関数は SVM によって得られた霊媒師らしさが最小のエージェントを返すように実装した．これを return 文で返すために str 型の値で返す．

- **狩人らしさが最大のエージェントを取得（getMaxBodyguardAgent）**

  str 型の値を返す．本関数は SVM によって得られた狩人らしさが最大のエージェントを返すように実装した．これを return 文で返すために str 型の値で返す．

- **狩人らしさが最小のエージェントを取得（getMinBodyguardAgent）**

  str 型の値を返す．本関数は SVM によって得られた狩人らしさが最小のエージェントを返すように実装した．これを return 文で返すために str 型の値で返す．

- **狂人らしさが最大のエージェントを取得（getMaxPossessedAgent）**

  str 型の値を返す．本関数は SVM によって得られた狂人らしさが最大のエージェントを返すように実装した．これを return 文で返すために str 型の値で返す．

- **狂人らしさが最小のエージェントを取得（getMinPossessedAgent）**

  str 型の値を返す．本関数は SVM によって得られた狂人らしさが最小のエージェントを返すように実装した．これを return 文で返すために str 型の値で返す．

- **ランダムなエージェントを取得する（getRandomAgent）**

str 型の値を返す．本関数は生存しているエージェントからランダムなエージェントを返すように実装した．これを return 文で返すために str 型の値で返す．

**終端子：対象候補となるエージェントの役職らしさの値を返す**

これらは主に算術演算子，比較演算子の引数として扱われる．以下に各終端子についての説明を記述する．それぞれの関数について，味方を含めたもの，味方を含めないもの，味方のみの 3 通りで取得を行う関数を作成した．

- **エージェントの中で最も高い人狼らしさの値を取得（getMaxWerewolfRate）**
  float 型の値を返す．本関数は SVM によって得られた人狼らしさの値のうち最大の値を返すように実装した．
- **エージェントの中で 2 番目に高い人狼らしさの値を取得（getMax2ndWerewolfRate）**
  float 型の値を返す．本関数は SVM によって得られた人狼らしさの値のうち 2 番目に大きい値を返すように実装した．
- **エージェントの中で 3 番目に高い人狼らしさの値を取得（getMax3rdWerewolfRate）**
  float 型の値を返す．本関数は SVM によって得られた人狼らしさの値のうち 3 番目に大きい値を返すように実装した．
- **エージェントの中で最も低い人狼らしさの値を取得（getMinWerewolfRate）**
  float 型の値を返す．本関数は SVM によって得られた人狼らしさの値のうち最小の値を返すように実装した．
- **エージェントの中で 2 番目に低い人狼らしさの値を取得（getMin2ndWerewolfRate）**
  float 型の値を返す．本関数は SVM によって得られた人狼らしさの値のうち 2 番目に小さい値を返すように実装した．
- **エージェントの中で 3 番目に低い人狼らしさの値を取得（getMin3rdWerewolfRate）**
  float 型の値を返す．本関数は SVM によって得られた人狼らしさの値のうち 3 番目を小さい値を返すように実装した．

- **エージェントの中で最も高い村人らしさの値を取得（getMaxVillagerRate）**

  float 型の値を返す．本関数は SVM によって得られた村人らしさの値のうち最大の値を返すように実装した．
- **エージェントの中で最も低い村人らしさの値を取得（getMinVillagerRate）**

  float 型の値を返す．本関数は SVM によって得られた村人らしさの値のうち最小の値を返すように実装した．
- **エージェントの中で最も高い占い師らしさの値を取得（getMaxSeerRate）**

  float 型の値を返す．本関数は SVM によって得られた占い師らしさの値のうち最大の値を返すように実装した．
- **エージェントの中で最も低い占い師らしさの値を取得（getMinSeerRate）**

  float 型の値を返す．本関数は SVM によって得られた占い師らしさの値のうち最小の値を返すように実装した．
- **エージェントの中で最も高い霊媒師らしさの値を取得（getMaxMediumRate）**

  float 型の値を返す．本関数は SVM によって得られた霊媒師らしさの値のうち最大の値を返すように実装した．
- **エージェントの中で最も低い霊媒師らしさの値を取得（getMinMediumRate）**

  float 型の値を返す．本関数は SVM によって得られた霊媒師らしさの値のうち最小の値を返すように実装した．
- **エージェントの中で最も高い狩人らしさの値を取得（getMaxBodyguardRate）**

  float 型の値を返す．本関数は SVM によって得られた狩人らしさの値のうち最大の値を返すように実装した．
- **エージェントの中で最も低い狩人らしさの値を取得（getMinBodyguardRate）**

  float 型の値を返す．本関数は SVM によって得られた狩人らしさの値のうち最小の値を返すように実装した．
- **エージェントの中で最も高い狂人らしさの値を取得（getMaxPossessedRate）**

  float 型の値を返す．本関数は SVM によって得られた狂人らしさの値のうち最大の値

を返すように実装した．

- **エージェントの中で最も低い狂人らしさの値を取得（getMinPossessedRate）**
  float 型の値を返す．本関数は SVM によって得られた狂人らしさの値のうち最小の値を返すように実装した．

#### 終端子：条件を満たすかどうかの判定を行う

これらは主に if_then_else 文の条件式の引数として扱われる．以下に各終端子についての説明を記述する．

- **全てのプレイヤがグレーである判定（allGray）**
  bool 型を返す．本関数は生存している全てのプレイヤが怪しいかを判定するように実装した．
- **全てのプレイヤがグレーではない判定（notAllGray）**
  bool 型を返す．本関数は生存している全てのプレイヤが怪しくないと判定するように実装した．

### 4.1.3　役職推定情報を用いない終端子

これらは役職推定情報を用いない終端子であり，返り値の型は str 型，float 型，bool 型である．以下では，それぞれの返り値の型ごとに説明している．

#### 終端子：対象となるエージェントを返す

これらは主に if_then_else の引数として扱われる．以下に各終端子についての説明を記述する．これらは，投票における投票先というような行動における対象となるものとなる．GP のコードを生成する箇所では str 型で返すが，実際のコードでは人狼知能特有の Agent クラスの関数である．

- **自分から人狼であるという嘘の判定を受けたエージェントを取得 (getFakeJudgeAgent)**

  str 型の値を返す．本関数は自分が占い師もしくは霊媒師として CO している時に人狼であるというと嘘の判定を行われたエージェントを返すように実装した．もし複数人いる場合は対象からランダムで返すように実装した．これを return 文で返すために str 型の値で返す．これは狂人または人狼の際のみ実装される．

- **対抗として CO したエージェントを取得（getOppositionAgent）**

  str 型の値を返す．本関数は自分が占い師もしくは霊媒師として CO している時に対抗として同じ役職を CO したエージェントを返すように実装した．もし複数人いる場合は対象からランダムで返すように実装した．これを return 文で返すために str 型の値で返す．これは狂人または人狼の際のみ実装される．

**終端子：対象候補となるエージェントの役職らしさの値を返す**

これらは主に算術演算子，比較演算子の引数として扱われる．以下に各終端子についての説明を記述する．また，これらの終端子の他に 1 刻みの int 型の定数（15 1）と 0.1 刻みの float 型の定数（1.0 0.0）を定義している．

- **現在の日にちを取得 (getDay)**

  int 型の値を返す．本関数はゲーム内で取得可能な現在の日にちの値を返すように実装した．

- **現在の村の残り人数を取得 (getPlayerNum)**

  int 型の値を返す．本関数はゲーム内で取得可能な現在の村の残り人数を返すように実装した．

- **現在の人狼の残り人数を取得 (getWerewolfNum)**

  int 型の値を返す．本関数はゲーム内で取得可能な現在の人狼の残り人数を返すように実装した．これは人狼の際のみ実装される．

**終端子：条件を満たすかどうかの判定を行う**

これらは主に if_then_else 文の条件式の引数として扱われる．以下に各終端子についての説明を記述する．

- **残り人数が半分以下かどうかの判定（isHalfPlayer）**

  bool 型の値を返す．本関数は生存している人数が最初の人数の半分以下であるかを判定するように実装した．15 人人狼なので，7 人以下であれば true，8 人以上であれば false を返す．
- **自分が占い師として嘘の CO をしているかの判定（isSeerCO）**

  bool 型を返す．本関数は自分が占い師として嘘の CO をしているかを判定するように実装した．占い師として嘘の CO をしていれば true，そうでなければ false を返す．
- **自分が霊媒師として嘘の CO をしているかの判定（isMediumCO）**

  bool 型を返す．本関数は自分が霊媒師として嘘の CO をしているかを判定するように実装した．霊媒師として嘘の CO をしていれば true，そうでなければ false を返す．
- **自分が他エージェントに対して人狼であるという嘘の判定をしたか（isFakeJudge）**

  bool 型の値を返す．本関数は自分が占い師または霊媒師を CO していて他エージェントに対して人狼であるという嘘の判定をしていれば true，そうでなければ false を返すように実装した

## 4.2　GP の適応度

本研究では，GP によって生成されたコードを元にした人狼知能を作成し，作成された人狼知能を使って対戦結果を適応度とした．この適応度の計算には 1000 ゲームを用いた．作成された個体の勝率を進化的計算手法の適応度とした研究には上田らのオセロ AI がある [14].

### 4.2.1　役職の固定と会話の偏り

人狼ゲームの勝率を測る際，人狼ゲームの持つ不確定性という性質の影響により，役職がランダムに分配されてしまい，適応度計算に使用するゲーム数が変動することがある．そこで人狼知能のサーバによる設定で作成したエージェントを対象とする役職で固定し，適応度計算に使用するゲーム数を固定しようと試みた．しかし，人狼の投票に関する GP を行っていた際に表 4.1 に示すように，人狼を固定したことで作成したエージェントの初日追放率が増加していた．

表 4.1　人狼固定時の勝率と初日追放率の変化

| | 人狼を固定する | 人狼を固定しない |
|---|---|---|
| 勝率 | 0.33 | 0.34 |
| 初日追放率 | 0.14 | 0.28 |

そこで，対戦相手の人狼知能エージェントについて分析したところ，人狼知能大会上位のエージェントは第二回人狼知能国際大会の takeda エージェントをもとにしているものが多かった．この takeda エージェントでは過去の会話の情報を記録し，記録した情報からエージェントの役職を推定している．

takeda エージェントを含めた過去大会のエージェント 6 種 15 エージェントを用いて 15 人人狼を 500000 ゲーム行い，その会話情報についての分析を行った．その結果が図 4.1 であり，この結果ではいづれのエージェントからも発言がなかった”AGREE”，”DISAGREE”，”DIVINATION”，”GUARD”，”ATTACK”及び，発言を飛ばすためだけの発言である”Skip”，”Over”については除外して示している．たとえば，”ESTIMATE”という発言は ioh，Sashimi，Basket の 3 エージェントは発言しているが，daisyo，Tomato，Takeda の 3 エージェントは発言していないことがわかる．このようにエージェントによって発言する内容が変わっており，takeda ベースのエージェントはこれらようなの会話の特徴を用いて判定していると考えられる．

## 4.2 GP の適応度

| Agent | Role | COMINGOUT | DIVINED | ESTIMATE | GUARDED | IDENTIFIED | REQUEST | VOTE | AND GUARDED |
|---|---|---|---|---|---|---|---|---|---|
| daisyo | BODYGUARD | 0 | 0 | 0 | 0 | 0 | 0 | 442547 | 0 |
| | MEDIUM | 66693 | 0 | 0 | 0 | 0 | 0 | 282430 | 0 |
| | POSSESSED | 66789 | 196275 | 0 | 0 | 0 | 0 | 246651 | 0 |
| | SEER | 66556 | 167239 | 0 | 0 | 0 | 0 | 226054 | 0 |
| | VILLAGER | 0 | 0 | 0 | 0 | 0 | 0 | 3612715 | 0 |
| | WEREWOLF | 199907 | 315984 | 0 | 0 | 0 | 0 | 1814652 | 0 |
| Tomato | BODYGUARD | 0 | 0 | 0 | 0 | 0 | 0 | 386787 | 0 |
| | MEDIUM | 66405 | 0 | 0 | 0 | 136548 | 0 | 286784 | 0 |
| | POSSESSED | 70253 | 185799 | 0 | 0 | 0 | 0 | 370178 | 0 |
| | SEER | 66146 | 183424 | 0 | 0 | 0 | 0 | 245956 | 0 |
| | VILLAGER | 0 | 0 | 0 | 0 | 0 | 0 | 3064110 | 0 |
| | WEREWOLF | 92284 | 288464 | 0 | 0 | 0 | 0 | 2105094 | 0 |
| ioh | BODYGUARD | 61002 | 0 | 933202 | 0 | 0 | 1248780 | 936463 | 64943 |
| | MEDIUM | 99933 | 0 | 1127393 | 0 | 210028 | 843118 | 847098 | 0 |
| | POSSESSED | 66107 | 5623 | 1051310 | 0 | 63465 | 889471 | 788840 | 36769 |
| | SEER | 100008 | 248402 | 854454 | 0 | 0 | 641847 | 639309 | 0 |
| | VILLAGER | 0 | 0 | 11869152 | 0 | 0 | 8906778 | 8904070 | 0 |
| | WEREWOLF | 299751 | 148453 | 2921977 | 139038 | 83143 | 2190833 | 2191745 | 0 |
| Sashimi | BODYGUARD | 66433 | 146239 | 38292 | 0 | 0 | 0 | 1225336 | 0 |
| | MEDIUM | 66571 | 148646 | 42548 | 0 | 0 | 0 | 1243675 | 0 |
| | POSSESSED | 66645 | 146946 | 48321 | 0 | 0 | 0 | 1346008 | 0 |
| | SEER | 67060 | 149602 | 50243 | 0 | 0 | 0 | 1233425 | 0 |
| | VILLAGER | 533505 | 1239477 | 323422 | 0 | 0 | 0 | 10414366 | 0 |
| | WEREWOLF | 199786 | 514191 | 285092 | 0 | 0 | 0 | 4157981 | 0 |
| Basket | BODYGUARD | 57233 | 0 | 923318 | 0 | 0 | 1231393 | 924565 | 61651 |
| | MEDIUM | 100129 | 0 | 1156281 | 0 | 217285 | 866493 | 866662 | 0 |
| | POSSESSED | 64504 | 5067 | 1038175 | 0 | 62362 | 876765 | 778445 | 36112 |
| | SEER | 99974 | 243655 | 837426 | 0 | 0 | 627954 | 627541 | 0 |
| | VILLAGER | 0 | 0 | 11708959 | 0 | 0 | 8781166 | 8780545 | 0 |
| | WEREWOLF | 298480 | 150392 | 2935820 | 138859 | 83948 | 2200978 | 2204943 | 0 |
| Takeda | BODYGUARD | 0 | 0 | 0 | 0 | 0 | 0 | 680992 | 0 |
| | MEDIUM | 99678 | 0 | 0 | 0 | 223513 | 0 | 431743 | 0 |
| | POSSESSED | 99766 | 225920 | 0 | 0 | 0 | 0 | 279611 | 0 |
| | SEER | 100256 | 198226 | 0 | 0 | 0 | 0 | 1683778 | 0 |
| | VILLAGER | 0 | 0 | 0 | 0 | 0 | 0 | 5493534 | 0 |
| | WEREWOLF | 300390 | 603933 | 0 | 0 | 0 | 0 | 2942157 | 0 |

図 4.1　会話情報の分析結果

### 4.2　GP の適応度

以上の分析結果から，人狼を固定してしまうと初日の昼の議論における会話から人狼と判定され，結果として初日追放率の増加を招いていたことがわかった．本研究では，純粋な役職推定情報の活用方法について明らかにしたかったため，対戦相手の人狼知能エージェントの会話情報を記録しないようにして，適応度の計算及び評価実験を実施する．

# 第 5 章

# 実験

本実験では，GP を用いてルールの構築し，完成したルールを用いた人狼知能を用いて対戦実験を行う．実験は，以下の 3 通りで実施した．

- **実験 1**

  15 人人狼を対象とし，GP は個体数 100，世代数 30 で 4.1 節の関数セットのうち CO 及び嘘の結果を扱っておらず，役職推定の結果については味方も含めた全員を対象としているもののみを扱っている．また，人狼エージェントを 1 体 SVM で固定し，残りの役職はランダムなエージェントを割り振りって適応度の計算を行う．実験 1 については，同様の実験を 4 回行い結果を確認した．

- **実験 2**

  15 人人狼を対象とし，個体数 50，世代数 30 で関数セットは 4.1 節のものを全て扱っている．また，人狼エージェントを 3 体全てを SVM で固定し，残りの役職はランダムなエージェントを割り振りって適応度の計算を行う．実験 2 については，同様の実験を 1 回行い結果を確認した．

- **実験 3**

  5 人人狼を対象とし，個体数 100，世代数 30 で 4.1 節の霊媒師及び狩人に関する関数セットを除いたものを扱っている．また，人狼エージェントを 1 体 SVM で固定し，残りの役職はランダムなエージェントを割り振りって適応度の計算を行う．実験 3 については，同様の実験を 4 回行い結果を確認した．

今回生成する意思決定ルールでは，SVM による役職推定の結果を用いる．今回使用する SVM エージェントは狩野らの「人狼知能で学ぶ AI プログラミング」をもとに作成した [15]．この SVM エージェントは村人の投票時の人狼の推定のみで SVM の推定結果を用いて，最も人狼らしい人に投票を行っている．そのため，人狼の投票や襲撃，占いや護衛等の行動では SVM の結果を用いていない．また，SVM の教師データには過去大会の人狼知能エージェントの対戦結果を使用し，500000 ゲーム行ったログより 1000 ゲーム分のログをランダムに選び使用した．

評価のための対戦実験では，GP によって生成された人狼知能のルールが各役職の勝率に与える影響を検証するために，15 人人狼のゲームにおいて人狼エージェントを全て GP によって作成したもので固定して対戦した．また，評価実験における比較対象として人狼が味方以外のプレイヤの中で最も人狼らしい人に投票するエージェント（max），味方も含めた全てのプレイヤのうちランダムに投票するエージェント（random）の 2 つを作成した．これらのエージェントは投票行動以外は GP のエージェントと同じものを使用している．

## 5.1　実験結果

### 5.1.1　実験 1

図 5.1 は実験 1 において投票行動に GP を適用した際の適応度の推移を表している．グラフは横軸が世代数，縦軸は適応度である勝率を表している．GP1-1，GP1-3，GP1-4 の結果では，適応度の更新が起こっていた．一方で，GP1-2 の結果では，初期化個体で生成されたルールの勝率が高く，適応度の更新が起こっていなかった．

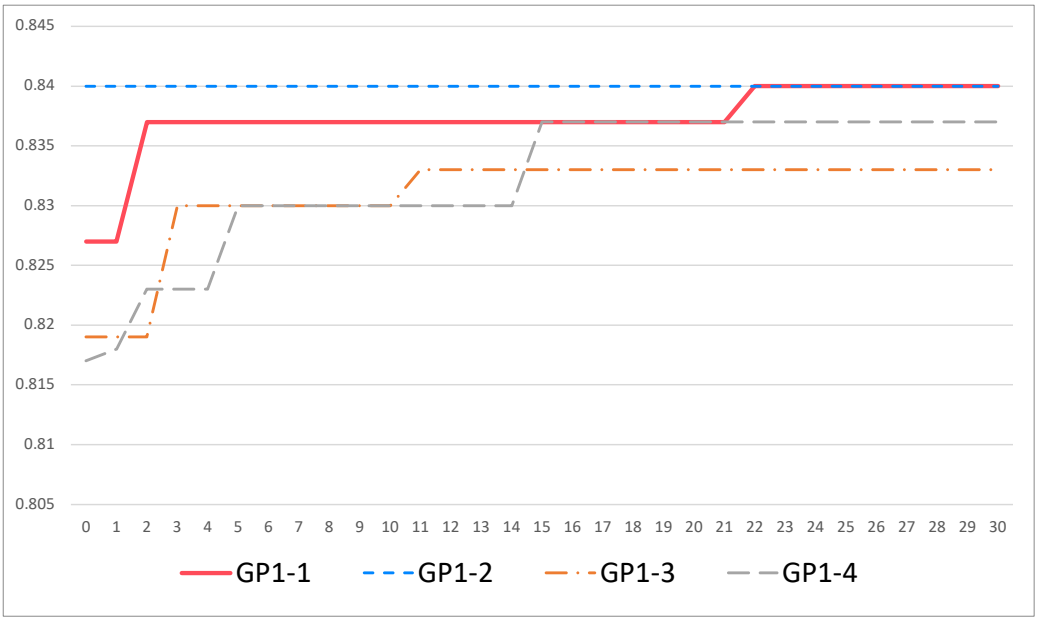


図 5.1　実験 1 における投票行動 GP の適応度の変化

表 5.1 は GP によって得られたルールで対戦実験を行った結果である．表 5.1 内のラベルは図 5.1 のグラフ内のラベルと一致している．GP1-2 は max と random の双方と比べて信頼区間 99%で統計的に有意な差が見られたが，GP1-1，GP1-3，GP1-4 は max と比べて信頼区間 95%でも統計的に有意な差が見られなかった．

表 5.1　実験 1 における評価実験の結果

| | max | random | GP1-1 | GP1-2 | GP1-3 | GP1-4 |
|---|---|---|---|---|---|---|
| 勝率 | 0.1793 | 0.1768 | 0.1839 | 0.1952 | 0.1786 | 0.1836 |

実験 1 で得られたルールをアルゴリズム 1，2，3，4 に示す.

**Algorithm 1** GP1-1 において最も適応度が高かったルール

```
if getMaxWerewolfRate <= getMinVillagerRate then
    if allGray then
        getMinMediumAgent
    else
        getMinWerewolfAgent
    end if
else
    if isHalfPlayer then
        getMaxBodyguardAgent
    else
        getMinVillagerAgent
    end if
end if
```

**Algorithm 2** GP1-2 において最も適応度が高かったルール

```
if isHalfPlayer then
    getMinVillagerAgent
else
    getMaxVillagerAgent
end if
```

**Algorithm 3** GP1-3 において最も適応度が高かったルール

```
if allGray then
    getMaxSeerAgent
else
    getMinVillagerAgent
end if
```

**Algorithm 4** GP1-4 において最も適応度が高かったルール

```
getMinBodyguardAgent
```

### 5.1.2　実験 2

図 5.2 は実験 2 において投票行動に GP を適用した際の適応度の推移を表している．グラフは横軸が世代数，縦軸は適応度である勝率を表している．図 5.2 の結果において，適応度が一度減少しているものの，全体の傾向としては徐々に増加していることがわかる．

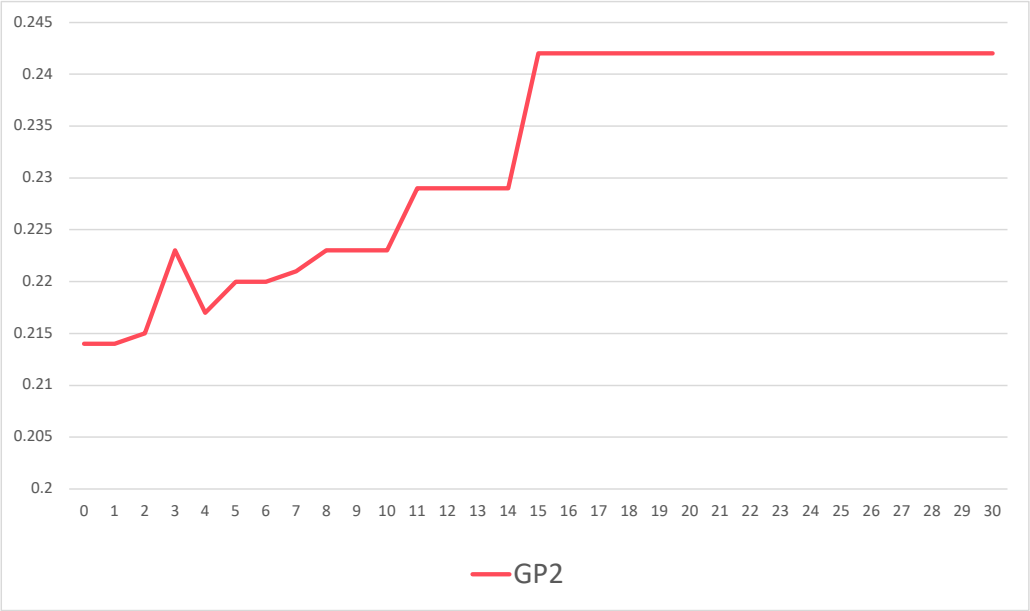


図 5.2　実験 2 における投票行動 GP の適応度の変化

表 5.2 は GP によって得られたルールで対戦実験を行った結果である．GP2 は max, random と比べて信頼区間 95%でも統計的に有意な差が見られなかった．

表 5.2　実験 2 における評価実験の結果

| | max | random | GP2 |
|---|---|---|---|
| 勝率 | 0.1793 | 0.1768 | 0.184 |

勝率のよかった方策をアルゴリズム 5 に示す．実験 2 からは，どのような状況においても「最も村人らしいエージェントに投票を行う」というルールが得られた．

**Algorithm 5** GP2 において最も適応度が高かったルール

```
getMaxVillagerAgent
```

### 5.1.3　実験 3

図 5.3 は実験 3 において投票行動に GP を適用した際の適応度の推移を表している．グラフは横軸が世代数，縦軸は適応度である勝率を表している．

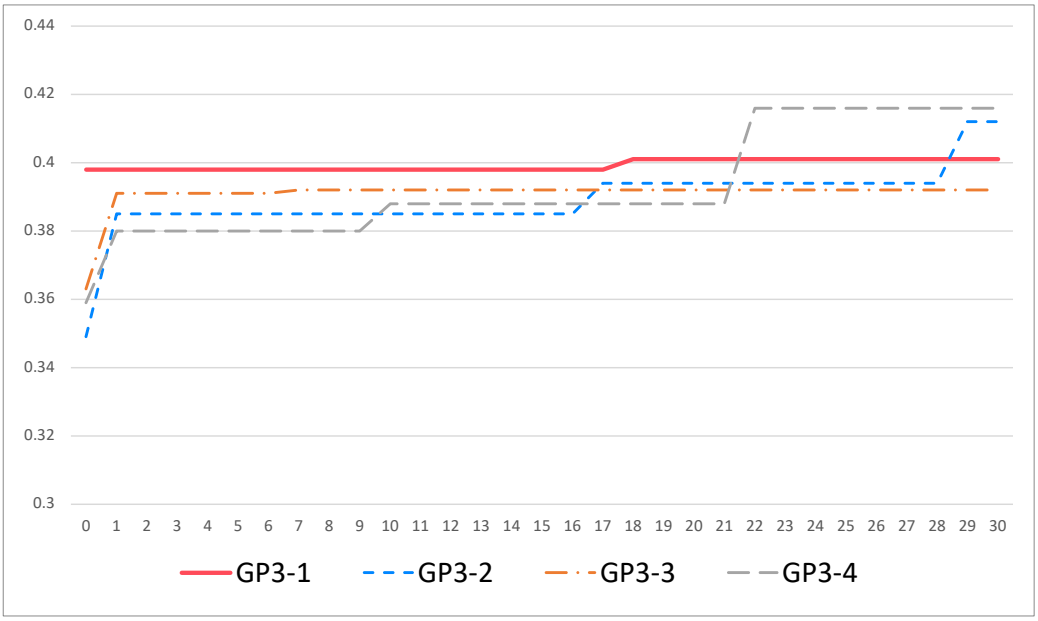


図 5.3　実験 3 における投票行動 GP の適応度の変化

表 5.3 は GP によって得られたルールで対戦実験を行った結果である．表 5.3 内のラベルは図 5.3 のグラフ内のラベルと一致している．これら全ての結果において，max と random の双方と比べて信頼区間 99%で統計的に有意な差が見られた．

表 5.3　実験 3 における評価実験の結果

| | max | random | GP3-1 | GP3-2 | GP3-3 | GP3-4 |
|---|---|---|---|---|---|---|
| 勝率 | 0.2356 | 0.1908 | 0.2859 | 0.2874 | 0.2666 | 0.2903 |

実験 3 で得られたルールをアルゴリズム 6，7，8，9 に示す．アルゴリズム 9 の後半箇所に関しては，else 以前と else 以降が完全に同じであったため省略している．また，実験 3 で得られたルールにおいては 1 つのルールの中で重複するルールや条件分岐によって実行されないルールが多く発生し，それらのコードに関しては人の手で排除する必要がある．

**Algorithm 6** GP3-1 において最も適応度が高かったルール

```
if isFakeJudge then
    getMinSeerNoAllyAgent
else
    getMaxPossessedAgent
end if
```

**Algorithm 7** GP3-2 において最も適応度が高かったルール

```
if getMax3rdWerewolfNoAllyRate <getMaxWerewolfRate then
    if allGray then
        getMinSeerNoAllyAgent
    else
        getMinPossessedAgent
    end if
else
    getMax3rdWerewolfAgent
end if
```

**Algorithm 8** GP3-3 において最も適応度が高かったルール

```
if getPlayerNum != 15 then
    if allGray then
        getMinVillagerAllyAgent
    else
        getMin2ndWerewolfAllyAgent
    end if
else
    if isSeerCO then
        getMax3rdWerewolfNoAllyAgent
    else
        getRandomNoAllyAgent
    end if
end if
```

**Algorithm 9** GP3-4 において最も適応度が高かったルール

```
if 0.7 != getMinSeerAllyRate then
    if allGray then
        if allNotGray ||isSeerCO then
            if isFakeJudge then
                getMinSeerAgent
            else
                getMaxPossessedNoAllyAgent
            end if
        else
            if isFakeJudge then
                getMax2ndWerewolfAllyAgent
            else
                getFakeJudgeAgent
            end if
        end if
    else
        getMaxPossessedAgent
    end if
else
    省略
end if
```

# 第 6 章

# 考察

## 6.1 GP について

### 6.1.1 15 人人狼

実験 1 の適応度においては，図 5.1 に示された適応度の推移では一度も変化していない結果が確認できる．これの原因として，関数セットの多様性の低さが考えられる．特に，図 5.1 の実験 1 では純粋な SVM の利用方法を確認したかったため，CO の有無や嘘の判定結果，味方を含めるか否かなどの差異を実装していなかったため，関数セットが十分に多様ではなかった．その結果，GP を行って各個体ごとに大きな差が生まれず，今回のように適応度の更新がほとんど行われない結果となったのではないかと考えられる．一方で，実験 2 の図 5.2 では全ての世代で更新されてはいないが，徐々に適応度が更新されていることが確認できる．本研究では SVM の推定情報をどのように使うのかを主目的としており，関数セットには SVM の情報を扱うものをメインに追加しているが，実際にはその他の情報も多岐に渡り，推定情報以外の情報についても扱うことができる．これらの推定を扱わない情報と本研究で得られたようなルールを組み合わせて扱うことでより優れたルールの生成ができると考えられる．

実験 1 の結果から，勝率が向上していたものの，統計的に有意な差は確認できず，GP のによって得られたルールで性能が向上したことはいえなかった．これらの原因の一つとして，適応度計算手法の問題が考えられる．実験 1 では人狼エージェントを 1 体 SVM で固定し，残りの役職はランダムなエージェントを割り振りって 1000 ゲームを行い，その結果を適応

度として用いている．しかし，この結果には他のプレイヤの影響が大きく，この 1000 ゲームで良い乱数を引き続けた場合は期待を上回る適応度を得られ，その個体を最適解としてしまう問題がある．この問題を解決するためには適応度計算手法を見直す必要があり，解決適応度の再計算を行い，古い適応度と新たな適応度の平均をとることで解決可能であると考える．これによりその個体における期待を上回る適応度や期待を下回る適応度が得られた場合にも極端な適応度をとらないような設計が可能であると考えられる．

### 6.1.2　5 人人狼

5 人人狼では実験 3 の結果から，GP のによって得られたルール性能が向上し，図 5.3 から適応度が徐々に向上している傾向が見受けられる．これらのことから 5 人人狼においては GP によって性能が向上することがわかる．即ち，人狼知能において GP を用いることで性能の向上を目指すことが可能であるといえる．一方で，15 人人狼と同様に関数セットをさらに多様化させることは可能であり，推定情報以外の情報についても扱うことでより優れたルールの生成ができると考えられる．

## 6.2　ルールについて

### 6.2.1　15 人人狼

実験 1 で得られたルールについての傾向を確認したところ，勝率の上位 2 つである GP1-1，GP1-2 において「人数が半分以下であるか」という共通の条件が使われていたことが明らかになった．アルゴリズム 1 では「人数が半分以下であれば狩人らしいプレイヤ」,「人数が半分以上であれば村人らしくないプレイヤ」というルールが生成され，アルゴリズム 2 では「人数が半分以下であれば村人らしくないプレイヤ」,「人数が半分以上であれば村人らしいプレイヤ」というルールが生成されている．また，GP1-3 においてアルゴリズム 3 では「全員がグレーであるか」という条件を使うルールが得られた．これらの「人数が半分以下であるか」と「全員がグレーであるか」という 2 つのルールは「怪しまれないような状況で

ある」という点が共通しており,「怪しまれない状況であればより追放したい役職に投票する」というルールが考えられる．ここで，人狼陣営として追放したい役職の重要度は表 6.1 に示されており，村人らしくないプレイヤであれば何らかの役職を持っていると判断できる．アルゴリズム 1 では「狩人らしいプレイヤ」が「村人らしくないプレイヤ」よりも重要であり，アルゴリズム 2 では「村人らしくないプレイヤ」が「村人らしいプレイヤ」よりも重要であり，これは表 6.1 の内容と一致している．以上のことから,「怪しまれない状況であればより追放したい役職に投票する」というルールを扱うことができ,「怪しまれない状況」としては「残りの人数」と「全員がグレーであるか」というルールが使えるということがいえる．しかし，今回得られたルールは統計的に有意な差が見られたわけではなく，今後より良いルールが得られた場合にはこのルールに固執しすぎないように注意する必要がある．

一方で，アルゴリズム 5 では，どのようの場合でも最も村人らしい人に投票するといった結果が得られた．この実験 2 では適応度計算の際に人狼全員を GP で生成されたエージェントで固定している．そのため，投票においては「票を固めること」が強いとされているのではないかと考えられる．その中で，役職を持つ人間に対して投票を集めると後の投票において怪しまれる可能性が高まってしまう恐れがある．そのため，表 6.1 の中で役職を持たない村人陣営である村人が重要度の高い対象となり，アルゴリズム 5 のようなルールが得られたのではないかと考えられる．

表 6.1　人狼の追放に関する重要度

| 重要度 | 役職 |
|---|---|
| 高い | 狩人 |
| | 占い師や霊媒師 |
| | 村人 |
| | 狂人 |
| 低い | 味方の人狼 |

### 6.2.2　5 人人狼

5 人人狼では実験 3 の結果から，GP3-2，GP3-3，GP3-4 において「全員がグレーであるか」という共通のルールが使われていたことが明らかになった．これは前述の 15 人人狼の場合と同様に「怪しまれない状況」を指していると考えられ，同様に「怪しまれない状況であればより追放したい役職に投票する」といったルールが使えるといえる．しかし，5 人人狼では村人陣営において特別な役職を持つプレイヤが占い師しかいないため，あまり重要なルールではないとも考えられる．これらについては実験を重ね，より重要なルールについて議論を重ねる必要がある．

# 第 7 章

# 終わりに

本研究では人狼知能において GP を用いて役職推定情報の有効な活用方法について調査し，意思決定の改善を行うことで人狼知能エージェントの性能向上を目指して実験を行った．GP で用いた役職推定情報は SVM より得られた情報を用い，GP の適応度には生成されたルールでの勝率を用いた．

GP の実験結果から，いくつかのルールで統計的に有意な性能向上が確認されたが，同時に一部のルールが解釈が難しかったり，統計的に有意な差が見られないルールが存在するなどの問題も浮かび上がった．生成されたルールの調査では，共通して残り人数が半分以下である時のルールの傾向が見られ，半分以下になればより追放したい役職のプレイヤに投票を行うことが確認された．

適応度の推移においては，適応度の更新が少ない傾向が見られた．この現象は関数セットの少なさが個体の多様性を低下させた可能性が原因として考えられた．本研究では役職推定情報をどのように扱うのかを主目的としており，役職推定情報に関する関数セット以外をあまり追加していない．そのため，関数セットが少なくなってしまい，個体の多様性の低下につながったと考えられた．しかし，人狼ゲームには役職推定情報以外にも多くの情報があり，今回得られたルールと役職推定情報以外を組み合わせることでより良いルールが生成されると考えられるため，さらなる調査が必要である．

# 謝辞

本研究を進めるにあたって，指導教員である竹内聖悟先生には研究室配属後 4 年間に渡り，研究活動や論文執筆をはじめとして多くのご指導をいただきました．心より感謝申し上げます．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，吉田真一教授に深く感謝いたします．

# 参考文献


[1] 梶原健吾, 鳥海不二夫, 稲葉通将, 大澤博隆, 片上大輔, 篠田孝祐, 松原仁, 狩野芳伸. 人狼知能大会における統計分析と svm を用いた人狼推定を行うエージェントの設計. 人工知能学会全国大会論文集, 第 JSAI2016 巻, pp. 2F41–2F41, 2016.

[2] 福田宗理, 穴田一. 15 人人狼ゲームにおける会話情報による役職推定. 人工知能学会全国大会論文集, 第 JSAI2020 巻, pp. 2F5OS20b01–2F5OS20b01, 2020.

[3] 萩原誠, 伊藤孝行, ムスタファアーメッド. Q 学習と役職推定に基づく人狼知能エージェントの作成. 情報処理学会論文誌, 第 60 巻, pp. 1728–1737, oct 2019.

[4] 小村友希, 坂本航, 尾崎知伸. 人狼ゲームにおける明示的役職・陣営推定理由の抽出. 人工知能学会全国大会論文集, 第 JSAI2019 巻, pp. 3F4OS14b04–3F4OS14b04, 2019.

[5] 塚本晴庸, 大村英史, 桂田浩一. 人狼ゲームにおける発言ベクトルを用いた役職推定. 人工知能学会全国大会論文集, 第 JSAI2020 巻, pp. 2F4OS20a04–2F4OS20a04, 2020.

[6] 清水大輔, 長谷部浩二. プレイヤーの発言内容に関するルールに基づいた人狼ゲームの役職推定. 人工知能学会全国大会論文集, 第 JSAI2020 巻, pp. 2F5OS20b05–2F5OS20b05, 2020.

[7] 伊藤毅志, 杉本磨美. 人狼プレイヤの意思決定過程. 人工知能学会全国大会論文集, 第 JSAI2020 巻, pp. 2F4OS20a01–2F4OS20a01, 2020.

[8] 大槻恭士, 今田佑生也. 人狼知能エージェントにおける説得・被説得機能の評価. 人工知能学会全国大会論文集, 第 JSAI2020 巻, pp. 1P3GS705–1P3GS705, 2020.

[9] 伊庭斉志. 進化論的計算の方法. 東京大学出版社, 1999.

[10] 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.

[11] 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 Inteligence in Games (CIG)*, pp. 1–8, 2013.

[12] 栃川純平. 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. master thesis, 高知工科大学大学院, 2022.

[13] 人狼知能プロジェクト. 第 5 回人狼知王国際大会プロトコル部門, 2023. `https://aiwolf.github.io/CompetitionProtocolDivision/ja/`.

[14] 上田陽平, 池田心. 遺伝的アルゴリズムによる人間のレベルに適応する多様なオセロ ai の生成. 研究報告ゲーム情報学（GI）, 第 2012-GI-27 巻, pp. 1–8, 2012.

[15] 狩野芳伸, 大槻恭士, 園田亜斗夢, 中田洋平, 箕輪峻, 鳥海不二夫. 人狼知能で学ぶ AI プログラミング. 株式会社マイナビ出版, 2017.