卒業研究報告書

題目

ミニチェスの対局を使ったアルゴリズムの研究

指導教員

石水　隆　　講師

報告者

09-1-037-0075

高尾　晃徳

近畿大学理工学部情報学科

平成26年1月31日　提出

概論

チェスは二人有限完全情報ゲームである。二人有限完全情報ゲームとは理論上は完全な先読みが可能であり、双方のプレイヤーが最善手を指せば、必ず先手必勝か後手必勝か引き分けかが決まるという点である。しかし、そのようなゲームは理論上は完全解析できるが、多くのゲームでは組み合わせが膨大になり計算機を駆使しても、完全解析は現実的ではない。チェスも可能な局面数が 10^{50} 通りあるとされており、現在の計算機性能では完全解析は不可能である。完全解析が不可能なゲームであっても、計算機で強い手を選択できる場合がある。計算機で着手を決定できる手法としては、一定手数までの局面の先読み、定跡データベース，対戦データベース、終盤での完全読み等が知られている。チェスでもそのような手法により強いコンピュータチェスが作られており、コンピュータチェスの大会なども開かれている。

本研究では、盤面サイズ 5×6 のミニチェスのプログラムを作成した。ある局面で指せる手が複数あるときに、どの手を採用するかを決定するには、その手を指した後にできる局面の評価が必要となる。本研究では、一手先の局面を生成し、その局面を評価し最善手を探索するプログラムを作成する。

目次

1　序論

1．1　二人零和有限確定完全情報ゲーム

将棋やリバーシ等に代表されるボードゲームは、二人零和有限確定完全情報ゲームに分類される。二人零和有限確定完全情報ゲームとは、二人または二チームでゲームを行い、ゲーム終了時双方のプレイヤーの利得合計が零、双方のプレイヤーの着手可能手が有限、プレイヤーの着手以外がゲームに影響を与える偶然の要素がはいらず、そして各プレイヤーの着手の意思決定の情報が知ることができるゲームである。二人零和有限確定完全情報ゲームに分類するゲームの特徴として、理論上完全な先読みが可能であり、双方のプレイヤーが最善手を打てば、先手必勝か後手必勝か引き分けが決まる。二人零和有限完全情報ゲームは、その性質上解析を行い易いため、ゲーム理論において様々な研究がなされてきた。また、人工知能の分野においても広く研究がなされている。

1．2　二人零和有限確定完全情報ゲームの完全解析に関する既知の結果

前述したように、二人零和有限確定完全情報ゲームは双方最善手を打った場合、先手勝ち、後手勝ち、引き分けのどれになるかはゲーム開始時点で決定しており、理論上、全ての可能な局面を解析することができれば最善の手を打つことができる。しかし多くのボードゲームでは、可能な局面の総数が極めて大きいため、完全解析を行うことは不可能である。例を挙げれば、可能な局面数はリバーシが 10^{28} 通り、チェスが 10^{50} 通り、将棋が 10^{69} 通り、囲碁が 10^{170} 通り程度あるとされており、現在の計算機の性能を越えている。一方、可能な局面数が少ないゲームでは完全解析されているものもある。連珠は双方最善手を打った場合、47 手で先手が勝つ[8][8)]。チェッカーは双方最善手を指すと引き分けとなる[9][9)]。

局面数が大きいゲームについては、ゲーム盤をより小さいサイズに限定した場合の解析も行われている。サイズ 6x6 のリバーシでは、双方最善手を打つと 16 対 20 で後手勝ちとなる[10][10)]。また、サイズ 4x4 の囲碁は双方最善手を打つと持碁(引き分け) 、5x5 の囲碁は黒の 25 目勝ちとなる[16][11)12)]。将棋については、盤面のサイズや使用する駒の種類を減らした5五将棋[21][13)]やゴロゴロ将棋[11][14)]などのミニ将棋がある。5五将棋はサイズ 5x5 の盤と 6 種類の駒、ゴロゴロ将棋はサイズ 5x6 の盤と 4 種類の駒を使用する。図 1，図 2 に5五将棋およびゴロゴロ将棋の盤と駒の初期配置を示す。これらは本将棋と比べて可能な局面数が少ない。しかしながら現在のところまだこれらは完全解析はされていない。

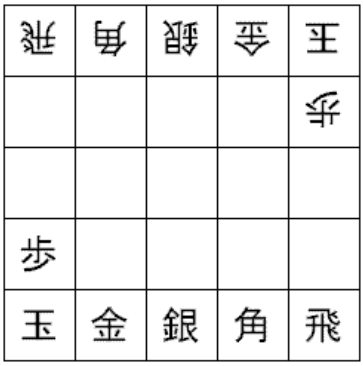
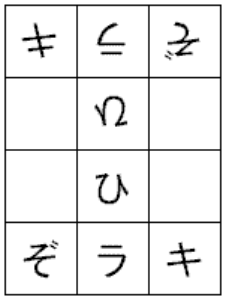


図1　5五将棋の盤面と駒の初期配置

図2ゴロゴロ将棋の盤と駒の初期配置

完全解析されているミニ将棋として、どうぶつしょうぎ[8][15)](以下動物将棋とする)がある。動物将棋はサイズ 3*4 の盤と、ライオン，象，キリン，ひよこの 4 種類の駒を使用する幼児向けのミニ将棋である。図 3 に動物将棋の盤と駒の初期配置を示す。動物将棋は完全解析により双方最善手を指した場合、78 手で後手が勝つことが判明している[12]。



ラ:ライオン
ぞ:象
キ:キリン
ひ:ひよこ

図3　どうぶつしょうぎの盤と駒の初期配置

チェスは将棋と違い取った相手の駒を使用することができない。そのためにチェスの終盤局面では駒の数が少ない場面がよく現れる。終盤において双方の駒が少なくなると、可能な局面数が減るため、完全解析可能となる。1980 年代には、残り駒数が 5 駒以下の場合の完全解析がされている。1984 年、Nefkens は、盤上に双方のキング以外はクイーンが 1 個のみある KQK エンディングにおける完全解析データベースを作成した[18]。Nefkens の結果によれば、KQK エンディングの可能な局面数は 40,960 通りであり、双方最善手を指せば最大 19 手でチェックメイトできる。

また、ルークが 1 個のみある KRK エンディングでは、可能な 40,960 局面全てで 31 手以内にチェックメイトできる。ポーンが 1 個のみある KPK エンディングについては、1989 年に Zennler により完全解析データベースが作られている[19]。Zennler の結果によれば、KPK エンディングでは、可能な局面数は 98,304 通りであり、そのうち 62,480 局面は白勝ちの局面であり、最大 37 手でクイーンに昇格でき、その後最大 17 手でチェックメイトできる[27]。5 駒以下の場合の解析については、1970 年代から特定の駒組み合わせに対して解析が行われてきた。例えば、1978 年には、Arlazarov らにより KROKR エンディングが、1986 年には、Komissarchik らにより KQPKQ エンディングに解析がなされている[29]。1989 年には、Stiller により、残り駒数が 5 駒以下の場合についての完全解析がなされた[17]た。現在では、6 駒以下のエンディングについては完全解析されており、フリーのデータベースとして利用できるようになっている。[30]このため、多くのチェスソフトでは、データベースを組み込むことにより残り駒数が少なくなると最善手を指せるようになっている。

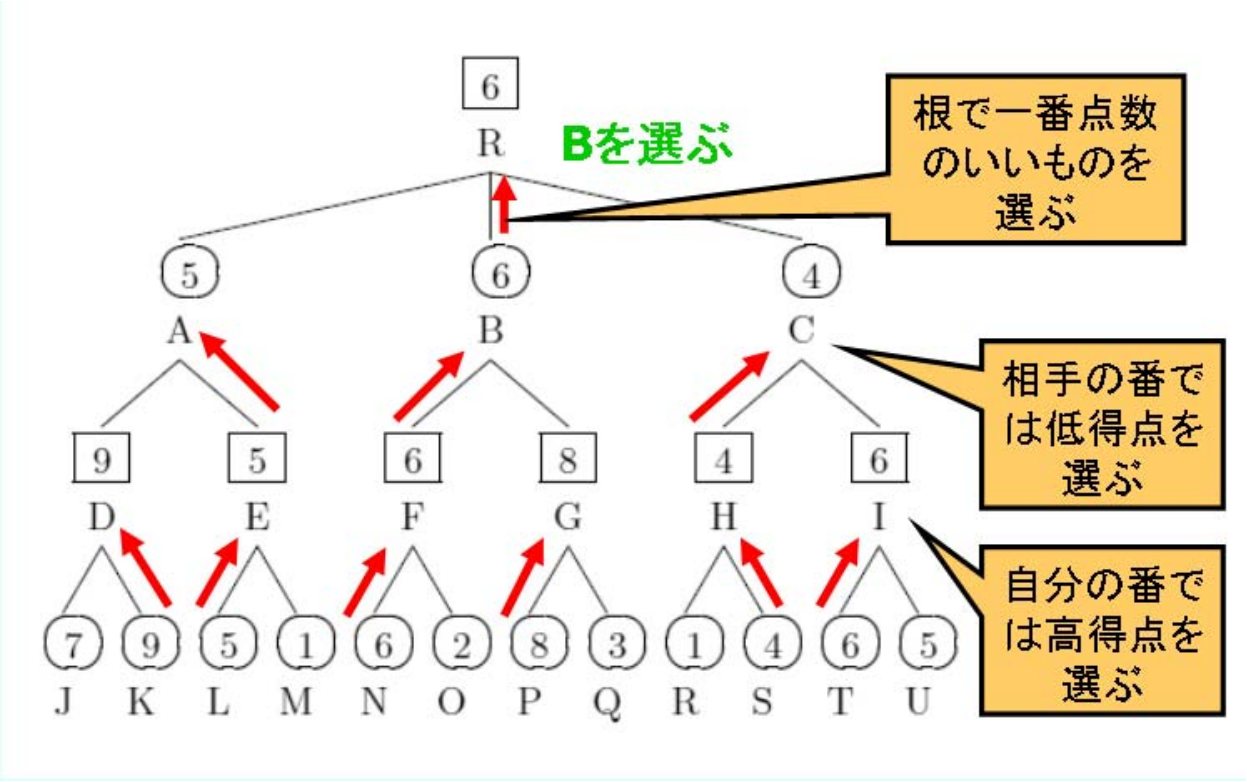
1．3　完全解析されていない二人零和有限確定完全情報ゲームに対する手法

可能な局面数が多いゲームに対して完全解析を行うことは困難である。そのようなゲームに対しては確実な最適手を得ることはできないが、局面の評価値計算、定跡データベース、一定手数の先読み、終盤での必勝読みと完全読み、モンテカルロ法[21)]などを用いてより有利だと思われる手を選択することができる。

定跡データベースとは、定跡をデータベース化し、各局面で有効な定跡があればそれに従って打つという手法である。定跡データベースを使用することで強いプログラムとなる。しかし、相手があえて定跡以外の手を打つなどして、データベースに無い局面が出てきたときにはこの手法は使えない。

モンテカルロ法とは、各着手可能手に対し、その手から先終局までをランダムに指し勝敗判定を行うという作業を数千～数万回繰り返し、最も勝率の高い着手可能手を採用するというものである。この手法は将棋ではあまり使われないが、局面数が極めて多い囲碁プログラムでは最近主流になっている。

一定手数の先読みとは、可能な範囲で一定数の先の手を読み、その手から作られる局面の評価値を求め、最も評価値が高い手を採用することである。局面の評価値は、盤上に置かれている駒の種類やその位置、着手可能手の数、各駒の稼働範囲等から計算される。こうした先読みを行う際には、min-max 法およびそれを改良した alpha-beta 法がよく用いられる。下記の図の未展開節点から①自分の番（MAX 節点）なら一番点数の高いものを選ぶ。②相手の番（MIN 節点）なら一番点数の低いものを選ぶ。ことにより、の節点を点数化していき根節点で一番点数の高くなるものを選ぶ。ミニマックス法の説明を図4に示す。



□ MAX節点（自分の番）　　○ MIN節点（相手の番）

図4 ミニマックスの例

ゲーム終盤になるそこから勝負が付くまでの手数が少なくなり、また指せる手が限定されてくるため、勝負が付くまで読み切ることが可能となる。終盤での読みは、必勝読みと完全読みがある。必勝読みとはゲーム終盤で勝敗のみを読み切り、必ず勝てる手を指すことを言う。完全読みとは、そこから得られる全ての局面を読み、最も点数の高くなる手を指すことを言う。必勝読みの方が計算時間が少なくてすむため、一般にまず必勝読みで勝ちを確定させた上で、残り手数が少なくなると完全読みに切り替えてより点数の高い勝ちを目指すことが多い。また、チェスでは 1.2 節で述べた通り、駒数が 6 個以下になれば、データベースを利用することにより、最善手を指すことができるようになる。

以上の手法を用いることにより、完全解析を行わなくてもある程度の強さのプログラムを作ることが可能であり、ゲームによってはプロに勝つこともできる。将棋では、将棋プログラムボンクラーズ[23][19]が 2012 年 1 月に元プロ棋士の米長邦雄永世棋聖と対戦しボンクラーズ先手 113 手で勝った[24][20]。リバーシでは、リバーシプログラム Logistello[25][22]が 2002 年 5 月に日本チャンピオン富永健太氏と対戦を行い、Logosttelp 先手で 38 対 26 で Logostello の 12 目勝ち、富永氏先手で 23 対 41 で Logistello の 18 目勝ちであった[26][23]。チェスでは、1996 年に IBM のコンピュータであるディープ・ブルー[20][17]がガルリ・カスパロフと対戦し、1 つのゲームとしては、初めて世界チャンピオンに勝利を収めた[10][18]。ただし、これは 6 戦中の 1 勝に過ぎず、全体ではカスパロフの 3 勝 1 敗 2 引き分けであった。しかし、翌 1997 年に、ディープ・ブルーは、2 勝 1 敗 3 引き分けとカスパロフ相手に雪辱を果たした[30]。現実的にはこれだけの試合数で実力は評価できない

が、世界チャンピオンと互角に戦えるだけの能力になったとIBMは宣伝したのがコンピュータチェスに関する既知の結果である。また、代表的なチェスプログラムとして、1912年に作成されたエル・アヘドレシスタ[2]である。1番最初に作成されたコンピュータチェスであるが、チェスの終盤だけを扱った限定的なためチェスの勝負をするものではなかった。人間相手に勝利したチェスプログラムの代表としては、フリッツ[7]である。フリッツはチェス専用コンピュータであるディープ・ブルーのプロトタイプ版に勝ったという実績を残している。

1．4　本研究の目的

本研究は、ミニチェスの完全解析を目標とする。しかし、本チェスより小さいとは言え、ミニチェスの可能な局面数は非常に多く、完全解析は難しい。そこで、完全解析の前準備として、本研究ではミニチェスプログラムを作成し、その動作を検証することにより、完全解析への足がかりとする。

1．5　本報告書の構成

本報告書の構成は以下の通りである。まず2章において、本研究の対象となるミニチェスについて説明する。続いて第3章において、本研究で作成したミニチェスプログラムについて述べる。第4章で本研究で得られた結果について述べ、第5章でまとめおよび考察を行う。

2.　ミニチェス

本研究では、一般的に使用されている盤面8×8のチェスとは違い、盤面5×6のミニチェスでゲームを行う。このミニチェスでは、各プレイヤーはキング，クイーン，ビショップ，ナイト，ルークを1個ずつと、ポーン5個を使用する。その他のルールは一般のチェスと同様であり、相手のキングをチェックメイトしたプレイヤー側の勝利となる。ミニチェスの盤面と初期配置を図5に示す。

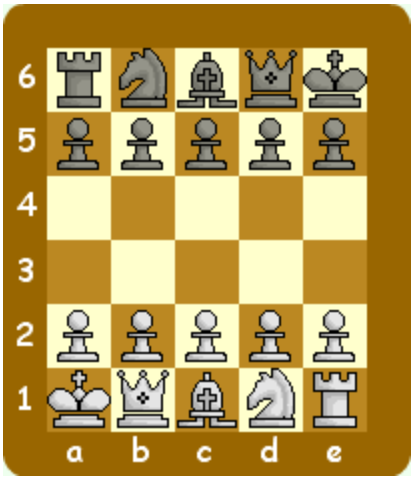


図 5 ミニチェスの盤面と駒の初期配置

2.1 駒の動き ポーン

ポーンは単純な進み方と、駒を取っての進み方が違う。ポーンの 1 マス前方のマスに駒が無い場合、ポーンはそのマスへ進むことができる。また、初期位置にいるポーンに限り、ポーンの前方 2 マスが空いていれば、1 手で 2 マス進むこともできる。斜め直前に相手の駒があればその駒を取りながらその駒のあった位置に進むことができる。図6および図7にポーンの動きを示す。ポーンが相手の最終段に到着すると、クイーン、ルーク、ナイト、ビショップの任意の駒に成れる。これを「成る」「昇格する」という。図8にポーンのポーンの昇格を示す。

ポーンに関するもうひとつの特殊ルールとして、アンパサンがある。アンパサンは、ポーン同士での特殊な取り方で、自分のポーンが最初の位置から3マス進んだ位置にいる時に、隣りの列のまだ動いていない相手のポーンが一気に2マス進んだ場合、1 マスだけ進んだ時と同じようにそのポーンを取る事ができるというルールである。図 9 にアンパサンでの動きを示す。 ただし、本研究のミニチェスのルール自体ではアンパサンは適用外とする。

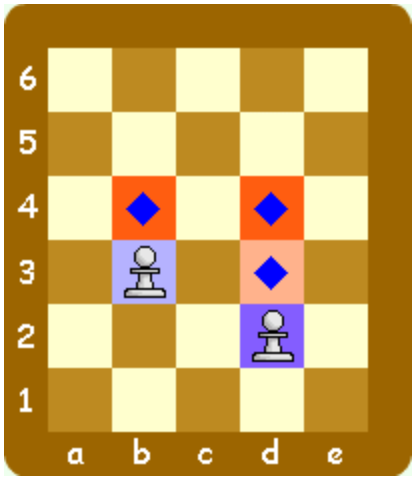


図6 ポーンの動き

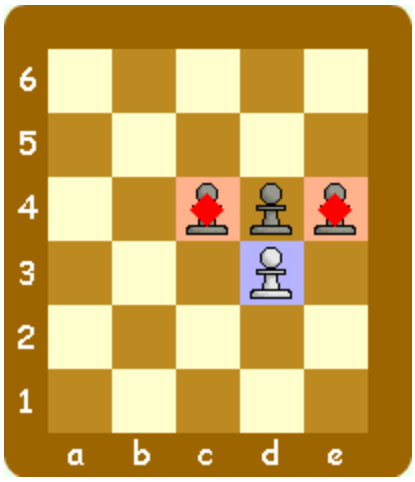
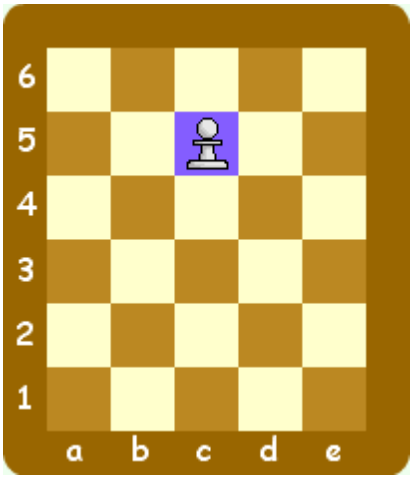


図7 ポーンの動き(敵の駒を取る場合)



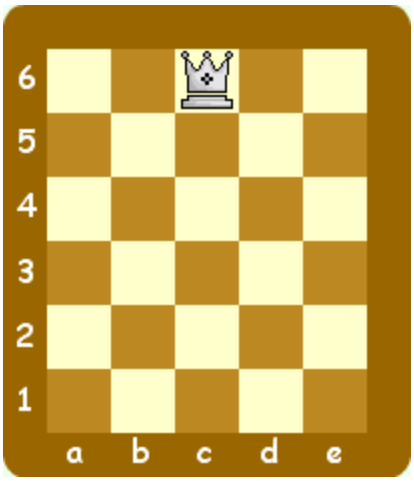


図8 ポーンの昇格

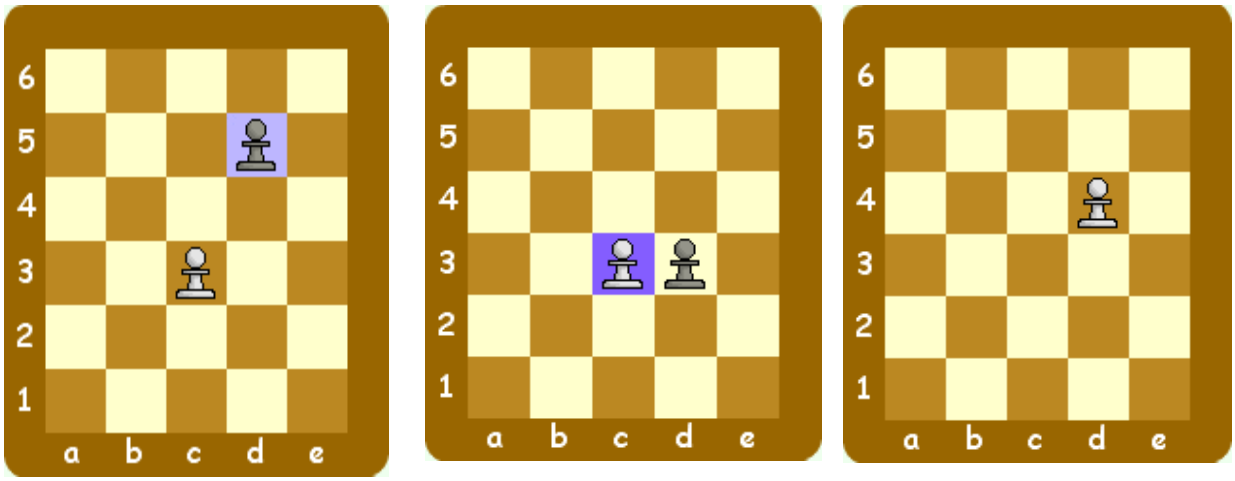


図 9 アンパサンでの動き

2．2 駒の動き ルーク

ルークは将棋の飛車と同じで縦横いくらでも進めることができる。ただし、駒を飛び越すことはできない。駒の動きを図 10 に示す。

2．3 駒の動き ナイト

ナイトは一つ進んだ盤の両斜め進行方向に進むことが可能、つまり将棋の桂馬に似ているが進行方向が八方向という点が特徴的な駒である。駒の動きを図 11 に示す。

2．4 駒の動き ビショップ

ビショップは将棋の角行と同じで斜め方向にいくらでも進める。駒の動きを図 12 に示す。

2．5 駒の動き クィーン

クィーンはルークとビショップを合わせた動きで、縦横斜めにいくらでも進むことができる。
ただし、駒を飛び越すことはできない。勝敗にもっとも効果的な駒である。駒の動きを図 13 に示す。

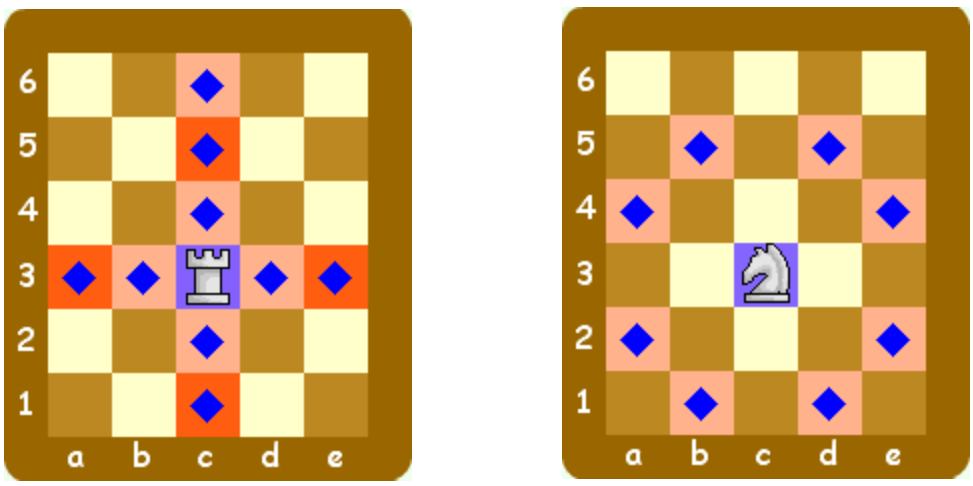


図 10 ルークの動き

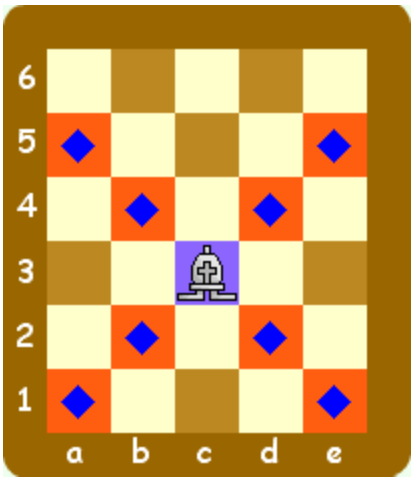


図 11 ナイトの動き

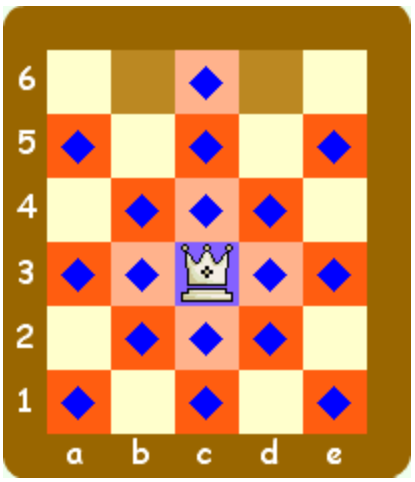


図 12 ビショップの動き

図 13 クィーンの動き

2．6 駒の動き キング

キングは将棋の王将と同じで縦横斜めに1個だけ進むことができるが、敵駒の効いている位置には進むことができない。 また、チェスにはキングとルークを一手で同時に動かす特殊な手を指すキャスリングというルールがある。キャスリングは以下の条件を満たすときに可能となる。

- キングとキャスリングさせるルークが共に初期配置から移動していない
- キングとキャスリングさせるルークとの間に駒が無い
- 現在キングがチェックされておらず、キングが通過するマスおよびキングの移動後のマスに敵の駒が利いていない
- キングとキャスリングさせるルークが同ランク上にある

上記の条件を満たしているとき、キングをルークに向かって 2 マス移動させ、そのルークをキングを飛び越えてキングの隣のマスに移動することができる。駒の動きを図14に示す。また、キャスリングでの動きを図 15 に示す。ただし、本研究のミニチェスのルール自体適用していない。

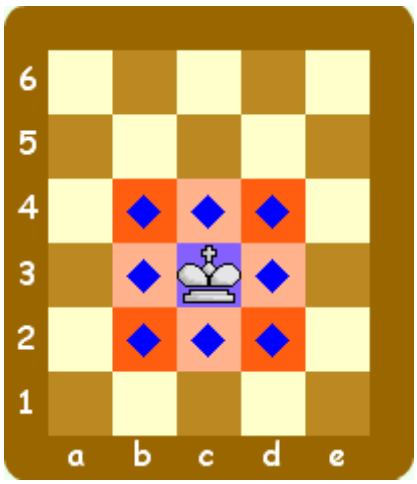


図14　キングの動き

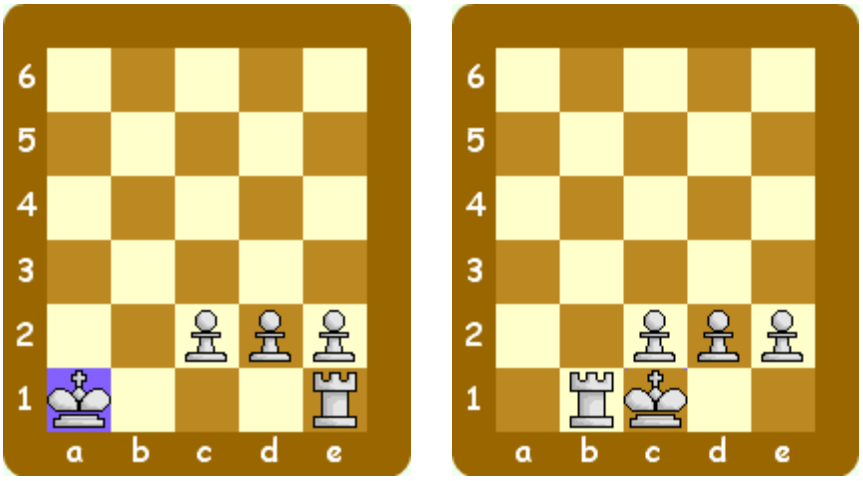


図 15　キャスリングでの動き

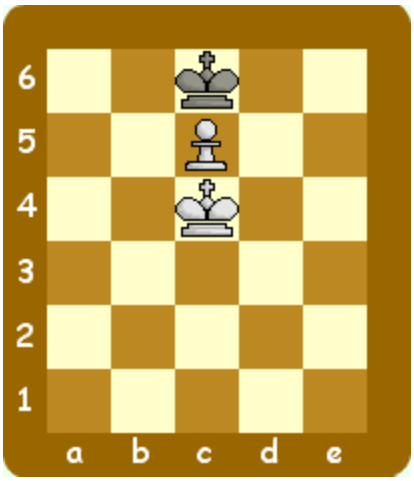


図 16　ステイルメイトの例

2．7　ステイルメイト

現在キングにチェックはかかっていないが、その状態で指せる合法手が 1 つもないときの状態をステイルメイトという。キングは敵駒が効いているマスには移動できない。その為、自駒がキングのみになった場合、相手の駒の配置によっては自駒をどこにも動かせなくなってしまう場合がある。また、キング以外の駒が残っていても、その駒に動ける地点が無かったり、動くとキングにチェックがかかってしまう状態であったりして、合法手が 1 つも無くなってしまう場合がある。図 16 にステイルメイトとなった例を示す。ステイルメイトになった場合は、引き分けとなる。

2．8　ミニチェスでは適用外の本チェスのルール

2．8．1　千日手　(パペチュアル)

千日手は別名スリーフォールド・レビティション（同形三復）と呼ばれている。相手の手で同一局

面が3回生じるときに引き分けとなる。ただし自動的に引き分けになるのではなく、自分の手番の時に指摘しなければならない。

2．8．2 50手引き分けルール

50 手ルールとは、50 手の間、ポーンに動きがなくキャプチャもされていない場合、ゲームはドローとなるルールである。

2．8．3 戦力不足

盤上に敵駒がキングのみになった場合、以下のいずれかの駒があればチェックメイトできる。

・クイーン　1 個
・ルーク　1 個
・ビショップ　2 個
・ビショップ 1 個とナイト 1 個
・昇格可能なポーン 1 個

逆に、上記の条件を満たさない場合、すなわち、優勢な側に残っている駒がビショップ 1 個のみ、あるいはナイトが 2 個以下のみとなった場合は、優勢なプレイヤーが残りのゲームでチェックメイトをかけることが不可能なためドローとなる。この状況で可能なチェックメイトは、劣勢側が 50 手ルールの途中で失敗した場合のみとなる。

3　ミニチェスプログラム

本研究では、Java を用いてミニチェスプログラムを作成した。付録 1 にプログラムのソースを示す。

3．1 ミニチェスプログラムの戦略

本節では、ミニチェスプログラムで着手を決定するために用いている戦略について述べる。

本研究で作成したプログラムは、一手先の局面において各駒に重みをつけ評価している。つまり、その局面の評価値を求め、もっとも高い評価値を持つ手を採用する。チェスでは、一般に盤上にある自駒が多い方が有利であるので、駒の種類に応じて自駒に正、敵駒に負の値を付加し、その合計値をその局面の評価値としている　表 1 に各駒に付加した評価値を示す。

表 1 各駒の評価値

駒	ポーン	ルーク	ナイト	ビショップ	クイーン	キング
評価値	1	5	3	3	9	1000

3．2　ミニチェスプログラムの詳細

本節では、ミニチェスプログラムの詳細について述べる。以下に本プログラムの各クラスについて説明する。

３．２．１　MiniChess クラス

MiniChessクラスは本研究において作成したクラスを利用し、実際に処理を行うクラスである。各クラスについては次項以降に記述する。

３．２．２　Computer クラス

Computer クラスは本研究で考案したコンピュータの動きを行うクラスである。表 2 に Computer クラスの各メソッドについてまとめる。

表 3 Computer クラスのメソッド

メソッド	処理内容
Computer(Phase, boolean)	コンストラクタ
int[] canMove()	現在の盤面より着手可能手を探索し、返り値として着手可能な盤面をリストで返す。
int checkpoint (Phase)	引数で与えた盤面より評価を行い点数を計算する。その点数を返り値とする。
Int[] select()	着手可能手のリストより最善の手を選択する。

３．２．３　Piece クラス

Piece クラスは各駒 の動きを記述する抽象クラスである。具体的な処理はサブクラスに記述している。

メソッド	処理内容
Piece(position,side)	コンストラクタ
boolean canMove(next,list)	引数で与えた盤面に着手できるかを判定。
void setPosition(position)	引数で与えた場所に駒の場所を移動する。
void setName(name)	名前を設定する。
boolean getSide()	どちらの番の駒であるかを返すゲッター。
String getName()	駒の名前を取得するゲッター。
public boolean istouch()	駒が一度でも操作されているかを返す。

３．２．４　Phase クラス

Phase クラスは盤面情報を持ち進行の判定を記述したクラスである。

メソッド	処理内容
Phase()	コンストラクタ
int getXX()	各駒の位置を取得するゲッター。
int getPieceNum()	盤面にある駒の数を返す。
ArrayList getList()	盤面を表す各駒を含むリストを返す。
void showPhase()	盤面を表示する。
void move()	ゲームの進行を記述したメソッド
int checkWinLose()	ゲームの勝敗がついているか確認する。
Phase clone()	盤面を複製する。

4. 結果・考察

本研究で作成したプログラムの評価を行うために、可能な手をランダムに指すプログラムとの対戦を 10000 回行った。その実行結果を表2に示す。

表2 ランダム戦の勝ち負け

先手	後手	先手勝ち	後手勝ち	先手勝率
本研究	ランダム	6242	3758	62%
ランダム	本研究	3872	6128	39%

試行回数約 10000 回

表 1 より、駒ごとの評価を用いたアルゴリズムで 62%の勝率という結果が示された。勝率 62%とは 3 回に 1 回は負けるということなので、本研究で作成されたプログラムは決して強い CPU プログラムだとは言えない。

5. 結論

本研究では、ミニチェスのプログラムを作成した。

本研究で作成したプログラムは各駒に評価値を割り振って盤面評価を行っているが、表 1 に示したようにランダムを相手に勝率 62%は決して高くない数字である。よって、本プログラムはまだまだ改良の余地がある。今後の課題としては探索の効率化を図り、より先の手まで先読みすること、駒

の価値の和以外の局面の評価関数を検討することが挙げられる。
　また、CPU の強さのレベルの選択ができるようにすること、一般的に使用されている8×8のチェスに拡張することも今後の課題の一つである。

謝辞

　本研究を行うにあたり、直接指導して頂いた近畿大学理工学部情報学科情報論理工学研究室石水講師には大変お世話になりました。日頃の研究に関する議論や研究のサポート、研究へのアドバイス、論文指導に対し適切なご助言と励ましをいただきましたので、ここに感謝の意を表します。

参考文献

1) ミニチェス, 松田道弘 編, 世界のゲーム辞典, p .134, 東 京堂出版, 1989
2) D.Levy, M.Newborn 著, 飯田弘之, 吉村信弘 訳, コンピュータチェス 世界チャンピオンへの挑戦,サイエンス社, 1994
3) 池泰弘 著, コンピュータ将棋のアルゴリズム―強アルゴリズムの探求とプログラミング, 工学社, 2005.

4) 池泰弘 著, Java 将棋のアルゴリズム, 工学社, 2007
5) 渡井美代子 著, 松本康司 監修, 図解早わかりチェス 初歩の定石と必勝のコツ, 日東書院, 2002
6) 湯川博士 著, 若島正 監修, 将棋ファンでも楽しめる初めてのチェス1手・2手詰集, 山海堂, 2003
7) Jacques Marie Pinoeu, ジャック・ピノーのダイナミックチェス入門, 山海堂, 1995.
8) Janos Wagner and Istvan Virag, Solving renju, ICGA Journal, Vol.24, No.1, pp.30-35 (2001),http://www.sze.hu/~gtakacs/download/wagnervirag_2001.pdf
9) Jonathan Schaeffer, Neil Burch, Yngvi Bjorsson, Akihiro Kishimoto, Martin Muller, RobertLake, Paul Lu, and Steve Suphen, Checkers is solved, Science Vol.317, No,5844, pp.1518-1522 (2007). http://www.sciencemag.org/content/317/5844/1518.full.pdf
10) Joel Feinstein, Amenor Wins World 6x6 Championships!, Forty billion noted under the tree (July 1993), pp.6-8, British Othello Federation's newsletter., (1993),http://www.britishothello.org.uk/fbnall.pdf
11) 清慎一, 川嶋俊:探索プログラムによる四路盤囲碁の解, 情報処理学会研究報告, GI-2000(98), pp.69--7(2000), http://ci.nii.ac.jp/naid/110006407446
12) Eric C.D. van der Welf, H.Jaap van den Herik, and Jos W.H.M.Uiterwijk, Solving Go on Small Boards, ICGA Journal, Vol.26, No.2, pp.92-107 (2003).
13) 日本 5 五将棋連盟, http://www.geocities.co.jp/Playtown-Spade/8662/
14) 「ごろごろどうぶつしょうぎ」発売開始！, お知らせ, 日本将棋連盟, 2012 年 11 月 26 日, (2012), http://www.shogi.or.jp/topics/2012/11/post-652.html
15) 北尾まどか, 藤田麻衣子, どうぶつしょうぎねっと, (2010), http://dobutsushogi.net/
16) 田中哲郎:「どうぶつしょうぎ」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/
17) IBM100 – Deep Blue, IBM, (1997), http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/
18) Michael Khodarkovsky and Leonid Shamkvoich, 人間対機械 ー チェス世界チャンピオンとスーパーコンピューターの闘いの記録, 毎日コミュニケーションズ, (1998)
19) 伊藤英紀, A 級リーグ差し手 1 号, (2013), http://aleag.cocolog-nifty.com/
20) 米長邦雄, われ敗れたり コンピュータ棋戦のすべてを語る, 中央公論社, (2012).
21) 美添一樹, 山下宏, 松原仁, コンピュータ囲碁—モンテカルロ法の理論と実践—, 共立出版, (2012).
22) Michael Buro , LOGISTELLO, 2002, https://skatgame.net/mburo/log.html
23) Michael Buro , Tominaga vs. Logistello, 2002, https://skatgame.net/mburo/iwec.html
24) Harry Nefkens, Constructing Data Bases to Fit a Microcomputer, ICCA Journal, Vol. 8, No. 4, pp. 219-224, 1985.
25) Hans Zellner, The KPK Database Revisited. ICCA Journal, Vol. 12, No. 2, pp.78-82, 1989.
26) Lewis Stiller, Parallel Analysis of Certain Endgames. ICCA Journal, Vol. 12, No. 2, pp. 55-64, 1989.
27) Vladimir Arlazarov and Aron Futer, Computer analysis of a rook endgame, Machine Intelligence 9, University of Edinburgh Press, 1978.
28) Edik Komisssarchink, Aron Futer and Vladimir Arlazarov, Computer anakysis of a Queen endgame, ICCA Journal, vol.9, No.4, pp.189-200, 1986.
29) Kirill Kryukov, Endgame Tablebases Online, 6-men endgame analysis free for everyone, 2013, http://kirill-kryukov.com/chess/tablebases-online/
30) Aaron Tay, A guide to Endgames Tablebase, 2006, http://horizonchess.com/FAQ/Winboard/egtb.html

付録

以下に本研究で作成したミニチェスのプログラムソースを示す。

```
package MiniChess;

public class MiniChess {
	public static void main(String[] args){
		/*
```

```
                初期配置
                false
        50 51 52 53 54
        Ro Kn Bi Ki Qe
        40 41 42 43 44
        P0 P1 P2 P3 P4

    10 11 12 13 14
        P4 P3 P2 P1 P0
        00 01 02 03 04
    Qe Ki Bi Kn Ro
         true
              */

              /*
              50 51 52 53 54
              40 41 42 43 44
            30 31 32 33 34
            20 21 22 23 24
            10 11 12 13 14
            00 01 02 03 04
               */

              Phase phase = new Phase(new King(01,true),new Qeen(00,true),new
Bishop(02,true),new Knight(03,true),new Rook(04,true),new Pawn(14,true),new
Pawn(13,true),new Pawn(12,true),new Pawn(11,true),new Pawn(10,true),
                              new King(53,false),new Qeen(54,false),new
Bishop(52,false),new Knight(51,false),new Rook(50,false),new Pawn(40,false),new
Pawn(41,false),new Pawn(42,false),new Pawn(43,false),new Pawn(44,false),true);

              int win=0;
              while((win=phase.checkWinLose())==0){
                    phase.showPhase();
                    phase.move();
              }

              phase.showPhase();
              if(win==-1){
                    System.out.println("後手の勝ち");
              }else{
                    System.out.println("先手の勝ち");
              }

        }
}
```

```
package MiniChess;
import java.util.*;

public class Computer {
        Phase phase;
        boolean side;
        ArrayList<Piece> list;
        final int[]
map={0,1,2,3,4,10,11,12,13,14,20,21,22,23,24,30,31,32,33,34,40,41,42,43,44,50,51,52,53,
```

```
54};

	public Computer (Phase phase,boolean side){
		this.phase = phase;
		this.list = phase.getList();
		this.side = side;
	}

	private int[][] canMove(){
		int[][] canMove;
		canMove=new int[list.size()][30];
		//初期化
		for(int[] can:canMove){
			Arrays.fill(can, -1);
		}

		Piece piece=null;
		int ii;
		for(int index=0;index<list.size();index++){
			ii=0;
			piece = list.get(index);
			if(piece.getSide()==side){
				for(int next:map){
					if(piece.canMove(next, list)){

	if(piece.getName().equals("Pa")&&side&&next>=50){
							//Kn
							canMove[index][ii]=next*100+1;
							ii++;
							//Bi
							canMove[index][ii]=next*100+2;
							ii++;
							//Qe
							canMove[index][ii]=next*100+3;
							ii++;
							//Ro
							canMove[index][ii]=next*100+4;
							ii++;
						}else
if(piece.getName().equals("Pa")&&!side&&next<=4){
							//Kn
							canMove[index][ii]=next*100+1;
							ii++;
							//Bi
							canMove[index][ii]=next*100+2;
							ii++;
							//Qe
							canMove[index][ii]=next*100+3;
							ii++;
							//Ro
							canMove[index][ii]=next*100+4;
							ii++;
						}else{
							canMove[index][ii]=next;
							ii++;
						}
					}
				}
			}
		}
		return canMove;
```

```
}

/**
 * 次の盤面を引数で与え、その盤面の得点を計算し、返す
 * 次の盤面@param nextPhase
 * 点数@return point
 * Qe9 Ro5 Bi3 Kn3 Pa1
 */
private int checkPoint(Phase nextPhase){
	int point=0;
	for(Piece piece:nextPhase.getList()){
		//自駒
		if(piece.getSide()==side){
			if(piece.getPosition()!=-1){//盤内
				if(piece.getName().equals("Qe")){
					point +=9;
				}else if(piece.getName().equals("Ro")){
					point += 5;
				}else if(piece.getName().equals("Bi")){
					point += 3;
				}else if(piece.getName().equals("Kn")){
					point += 3;
				}else if(piece.getName().equals("Pa")){
					point += 1;
				}
			}
		}
		//敵駒
		else{
			if(piece.getPosition()!=-1){//盤内
				if(piece.getName().equals("Qe")){
					point -=9;
				}else if(piece.getName().equals("Ro")){
					point -= 5;
				}else if(piece.getName().equals("Bi")){
					point -= 3;
				}else if(piece.getName().equals("Kn")){
					point -= 3;
				}else if(piece.getName().equals("Pa")){
					point -= 1;
				}
			}
			//敵のキングが盤外(勝ち確定)
			if(piece.getName().equals("Ki")&&piece.getPosition()==-1){
				point +=1000;
			}
		}
	}
	point*=10;
	point+=(new Random().nextInt(10));
	return point;

}

public int[] serect(){
	System.out.println();
	int[][] canMove= canMove();
	int Max=-10000;
	int point;
	int[] select={-1,-1};//{listの何番目の駒か,駒の移動場所}
	Phase nextPhase;
```

```
		int index=-1;
		for(int[] move:canMove){
			index++;
			for(int next:move){
				if(next!=-1){
					nextPhase=phase.clone();
					if(next>100){
						switch(next%100){
						case 1:

	nextPhase.getList().get(index).setName("Kn");

	nextPhase.getList().get(index).setPosition(next/100);
							break;
						case 2:

	nextPhase.getList().get(index).setName("Bi");

	nextPhase.getList().get(index).setPosition(next/100);
							break;
						case 3:

	nextPhase.getList().get(index).setName("Qe");

	nextPhase.getList().get(index).setPosition(next/100);
							break;
						case 4:

	nextPhase.getList().get(index).setName("Ro");

	nextPhase.getList().get(index).setPosition(next/100);
							break;
						}
					}else{

	nextPhase.getList().get(index).setPosition(next);
					}
					point = checkPoint(nextPhase);

	System.out.println("index:"+list.get(index).name()+",next:"+next+",point:"+point)
;
					if(point>Max){
						Max=point;
						select[0]=index;
						select[1]=next;
					}
				}
			}
		}
		System.out.println("*****************");

	System.out.println("index:"+phase.getList().get(select[0]).name()+",next"+select[
1]);
		return select;
	}




}
```

```
package MiniChess;
import java.util.*;

public class Computer {
	Phase phase;
	boolean side;
	ArrayList<Piece> list;
	final int[]
map={0,1,2,3,4,10,11,12,13,14,20,21,22,23,24,30,31,32,33,34,40,41,42,43,44,50,51,52,53,
54};


	public Computer (Phase phase,boolean side){
		this.phase = phase;
		this.list = phase.getList();
		this.side = side;
	}

	private int[][] canMove(){
		int[][] canMove;
		canMove=new int[list.size()][30];
		//初期化
		for(int[] can:canMove){
			Arrays.fill(can, -1);
		}

		Piece piece=null;
		int ii;
		for(int index=0;index<list.size();index++){
			ii=0;
			piece = list.get(index);
			if(piece.getSide()==side){
				for(int next:map){
					if(piece.canMove(next, list)){

	if(piece.getName().equals("Pa")&&side&&next>=50){
							//Kn
							canMove[index][ii]=next*100+1;
							ii++;
							//Bi
							canMove[index][ii]=next*100+2;
							ii++;
							//Qe
							canMove[index][ii]=next*100+3;
							ii++;
							//Ro
							canMove[index][ii]=next*100+4;
							ii++;
						}else
if(piece.getName().equals("Pa")&&!side&&next<=4){
							//Kn
							canMove[index][ii]=next*100+1;
							ii++;
							//Bi
							canMove[index][ii]=next*100+2;
							ii++;
							//Qe
							canMove[index][ii]=next*100+3;
							ii++;
```

```
                                        //Ro
                                        canMove[index][ii]=next*100+4;
                                        ii++;
                                  }else{
                                        canMove[index][ii]=next;
                                        ii++;
                                  }
                            }
                      }
                }
          }
          return canMove;
    }

    /**
     * 次の盤面を引数で与え、その盤面の得点を計算し、返す
     * 次の盤面@param nextPhase
     * 点数@return point
     * Qe9 Ro5 Bi3 Kn3 Pa1
     */
    private int checkPoint(Phase nextPhase){
          int point=0;
          for(Piece piece:nextPhase.getList()){
                //自駒
                if(piece.getSide()==side){
                      if(piece.getPosition()!=-1){//盤内
                            if(piece.getName().equals("Qe")){
                                  point +=9;
                            }else if(piece.getName().equals("Ro")){
                                  point += 5;
                            }else if(piece.getName().equals("Bi")){
                                  point += 3;
                            }else if(piece.getName().equals("Kn")){
                                  point += 3;
                            }else if(piece.getName().equals("Pa")){
                                  point += 1;
                            }
                      }
                }
                //敵駒
                else{
                      if(piece.getPosition()!=-1){//盤内
                            if(piece.getName().equals("Qe")){
                                  point -=9;
                            }else if(piece.getName().equals("Ro")){
                                  point -= 5;
                            }else if(piece.getName().equals("Bi")){
                                  point -= 3;
                            }else if(piece.getName().equals("Kn")){
                                  point -= 3;
                            }else if(piece.getName().equals("Pa")){
                                  point -= 1;
                            }
                      }
                      //敵のキングが盤外(勝ち確定)
                      if(piece.getName().equals("Ki")&&piece.getPosition()==-1){
                            point +=1000;
                      }
                }
          }
          point*=10;
```

```
		point+=(new Random().nextInt(10));
		return point;

	}

	public int[] serect(){
		System.out.println();
		int[][] canMove= canMove();
		int Max=-10000;
		int point;
		int[] select={-1,-1};//{listの何番目の駒か,駒の移動場所}
		Phase nextPhase;
		int index=-1;
		for(int[] move:canMove){
			index++;
			for(int next:move){
				if(next!=-1){
					nextPhase=phase.clone();
					if(next>100){
						switch(next%100){
						case 1:

	nextPhase.getList().get(index).setName("Kn");

	nextPhase.getList().get(index).setPosition(next/100);
							break;
						case 2:

	nextPhase.getList().get(index).setName("Bi");

	nextPhase.getList().get(index).setPosition(next/100);
							break;
						case 3:

	nextPhase.getList().get(index).setName("Qe");

	nextPhase.getList().get(index).setPosition(next/100);
							break;
						case 4:

	nextPhase.getList().get(index).setName("Ro");

	nextPhase.getList().get(index).setPosition(next/100);
							break;
						}
					}else{

	nextPhase.getList().get(index).setPosition(next);
					}
					point = checkPoint(nextPhase);

	System.out.println("index:"+list.get(index).name()+",next:"+next+",point:"+point)
;
					if(point>Max){
						Max=point;
						select[0]=index;
						select[1]=next;
					}
				}
			}
		}
		System.out.println("*****************");
```

```
		System.out.println("index:"+phase.getList().get(select[0]).name()+",next"+select[
1]);
			return select;
		}




}


package MiniChess;
import java.util.*;

public class King extends Piece{
	public King(int position,boolean side) {
		super(position,side);
		this.side=side;
		pieceName="Ki";
	}

	/**
	 * 8‹ß–T,É,Pƒ}ƒX“®,¯,é(文字化けしています)
	 */
	@Override
	public boolean canMove(int next,ArrayList<Piece> list) {
		boolean result = false;
		int dist = next - position;
		if(next==position){
			return false;
		}
		if(side){
			if( dist == -1 ) { /*  ¶ */(文字化けしています)
				result = true;
			}
			else if( dist == 1 ) { /* %E */
				result = true;
			}
			else if( dist == 9 ) { /* ^¶‘O */
				result = true;
			}
			else if( dist == 10 ) { /* ‘O */
				result = true;
			}
			else if( dist == 11 ) { /* %E‘O */
				result = true;
			}
			else if( dist == -11 ) { /* ^¶Œã */
				result = true;
			}
			else if( dist == -10 ) { /* Œã */
				result = true;
			}
			else if( dist == -9 ) { /* %EŒã */
				result = true;
			}
		}else{
			if( dist == 1 ) { /* ^¶ */
```

```
                result = true;
            }
            else if( dist == -1 ) { /* %E */
                result = true;
            }
            else if( dist == -9 ) { /* ^¶‘O */
                result = true;
            }
            else if( dist == -10 ) { /* ‘O */
                result = true;
            }
            else if( dist == -11 ) { /* %E‘O */
                result = true;
            }
            else if( dist == 11 ) { /* ^¶Œã */
                result = true;
            }
            else if( dist == 10 ) { /* Œã */
                result = true;
            }
            else if( dist == 9 ) { /* %EŒã */
                result = true;
            }
        }

        //-Ú“I’n,ÉŽ©‹î,ª, ,ê,Î•s‰Â
        for(Piece p:list){
            if(p.getPosition()==next&&p.getSide()==side){
                result = false;
            }
        }
        return (result);
}

/**
 * ƒLƒƒƒXƒŠƒ“ƒO,Å,«,é,©,Ç,¤,©(文字化けしています)
 * @return
 */
public boolean isCasling(ArrayList<Piece> list){
        boolean check=false;
        Rook rook=null;
        for(Piece p:list){
            if(p.name().charAt(0)=='R'&&side==p.getSide()){
                rook=(Rook)  p;
            }
        }
        int i=1;
        if(!side){
            i=-1;
        }
        if(touch&&rook.istouch()&&rook.canMove(position+i, list)){
            check=true;
        }
        return check;
}

public void casling(List<Piece> list){
        Rook rook=null;
        for(Piece p:list){
            if(p.name().charAt(0)=='R'&&side==p.getSide()){
                rook=(Rook)  p;
```

```
                }
            }

            if(side){
                setPosition(03);
                rook.setPosition(02);
            }else{
                setPosition(51);
                rook.setPosition(52);
            }

        }


}



package MiniChess;
import java.util.*;

public class Knight extends Piece{
    public Knight(int position,boolean side) {
        super(position,side);
        this.side=side;
        pieceName="Kn";
    }

    @Override
    /**
     * 8•ûŒü,ÉŒj”n,Ì,æ,¤,É“®,
     */
    public boolean canMove(int next,ArrayList<Piece> list) {
        boolean result = false;
        int dist = next - position;

        if(dist==19){
            result = true;
        }else if(dist ==21){
            result = true;
        }else if(dist ==8){
            result = true;
        }else if(dist ==-12){
            result = true;
        }else if(dist ==12){
            result = true;
        }else if(dist ==-8){
            result = true;
        }else if(dist ==-21){
            result = true;
        }else if(dist ==-19){
            result = true;
        }

        for(Piece p:list){
            if(p.getSide()==side&&p.getPosition()==next){
                result=false;
            }
        }
```

```
			return (result);
		}
}



package MiniChess;
import java.util.*;

public class Pawn extends Piece{
	Scanner kbs=new Scanner(System.in);
	final int firstPosition;
	int
mode;//‘Š è,Ì Å I’i,É“ ’…,µ,½,Æ,« AfiƒCƒg(1) AƒrƒVƒ‡ƒbƒv(2) AƒNƒC [ƒ“(3) Aƒ‹ [ƒN(4),Æ,È
,é
	public Pawn(int position,boolean side) {
		super(position,side);
		this.side=side;
		pieceName="Pa";
		this.firstPosition = position;
		mode=0;
	}

	@Override
	/**
	 * ‘O•û,Pƒ}ƒX,É i,ß,é(‘O•û,Pƒ}ƒX,É‹î,ª, ,é,Æ“®,¯,È,¢)
	 *  `“Á ê `
	 *  ‰ŠúˆÊ’u,Ìƒ| [ƒ“,Í‘O•û,Qƒ}ƒX,ÌˆÊ’u,É i,Þ
	 *  Î,ß‘O•û,Pƒ}ƒX,É“G‹î,ª, ,é ê ‡,Ì,Ý Î,ß‘O•û,Pƒ}ƒX,É“®,« A“G‹î,ð,Æ,ê,é
	 */
	public boolean canMove(int next,ArrayList<Piece> list) {
		boolean result = false;
		int dist = next - position;
		if(next==position){
			return false;
		}
		if(mode==0){
			if(side){
				if(dist==10){
					result = true;
					for(Piece p:list){
						if(next==p.getPosition()){
							result=false;
						}
					}
				}else if(position==firstPosition&&dist==20){
					result=true;
					for(Piece p:list){
						if(next==p.getPosition()){
							result=false;
						}else if(position+10==p.getPosition()){
							result=false;
						}
					}
				}
			}else{
				if(dist==-10){
					result = true;
					for(Piece p:list){
						if(next==p.getPosition()){
							result=false;
```

```
                                        }
                                }
                        }else if(position==firstPosition&&dist==-20){
                                result=true;
                                for(Piece p:list){
                                        if(next==p.getPosition()){
                                                result=false;
                                        }else if(position-10==p.getPosition()){
                                                result=false;
                                        }
                                }
                        }
                }

                // Î,ß‘O•û,É“G‹î,ª, ,ê,Î‰Â
                for(Piece p:list){
                        if(side){

        if((next==position+9||next==position+11)&&next==p.getPosition()&&side!=p.getSide(
)){
                                        result=true;
                                }
                        }else{
                                if((next==position-9||next==position-
11)&&next==p.getPosition()&&side!=p.getSide()){
                                        result=true;
                                }
                        }
                }
        }
        else if(mode==1){
                result = (new Knight(position,side)).canMove(next, list);
        }
        else if(mode==2){
                result = (new Bishop(position,side)).canMove(next, list);
        }
        else if(mode==3){
                result = (new Qeen(position,side)).canMove(next, list);
        }
        else if(mode==4){
                result = (new Rook(position,side)).canMove(next, list);
        }


        return (result);
    }

    public void setPosition(int position,ArrayList<Piece>list){
        if(side){
                if(position>=50){
                        // ¸Ši æ,Ì“ü—Í

    System.out.print("Pawn ¸Ši:(1)Knight,(2)Bishop,(3)Qeen,(4)Rook:");
                        int k;
                        while((k=kbs.nextInt())>4||k<=0){

    System.out.print("Pawn ¸Ši:(1)Knight,(2)Bishop,(3)Qeen,(4)Rook:");
                        }
                        mode=k;
                        switch(k){
                        case 1://Knight
                                pieceName="Kn";
```

```
                        break;
                    case 2://Bishop
                        pieceName="Bi";
                        break;
                    case 3://Qeen
                        pieceName="Qe";
                        break;
                    case 4://Rook
                        pieceName="Ro";
                        break;
                    }
                }
            }else{
                if(position<=4){
                    // ¸Ši æ,Ì‘ü—Í

        System.out.print("Promotion:(1)Knight,(2)Bishop,(3)Qeen,(4)Rook:");
                    int k;
                    while((k=kbs.nextInt())>4||k<=0){

        System.out.print("Promotion:(1)Knight,(2)Bishop,(3)Qeen,(4)Rook:");
                    }
                    mode=k;
                    switch(k){
                    case 1://Knight
                        pieceName="Kn";
                        break;
                    case 2://Bishop
                        pieceName="Bi";
                        break;
                    case 3://Qeen
                        pieceName="Qe";
                        break;
                    case 4://Rook
                        pieceName="Ro";
                        break;
                    }
                }
            }
            this.position=position;

            for(Piece p:list){
                if(p.getPosition()==position&&p.getSide()!=this.side){
                    p.setPosition(-1);
                }
            }
        }
}


package MiniChess;
import java.util.*;

public class Phase {
        private King king1;
        private King king2;
        private Qeen qeen1;
        private Qeen qeen2;
        private Bishop bishop1;
        private Bishop bishop2;
        private Knight knight1;
```

```
	private Knight knight2;
	private Rook rook1;
	private Rook rook2;
	private Pawn pawn10;
	private Pawn pawn11;
	private Pawn pawn12;
	private Pawn pawn13;
	private Pawn pawn14;
	private Pawn pawn20;
	private Pawn pawn21;
	private Pawn pawn22;
	private Pawn pawn23;
	private Pawn pawn24;

	private ArrayList<Piece> list;
	private Scanner kbs;
	private boolean side;
	Computer com;


	private int pieceNum=0;

	public Phase(King king1,Qeen qeen1,Bishop bishop1,Knight knight1,Rook rook1,Pawn pawn10,Pawn pawn11,Pawn pawn12,Pawn pawn13,Pawn pawn14,
			King king2,Qeen qeen2,Bishop bishop2,Knight knight2,Rook rook2,Pawn pawn20,Pawn pawn21,Pawn pawn22,Pawn pawn23,Pawn pawn24,boolean turn){
		this.king1=king1;
		this.king2=king2;
		this.qeen1=qeen1;
		this.qeen2=qeen2;
		this.bishop1=bishop1;
		this.bishop2=bishop2;
		this.knight1=knight1;
		this.knight2=knight2;
		this.rook1=rook1;
		this.rook2=rook2;
		this.pawn10=pawn10;
		this.pawn11=pawn11;
		this.pawn12=pawn12;
		this.pawn13=pawn13;
		this.pawn14=pawn14;
		this.pawn20=pawn20;
		this.pawn21=pawn21;
		this.pawn22=pawn22;
		this.pawn23=pawn23;
		this.pawn24=pawn24;

		list= new ArrayList<Piece>();
		list.add(king1);
		list.add(king2);
		list.add(qeen1);
		list.add(qeen2);
		list.add(knight1);
		list.add(knight2);
		list.add(rook1);
		list.add(rook2);
		list.add(bishop1);
		list.add(bishop2);
		list.add(pawn10);
		list.add(pawn11);
		list.add(pawn12);
		list.add(pawn13);
```

```
        list.add(pawn14);
        list.add(pawn20);
        list.add(pawn21);
        list.add(pawn22);
        list.add(pawn23);
        list.add(pawn24);
        side = true;
}

public King getKing1() {
        return king1;
}

public King getKing2() {
        return king2;
}

public Qeen getQeen1() {
        return qeen1;
}

public Qeen getQeen2() {
        return qeen2;
}

public Bishop getBishop1() {
        return bishop1;
}

public Bishop getBishop2() {
        return bishop2;
}

public Knight getKnight1() {
        return knight1;
}

public Knight getKnight2() {
        return knight2;
}

public Rook getRook1() {
        return rook1;
}

public Rook getRook2() {
        return rook2;
}

public Pawn getPawn10() {
        return pawn10;
}

public Pawn getPawn11() {
        return pawn11;
}

public Pawn getPawn12() {
        return pawn12;
}

public Pawn getPawn13() {
```

```
		return pawn13;
	}

	public Pawn getPawn14() {
		return pawn14;
	}

	public Pawn getPawn20() {
		return pawn20;
	}

	public Pawn getPawn21() {
		return pawn21;
	}

	public Pawn getPawn22() {
		return pawn22;
	}

	public Pawn getPawn23() {
		return pawn23;
	}

	public Pawn getPawn24() {
		return pawn24;
	}

	public int getPieceNum() {
		return pieceNum;
	}

	public ArrayList<Piece> getList(){
		return list;
	}


	/**
	 * ボードの表示
	 */
	public void showPhase(){
		String[][] board = new String[6][5];
		for(int y=0;y<board.length;y++){
			for(int x=0;x<board[y].length;x++){
				board[y][x]="	"+y+x+"	";
			}
		}
		for(Piece p:list){
			if(p.getPosition()!=-1){
				if(p.side){

	board[p.getPosition()/10][p.getPosition()%10]="◇"+p.name()+"◇";
				}else{

	board[p.getPosition()/10][p.getPosition()%10]="◆"+p.name()+"◆";
				}
			}
		}
		for(int y=board.length-1;y>=0;y--){
			for(int x=0;x<board[y].length;x++){
				System.out.print(board[y][x]);
			}
			System.out.println();
```

```
	}

}

/**
 * 駒の移動
 */
public void move(){
	boolean isCasle = false;
	if(side){
		if(side){
			System.out.println("◇先手の手番");
			isCasle = king1.isCasling(list);
		}else{
			System.out.println("◆後手の手番");
			isCasle = king2.isCasling(list);
		}

		if(isCasle){
			System.out.print("キャスリングしますか？y/n : ");
			if(kbs.next().charAt(0)=='n'){
				isCasle=false;
			}
		}

		if(isCasle){
			if(side){
				king1.casling(list);
			}else{
				king2.casling(list);
			}
		}else{
			//移動する駒の選択
			Piece targetPiece=null;
			kbs = new Scanner(System.in);
			int targetNum;
			boolean check;
			do{
				check=true;
				do{
					System.out.print("移動する駒:");
					targetNum=kbs.nextInt();
					for(Piece p:list){

if(p.getPosition()==targetNum&&side==p.getSide()){
							targetPiece=p;
						}
					}
				}while(targetPiece==null);
				//選択した駒が移動できる駒か
				for(int y=0;y<6;y++){
					for(int x=0;x<5;x++){
						if(targetPiece.canMove(y*10+x, list)){
							check=false;
						}
					}
				}
			}while(check);

			//移動するマスの選択
			int next;
```

```
                        do{
                                System.out.print("マスの選択:"+targetPiece.name()+"→");
                                next=kbs.nextInt();
                        }while(!targetPiece.canMove(next, list));

                        //駒の移動
                        //駒がポーンのとき

        if(targetPiece.name().equals("Pa"+targetPiece.getPosition())){
                                ((Pawn)targetPiece).setPosition(next,list);
                        }
                        else{
                                for(Piece p:list){
                                        if(p.getPosition()==next){
                                                p.setPosition(-1);
                                        }
                                }
                                targetPiece.setPosition(next);
                        }
                }
        }else{
                com = new Computer(this,side);
                int[] next = com.serect();
                for(Piece p:list){
                        if(p.getPosition()==next[1]){
                                p.setPosition(-1);
                        }
                }
                list.get(next[0]).setPosition(next[1]);
        }

        //ターン交代
        if(side){
                side=false;
        }else{
                side=true;
        }
}

        /**
         * 勝敗判断
         * check=0 未確定
         * check=1 先手勝ち
         * check=-1 後手勝ち
         * @return check
         */
        public int  checkWinLose(){
                int check=0;
                if(king1.getPosition()==-1){
                        check=-1;
                }else if(king2.getPosition()==-1){
                        check=1;
                }
                return check;
        }

        public Phase clone(){
                Phase phase = new Phase(new King(king1.getPosition(),true),new
Qeen(qeen1.getPosition(),true),new Bishop(bishop1.getPosition(),true),new
Knight(knight1.getPosition(),true),new Rook(rook1.getPosition(),true),new
Pawn(pawn10.getPosition(),true),new Pawn(pawn11.getPosition(),true),new
```

```
Pawn(pawn12.getPosition(),true),new Pawn(pawn13.getPosition(),true),new
Pawn(pawn14.getPosition(),true),
                        new King(king2.getPosition(),false),new
Qeen(qeen2.getPosition(),false),new Bishop(bishop2.getPosition(),false),new
Knight(knight2.getPosition(),false),new Rook(rook2.getPosition(),false),new
Pawn(pawn20.getPosition(),false),new Pawn(pawn21.getPosition(),false),new
Pawn(pawn22.getPosition(),false),new Pawn(pawn23.getPosition(),false),new
Pawn(pawn24.getPosition(),false),side);

                return phase;
        }


}
```

```
package MiniChess;
import java.util.*;

public class Qeen extends Piece{

        public Qeen(int position,boolean side) {
                super(position,side);
                this.side=side;
                pieceName="Qe";
        }

        /**
         * 縦、横、クロスに任意のマスだけ動ける
         */
        @Override
        public boolean canMove(int next,ArrayList<Piece> list) {

                boolean result = false;
                int dist = next - position;

                if(next==position){
                        return false;
                }

                if( dist == -1||dist == -2||dist == -3||dist == -4 ) { /* 左 */
                        result = true;
                        for(int i=position-1;i>next;i--){
                                for(Piece p:list){
                                        if(i==p.getPosition()){
                                                result=false;
                                        }
                                }
                        }
                }
                else if( dist == 1||dist == 2||dist == 3||dist == 4 ) { /* 右 */
                        result = true;
                        for(int i=position+1;i<next;i++){
                                for(Piece p:list){
                                        if(i==p.getPosition()){
                                                result=false;
                                        }
                                }
                        }
```

```
}

else if( (dist)%10 == 0 ) { /* 縦 */
	result = true;
	//上
	if(position<next){
		for(int i=position+10;i<next;i+=10){
			for(Piece p:list){
				if(i==p.getPosition()){
					result=false;
				}
			}
		}
		//下
	}else{
		for(int i=position-10;i>next;i-=10){
			for(Piece p:list){
				if(i==p.getPosition()){
					result=false;
				}
			}
		}
	}
}
else if( (dist%9) == 0 ) { /*  左上ー右下  */
	result = true;
	if((position==0&&next==54)||(position==54&&next==0)){
		result=false;
	}
	//左上方向
	if(position<next){
		for(int i=position+9;i<next;i+=9){
			for(Piece p:list){
				if(i==p.getPosition()){
					result=false;
				}
			}
		}

	}
	//右下方向
	else{
		for(int i=position-9;i>next;i-=9){
			for(Piece p:list){
				if(i==p.getPosition()){
					result=false;
					break;
				}
				if(!result){
					break;
				}
			}
		}
	}
}

else if( (dist%11) == 0 ) { /* 右上ー左下 */
	result = true;
	//右上方向
	if(position<next){
		for(int i=position+11;i<next;i+=11){
```

```
                                for(Piece p:list){
                                        if(i==p.getPosition()){
                                                result=false;
                                        }
                                }
                        }
                }else
                //左下
                {
                        for(int i=position-11;i>next;i-=11){
                                for(Piece p:list){
                                        if(i==p.getPosition()){
                                                result=false;
                                        }
                                }
                        }
                }
        }

        //移動先に自分の駒があれば動けない
        for(Piece p:list){
                if(side==p.getSide()&&next==p.getPosition()){
                        result=false;
                }
        }
        if(next==-1){
                result=false;
        }


        return (result);
    }


}


package MiniChess;
import java.util.*;

public class Rook extends Piece{
    public Rook(int position,boolean side) {
        super(position,side);
        this.side=side;
        pieceName="Ro";
    }

    @Override
    /**
     * ‰¡ A c‚É”C‚Ó‚Ìƒ}ƒX‚¾‚¯“®‚¯‚é
     */
    public boolean canMove(int next,ArrayList<Piece> list) {
        boolean result = false;
        int dist = next - position;
        if(next==position){
                return false;
        }
        if( dist == -1||dist == -2||dist == -3||dist == -4 ) { /*  ¶ */
                result = true;
                for(int i=position-1;i>next;i--){
                        for(Piece p:list){
```

```
                                if(i==p.getPosition()){
                                        result=false;
                                }
                        }
                }
                else if( dist == 1||dist == 2||dist == 3||dist == 4 ) { /* %E */
                        result = true;
                        for(int i=position+1;i<next;i++){
                                for(Piece p:list){
                                        if(i==p.getPosition()){
                                                result=false;
                                        }
                                }
                        }
                }
                else if( (dist)%10 == 0 ) { /*  c */
                        result = true;
                        // ã
                        if(position<next){
                                for(int i=position+10;i<next;i+=10){
                                        for(Piece p:list){
                                                if(i==p.getPosition()){
                                                        result=false;
                                                }
                                        }
                                }
                        //%º
                        }else{
                                for(int i=position-10;i>next;i-=10){
                                        for(Piece p:list){
                                                if(i==p.getPosition()){
                                                        result=false;
                                                }
                                        }
                                }
                        }
                }

                for(Piece p:list){
                        if(side==p.getSide()&&next==p.getPosition()){
                                result=false;
                        }
                }

                return (result);
        }
}


package MiniChess;
import java.util.*;

public abstract class Piece {
        protected int position;
        protected String pieceName;
        protected boolean touch;
        /*
        50 51 52 53 54
        40 41 42 43 44
    30 31 32 33 34
```

```
    20 21 22 23 24
    10 11 12 13 14
    00 01 02 03 04
 */
 //‰º,ª æ èside = true


	protected boolean side;

	public Piece(int position ,boolean side){
		this.position = position;
		touch=true;
	}

	public abstract boolean canMove(int next,ArrayList<Piece> list);

	public int getPosition(){
		return position;
	}

	public void setPosition(int position){
		touch=false;
		this.position=position;
	}

	public void setName(String name){
		this.pieceName = name;
	}

	public boolean getSide(){
		return side;
	}

	public String name(){
		if(position<10){
			return pieceName+"0"+position;
		}else
		return pieceName+position;
	}

	public String getName(){
		return pieceName;
	}

	public boolean istouch(){
		return touch;
	}
}
```