卒業研究報告書

題目

モンテカルロ法による５五将棋

指導教員

石水 隆　講師

報告者

21-1-037-0032

寺田 一穂

近畿大学理工学部情報学科

令和 07 年 2 月 2 日提出

概要

本研究では, モンテカルロ法を５五将棋に応用して強い将棋ゲーム AI の作成を目指す. 5 五将棋とは,将棋のルールを変更したゲームで,1970 年に楠本茂信によって発表されたミニ将棋の一種であり，コンピュータ同士の対戦による大会なども開かれている.5 五将棋は 5 × 5 の盤面と王, 金, 銀, 角, 飛, 歩の 6 種の駒を用いる.５五将棋の AI については,様々な評価関数の機械学習を用いた手法が提案されているが,本研究ではモンテカルロ法を用いた手法や学習により局面の評価関数を求める手法を応用することで,各局面で最善手を確定させていく.

目次

1. 序論

1.1 本研究の背景

将棋は 2 人のプレイヤーがそれぞれ盤の上で交互に駒を動かし，相手の玉将を詰ませることが勝利条件の日本古来のボードゲームである．将棋には本将棋以外にも様々なバリエーションがあり，そのバリエーションによって盤の大きさやコマの動き，勝利条件といったルールが異なるため，それぞれで違った戦略が必要になる．

将棋のバリエーションの一つに 5 五将棋がある．5 五将棋とは, 将棋のルールを変更したゲームの, 1970 年に楠本茂信によって発表されたミニ将棋の 1 種である．5 五将棋は５×５の盤面と王, 金, 銀, 角, 飛, 歩の６種,全部で 12 枚の駒を用いる.敵陣の１段目に駒を進めた時のみに成ることができ,その他のルールは本将棋のルールに準拠する.図１に５五将棋の初期盤面を示す.

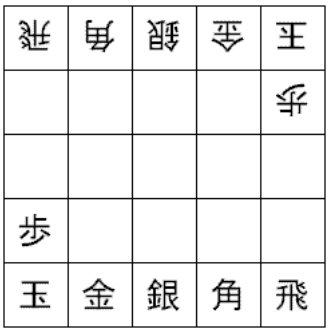

図１　５五将棋の初期盤面

1.2 5 五将棋に関する既知の結果

5 五将棋の大会である UEC 杯 5 五将棋大会は，電気通信大学(University of Electro-Communications)エンターテイメントと認知科学研究ステーション主催で 2007 年より毎年 11 月下旬から 12 月上旬に開催されている. 5 五将棋は，通常の将棋に比べて盤の広さが狭いので, 5 五将棋は本将棋よりも先に人間を上回る強さとなった．5 五将棋は，2009 年に人間が敗れて以降，コンピュータと人間のエキシビションマッチは行われなくなっている．普通の将棋の大会ではやりにくい試みも比較的手軽にできるという特徴がある 5 五将棋の大会は年々広がりを見せている[2]．また，「千日手後手勝ち」というルールのもとコンピュータ対戦を行うと，後手が勝ちやすいゲームであることが推察されている．

既存の将棋 AI は,主に評価関数と,MiniMax 法などの探索アルゴリズムを組み合わされたものであり,探索の効率を上げるためにα β法が採用されたり,終盤では詰将棋ルーチンを組み込んだり等，様々な枝刈りのアイデアが導入されている．

1.3 本研究の目的

５五将棋の AI については,様々な評価関数の機械学習を用いた手法が提案されている.本研究ではモンテカルロ法を用いた手法や学習により局面の評価関数を求める手法を応用することで,各局面で最善手を確定させていく．

1.4　本報告書の構成

本報告書の構成を以下に示す. 2 章でモンテカルロ法について, 3 章で本研究の研究内容に関する説明，４章で結果と考察，５章で結論と今後の課題について述べる．

2　5 五将棋

本章では，本研究の対象である 5 五将棋について述べる．

2.1 将棋のバリエーション

将棋には本将棋以外にも様々なバリエーションがある．中将棋や大将棋といった本将棋よりも大きな盤面を用いる将棋もあるが，バリエーションの多くは将棋の初心者向けの小さい盤面を用いるミニ将棋であり，5 五将棋もミニ将棋の一つである．5 五将棋は盤面が小さく，使用する駒の種類も少ないため，将棋の初心者が気軽に楽しめるゲームとなっている．5 五将棋以外にも，ミニ将棋には，どうぶつしょうぎやごろごろどうぶつしょうぎ，アンパンマンはじめて将棋などがある．図 2 にこれらのミニ将棋の初期盤面を示す．

ミニ将棋のうち，どうぶつしょうぎとアンパンマンはじめてしょうぎは完全解析されており，双方最善手を指すとどうぶつしょうぎは 78 手で後手勝ち，アンパンマンはじめてしょうぎは引き分けとなる[6][7]．

5 五将棋は，現在のところまだ完全解析はされていない．しかし，5 五将棋は本将棋よりも可能な局面数が少ないため，本将棋よりもコンピュータによる探索が行いやすく，コンピュータ 5 五将棋の大会なども開かれている．

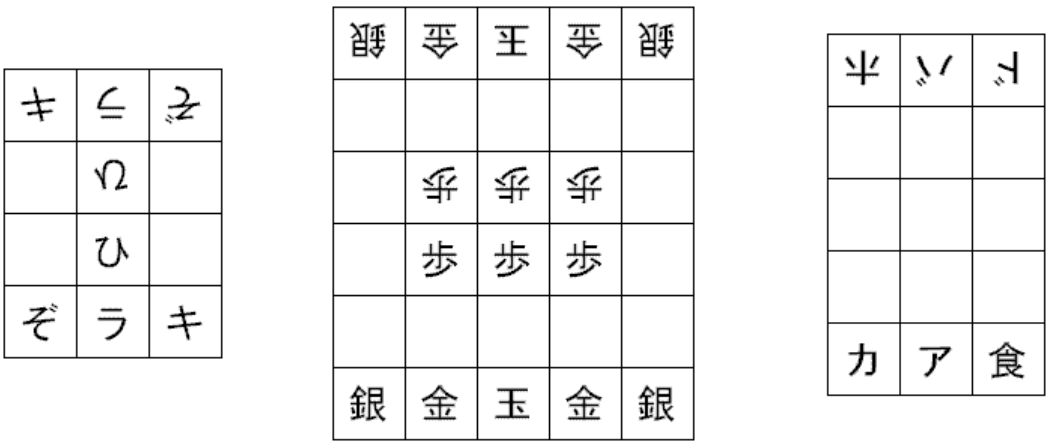

図 2　どうぶつしょうぎ，ごろごろどうぶつしょうぎ，アンパンマンはじめてしょうぎの初期盤面

2.2 5 五将棋のルール

本節では 5 五将棋のルールについて述べる．

５五将棋は５×５の盤面と王, 金, 銀, 角, 飛, 歩の６種,全部で 12 枚の駒を用いる.敵陣の１段目に駒を進めた時のみに成ることができ,その他のルールは本将棋のルールに準拠する

5 五将棋は盤面が小さいため，初期盤面から自駒と相手の玉将の位置が近い．また，初手から飛車角の大駒を動かすことができる．このため，5 五将棋は急戦になりやすく，自玉を守ることよりも相手玉を攻めることが重要となる．

3　モンテカルロ法

本章では, 本研究で作成した 5 五将棋 AI において着手選択に用いるモンテカルロ法について説明する．

3.1 モンテカルロ法とは

モンテカルロ法とは,乱数を用いたシミュレーションを何度も繰り返し行うことで,近似解を求める計算手法である.例えば円周率を求める場合,１×１の正方形の内部にランダムに点を打ち,中心からの距離を計算する.この距離が 0.5 以下なら,その中心を原点とする半径 0.5 の円の中にあり,距離が 0.5 を超えていれば円の外にある.これを

何度も繰り返し行う.円の面積は $r^2\pi = 0.25\pi$,正方形の面積は 1 になので,無限に点を打つと
<円の中の点の個数 : 全ての点の個数 $= 0.25\pi : 1$ >となり,円の中の点の数×4÷全ての点の数 = π となる.シミュレーション回数が増えると実際の円周率にかなり近い数字が得られる.これがモンテカルロ法である.

3.2 モンテカルロ法による着手選択

本節では，将棋，囲碁などのボードゲームにおいてモンテカルロ法による着手選択の方法について述べる.

最も単純な方法は,その時点での有効な選択肢に対し,同数ずつ勝負がつくまでシミュレーションを行い,最も勝率の高かった手を選択する方法である.

3.3 将棋におけるモンテカルロ法

将棋において,モンテカルロ法を用いる利点について以下のようなものが考えられる.

まず,将棋での戦法に関する深い知識が必要ないという点である.将棋には様々な戦法が存在しており,通常の将棋 AI でも,これらの戦法に則って駒を進めた場合の評価を高くするといった手法が取られている.一方でモンテカルロ法を用いた手法では,乱数によって手が選択されるので,戦法に基づいて評価する部分は必要ない.

また,他の利点として,新たに駒の種類を追加しやすいという点がある.先ほども述べた通りモンテカルロ法は個々のゲーム特有の戦法については考慮せずに手を選択できるので,新たに駒を追加したり，ゲームのルールを変更したりしてもプログラムのコードの修正は最低限に止めることが出来る.

3.4 本研究でのモンテカルロ法の応用方法

本研究ではモンテカルロ法を以下のようにして将棋に応用する方法した.

まず, ある局面において, 自分の合法手の中から 1 つ手を選びそこから勝負が付くまでランダムに双方の手を選択する. この操作を十分な回数行い勝率を求める. 図 3 にモンテカルロ法による局面の勝率推定法を示す. この方法を全ての合法手に対して同様の操作を繰り返し, 最も勝率が高かった手を次の手として選択する. ただし, 全ての合法手に対して勝負が付くまでシミュレーションをすると, 途中の段階で既にほぼ負けが決まっている局面や, ここから巻き返すのが難しい段階になってしまっている局面についても, ここからさらに何手も読む必要があり, 無駄が生じてしまう. そこでこの無駄なシミュレーションを減らすための取り組みを行う.

まず, 全ての合法手に対して 10 手先まで読み, その時点の局面や持ち駒から勝敗を判定する. このシミュレーションを 10 回を行い, それぞれの勝率を出した後, 勝率が高かったものだけを選び, 勝負が付くまでさらに 100 回シミュレーションを行うようにする. こうすることで, 10 回シミュレーションを行った段階で勝率の高かった信頼性のある合法手のみを勝敗がつくまでシミュレーションを行うことができる. 図 4 に本研究で用いている戦略を示す.

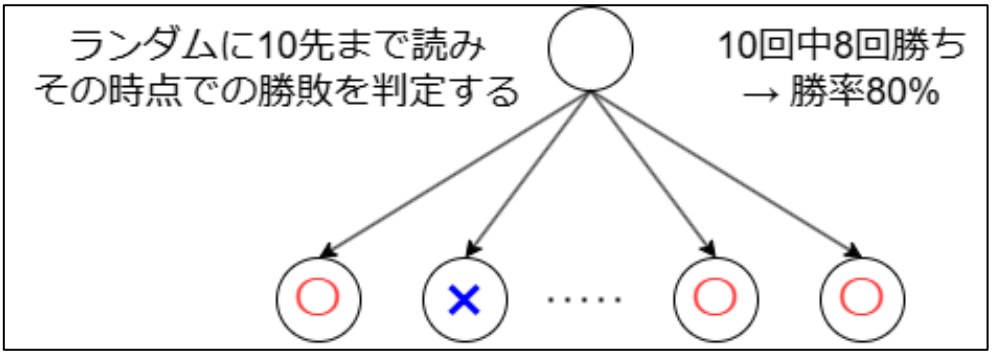

図 3　モンテカルロ法による局面の勝率推定法

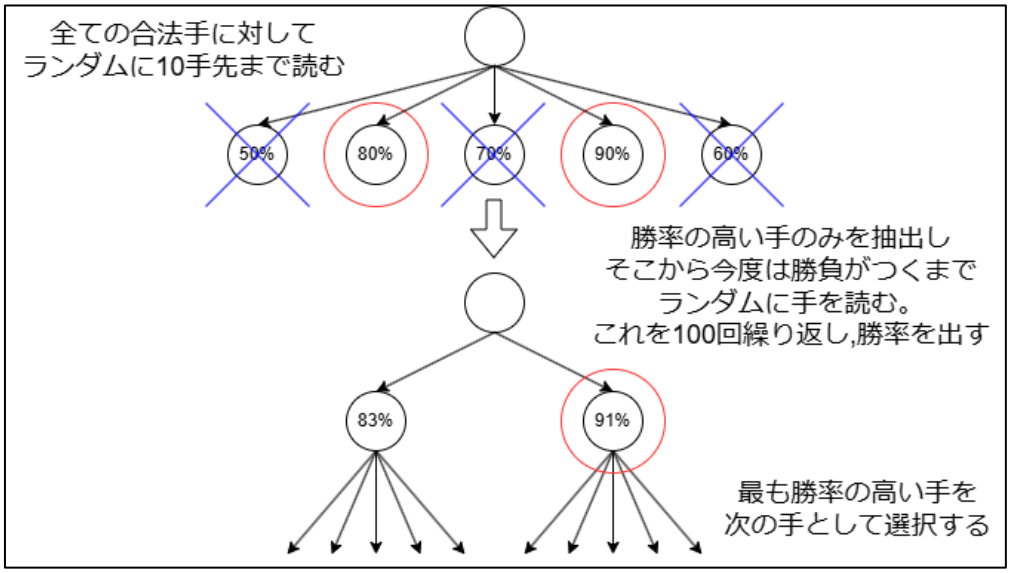

図 4　本研究で用いている戦略

4 プログラム

本章では,モンテカルロ法を利用した将棋 AI に用いたプログラムについて述べる．付録に本研究で作成したプログラムのソースコードを示す.

以降の節では，本研究で作成した 5 五将棋プログラムの各クラスについて説明する.

4.1 Constants クラス

Constants クラスは各種定数を定義するクラスである.図 2 に Constants クラスのクラス図を示す.

Constants クラス	
- SENTE : int	先手
- GOTE : int	後手
- sujiStr : String	筋
- danStr : String	段

図 2 Constants クラスのクラス図

4.2 Koma クラス

Koma クラスは駒の種類を表すクラスである.図 3 に Koma クラスのクラス図を示す.

Koma クラス	
- EMPTY: int	空
- EMP : int	空の予備
- PROMOTE : String	成りフラグ
- FU : int	歩

- GI : int	銀
- KI : int	金
- KA : int	角
- HI : int	飛車
- OU : int	王
- TO : int	と金
- NG : int	成り銀
- UM : int	馬
- RY : int	竜
- SFU : int	先手の歩
- SGI : int	先手の銀
- SKI : int	先手の金
- SKA : int	先手の角
- SHI : int	先手の飛車
- SOU : int	先手の王
- STO : int	先手のと金
- SNG : int	先手の成り銀
- SUM : int	先手の馬
- SRY : int	先手の竜
- GFU : int	後手の歩
- GGI : int	後手の銀
- GKI : int	後手の金
- GKA : int	後手の角
- GHI : int	後手の飛車
- GOU : int	後手の王
- GTO : int	後手のと金
- GNG : int	後手の成り銀
- GUM : int	後手の馬
- GRY : int	後手の竜
- WALL : int	壁
- komaString : String[]	駒の出力用の文字列
- canPromote : boolean[]	成ることが出来る駒の種類
+ isSente(int) : boolean	先手の駒かどうかを判定
+ isGote(int) : boolean	後手の駒かどうかを判定
+ isSelf(int,int) : boolean	自分の駒かどうかを判定
+ isEnemy(int,int) : boolean	相手の駒かどうかを判定
+ getKomashu(int) : int	駒の種類を取得
+ toBanString(int) : String	盤面を表示する

+ toString(int) : String	持ち駒、手を表示する
+ canPromote(int) : boolean	駒が成れるかどうかを表す

図 3 Koma クラスのクラス図

4.3 KomaDirections クラス

KomaDirections クラスは各種駒が動ける方向を設定するクラスである.図 4 に KomaDirections クラスのクラス図を示す.

KomaDirections クラス	
- diffDan : int[]	段の移動の定義
- diffSuji : int[]	筋の移動の定義
- canMove: boolean[]	ある方向に駒が動けるかどうかを表すテーブル
- canJump : boolean[]	ある方向に駒が飛べるかどうかを表すテーブル

図 4 KomaDirections クラスのクラス図

4.4 Te クラス

Te クラスは各種定数を定義するクラスである.図 5 に Te クラスのクラス図を示す.

Te クラス	
+ koma : int	駒の種類
+ from : Position	動く前の位置
+ to : Position	動いた後の位置
+ promote : boolean	成るかどうか
+ Te(int,Position,Position,boolean)	コンストラクタ
+ equals(Te) : boolean	比較用
+ equals(Object) : boolean	比較用
+ clone() : Object	コピーを返す
+ toString() : String	手を文字列で表現する

図 5 Te クラスのクラス図

4.5 Position クラス

Position クラスは駒の位置を表すクラスである.図 6 に Position クラスのクラス図を示す.

Position クラス	
+ suji : int	筋
+ dan : int	段

+ Position()	コンストラクタ
+ Position(int,int)	コンストラクタ
+ equals(Position) : boolean	比較用
+ equals(Object) : boolean	比較用
+ clone(): Object	コピーを返す
+ add(int,int) : void	ある方向へ動く
+ sub(int,int) : void	ある方向へ逆向きに動く
+ add(int) : void	ある方向へ動く
+ add(int) : void	ある方向へ逆向きに動く

図 6 Position クラスのクラス図

4.6 Kyokumen クラス

Kyokumen クラスは局面を表すクラスである.図 7 に Kyokumen クラスのクラス図を示す.

Kyokumen クラス	
+ ban : int[][]	盤面
+ senteHand : ArrayList<Integer>	先手番の持ち駒
+ goteHand : ArrayList<Integer>	後手番の持ち駒
+ teban : int	手番
+ csaKomaTbl : String[]	初期局面用の配列
+ komaValue : int[]	駒の点数
+ Kyokumen()	コンストラクタ
+ clone() : Object	コピーを返す
+ equals(Obejct) : boolean	比較用
+ equals(Kyokumen) : boolean	比較用
+ get(Position) : int	ある位置にある駒を取得する
+ put(Position,int) : void	ある位置にある駒を置く
+ move(Te) : void	与えられた手で一手進める
+ searchGyoku(int) : Position	王の位置を探して位置を返す
+ ReadCsaKifu(String[]) : void	局面を読み込む
+ toString() : String	局面を文字列化する
+ evaluationFun() : int	駒の点数を合算する
+ haveOu() : int	持ち駒に王があるかを確認する

図 7 Kyokumen クラスのクラス図

4.7 GenerateMoves クラス

GenerateMoves クラスは合法手を生成するクラスである.図 8 に GenerateMoves クラスのクラス図を示す.

GenerateMoves クラス	
- removeSeleMate(Kyokumen,List<Te>) : List<Te>	王手がかかっている手を除く
- addTe(List<Te>,int,int,Position,Position) : void	生成した手を追加する
- isUtiFuDume(Kyokumen,Te) : boolean	打ち歩詰めになっていないかを確認
- generateLegalMoves(Kyokumen) : List<Te>	その局面での合法手を生成する

図 8 GenerateMoves クラスのクラス図

4.8 Human クラス

Human クラスは手動で次の手を決めるためのクラスである.図 9 に Human クラスのクラス図を示す.

Human クラス	
- reader : BufferdReader	文字列入力用
+ getNextTe(Kyokumen) : Te	入力した文字列から,次の手を生成する

図 9 Human クラスのクラス図

4.9 Cpu クラス

Cpu クラスはコンピュータが次の手を決めるためのクラスである.図 10 に Cpu クラスのクラス図を示す.

KomaDirections クラス	
- random : Random	乱数生成
+ Cpu()	コンストラクタ
+ getRandomTe(Kyokumen) : Te	次の手をランダムに選択する

図 10 Cpu クラスのクラス図

4.10 Main クラス

Main クラスはメインの実行を行うクラスである.図 11 に Main クラスのクラス図を示す.

KomaDirections クラス	
- ShokiHaiti: int[][]	初期局面
- cpu : Cpu[]	コンピュータ
- human : Human[]	人間
- kyokumenRireki : ArrayList<Object>	局面の履歴
- middleKyokumenRireki : ArrayList<Object>	途中段階での局面の履歴
- gohoTeNumber : ArrayList<Integer>	勝率の高かった手の管理用
- bestTes : ArrayList<Te>	勝率が同じ手が複数あった時用のリスト
- random : Random	乱数生成

+ main(String[]) : void	メインの実行を行う

図 11 Main クラスのクラス図

4.11 プログラムの実行方法

本節では，本研究で作成したプログラムの実行方法を述べる．プロクラムの流れは以下の通りである．

1. 初期盤面を用意する．
2. 3,4,5,6 を勝負が付くまで繰り返す．
3. その局面における全ての合法手を生成する．
4. 全ての合法手に対して 10 手先まで読み,その時点での勝敗を判定する.これを 10 回繰り返す．
5. 4.で勝率の高かった手のみを抽出し,さらに勝負が付くまでシミュレーションを行う.これを 100 回繰り返す．
6. 5.で最も勝率の高かった手を次の手として選択する．

5 結果・考察

モンテカルロ法を用いて手を選ぶ AI(以下 Monte とする)と、合法手の中から完全にランダムに手を選ぶ AI（以下 Rand とする）を先手後手を入れ替えてそれぞれ 10 回対戦させた結果,Monte 同士を 10 回対戦させた結果,Monte と既存の AI[5]を先手後手を入れ替えてそれぞれ 10 回対戦させた結果を表 1 に示す．

表 1 の結果から Monte と Rand の対戦では Monte 側の勝率が高いことがわかった.また,Monte 同士の対戦は後手側の勝率がかなり高いことがわかった.一方で,既存 AI との対戦は先手後手に関わらず,ほぼ負けてしまったことがわかった．

モンテカルロ法は,そのゲームに適した評価関数を実装し探索する方法と比べて強くなりにくい傾向がある.そのため,通常のルールではなく,例えば駒の種類を増やしたり,特殊なルールが追加された状態であったりといった,数多くある将棋のバリエーションのそれぞれに対して AI を作る場合など，それぞれのバリエーションに対して適切な評価関数を実装するのが難しい場合には,モンテカルロ法が有用であると考えられる．

表 1　AI 同士の対戦結果

先手	後手	先手勝	後手勝
Rand	Monte	8 回	2 回
Monte	Rand	0 回	10 回
Monte	Monte	2 回	8 回
Monte	[5]	0 回	10 回
[5]	Monte	1 回	9 回

6 結論・今後の課題

本研究ではモンテカルロ法を用いた 5 五将棋 AI を作成し,ある程度の強さがあることがわかった.今後は,ありえない手を除く処理や,相手の駒を減らす手を選ぶ確率を増やすことでより強い AI になると考えられる．

また,モンテカルロ法以外の手法を組み合わせることで,より効率よく勝率の高い手を選択できるようになると考えられる.例えば Mini-Max 法と組み合わせる場合,ある局面で次の手を決める際,現在シミュレーションを繰り返し行なっている手の敗北数が,その時点で最も勝率の高い手とされている手の敗北数を超えた時点でシミュレーションを中断するといったものが考えられる．

謝辞

本研究をするにあたり，適切な助言と指導をいただいた石水先生に感謝しています．ありがとうございました．

参考文献

[1] 伊藤毅志, 新沢剛:モンテカルロ法を用いた 5 五将棋システム, 情報処理学会 研究報告ゲーム情報学 Vol.2007, No.6, pp.1-6 (2007). http://id.nii.ac.jp/1001/00058488/

[2] 伊藤毅志:5五将棋大会の動向(2013 年～2014 年), 情報処理学会 研究報告, Vol.2015-GI-33, No.1, pp.1--5 (2015) http://id.nii.ac.jp/1001/00113628/

[3] 塩田雅弘, 伊藤毅志:5五将棋における自動対戦を用いた評価関数の学習, 情報処理学会研究報告, Vol.2020-GI-44, NO.3, pp.1--6 (2020) http://id.nii.ac.jp/1001/00204861/

[4] 石井颯太郎, 田中哲朗:5 五将棋の実現可能局面数の推計, 情報処理学会 研究報告ゲーム情報学 Vol.2024-GI-53, No.1, pp.1-6 (2024) http://id.nii.ac.jp/1001/00238391/

[5] 5 五将棋, スマホ Web ゲーム中心(2014) https://sfw.gamebatake.info/game/sg55shogi.html

[6] 田中哲郎:「どうぶつしょぎ」の完全解析, 情報処理学会研究報告, Vol.2009-GI-22 No.3, pp.1-8 (2009). http://id.nii.ac.jp/1001/00062415/

[7] 塩田好, 石水隆, 山本博史:「アンパンマンはじめてしょうぎ」の完全解析, 情報処理学会関西支部 支部大会講演論文集 (2013). http://id.nii.ac.jp/1001/00096792/

付録 プログラムのソースコード

以下に本研究で作成したConstantsクラスのソースコードを示す.

```
package syougi;

public interface Constants {
public final static int SENTE = 12;
public final static int GOTE = 24;
public final static String sujiStr[] = {"","1","2","3","4","5"};
public final static String danStr[] = {"","一","二","三","四","五"};
}
```

以下に本研究で作成したKomaクラスのソースコードを示す.

```
package syougi;

public class Koma implements Constants,Cloneable{
public static final int EMPTY = 0;
public static final int EMP = EMPTY;
public static final int PROMOTE = 6;

public static final int FU = 1;
public static final int GI = 2;
public static final int KI = 3;
public static final int KA = 4;
public static final int HI = 5;
public static final int OU = 6;
public static final int TO = FU + PROMOTE;
public static final int NG = GI + PROMOTE;
public static final int UM = KA + PROMOTE;
public static final int RY = HI + PROMOTE;

public static final int SFU = SENTE + FU;
public static final int SGI = SENTE + GI;
public static final int SKI = SENTE + KI;
public static final int SKA = SENTE + KA;
public static final int SHI = SENTE + HI;
public static final int SOU = SENTE + OU;
public static final int STO = SENTE + TO;
public static final int SNG = SENTE + NG;
```

```
public static final int SUM = SENTE + UM;
public static final int SRY = SENTE + RY;

public static final int GFU = GOTE + FU;
public static final int GGI = GOTE + GI;
public static final int GKI = GOTE + KI;
public static final int GKA = GOTE + KA;
public static final int GHI = GOTE + HI;
public static final int GOU = GOTE + OU;
public static final int GTO = GOTE + TO;
public static final int GNG = GOTE + NG;
public static final int GUM = GOTE + UM;
public static final int GRY = GOTE + RY;

public static final int WALL = 64;

static public boolean isSente(int koma) {
        if(13<=koma&&koma<=23) return true;
        else return false;

}

static public boolean isGote(int koma) {
        if(25<=koma&&koma<=35) return true;
        else return false;
}

static public boolean isSelf(int teban,int koma) {
        if(teban == SENTE) {
                return isSente(koma);
        }
        else {
                return isGote(koma);
        }
}

static public boolean isEnemy(int teban,int koma) {
        if(teban == SENTE) {
                return isGote(koma);
```

```
        }else {
                return isSente(koma);
        }
}

static public int getKomashu(int koma) {
        if(13<=koma&&koma<=23) {
                return koma - 12;
        }else if(25<=koma&&koma<=35) {
                return koma - 24;
        }else {
        return koma;
        }
}

static public final String komaString[]= {
                "",
                "歩",
                "銀",
                "金",
                "角",
                "飛",
                "王",
                "と",
                "全",
                "　",
                "馬",
                "龍"

};

static public String toBanString(int koma) {
        if(koma==EMPTY) {
                return "　 ";
        }else if(13<=koma&&koma<=23) {
                return " " + komaString[getKomashu(koma)];
        }else {
                return "v" + komaString[getKomashu(koma)];
        }
```

```
}

static public String toString(int koma) {
        return komaString[getKomashu(koma)];
}

public static final boolean canPromote[] = {
                false,
                false,false,false,false,false,false,
                false,false,false,false,false,false,
                true,true,false,true,true,false,
                false,false,false,false,false,false,
                true,true,false,true,true,false,
                false,false,false,false,false,false
};

static public boolean canPromote(int koma) {
        return canPromote[koma];
}
}
```

以下に本研究で作成した KomaDirections クラスのソースコードを示す.

```
package syougi;

public interface KomaDirections {
public static final int diffDan[]= {
                1,1,1,0,0,-1,-1,-1
};

public static final int diffSuji[]= {
                -1,0,1,1,-1,1,0,-1
};

public static final boolean canMove[][]= {
                //斜め左下
                {
                        false,
                        false,false,false,false,false,false,
```

```
        false, false, false, false, false, false,
        false, true, false, false, false, true,  //先手の歩銀金角飛王
        false, false, false, false, true, false,  //先手の、と、成り銀、空、馬龍、空
        false, true, true, false, false, true,  //後手の歩銀金角飛王
        true, true, false, false, true, false //後手の、と、成り銀、空、馬龍、空
},

//真下
{
        false,
        false, false, false, false, false, false,
        false, false, false, false, false, false,
        false, false, true, false, false, true,  //先手の歩銀金角飛王
        true, true, false, true, false, false,  //先手の、と、成り銀、空、馬龍、空
        true, true, true, false, false, true,  //後手の歩銀金角飛王
        true, true, false, true, false, false //後手の、と、成り銀、空、馬龍、空
},

//斜め右下
{
        false,
        false, false, false, false, false, false,
        false, false, false, false, false, false,
        false, true, false, false, false, true,  //先手の歩銀金角飛王
        false, false, false, false, true, false,//先手の、と、成り銀、空、馬龍、空
        false, true, true, false, false, true,  //後手の歩銀金角飛王
        true, true, false, false, true, false //後手の、と、成り銀、空、馬龍、空
},

//左
{
        false,
        false, false, false, false, false, false,
        false, false, false, false, false, false,
        false, false, true, false, false, true,  //先手の歩銀金角飛王
        true, true, false, true, false, false,  //先手の、と、成り銀、空、馬龍、空
        false, false, true, false, false, true,  //後手の歩銀金角飛王
        true, true, false, true, false, false //後手の、と、成り銀、空、馬龍、空
},
```

```
//右
{
        false,
        false, false, false, false, false, false,
        false, false, false, false, false, false,
        false, false, true, false, false, true,  //先手の歩銀金角飛王
        true, true, false, true, false, false,  //先手の、と、成り銀、空、馬龍、空
        false, false, true, false, false, true,  //後手の歩銀金角飛王
        true, true, false, true, false, false//後手の、と、成り銀、空、馬龍、空
},

//斜め左上
{
        false,
        false, false, false, false, false, false,
        false, false, false, false, false, false,
        false, true, true, false, false, true,  //先手の歩銀金角飛王
        true, true, false, false, true, false,  //先手の、と、成り銀、空、馬龍、空
        false, true, false, false, false, true,  //後手の歩銀金角飛王
        false, false, false, false, true, false //後手の、と、成り銀、空、馬龍、空
},

//真上
{
        false,
        false, false, false, false, false, false,
        false, false, false, false, false, false,
        true, true, true, false, false, true,  //先手の歩銀金角飛王
        true, true, false, true, false, false,  //先手の、と、成り銀、空、馬龍、空
        false, false, true, false, false, true,  //後手の歩銀金角飛王
        true, true, false, true, false, false//後手の、と、成り銀、空、馬龍、空
},

//斜め右上
{
        false,
        false, false, false, false, false, false,
```

```
                false,false,false,false,false,false,
                false,true,true,false,false,true,  //先手の歩銀金角飛王
                true,true,false,false,true,false,//先手の、と、成り銀、空、馬龍、空
                false,true,false,false,false,true,  //後手の歩銀金角飛王
                false,false,false,false,true,false //後手の、と、成り銀、空、馬龍、空
        }

};

static final public boolean canJump[][] = {

        //斜め左下
        {
                false,
                false,false,false,false,false,false,
                false,false,false,false,false,false,
                false,false,false,true,false,false,  //先手の歩銀金角飛王
                false,false,false,true,false,false,  //先手の、と、成り銀、空、馬龍、空
                false,false,false,true,false,false,  //後手の歩銀金角飛王
                false,false,false,true,false,false //後手の、と、成り銀、空、馬龍、空
        },

        //真下
        {
                false,
                false,false,false,false,false,false,
                false,false,false,false,false,false,
                false,false,false,false,true,false,  //先手の歩銀金角飛王
                false,false,false,false,true,false,  //先手の、と、成り銀、空、馬龍、空
                false,false,false,false,true,false,  //後手の歩銀金角飛王
                false,false,false,false,true,false,  //後手の、と、成り銀、空、馬龍、空
        },

        //斜め右下
        {
                false,
                false,false,false,false,false,false,
                false,false,false,false,false,false,
                false,false,false,true,false,false,  //先手の歩銀金角飛王
```

```
        false, false, false, true, false, false,  //先手の、と、成り銀、空、馬龍、空
        false, false, false, true, false, false,  //後手の歩銀金角飛王
        false, false, false, true, false, false //後手の、と、成り銀、空、馬龍、空
},

//左
{
        false,
        false, false, false, false, false, false,
        false, false, false, false, false, false,
        false, false, false, false, true, false,  //先手の歩銀金角飛王
        false, false, false, false, true, false,  //先手の、と、成り銀、空、馬龍、空
        false, false, false, false, true, false,  //後手の歩銀金角飛王
        false, false, false, false, true, false //後手の、と、成り銀、空、馬龍、空
},

//右
{
        false,
        false, false, false, false, false, false,
        false, false, false, false, false, false,
        false, false, false, false, true, false,  //先手の歩銀金角飛王
        false, false, false, false, true, false,  //先手の、と、成り銀、空、馬龍、空
        false, false, false, false, true, false,  //後手の歩銀金角飛王
        false, false, false, false, true, false //後手の、と、成り銀、空、馬龍、空
},

//斜め左上
{
        false,
        false, false, false, false, false, false,
        false, false, false, false, false, false,
        false, false, false, true, false, false,  //先手の歩銀金角飛王
        false, false, false, true, false, false,  //先手の、と、成り銀、空、馬龍、空
        false, false, false, true, false, false,  //後手の歩銀金角飛王
        false, false, false, true, false, false //後手の、と、成り銀、空、馬龍、空
},

//真上
```

```
                {
                        false,
                        false,false,false,false,false,false,
                        false,false,false,false,false,false,
                        false,false,false,false,true,false, //先手の歩銀金角飛王
                        false,false,false,false,true,false, //先手の、と、成り銀、空、馬龍、空
                        false,false,false,false,true,false, //後手の歩銀金角飛王
                        false,false,false,false,true,false //後手の、と、成り銀、空、馬龍、空
                },

                //斜め右上
                {
                        false,
                        false,false,false,false,false,false,
                        false,false,false,false,false,false,
                        false,false,false,true,false,false, //先手の歩銀金角飛王
                        false,false,false,true,false,false, //先手の、と、成り銀、空、馬龍、空
                        false,false,false,true,false,false, //後手の歩銀金角飛王
                        false,false,false,true,false,false //後手の、と、成り銀、空、馬龍、空
                }
};
}
```

以下に本研究で作成したTeクラスのソースコードを示す.

```
package syougi;

public class Te implements Constants,Cloneable{
int koma;
Position from;
Position to;
boolean promote;

public Te(int koma,Position from,Position to,boolean promote) {
        this.koma = koma;
        this.from = from;
        this.to = to;
        this.promote = promote;
}
```

```
public boolean equals(Te te) {
        return (te.koma == koma && te.from.equals(from) && te.to.equals(to) && te.promote == promote);
}

public boolean equals(Object te_) {
        Te te = (Te)te_;
        if(te == null) return false;
        return equals(te);
}

public Object clone() {
        return new Te(koma,from,to,promote);
}

public String toString() {
        return sujiStr[to.suji] + danStr[to.dan]
                        + Koma.toString(koma)+(promote?"成":"")
                        + (from.suji==0?"打 ":"("+sujiStr[from.suji]+danStr[from.dan]+")")
                        + (promote?"":" ");
}

}
```

以下に本研究で作成した Position クラスのソースコードを示す.

```
package syougi;

public class Position implements Cloneable,KomaDirections{
public int suji;
public int dan;

public Position() {
        suji=0;
        dan=0;
}

public Position(int sujiA,int danA) {
        suji = sujiA;
```

```
        dan = danA;
}
public boolean equals(Position p) {
        return (p.suji==suji && p.dan == dan);
}

public boolean equals(Object o) {
        Position p =(Position)o;
        if(p == null) return false;
        return equals(p);
}

public Object clone() {
        return new Position(suji,dan);
}

public void add(int diffSuji,int diffDan) {
        suji+=diffSuji;
        dan+=diffDan;
}

public void sub(int diffSuji,int diffDan) {
        suji-=diffSuji;
        dan-=diffDan;
}

public void add(int direct) {
        add(diffSuji[direct],diffDan[direct]);
}

public void sub(int direct) {
        sub(diffSuji[direct],diffDan[direct]);
}

}
```

以下に本研究で作成した Kyokumen クラスのソースコードを示す.

```
package syougi;
```

```
import java.util.ArrayList;

public class Kyokumen implements Constants,Cloneable{

int ban[][];

ArrayList<Integer> senteHand;
ArrayList<Integer> goteHand;


int teban = SENTE;

public Kyokumen() {
        ban = new int[7][7];
        senteHand = new ArrayList<>();
        goteHand = new ArrayList<>();
}

@SuppressWarnings("unchecked")
public Object clone() {
        Kyokumen k = new Kyokumen();

        for(int suji=0;suji<7;suji++) {
                for(int dan=0;dan<7;dan++) {
                        k.ban[suji][dan] = ban[suji][dan];
                }
        }
        k.senteHand = (ArrayList<Integer>) senteHand.clone();
        k.goteHand = (ArrayList<Integer>) goteHand.clone();
        k.teban = teban;

        return k;
}

public boolean equals(Object o) {
        Kyokumen k = (Kyokumen)o;
        if(k==null) return false;
        return equals(k);
```

```
}

public boolean equals(Kyokumen k) {
        if (teban != k.teban) {
                return false;
        }
        for(int suji=1;suji<=5;suji++) {
                for(int dan=1;dan<=5;dan++) {
                        if(!(ban[suji][dan]==k.ban[suji][dan])) {
                                return false;
                        }

                }
        }

        int handSente[] = new int[Koma.HI+1];
        int handGote[] = new int[Koma.HI+1];
        int compareHandSente[] = new int[Koma.HI+1];
        int compareHandGote[] = new int[Koma.HI+1];

        for(int i=0;i<senteHand.size();i++) {
                int koma = senteHand.get(i);
                int komaShu = Koma.getKomashu(koma);
                handSente[komaShu]++;
        }

        for(int i=0;i<goteHand.size();i++) {
                int koma = goteHand.get(i);
                int komaShu = Koma.getKomashu(koma);
                handGote[komaShu]++;
        }

        for(int i=0;i<k.senteHand.size();i++) {
                int koma = k.senteHand.get(i);
                int komaShu = Koma.getKomashu(koma);
                compareHandSente[komaShu]++;
        }
```

```
        for(int i=0;i<k.goteHand.size();i++) {
                int koma = k.goteHand.get(i);
                int komaShu = Koma.getKomashu(koma);
                compareHandSente[komaShu]++;
        }

        for(int i=Koma.FU;i<=Koma.HI;i++) {
                if(handSente[i]!=compareHandSente[i]) return false;
                if(handGote[i]!=compareHandGote[i]) return false;
        }
        return true;
}

public int get(Position p) {
        if(p.suji<1||5<p.suji||p.dan<1||5<p.dan) {
                return Koma.WALL;
        }
        return ban[p.suji][p.dan];
}

public void put(Position p,int koma) {
        ban[p.suji][p.dan] = koma;
}

public void move(Te te) {
        if(get(te.to) != 0) {
                if(Koma.isSente(get(te.to))) {
                        int koma=get(te.to);

                        if(13<=koma&&koma<=23) {
                                koma -= 12;
                        }else if(25<=koma&&koma<=35) {
                                koma -= 24;
                        }else {}

                        if(7<=koma&&koma<=11) {
                                koma -= 6;
                        }
                        koma = koma + 24;
```

```
                goteHand.add(koma);
        }else {
                int koma=get(te.to);

                if(13<=koma&&koma<=23) {
                        koma -= 12;
                }else if(25<=koma&&koma<=35) {
                        koma -= 24;
                }else {}

                if(7<=koma&&koma<=11) {
                        koma -= 6;
                }
                koma = koma + 12;
                senteHand.add(koma);
        }
}

if(te.from.suji==0) {
        if(Koma.isSente(te.koma)) {
                for(int i=0;i<senteHand.size();i++) {
                        int koma = senteHand.get(i);
                        if(koma==te.koma) {
                                senteHand.remove(i);
                                break;
                        }
                }
        }else {
                for(int i=0;i<goteHand.size();i++) {
                        int koma = goteHand.get(i);
                        if(koma==te.koma) {
                                goteHand.remove(i);
                                break;
                        }
                }
        }
}else {
        put(te.from,Koma.EMPTY);
}
```

```
        int koma=te.koma;
        if(te.promote) {
                koma=koma + 6;
        }
        put(te.to,koma);
}

public Position searchGyoku(int teban) {
        int toSearch = 9999;
        if(teban == SENTE) {
                if(teban - 12 == Koma.OU) {
                        toSearch = teban - 12;
                }
        }else {
                if(teban - 24 == Koma.OU) {
                        toSearch = teban - 24;
                }
        }
        for(int suji=1;suji<=5;suji++) {
                for(int dan=1;dan<=5;dan++) {
                        if(ban[suji][dan]==toSearch) {
                                return new Position(suji,dan);
                        }
                }
        }
        return new Position(-2,-2);
}

static final String csaKomaTbl[] = {
            "   ","FU","GI","KI","KA","HI","OU","TO","NG","","UM","RY",
            "   ","+FU","+GI","+KI","+KA","+HI","+OU","+TO","+NG","","+UM","+RY",
            "   ","-FU","-GI","-KI","-KA","-HI","-OU","-TO","-NG","","-UM","-RY"
          };

          public void ReadCsaKifu(String[] csaKifu) {
            int motigoma[][]=new int[2][Koma.HI+1];

            int restKoma[]=new int[Koma.HI+1];
```

```
for (int i = 0; i <= Koma.HI; i++) {
  motigoma[0][i] = 0;
  motigoma[1][i] = 0;
}

restKoma[Koma.FU]=2;
restKoma[Koma.GI]=2;
restKoma[Koma.KI]=2;
restKoma[Koma.KA]=2;
restKoma[Koma.HI]=2;

for(int suji=1;suji<=5;suji++) {
  for(int dan=1;dan<=5;dan++) {
    ban[suji][dan]=Koma.EMPTY;
  }
}

for(int i=0;i<csaKifu.length;i++) {
  String line=csaKifu[i];
  System.out.println(""+i+" :"+line);
  if (line.startsWith("P+")) {
    if (line.equals("P+00AL")) {
      for(int koma=Koma.FU;koma<=Koma.HI;koma++) {
        motigoma[0][koma]=restKoma[koma];
      }
    } else {
      for(int j=0;j<=line.length()-6;j+=4) {
        int koma=0;
        String komaStr=line.substring(j+2+2,j+2+4);
        for(int k=Koma.FU;k<=Koma.HI;k++) {
          if(komaStr.equals(csaKomaTbl[k])) {
            koma=k;
            break;
          }
        }
        motigoma[0][koma]++;
      }
    }
```

```
} else if (line.startsWith("P-")) {
  if (line.equals("P-00AL")) {
    for(int koma=Koma.FU;koma<=Koma.HI;koma++) {
      motigoma[1][koma]=restKoma[koma];
    }
  } else {
    for(int j=0;j<line.length();j+=4) {
      int koma=0;
      for(int k=Koma.FU;k<=Koma.HI;k++) {
        if(line.substring(j+2,j+4).equals(csaKomaTbl[k])) {
          koma=k;
          break;
        }
      }
      motigoma[1][koma]++;
    }
  }
} else if (line.startsWith("P")) {
  String danStr=line.substring(1,2);
  int dan=0;
  try {
    dan=Integer.parseInt(danStr);
  } catch(Exception e) {
  }
  String komaStr;
  for(int suji=1;suji<=5;suji++) {
    komaStr=line.substring(2+(5-suji)*3,2+(5-suji)*3+3);
    int koma=Koma.EMPTY;
    for(int k=Koma.EMPTY;k<=Koma.GRY;k++) {
      if (komaStr.equals(csaKomaTbl[k])) {
        koma=k;
        restKoma[(Koma.getKomashu(koma) & ~Koma.PROMOTE)]--;
        break;
      }
    }
    ban[suji][dan]=koma;
  }
} else if (line.equals("-")) {
  teban=GOTE;
```

```
    } else if (line.equals("+")) {
      teban=SENTE;
    }
  }
  for(int i=Koma.FU;i<Koma.HI;i++) {
    for(int j=0;j<motigoma[0][j];j++) {
      senteHand.add(i);
    }
    for(int j=0;j<motigoma[1][j];j++) {
      goteHand.add(i);
    }
  }
}

public String toString() {
  String s="";
  s+="後手持ち駒:";
  if(goteHand.size()!=0) {
  for(int i=0;i<goteHand.size();i++) {
    s+=Koma.toString((goteHand.get(i)));
  }
  }
  s+="¥n";
  s+="  5  4  3  2  1¥n";
  s+="+---+---+---+---+---+¥n";
  for(int dan=1;dan<=5;dan++) {
    for(int suji=5;suji>=1;suji--) {
      s+="|";
      s+=Koma.toBanString(ban[suji][dan]);
    }
    s+="|";
    s+=danStr[dan];
    s+="¥n";
    s+="+---+---+---+---+---+¥n";
  }
  s+="先手持ち駒:";
  if(senteHand.size()!=0) {
  for(int i=0;i<senteHand.size();i++) {
    s+=Koma.toString(senteHand.get(i));
```

```
            }
            }
            s+="¥n";
            return s;
          }

static final int komaValue[]= {
                0,
                0,0,0,0,0,0,
                0,0,0,0,0,0,
                100,1000,1200,1800,2000,10000,
                1200,1200,2000,2200,0,0,
                -100,-1000,-1200,-1800,-2000,-10000,
                -1200,-1200,-2000,-2200,0,0
};

public int evaluationFun() {
        int sum = 0;
        for(int suji=1;suji<=5;suji++) {
                for(int dan=1;dan<=5;dan++) {
                        int koma = ban[suji][dan];
                        sum = sum + komaValue[koma];
                }
        }

        for(int i=0;i<senteHand.size();i++) {
                int koma = senteHand.get(i);
                sum = sum + komaValue[koma];
        }
        for(int i=0;i<goteHand.size();i++) {
                int koma = goteHand.get(i);
                sum = sum + komaValue[koma];
        }

        return sum;
}


public int haveOu() {
```

```
        for(int i=0;i<senteHand.size();i++){
                if(Koma.toString(senteHand.get(i)).equals("王")) {
                        return 1;
                }
        }

        for(int i=0;i<goteHand.size();i++){
                if(Koma.toString(goteHand.get(i)).equals("王")) {
                        return 2;
                }
        }

        return 0;
}

}
```

以下に本研究で作成した GenerateMoves クラスのソースコードを示す.

```
package syougi;
import java.util.ArrayList;
import java.util.List;

public class GenerateMoves implements Constants,KomaDirections {

  public static List<Te> removeSelfMate(Kyokumen k,List<Te> tes) {
    List<Te> removedTe = new ArrayList<>();
    for(int i=0;i<tes.size();i++) {

      Te te=tes.get(i);

      Kyokumen test=(Kyokumen)k.clone();
      test.move(te);

      Position gyokuPosition=test.searchGyoku(k.teban);

      boolean isOuteHouchi=false;

      for(int direct=0;direct<8 && !isOuteHouchi;direct++) {
```

```
      Position pos=(Position)gyokuPosition.clone();
      pos.sub(direct);
      int koma=test.get(pos);
      if (Koma.isEnemy(test.teban,koma) && canMove[direct][koma]) {
        isOuteHouchi=true;
        break;
      }
    }

    for(int direct=0;direct<8 && !isOuteHouchi;direct++) {
      Position pos=(Position)gyokuPosition.clone();
      int koma;
      for(pos.sub(direct),koma=test.get(pos);
       koma!=Koma.WALL;pos.sub(direct),koma=test.get(pos)) {
        if (Koma.isSelf(test.teban,koma)) break;
        if (Koma.isEnemy(test.teban,koma) && canJump[direct][koma]) {
          isOuteHouchi=true;
          break;
        }
        if (Koma.isEnemy(test.teban,koma)) {
          break;
        }
      }
    }
    if (!isOuteHouchi) {
      removedTe.add(te);
    }
  }
  return removedTe;
}

public static void addTe(List<Te> tes,int teban,int koma,Position from,Position to) {
  if (teban==SENTE) {
    if ((Koma.getKomashu(koma)==Koma.FU) && to.dan==1) {
      Te te=new Te(koma,from,to,true);
      tes.add(te);
    } else if ((to.dan<=2 || from.dan<=2) && Koma.canPromote(koma)) {
      Te te=new Te(koma,from,to,true);
      tes.add(te);
```

```
      te=new Te(koma,from,to,false);
      tes.add(te);
    } else {
      Te te=new Te(koma,from,to,false);
      tes.add(te);
    }
  } else {
    if ((Koma.getKomashu(koma)==Koma.FU) && to.dan==5) {
      Te te=new Te(koma,from,to,true);
      tes.add(te);
    } else if ((to.dan>=4 || from.dan>=4) && Koma.canPromote(koma)) {
      Te te=new Te(koma,from,to,true);
      tes.add(te);
      te=new Te(koma,from,to,false);
      tes.add(te);
    } else {
      Te te=new Te(koma,from,to,false);
      tes.add(te);
    }
  }
}

public static boolean isUtiFuDume(Kyokumen k,Te te) {
  if (te.from.suji!=0 && te.from.dan!=0) {
    return false;
  }
  if (Koma.getKomashu(te.koma)!=Koma.FU) {
    return false;
  }
  int teban;
  int tebanAite;
  if ((te.koma&SENTE)!=0) {
    teban=SENTE;
    tebanAite=GOTE;
  } else {
    teban=GOTE;
    tebanAite=SENTE;
  }
  Position gyokuPositionAite=k.searchGyoku(tebanAite);
```

```
    if (teban==SENTE) {
      if (gyokuPositionAite.suji!=te.to.suji || gyokuPositionAite.dan!=te.to.dan-1) {
        return false;
      }
    } else {
      if (gyokuPositionAite.suji!=te.to.suji || gyokuPositionAite.dan!=te.to.dan+1) {
        return false;
      }
    }

    Kyokumen test=(Kyokumen)k.clone();
    test.move(te);
    test.teban=tebanAite;
    List<Te> testTes =generateLegalMoves(test);
    if (testTes.size()==0) {
      return true;
    }
    return false;
  }

  @SuppressWarnings("unchecked")
public static List<Te> generateLegalMoves(Kyokumen k) {
    List<Te> gohote = new ArrayList<>();

    for(int suji=1;suji<=5;suji++) {
      for(int dan=1;dan<=5;dan++) {
        Position from=new Position(suji,dan);
        int koma=k.get(from);
        if (Koma.isSelf(k.teban,koma)) {
          for(int direct=0;direct<8;direct++) {
            if (canMove[direct][koma]) {
              Position to=new Position(suji+diffSuji[direct],dan+diffDan[direct]);
              if (1<=to.suji && to.suji<=5 && 1<=to.dan && to.dan<=5) {
                if (Koma.isSelf(k.teban,k.get(to))) {
                  continue;
                }
                addTe(gohote,k.teban,koma,from,to);
              }
            }
```

```
      }
      for(int direct=0;direct<8;direct++) {
        if (canJump[direct][koma]) {
          for(int i=1;i<5;i++) {
            Position to=new Position(suji+diffSuji[direct]*i,dan+diffDan[direct]*i);
            if (k.get(to)==Koma.WALL) break;
            if (Koma.isSelf(k.teban,k.get(to))) break;
            addTe(gohote,k.teban,koma,from,to);
            if (k.get(to)!=Koma.EMPTY) break;
          }
        }
      }
    }
  }
}

boolean isPutted[]={false,false,false,false,false,false,false,false};

ArrayList<Integer> motigoma = new ArrayList<>();
if (k.teban==SENTE) {
  motigoma=(ArrayList<Integer>) k.senteHand.clone();
} else {
  motigoma=(ArrayList<Integer>) k.goteHand.clone();
}

for(int i=0;i<motigoma.size();i++) {
  int koma=motigoma.get(i);
  int komashu=Koma.getKomashu(koma);
  if (isPutted[komashu]) {
    continue;
  }
  isPutted[komashu]=true;
  for(int suji=1;suji<=5;suji++) {
    if (komashu==Koma.FU) {
      boolean isNifu=false;
      for(int dan=1;dan<=5;dan++) {
        Position p=new Position(suji,dan);
        if (k.get(p)==(k.teban|Koma.FU)) {
          isNifu=true;
```

```
            break;
          }
        }
        if (isNifu) {
          continue;
        }
      }
      for(int dan=1;dan<=5;dan++) {
        if (komashu==Koma.FU) {
          if (k.teban==SENTE && dan==1) {
            continue;
          } else if (k.teban==GOTE && dan==5) {
            continue;
          }
        }
        Position from=new Position(0,0);
        Position to=new Position(suji,dan);
        if (k.get(to)!=Koma.EMPTY) {
          continue;
        }
        Te te=new Te(koma,from,to,false);
        if (isUtiFuDume(k,te)) {
          continue;
        }
        gohote.add(te);
      }
    }
  }

  gohote=removeSelfMate(k,gohote);

  return gohote;
}
      }
```

以下に本研究で作成した Human クラスのソースコードを示す.

```
package syougi;
import java.io.BufferedReader;
```

```
import java.io.InputStreamReader;
import java.util.List;

public class Human implements Constants{

static BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

public Te getNextTe(Kyokumen k) {
        List<Te> gohoTe = GenerateMoves.generateLegalMoves(k);

        Position toryoPos = new Position(0,0);
        Te te = new Te(0,toryoPos,toryoPos,false);

        do {
                if(k.teban==SENTE) {
                        System.out.println("先手番です");
                }else {
                        System.out.println("後手番です");
                }
                System.out.println("差し手を入力してください");

                String s ="";

                try {
                        s = reader.readLine();
                }catch(Exception e){
                        e.printStackTrace();
                        break;
                }

                if(s.equals("%TORYO")) {
                        break;
                }

                boolean promote = false;

                if(s.length()==5) {
                        if(s.substring(4,5).equals("*")) {
                                promote = true;
```

```
                }else {
                        continue;
                }
        }

        int fromSuji = 0,fromDan = 0,toSuji = 0,toDan = 0;
        try {
                fromSuji = Integer.parseInt(s.substring(0,1));
                fromDan  = Integer.parseInt(s.substring(1,2));
                toSuji   = Integer.parseInt(s.substring(2,3));
                toDan    = Integer.parseInt(s.substring(3,4));
        }catch(Exception e){
                continue;
        }

        int koma = 0;

        if(fromSuji==0) {
                koma = fromDan | k.teban;

                fromDan = 0;
        }
        Position from = new Position(fromSuji,fromDan);
        Position to   = new Position(toSuji,toDan);

        if(fromSuji!=0) {
                koma = k.get(from);
        }

        te = new Te(koma,from,to,promote);

    }while(!gohoTe.contains(te));

    return te;
}
}
```

以下に本研究で作成した Cpu クラスのソースコードを示す.

```
package syougi;

import java.util.List;
import java.util.Random;

public class Cpu implements Constants{
Random random;

public Cpu() {
        random = new Random();
}

public Te getRandomTe(Kyokumen k) {
        List<Te> gohoTe = GenerateMoves.generateLegalMoves(k);
        Te te = gohoTe.get(random.nextInt(gohoTe.size()));
        return te;
}

}
```

以下に本研究で作成したMainクラスのソースコードを示す.

```
package syougi;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class Main implements Constants{

static final int ShokiHaiti[][] = {
        {Koma.GHI,Koma.GKA,Koma.GGI,Koma.GKI,Koma.GOU},
        {Koma.EMP,Koma.EMP,Koma.EMP,Koma.EMP,Koma.GFU},
        {Koma.EMP,Koma.EMP,Koma.EMP,Koma.EMP,Koma.EMP},
        {Koma.SFU,Koma.EMP,Koma.EMP,Koma.EMP,Koma.EMP},
        {Koma.SOU,Koma.SKI,Koma.SGI,Koma.SKA,Koma.SHI},
};

static Cpu cpu[] = new Cpu[2];
```

```
static Human human[] = new Human[2];

static ArrayList<Object> kyokumenRireki = new ArrayList<>();

static ArrayList<Object> middleKyokumenRireki = new ArrayList<>();

static ArrayList<Integer> gohoTeNumber = new ArrayList<>();

static ArrayList<Te> bestTes = new ArrayList<>();

static Random random = new Random();

public static void main(String argv[]) {
		human[1] = new Human();
		cpu[0] = new Cpu();
		cpu[1] = new Cpu();

		Kyokumen k = new Kyokumen();
		k.teban=SENTE;
		for(int dan=1;dan<=5;dan++) {
			for(int suji=5;suji>=1;suji--) {
				k.ban[suji][dan] = ShokiHaiti[dan-1][5-suji];
			}
		}

		System.out.println(k.toString());

		Te bestTe = null;
		while(true) {
			Kyokumen fKyokumen = (Kyokumen)k.clone();

			if(k.teban==GOTE) {
				System.out.println("[[後手番]]");

				List<Te> gohoTe = GenerateMoves.generateLegalMoves(k);

				//Te te=cpu[1].getRandomTe(k);
				Te te = human[1].getNextTe(k);
```

```
            if(!gohoTe.contains(te)) {
                    System.out.println("合法手ではありません");
                    if(k.teban==SENTE) {
                            System.out.println("後手の勝利");
                    }else {
                            System.out.println("先手の勝利");
                    }
                    break;
            }

            k.move(te);

            if(k.teban==SENTE) {
                    k.teban=GOTE;
            }else{
                    k.teban=SENTE;
            }

            System.out.println(k.toString());
    }else{
            System.out.println("[[先手番]]");
            List<Te> fGohoTe = GenerateMoves.generateLegalMoves(k);

            for(int n=0;n<fGohoTe.size();n++) {
                    Te te = fGohoTe.get(n);
                    System.out.println(te.toString());
            }

            for(int j=0;j<fGohoTe.size();j++) {

                    Te firstTe = fGohoTe.get(j);

                    int wins = 0;

                    k = (Kyokumen)fKyokumen.clone();

                    k.move(firstTe);
```

```
if(k.teban==SENTE) {
        k.teban=GOTE;
}else {
        k.teban=SENTE;
}

int fTeban = k.teban;

Kyokumen nextK = (Kyokumen)k.clone();
for(int l=0;l<10;l++) {
        k = (Kyokumen)nextK.clone();
        for(int m=0;m<10;m++) {
        //while(true) {
                List<Te>              gohoTe              =
GenerateMoves.generateLegalMoves(k);

                if(gohoTe.size()==0) {
                        if(k.teban==SENTE) {
                        }else {
                                wins++;
                        }
                        break;
                }

                int sameKyokumen = 0;
                for(int n=0;n<kyokumenRireki.size();n++) {
                        if(kyokumenRireki.get(n).equals(k))
{
                                sameKyokumen++;
                        }
                }

                if(sameKyokumen>=4) {
                        System.out.println("千日手です");
                        break;
                }

                //System.out.println(k.toString());
```

```
                Te te;
                if(k.teban==SENTE) {
                        te=cpu[0].getRandomTe(k);
                }else{
                        te=cpu[1].getRandomTe(k);
                }

                //System.out.println(te.toString());

                if(!gohoTe.contains(te)) {
                        System.out.println("合法手ではありません");

                        if(k.teban==SENTE) {
                                //System.out.println("後手の勝利");

                        }else {
                                //System.out.println("先手の勝利");

                        }
                        break;
                }

                k.move(te);
                if(k.haveOu()==1) {
                        wins++;
                        break;
                }

                if(k.haveOu()==2) {
                        break;
                }

                if(k.teban==SENTE) {
                        k.teban=GOTE;
                }else {
                        k.teban=SENTE;
                }
        }
```

```
                        if(k.haveOu()==0&&k.evaluationFun()>0) {
                                wins++;
                        }

                        k.teban = fTeban;
                }
                System.out.println("j="+j+",wins="+wins);
                if(wins>=8) {
                        middleKyokumenRireki.add(nextK.clone());
                        gohoTeNumber.add(j);
                }
        }

        if(middleKyokumenRireki.size()!= 0) {
                int mostWins = 0;
                Te te = null;
                for(int l=0;l<middleKyokumenRireki.size();l++) {

                        int wins = 0;

                        for(int m=0;m<50;m++) {
                                k = (Kyokumen)middleKyokumenRireki.get(l);
                                //for(int n=0;n<10;n++) {
                                int fTeban = k.teban;
                                while(true) {
                                        List<Te>          gohoTe          =
GenerateMoves.generateLegalMoves(k);

                                        if(gohoTe.size()==0) {
                                                if(k.teban==SENTE) {

  System.out.println("後手の勝利");

                                                }else {

  System.out.println("先手の勝利");

                                                        wins++;
                                                }
                                                break;
                                        }
```

```
o=0;o<kyokumenRireki.size();o++) {

  if(kyokumenRireki.get(o).equals(k)) {




です");




  //System.out.println(te.toString());


ではありません");

  System.out.println("後手の勝利");

  System.out.println("先手の勝利");
```

```
int sameKyokumen = 0;
for(int

                sameKyokumen++;
        }
}

if(sameKyokumen>=4) {
        System.out.println("千日手

        break;
}

//System.out.println(k.toString());

if(k.teban==SENTE) {
        te=cpu[0].getRandomTe(k);
}else {
        te=cpu[1].getRandomTe(k);
}



if(!gohoTe.contains(te)) {
        System.out.println("合法手

        if(k.teban==SENTE) {

        }else {

        }
        break;
}
```

```
                k.move(te);

                if(k.haveOu()==1) {
                        //System.out.println("先手
の勝利");

                        wins++;
                        break;
                }

                if(k.haveOu()==2) {
                        break;
                }

                if(k.teban==SENTE) {
                        k.teban=GOTE;
                }else {
                        k.teban=SENTE;
                }


        }


        k.teban=fTeban;

}


if(mostWins < wins) {
        mostWins = wins ;
        System.out.println(gohoTeNumber.get(l));
        bestTe = fGohoTe.get(gohoTeNumber.get(l));
}
if(wins==50) {

        bestTe = fGohoTe.get(gohoTeNumber.get(l));
        bestTes.add(bestTe);
```

```
                }


            }
            middleKyokumenRireki.clear();

            k = (Kyokumen)fKyokumen.clone();
            k.teban=SENTE;

            System.out.println(bestTe.toString());

            if(bestTes.size()!=0) {
                bestTe = bestTes.get(random.nextInt(bestTes.size()));
            }

            k.move(bestTe);

            if(k.teban==SENTE) {
                k.teban=GOTE;
            }else {
                k.teban=SENTE;
            }
            System.out.println(k.toString());

            bestTes.clear();
            gohoTeNumber.clear();

        }else {
            Te te = cpu[0].getRandomTe(fKyokumen);

            k = (Kyokumen)fKyokumen.clone();
            k.teban=SENTE;

            k.move(te);

            if(k.teban==SENTE) {
                k.teban=GOTE;
            }else {
```

```
                                        k.teban=SENTE;
                                }
                                System.out.println(k.toString());
                        }


                }

                if(k.haveOu()!=0) {
                        break;
                }
        }
        if(k.haveOu()==1) {
                System.out.println("先手の勝利");
        }else if(k.haveOu()==2) {
                System.out.println("後手の勝利");
        }else {
        System.out.println(k.evaluationFun());
        }
    }
}
```