卒業研究報告書

題目

四路の碁 アプリ制作

指導教員

石水 隆　講師

報告者

09-1-037-0169

高倉 秀斗

近畿大学理工学部情報学科

平成 25 年 1 月 31 日提出

概要

本研究では「よんろのご[5]」と呼ばれる2011年7月に発売された張栩九段(棋聖・王座)が考案した4*4の盤面と囲碁のルールを用いたアプリケーションを制作した。本研究テーマである4*4の囲碁は完全解析されており、両者最善手を打てば持碁(引き分け) となることが示されている[6]。四路の碁で可能な局面の総数は 3,047,783 通りであるので、 その全ての局面に対する最善手をデータベースに持てば真に最強のCPU を作成できる。また現在強い囲碁AIを制作するにはモンテカルロ法が有望だと言われている。しかし、本研究に4*4の囲碁を研究テーマと決定し、制作を始める際に完全解析されていることや、モンテカルロ法のことを知り、ならば相手の手をその都度計算し先読みするCPUを制作すればどれほどに強くなるのか、又は処理時間がどれほどかかるのかと考え本研究テーマを、CPUが先読みするAIの四路の碁のアプリ制作、することに決定した。

本研究では相手の手を先読みをし、処理時間ができるだけかからないような強いCPUを制作することを目標にする。

目次

1　序論

1.1.　本研究の背景

将棋や囲碁, チェス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。零和とは、ゲーム上プレイしているプレイヤーの利得が常にゼロ、または個々のプレイヤーの指す手の組み合わせに対する利得が常に一定の数値となることである。利得とは、プレイヤーがゲーム終了時、またはターン終了時に獲得する状況に対する評価である。有限とは、そのゲームにおける各プレイヤーの可能な手の組み合わせの総数が有限であることである。確定とは、プレイヤーの着手以外にゲームに影響を与える偶然が全く入り込まないという意味である。完全情報ゲームとは、各プレイヤーが自分の手番において、これまで各プレイヤーが行った選択、または意思決定についてのすべての情報を完全に知ることができるゲームである。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

二人零和有限確定完全情報ゲームは双方最善手を指した場合、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[5]。チェッカーは双方最善手を指すと引き分けとなる[9]。その他にも「シンペイ」と呼ばれるバンダイが発売したゲームは勝ちに要する最長手数が 49 手で、後攻が勝つとわかっている[6]など多数のゲームが完全解析されている。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]。将棋では、盤サイズ 3x4 に減らし、駒の種類を 4 つに減らしたどうぶつしょうぎ[11]が完全解析されており、双方最善手を指すと 78 手で後手が勝つことが判明している[12]。

囲碁の場合は、サイズ 4x4 では双方最善手を打つと持碁(引き分け)になり[2]、5x5 の囲碁は黒の 25 目勝ちとなる[13]。しかしサイズ 6x6 以上については、現在のところまだ解析されていない。

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、定石データベース、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[5]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。囲碁ではモンテカルロ法を用いた AI が 2006 年から成果をだしてきている。モンテカルロ法とは、終局までをシミュレーションし勝率の最も高い着手を選択する方法である。より差を広げて勝つことよりもいかに無難に勝てるかという考え方である。

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。

チェスでは、1997 年 5 月にチェスプログラム Deep Blue[14]が世界チャンピオン Garry Kimovich Kasparov と

対戦を行い 2 勝 1 敗 3 引き分けで勝った[15]。将棋では、将棋プログラムボンクラーズ[16]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[17]。囲碁では 2012 年 3 月に、第５回ＵＥＣ杯コンピュータ囲碁大会のエキシビションで優勝した Zen がハンデ付きながらも武宮九段との対戦に勝った[45]。また同大会で準優勝している ERICA も小林千寿五段との対戦に負けはした[46]が有名である。

1.4. 本研究の目的

本研究テーマである 4*4 の囲碁は上で述べた通り完全解析が既に行われており、持碁となることが判明している。よって本研究は盤面の情報のデータベースを用いず、またモンテカルロ法も用いずに、その都度、盤面の状態から相手の次の手を計算し、先読みする CPU を制作することにした。囲碁などが強い名人と聞くと、何手先まで読める、先ほどの手はこのための布石であった、等の話を聞いたことがある。CPU が相手の手を先読みする AI は何処に問題があるのか、強くはならないのか、と考え本研究では先読みする CPU を制作することにした。

本研究は相手の手を先読みし強く、プレイするにあたりストレスがかからないようになるべく処理時間を短い CPU を制作することを目標とする。

1.5. 本報告書の構成

本報告書の第 2 節以降、本研究で制作したプログラム内容について説明する。本研究では goApplet クラス、List クラス、CPU クラスの 3 つのクラスを制作した。第 2 節ではまず囲碁について説明し、その後それぞれのクラスについて説明、第 3 章から本研究の結果・結論を述べる。

2. 囲碁について

本章では囲碁について説明する。

2.1. 囲碁のルール

囲碁の基本的なルールは以下の通りである。

- 二人で黒と白の石を交互に碁盤の交点に打っていく。
- 上下左右相手の石に囲まれている（呼吸点のない）石は碁盤から取り除かれる。
- 呼吸点のなくなるところへは打てない。
- 一手前と全く同じ状況になるところへは打てない。(劫)
- 勝敗は地の多少で決める。

2.2. 囲碁の戦術・定石

囲碁には多数の定石が存在する。ここでは星の定石と呼ばれる定石の一部を紹介する。

小ゲイマガカリ-ツケノビ[7]

ツケノビは置碁定石の代表格と言われている。形を決めるのが目的だが、右辺に勢力を向ける利点もある。

- ノビ

置碁定石の代表格であるノビ。図 1 では白 8 で A の 1 間飛びも立派な一手である。図 2 では黒 7 と堅くカケツいできたら、白も白 8 と堅く応じる。

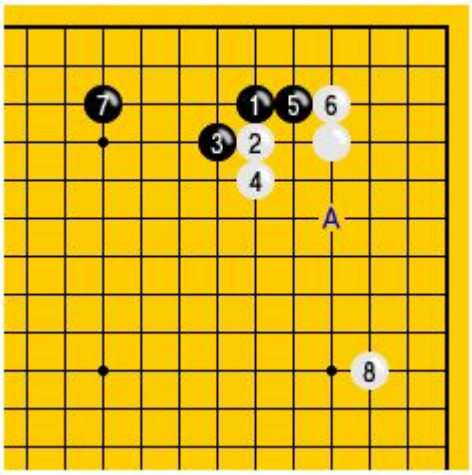


図 1　ノビ1

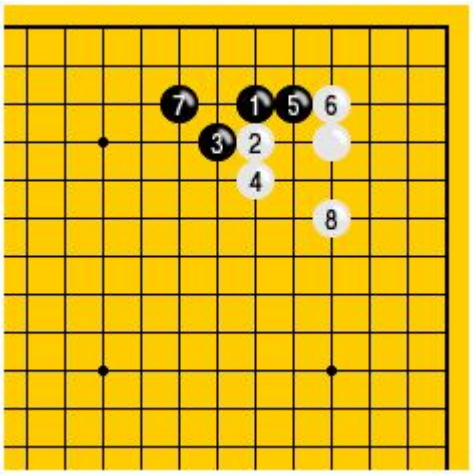
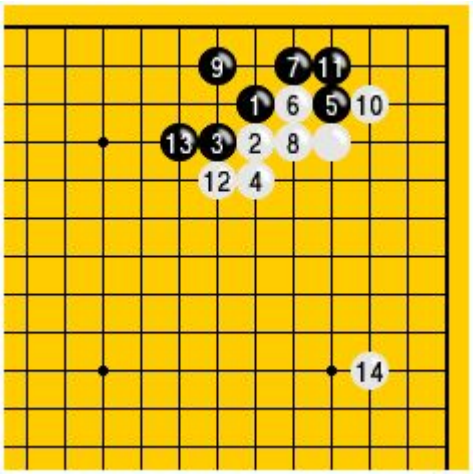


図 2　ノビ2

- トビツケ



黒 5 のトビツケは少しでも隅に食い込もうというもの。
白 6 の割り込みと 10、12 と利かして白 14 と大きく開くのが良い。

図 3　トビツケ

- コスミ

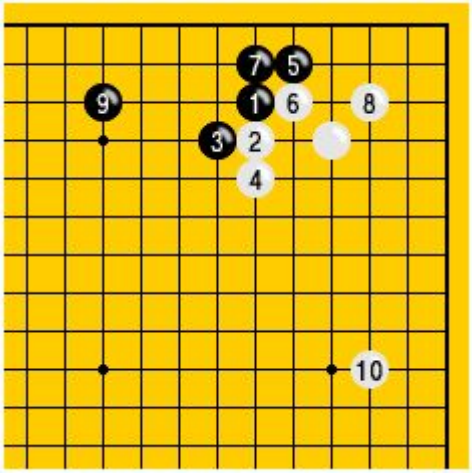


図 4　コスミ 1

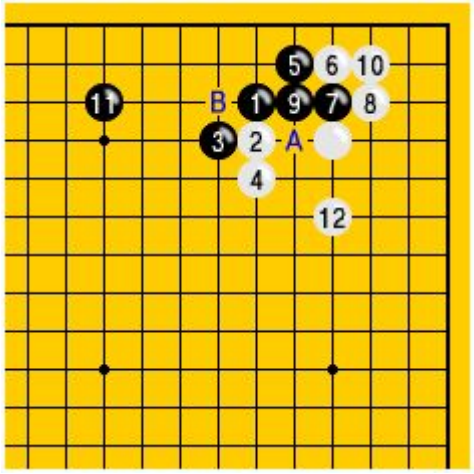


図 5　コスミ 2

図 4 では黒 5 のコスミには白 6 アテコミがわかりやすい。黒 7 と堅くつけば白 8 とコスんでいるのが本手。図 5 では黒 5 のコスミに白 6 とツケるのは隅を大事にした打ち方。白 11 で A の出切りには黒 B。

ノビ、トビツケ、コスミと 3 つ紹介したが、これはツケノビと呼ばれる小ゲイマガカリの一つで小ゲイマガカリには他にもツケオサエ、一間バサミ、二間高バサミ、大ゲイマウケなど幾つもあり、さらには定石にも小目の定石、高めの定石、目はずしの定石など多くの定石が存在する。囲碁には「定石を覚えて二目弱くなり」という定石を覚えても、その都度状況を考えて打たなければ逆に弱くなるといった意味の言葉も存在する。

3. 研究内容

本章では本研究で作成した四路の碁のアプリケーションについて説明する。

3.1. 四路の碁アプリケーションで用いた手法

1.3 節で述べたように、次に打つべき手をどのように選択するかは様々な手法がある。本節では四路の碁アプリケーションで用いた手法について説明する。制作したプログラムでは 1,3 節で述べた局面の評価値計算、一定手数の先読みを行う囲碁 AI を制作した。モンテカルロ法は現在強い AI を制作するのには有効とされているが、本研究では一定手数の先読みだけでどこまで強くなるのか、または処理時間がどれほどまでにかかるのかを調べるため使用しなかった。

本アプリケーションの囲碁 AI では、現在着手できる座標に全て置いたパターンの後、得られる評価値が最も高い場所に着手するようになっている。得られた評価値で同じ評価値が複数ある場合、それ以降の先読みをそのパターン分していき、その後の評価値が最も高い座標へ着手する。一定手数先読みした後も同じ評価値が複数ある場合は、その評価値が得られた座標の中からランダムで着手する。

3.1.1. 着手の判定

四路の碁アプリケーションでは、着手可能な手の判定に、着手しようとしている座標に既に石は置かれていないか。その座標へ着手後、着手した石を含む連が呼吸点のない状況にならないかどうか、1 手前と全く同じ局面に戻らないかどうかを判定し、決定している。着手可能な手を打った場合、呼吸点がなくなっていない連が存在するかどうかを判定、存在するならその連に含まれる石を全て碁盤から取り除く。

3.1.2. 局面の評価関数

着手、呼吸点が無くなっていない連が存在するかを判定し石を取り除いた後、地を数え、黒と白どちらの地が多いかで評価値は決定される。

3.1.3. 局面の先読み

とある局面から先読みをする際には、現在どの座標に着手することができるのかをまず判定し、着手可能な座標全てに着手した際の評価値を求め、最も高い評価値の座標に着手する。そこからもう 1 手先読みし、得られた高い評価値の座標を決定する。この処理を予め定めた一定回数分ループし先読みを行う。

3.1.4. 着手の選択

3.1.1.の着手の判定、3.1.2.の局面の評価、3.1.3.の局面の先読みを行った後、もし同じ評価値の座標が複数

存在する場合、その座標の中からランダムで着手する手を決定する。

3.1.5. 勝敗の判定

囲碁の勝敗の判定は投了か、地を数え黒と白双方の地の多少で勝敗を決定する。四路の碁アプリケーションでは CPU に投了を決定する機能を搭載させていないので、勝敗がつくまで打つことになる。黒と白のいずれでもない地がなくなった際、ゲームは終了し地の多少で勝敗を決定する。

3.1.6. 劫の判定

囲碁には劫と呼ばれるルールが存在する。これは 1 手前と全く同じになる手は打てないというものである。劫を判定するために、四路の碁アプリケーションでは劫用の配列を一つ用意し、常に 1 前の局面の状態を記録している。着手可能な座標を決定する際にこの劫も考慮し、ルールを破ってしまう着手になる座標は、そもそも着手不可能な座標として判定され着手することは出来ない。

3.2. 四路の碁アプリケーションプログラム

四路の碁のアプリケーションを制作するために本研究では Java 言語を用いた。付録 1 に本研究で作成した四路の碁アプリケーションプログラムのソースを示す。本研究で制作した以下の 3 つのクラスから成る。

- goApplet クラス

  goApplet クラスは、画面に碁盤を表示し、また、ユーザからのマウス入力を読み取るクラスである。本研究はあくまでアプリケーションを制作することなのでユーザーインターフェースの制作は欠かすことが出来ない。よって Applet クラスを継承した goApplet クラスを制作し、ボタンによって碁の石を各位置に置いていけるようにした。

- List クラス

  List クラスは碁盤上の石の配置を管理するクラスである。

  囲碁の盤面の情報を持つことになる配列のほとんどはここで生成している。他にも、盤面の評価・判定もこのクラスで行うことになっている。

- CPU クラス

  CPU クラスでは CPU の動作に関するプログラムが記述されている。現段階では 5 手先まで読むロジックになっているが、グローバル変数である Loop という名の数値を変更すると先読みする数が変更される仕様となっている。

以下では、各クラスについての詳細を記述する。

3.2.1. goApplet クラス

goApplet クラスではユーザーインターフェースを作成する。プレイヤーはボタンによって碁盤となる配列に石を着手していく。プレイヤーは最初、先攻か後攻かを選ぶ。選んだボタンによって先攻ならばプレイヤーは黒色の石、後攻なら CPU が黒色の石を置いた後に白色の石を置いていくことになる。先攻か後攻のいずれかを選んだプレイヤーは 4*4 の盤面のいずれかに対応したボタンを押すことによって、自分の石を置いていくことになる。16 個のいずれかのボタンを押すと次に説明する List クラスのメソッドによってその場所に置けるかどうかの判定がなされ、置けるならそのボタンに対応した場所に石を置くことが可能で次の処理に入ってい

く。置けないならば、表示された盤面の下側に「その場所には置けません。」と表示される仕様になっている。石を置いた後は List クラスによって盤面の評価・判定が行われ、相手の石が取れるようなら取る。評価・判定が行われた後、CPU クラスによる先読みがなされ CPU によって自動的に石が置かれる様になっている。この一連の処理が行われた後アプレットが再描画され、プレイヤーが再びボタンを押していくことになる。図 6 は本アプレットを実行した際の画像である。

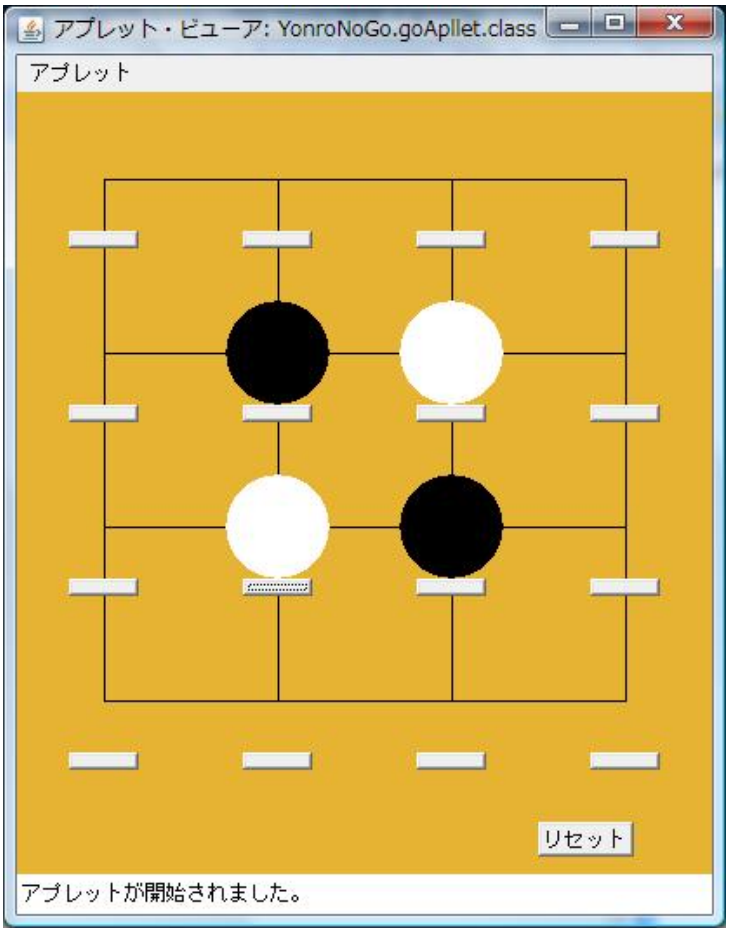


図 6 囲碁アプリ実行画像

3.2.2. List クラス

この List クラスでは、囲碁の盤面のほとんどの情報を保持、評価・判定するクラスである。
このクラスでは以下の様な配列を生成している。

- List[6][6]

  盤面の何処に石が置いてあるかどうかの情報を保持する配列。先攻である黒色の石は 1、後攻である白色の石は 2 という要素が、何も置かれていないところは 0、盤面の縁に当たる 0 行目、5 行目、0 列目、5 列目は始めから 3 という要素が格納されている。

- RenList[6][6]

  同じ色の石が上下左右に接している塊を連と呼ぶ。配列 List[][]から判定された連が、連ごとに番号を付けられ、その番号がこの配列に格納される。相手の石を取る、または相手の石に挟まれ取られるという判定はこの連ごとに決定される。

- JudgeList[6][6]

連ごとに石を取られるかどうか判定される際、連の上下左右全て相手の石で囲まれているかどうかを見ることになるが、このプログラムではとある石の隣が同じ色の石だった場合更にその隣がどのような状況になっているか再帰的プログラムによって判断されるようになっている。その際、一度見た石が判定済みかどうかがわからなければまたその隣を判定し無限ループに陥ってしまう。それを回避するために判定済みの石がわかるようにした配列がこの JudgeList[][]である。

- PutJudgeList[6][6]

  この PutJudgeList[][]では、後述する PutJudge メソッドと名付けたメソッドで判定された、そのターンその色の石が何処になら置けるのかを示す数値が格納される配列である。例えば先攻なら、後攻が石を置き終わり盤面の評価・判定も終了すると先攻が石を置ける状態になる前に PutJudge メソッドが実行され、この PutJudgeList[][]にその時の状態に応じた数値が格納される。また後述する BforeList[][]の数値もこの配列を決定するにあたって重要である。この配列は前述したボタンを押してもその場所に置けないようなら「そこには置けません。」というメッセージが表示される仕様にも使われている。

- BforeList[6][6]

  囲碁には同じ箇所には 2 度続けて置いてはならないというルールが存在する。この配列はそのルールを破らないために 1 手前に置いた所に、先攻なら 1 を後攻なら 2 を入れるようになっており 2 度続けては置けないようになっている。

上記の全てが 6*6 の配列になっている理由としては、盤面の縁の部分、List[][]の 0 行目、5 行目、0 列目、5 列目が 3 としているため、それに合わせ石の状態を 1 行目から 4 行目又は 1 列目から 4 列目に格納し、ズレが生じないようにしているためである。

List クラスには以下のメソッドが用意されている。

- void renListReset(){}

  配列 RenList[][]の要素を全て 0 にリセットするメソッド。

- void judgeListReset() {}

  配列 JudgeList[][]の要素を全て 0 にリセットするメソッド。

- void putJudgeListReset(){}

  配列 PutJudgeList[][]の要素を全て 0 にリセットするメソッド。

- void listReset(){}

  上記の Reset メソッドを 1 度に行うメソッド。

- void RenSet(int, int, int){}

  連番号を設定するために使われるメソッド。一つの連がどこまでつづいているのかを判定するために、引数で渡されてきた連番号と判定する石の座標を元にその上下左右がどのような状態になっているかを見る。もし指定された座標の隣も同じ色の石ならば自身を呼び出し、引数に今見ている連の番号と隣の同じ色の石の座標とを渡す再帰的プログラムになっている。

- void RenSet(){}

連番号を設定するために使われるメソッド。配列 List[][]を 1 行 1 列目から順に見ていき、未だ連番号が 0 の状態の石があれば、その石にあてがわれる連番号と石の座標を上記の void RenSet(int, int, int){}に引数として渡し実行する。

- int SteelJudge(int){}

  このメソッドでは、引数で渡された連番号に対応した連が相手の石に上下左右囲まれているかを判定する。もし上下左右全て相手の石で囲まれているなら、その取られる石の色、黒色なら 1 を白色なら 2 を戻り値で返すようになっている。

- int SteelJudge(int, int, int){}

  このメソッドの引数にはたった今石が置かれた盤面の座標とその色の数値が渡される。たった今置かれた石が取られるか否かの判定は後回しにされ、まずその石が含まれる連以外が取られるかを判定するようになっている。もし上下左右相手の石で囲まれていたとしても、そこに石を置くことによって相手の石が取れるなら置ける。取れないのなら自分の石が取られるだけなので置けないといった判定にもこのメソッドは関係してくる。よって今置かれた石が含まれる連以外を見つけ出し、見つけた連番号を引数に上記の int SteelJudge(int){}メソッドの処理が行われる。全ての連の処理が行われて後、後回しにされていた連の処理が実行される。Int SteelJudge(int){}によって取られる連が存在するとわかった場合、下に記述する int Steel(int, int)によってその連は何も置かれていない状態に戻される。

- int Steel(int, int){}

  このメソッドは上に記述した int SteelJudge(int, int, int){}によって取られてしまう連が判明した際に実行される。引数として渡されてきた連番号が格納されている座標全ての石の状態が何も置かれていない状態に戻す。戻した石の個数を戻り値として返す。

- void PutJudge(int){}

  このメソッドではそのターンに応じた色の石がどこになら置くことが可能なのかを判定する。PutJudgeList[][]へ既に石が置かれている座標の他に、Int SteelJudge(int)によって相手の石を取ることも出来ずたった今置かれた石が取られてしまう状態、または beforeList[][]に格納された前の手に打った石の座標以外の場所には 0 を格納する。問題なく石を置ける座標にはその石の色に対応する数値を格納する。

- void before(int, int, int){}

  このメソッドでは石を置いた際、既に beforeList[][]に格納されていた自分の石の色の数値を 0 に戻し、今回置いた座標にその数値を格納する。

- int input(int, int, int)

  このメソッドでは引数として渡されてきた今置こうとしている座標とその石の色を
  void PutJudge(int){}によってセットされた PutJudgeList[][]より置けるかどうか判断し、置けるのなら List[][]の座標に格納する。またその際に befor(int, int, int)、RenSet()、SteelJudge(int, int, int)を実行しておく。これらの処理がなされたなら戻り値として 1 を、出来ないのなら 0 を返す。

3.2.3. CPU クラス

CPU クラスでは囲碁 AI を制作した。囲碁 AI は直接 List.List[][]を操作し手を先読みしていくため操作をする事前に Lisi.List[][]の現状のバックアップをとっていないといけない。そのためにこの CPU クラスでは複数のバックアップ用の配列を生成している。

このクラスで生成したメソッドは以下の通りである。

- void buckup(int){}
  このメソッドでは引数として渡されてきた整数に基づき、配列 List.List[][]の現状をどの配列にバックアップをとるかを決定し、格納する。
- void buckupR(int){}
  このメソッドはバックアップがとってある配列の通りに配列 List.List[][]をリセットするメソッドである。引数として渡されてきた整数によりどの配列の通りに復元するか決定される。
- int rmcpu(int){}
  このメソッドでは、事前に List.PutJudgeList[][]をセットしていることを前提に、そのターン置ける位置にランダムでどの位置に置くのかを決定する。しかし、このメソッドで直接配列に格納するのではなく、決定された座標に相当する値を戻り値として返す。
- int cpujudge(int){}
  このメソッドでは、そのターンに盤面の何処に置くと相手の石がいくつ取れるかを判定する。最も高い評価値を戻り値として返す。
- void judge1(int, int){}
  このメソッドは、引数として渡されてきた整数がグローバル変数で宣言された Loop の数を超えるまで、自身を呼び出しループし先読みしていくメソッドである。バックアップをとった後、最も評価値が高い座標に設置していく。同じ評価値が複数ある場合、そのパターン分石を置いた後の状態がどうなっていくかを見ていくことになる。先読み回数が Loop 分に達した時、その時点の一番高い評価値を記録する。
- void judge(int){}
  このメソッドでは先読みを行う前にバックアップをとり、 judge1(int, int)を実行するメソッドである。Judge1(int, int)が実行され、何処に置くのが一番良いのかが判断されると、変更された List.List[][]を元に戻し List.input(int)で配列にその場所に値を格納する。
- void player_turn(int){}
  このメソッドは殆ど judge(int)と同様でプレイヤー側の石の配置を計算する。

以上のメソッドを用い、囲碁アプリケーションを作成した。

4. 結果

本研究で作成した四路の碁アプリケーションの強さを検証するために、ランダムに石を設置する CPU と制作した先読み数が 1～5 回までの囲碁 AI を 100 回対戦させた。表 1 にその勝率と 1 手目の処理時間を示す。

表 1 四路の碁アプリケーションプログラムの勝率と実行時間(試行回数 100 回)

先読み回数	1	2	3	4	5
勝率(%)	73	88	99	100	100
処理時間(ms)	63	110	390	4134	57267

表 1 より、本研究にて制作したアプリケーションを実行すると、特に 1 手目に非常に処理時間がかかることが示される。特に 4 手先を読む場合と 5 手先を読む場合との間に飛躍的に処理時間が伸びていることが表からわかる。また、先読み回数を増やすことによって勝率も上昇している。もちろん相手はランダムなのでこの結果も偶然のものかもしれないが劇的に変わることはないと思わる。

5. 結論および今後の課題

本研究では四路の碁アプリケーションプログラムを作成した。本研究で作成したプログラムは、ランダム CPU に対して先読み回数を増やすほど勝率が上昇した。しかし、それに伴い処理時間が非常にかかりゲームとしては破綻している。本研究で制作した四路の碁アプリケーションは作り始める際に入念に設計したものではなく、製作途中で思いついたものを入れたり、書きなおしたりして制作したものであるので無駄なコードや処理が多々あると思われる。これほどの処理時間を要したのもこういった製作姿勢のためかと思われる。今後の課題としては、まず入念な設計段階を経たうえでコーディングすることが挙げられる。今回は 4＊4 の盤面での囲碁プログラムを制作したが、今後の課題として 19＊19 の正式なマス目の強い囲碁 AI を製作することが挙げられる。19 路盤においては、現在、モンテカルロ法を用いて制作することが有望であると考えられており、盤面の評価関数や先読みと組み合わせることでより強い囲碁 AI を作り出すことが、今後の課題として挙げられる。

謝辞

2 年間指導してくださった石水隆先生には、度々迷惑をかけたこともあり非常に申し訳なく、且つ非常に感謝してもしきれない思いでいっぱいです。今まで呆れたことも多々あると思われますが、見捨てることなく温かい目で見守り指導して下さって本当にありがとうございました。

参考文献

[1] よんろのご, 幻冬舎エデュケーション一 (2011) http://www.gentosha-edu.co.jp/products/post-95.html

[2] 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, Vol. 2000-GI-004, pp.69-76, (2000), http://id.nii.ac.jp/1001/00058633/

[3] 清慎一, 佐々木宣介, 山下宏：コンピュータ囲碁の入門, 共立出版(2005)

[4] GORO：囲碁ソフトの作り方, http://hp.vector.co.jp/authors/VA002290/Igo/create/gocreate.html

[5] 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).

[6] 田中哲朗, ボードゲーム「シンペイ」完全解析（修正版）, http://media.itc.u-tokyo.ac.jp/ktanaka/simpei/20060307-rev.pdf

[7] 目指せ！初級突破！！囲碁定石シミュミレーション, http://joseki-sim.ciao.jp/contents/list/hoshi/hoshi_tukenobi.php

[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, Robert Lake, 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] 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/

[12] 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/

[13] 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).

[14] IBM100 – Deep Blue, IBM, (1997),http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/

[15] Michael Khodarkovsky and Leonid Shamkvoich, 人間対機械－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)

[16] 伊藤英紀, A 級リーグ差し手 1 号, (2013), http://aleag.cocolog-nifty.com/

[17] 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).

[18] Commutative Weblog2 http://commutative.world.coocan.jp/blog2/2010/10/post-817.html

[19] 小島琢矢, 吉川厚, 囲碁における空間的チャンクと時間的チャンク, 情報処理学会研究報告, Vol. 1999-GI-002, pp.41-47, (2000), http://id.nii.ac.jp/1001/00058650/

[20] 吉川 , 小島琢矢, 囲碁対局プロトコルの諸相, 情報処理学会研究報告, Vol. 1999-GI-002, pp.49-54, (2000), http://id.nii.ac.jp/1001/00058651/

[21] 渡辺聡, 小谷善行, 棋譜からの捕獲パターンの獲得, 情報処理学会研究報告, Vol. 2000-GI-004, pp.55-60, (2000), http://id.nii.ac.jp/1001/00058631/

[22] 谷口和友 , 小島琢矢 , 吉川厚 , 植田一博, 詰碁におけるパターンの分布と着手の頻度, 情報処理学会研究報告, Vol. 2000-GI-004, pp.61-68, (2000), http://id.nii.ac.jp/1001/00058632/

[23] 梶山貴司 , 中村貞吾, 囲碁棋譜からの順序パターンとその共有知識の獲得, 情報処理学会研究報告, Vol.

2000-GI-004, pp.77-84, (2000), http://id.nii.ac.jp/1001/00058634/

[24] 阿部能明, 小谷善行, 囲碁における決定木を使った連の強度の学習, 情報処理学会研究報告, Vol. 2000-GI-005, pp.1-8, (2001), http://id.nii.ac.jp/1001/00058614/

[25] 佐々木宣介, ニューラルネットワークによる詰碁解答能力, 情報処理学会研究報告, Vol. 2002-GI-008, pp.45-51, (2002), http://id.nii.ac.jp/1001/00058591/

[26] 中村貞吾, 組み合わせゲーム理論を用いた囲碁の攻合いの解析, 情報処理学会研究報告, Vol. 2002-GI-009, pp.27-34, (2003), http://id.nii.ac.jp/1001/00058582/

[27] 鎌田真人, 松坂昇子 , 松原仁, 基力認定問題によるコンピュータ囲碁の評価(その 1), 情報処理学会研究報告, Vol. 2003-GI-010, pp.39-46, (2003), http://id.nii.ac.jp/1001/00058573/

[28] 福井真人, 竹内義則, 松本哲也, 工藤博章, 山村毅, 大西昇, 囲碁の中盤における盤面の評価関数, 情報処理学会研究報告, Vol. 2003-GI-010, pp.47-54, (2003), http://id.nii.ac.jp/1001/00058574/

[29] 田島守彦, 実近憲昭, 囲碁における族と領域, 情報処理学会研究報告, Vol. 2003-GI-010, pp.55-61, (2003), http://id.nii.ac.jp/1001/00058575/

[30] 福井真人, 竹内義則, 松本哲也, 工藤博章, 山村毅, 大西昇, 囲碁盤面の評価方法, 情報処理学会研究報告, Vol. 2003-GI-011, pp.21-26, (2004), http://id.nii.ac.jp/1001/00058555/

[31] 二宮勘輔, 囲碁ヨセ計算における近似法, 情報処理学会研究報告, Vol. 2004-GI-012, pp.55-62, (2004), http://id.nii.ac.jp/1001/00058549/

[32] 鎌田真人, 松坂昇子 , 松原仁, 基力認定問題によるコンピュータ囲碁の評価(その 2), 情報処理学会研究報告, Vol. 2004-GI-013, pp.35-42, (2005), http://id.nii.ac.jp/1001/00058540/

[33] 高濱昌孝, 大沢英一, 囲碁における注意決定と分節化について, 情報処理学会研究報告, Vol. 2004-GI-013, pp.43-48, (2005), http://id.nii.ac.jp/1001/00058541/

[34] 美添一樹 , 今井浩, 囲碁の部分問題における両利きの探索, 情報処理学会研究報告, Vol. 2005-GI-014, pp.63-70, (2005), http://id.nii.ac.jp/1001/00058533/

[35] 鎌田真人, 松坂昇子 , 松原仁, 基力認定問題によるコンピュータ囲碁の評価(その 3), 情報処理学会研究報告, Vol. 2006-GI-015, pp.1-8, (2006), http://id.nii.ac.jp/1001/00058516/

[36] 平本直之 , 成瀬正, 囲碁の中盤における盤面の評価関数に関する検討, 情報処理学会研究報告, Vol.2007-GI-018, pp.23-30, (2007), http://id.nii.ac.jp/1001/00058491/

[37] 野口陽来 , 松井利樹, 橋本隼一 , 橋本剛, モンテカルロ碁で用いるパターンの大きさに関する研究, 情報処理学会研究報告, Vol.2008-GI-020, pp.31-35, (2008), http://id.nii.ac.jp/1001/00058475/

[38] 鎌田真人, プロプロ 9 路盤囲碁の序盤の変化(その一), 情報処理学会研究報告, Vol. 2009-GI-21, pp.25-32 , (2009), http://id.nii.ac.jp/1001/00061656/

[39] 松井利樹 , 野口陽来 , 土井佑紀 , 橋本剛, 囲碁における勾配法を用いた確立関数の学習, 情報処理学会研究報告, Vol. 2009-GI-21, pp.33-40 , (2009), http://id.nii.ac.jp/1001/00061657/

[40] 矢島敏雄, 石の働きと盤の効果, 情報処理学会研究報告, Vol. 2009-GI-21, pp.41-46 , (2009), http://id.nii.ac.jp/1001/00061658/

[41] 松本祐輔 , 小谷善行, 溢れ碁ルールの提案とそれを用いた囲碁の小路探索, 情報処理学会研究報告, Vol. 2010-GI-23, No.1 ,pp.1-8 , (2010), http://id.nii.ac.jp/1001/00068061/

[42] 豊田琢磨, 松本祐輔, 佐々木健太 , 小谷善行, 囲碁におけるシミュレーション結果の継承を用いたモンテカルロ法の改良, 情報処理学会研究報告, Vol.2010-GI-23, No.2, pp.1-7, (2010),

http://id.nii.ac.jp/1001/00068062/

[43] 岩川夏季, 成見哲, 村松正和, GPGPU によるモンテカルロ碁のシミュレーションの並列処理, 情報処理学会研究報告, Vol.2011-GI-26, No.10, pp.1-6, (2011), http://id.nii.ac.jp/1001/00074629/

[44] (株)マグノリア, バリュー囲碁 4【無料版】, http://www.vector.co.jp/soft/win95/game/se428339.html

[45] 毎日 JP, ルポ・コンピューター囲碁ソフト最新事情, 毎日新聞, 2012 年 11 月 12 日, 東京夕刊, http://mainichi.jp/feature/news/20121112dde012040083000c.html

付録

以下に本研究で作成したプログラムのソースを示す。

goApplet クラス

```
package YonroNoGo;

import java.applet.Applet;

import java.awt.*;

import java.awt.event.*;


public    class goApllet extends Applet implements ActionListener{

                private static final long serialVersionUID = 1L;

                int turn = 1,ok2=1;

                boolean E = false;

                Button btn11,btn12,btn13,btn14,btn21,btn22,btn23,btn24,btn31,btn32,btn33,btn34,btn41,btn42,btn43,btn44;

                Button D,R,seclet;

                Button firstturn,secondturn;

                boolean first= true;

                List list = new List();

                CPU cpu = new CPU();


                public void init() {

                                this.setLayout(null);

                                btn11 =new Button();

                                btn12 =new Button();

                                btn13 =new Button();

                                btn14 =new Button();

                                btn21 =new Button();

                                btn22 =new Button();

                                btn23 =new Button();

                                btn24 =new Button();

                                btn31 =new Button();

                                btn32 =new Button();

                                btn33 =new Button();

                                btn34 =new Button();

                                btn41 =new Button();

                                btn42 =new Button();

                                btn43 =new Button();

                                btn44 =new Button();
```

```
R=new Button("リセット");
seclet=new Button();
firstturn=new Button("先攻");
secondturn=new Button("後攻");

btn11.setBounds(30,80,40,10);
btn12.setBounds(130,80,40,10);
btn13.setBounds(230,80,40,10);
btn14.setBounds(330,80,40,10);
btn21.setBounds(30,180,40,10);
btn22.setBounds(130,180,40,10);
btn23.setBounds(230,180,40,10);
btn24.setBounds(330,180,40,10);
btn31.setBounds(30,280,40,10);
btn32.setBounds(130,280,40,10);
btn33.setBounds(230,280,40,10);
btn34.setBounds(330,280,40,10);
btn41.setBounds(30,380,40,10);
btn42.setBounds(130,380,40,10);
btn43.setBounds(230,380,40,10);
btn44.setBounds(330,380,40,10);
R.setBounds(300,420,55,20);
seclet.setBounds(600,420,10,10);
firstturn.setBounds(80,420,100,20);
secondturn.setBounds(230,420,100,20);

if(first == false) {
        add(btn11); add(btn12); add(btn13); add(btn14);
        add(btn21); add(btn22); add(btn23); add(btn24);
        add(btn31); add(btn32); add(btn33); add(btn34);
        add(btn41); add(btn42); add(btn43); add(btn44);
        add(R);
        add(seclet);
} else {
        add(firstturn);
        add(secondturn)
}
btn11.addActionListener(this);
btn12.addActionListener(this);
```

```
		btn13.addActionListener(this);
		btn14.addActionListener(this);
		btn21.addActionListener(this);
		btn22.addActionListener(this);
		btn23.addActionListener(this);
		btn24.addActionListener(this);
		btn31.addActionListener(this);
		btn32.addActionListener(this);
		btn33.addActionListener(this);
		btn34.addActionListener(this);
		btn41.addActionListener(this);
		btn42.addActionListener(this);
		btn43.addActionListener(this);
		btn44.addActionListener(this);
		 R.addActionListener(this);
		seclet.addActionListener(this);
		firstturn.addActionListener(this);
		secondturn.addActionListener(this);
}

public void paint(Graphics g) {
		setBackground(new Color(230,180,50));

		for(int i = 0; i <4; i++) {
			g.drawLine(50, i*100+50, 350, i*100+50);
			g.drawLine(i*100+50, 50, 50+i*100, 350);
		}
		for(int i=1;i<5;i++) {
			for(int j=1;j<5;j++) {
				if(List.List[i][j]==1) {
					g.setColor(Color.black);
					g.fillOval(20+((j-1)*100), 20+((i-1)*100), 60, 60);
				} else if(List.List[i][j]==2) {
					g.setColor(Color.white);
					g.fillOval(20+((j-1)*100), 20+((i-1)*100), 60, 60);
				}
			}
		}
		if(first==false){
```

```
                    if(turn==1) g.setColor(Color.black);
                    else g.setColor(Color.white);
                    g.drawString("あなたのターンです", 150,10 );
            }
            if(ok2==0) {
                    if(turn == 1) g.setColor(Color.black);
                    else g.setColor(Color.white);
                    g.drawString("そこには置けません", 150,420 );
            }
}

void turnchange() {
            if(turn==1) turn=2; else turn=1;
}

public void actionPerformed(ActionEvent e)  {
            long start = System.currentTimeMillis();
            list.listReset();
            if(first == false) {
                    int ok1=0;
                    list.putJudge(turn);
                    if(e.getSource() == btn11) {
                            if(List.PutJudgeList[1][1]==turn) {
                                    ok1=list.input(1, 1, turn);
                                    ok2=1;
                            } else {
                                    ok1=-1;
                                    ok2=0;
                            }
                    } else if(e.getSource() == btn12) {
                            if(List.PutJudgeList[1][2]==turn) {
                                    ok1=list.input(1, 2, turn);
                                    ok2=1;
                            } else {
                                    ok1=-1;
                                    ok2=0;
                            }
                    } else if(e.getSource() == btn13) {
                            if(List.PutJudgeList[1][3]==turn) {
```

```
                    ok1=list.input(1, 3, turn);
                    ok2=1;
                } else {
                    ok2=0;
                    ok1=-1;
                }
            } else if(e.getSource() == btn14) {
                if(List.PutJudgeList[1][4]==turn) {
                    ok1=list.input(1, 4, turn);
                    ok2=1;
                } else {
                    ok1=-1;
                    ok2=0;
                }
            } else if(e.getSource() == btn21) {
                if(List.PutJudgeList[2][1]==turn) {
                    ok1=list.input(2, 1, turn);
                    ok2=1;
                } else {
                    ok1=-1;
                    ok2=0;
                }
            } else if(e.getSource() == btn22) {
                if(List.PutJudgeList[2][2]==turn) {
                    ok1=list.input(2, 2, turn);
                    ok2=1;
                } else {
                    ok1=-1;
                    ok2=0;
                }
            } else if(e.getSource() == btn23) {
                if(List.PutJudgeList[2][3]==turn) {
                    ok1=list.input(2, 3, turn);
                    ok2=1;
                } else {
                    ok1=-1;
                    ok2=0;
                }
            } else if(e.getSource() == btn24) {
```

```
                if(List.PutJudgeList[2][4]==turn ) {
                        ok1=list.input(2, 4, turn);
                        ok2=1;
                } else{
                        ok1=-1;
                        ok2=0;
                }
        } else if(e.getSource() == btn31) {
                if(List.PutJudgeList[3][1]==turn ) {
                        ok1=list.input(3, 1, turn);
                        ok2=1;
                } else {
                        ok1=-1;
                        ok2=0;
                }
        } else if(e.getSource() == btn32) {
                if(List.PutJudgeList[3][2]==turn) {
                        ok1=list.input(3, 2, turn);
                        ok2=1;
                } else {
                        ok1=-1;
                        ok2=0;
                }
        } else if(e.getSource() == btn33) {
                if(List.PutJudgeList[3][3]==turn) {
                        ok1=list.input(3, 3, turn);
                        ok2=1;
                } else {
                        ok1=-1;
                        ok2=0;
                }
        } else if(e.getSource() == btn34) {
                if(List.PutJudgeList[3][4]==turn) {
                        ok1=list.input(3, 4, turn);
                        ok2=1;
                } else {
                        ok1=-1;
                        ok2=0;
                }
```

```
} else if(e.getSource() == btn41) {
		if(List.PutJudgeList[4][1]==turn) {
			ok1=list.input(4, 1, turn);
			ok2=1;
		} else {
			ok1=-1;
			ok2=0;
		}

} else if(e.getSource() == btn42) {
		if(List.PutJudgeList[4][2]==turn) {
			ok1=list.input(4, 2, turn);
			ok2=1;
		} else {
			ok1=-1;
			ok2=0;
		}
} else if(e.getSource() == btn43) {
		if(List.PutJudgeList[4][3]==turn) {
			ok2=1;
			ok1=list.input(4, 3, turn);
		} else {
			ok1=-1;
			ok2=0;
		}
} else if(e.getSource() == btn44) {
		if(List.PutJudgeList[4][4]==turn) {
			ok1=list.input(4, 4, turn);
			ok2=1;
		} else {
			ok1=-1;
			ok2=0;
		}
} else if(e.getSource()==R) {
		for(int i=1;i<5;i++) {
			for(int j=1;j<5;j++) {
				List.List[i][j]=0;
			}
		}
```

```
                    list.beforeListReset();
                    first=true;
                    removeAll();
                    ok2=1;
                    repaint();
                    init();
                    ok1=-1;
                    turn = 1;
                    E=false;
            } else if(e.getSource()==seclet) {
                    for(int i=1;i<5;i++) {
                            for(int j=1;j<5;j++) {
                                    if(List.List[i][j]!=turn)
                                    List.List[i][j]=0;
                            }
                    }
                    repaint();
                    ok1=-1;
            } else if(e.getSource()==D)
                    System.exit(0);
            if(ok1==1) {
                    turnchange();
                    cpu.judge(turn);
                    list.input(cpu.n, cpu.m, turn);
                    turnchange();
                    repaint();
                    long end=System.currentTimeMillis();
                    System.out.println((end - start)+"ms");
            } else if(ok1==0) {
                    ok2=0;
                    repaint();
            } else repaint();
    } else {
            if(e.getSource()==firstturn)
                    turn = 1;
            else {
                    List.List[2][2]=1;
                    turn = 2;
            }
```

```
                    first=false;
                    remove(firstturn);
                    remove(secondturn);
                    init();
                    repaint();
                }
        }
}
```

List クラス

```
package YonroNoGo;


public class List {
        public static int[][] List = new int[6][6];
        public static int[][] RenList = new int[6][6];
        public static int[][] JudgeList = new int[6][6];
        public static int[][] PutJudgeList = new int[6][6];
        public static int[][] beforeList = new int[6][6];
        public static int[][] ji = new int[6][6];
        public static int jiA=0,jiB=0;
        public int[][] List2;
        public List(){
                for(int i=0;i<6;i++) {
                        for(int j=0;j<6;j++) {
                                if(i==0 || i==5 || j==0 || j==5) List[i][j]=3;
                                else List[i][j]=0;
                                beforeList[i][j]=0;
                                ji[i][j]=0;
                        }
                }
        }

        void beforeListReset() {
                for(int i=0;i<6;i++) {
                        for(int j=0;j<6;j++) {
                                beforeList[i][j] = 0;
                        }
                }
        }
```

```
void renListReset() {
		for(int i=0;i<6;i++) {
			for(int j=0;j<6;j++) {
				RenList[i][j] = 0;
			}
		}
}

void judgeListReset() {
		for(int i=0;i<6;i++) {
			for(int j=0;j<6;j++) {
				JudgeList[i][j] = 0;
			}
		}
}

void putJudgeListReset() {
		for(int i=0;i<6;i++) {
			for(int j=0;j<6;j++) {
				PutJudgeList[i][j] = 0;
			}
		}
}

public void listReset() {
		renListReset();
		judgeListReset();
		putJudgeListReset();
}

public void jiSet() {
		for(int i=1;i<5;i++) {
			for(int j=1;j<5;j++) {
				if(List[i][j]==1) ji[i][j]=1;
				else if(List[i][j]==2) ji[i][j]=2;
				else if(List[i][j]==0) {
					if(List[i-1][j]==1 || List[i-1][j]==3) {
						if(List[i+1][j]==1 || List[i+1][j]==3) {
```

```
                                        if(List[i][j-1]==1 || List[i][j-1]==3) {
                                                if(List[i][j+1]==1 || List[i][j+1]==3)
                                                        ji[i][j]=1;
                                        }
                                }
                        }
                        if(List[i-1][j]==2 || List[i-1][j]==3) {
                                if(List[i+1][j]==2 || List[i-1][j]==3) {
                                        if(List[i][j-1]==2 || List[i][j-1]==3) {
                                                if(List[i][j+1]==2 || List[i][j+1]==3)
                                                        ji[i][j]=2;
                                        }
                                }
                        }
                }
        }
    }

    public int hyoukachi_hantei(){
        jiA=0;
        jiB=0;
        for(int i=1;i<5;i++) {
                for(int j=1;j<5;j++) {
                        if(ji[i][j]!=0){
                                if(ji[i][j]==1) jiA+=1;
                                else jiB+=1;
                        }
                }
        }
        return (jiA+jiB);
    }

    public void RenSet(int RenNum,int x,int y) {
        RenList[x][y]=RenNum;
        if(List[x][y+1]==List[x][y] && RenList[x][y+1]==0)
                RenSet(RenNum,x,y+1);
        if(List[x][y-1]==List[x][y] && RenList[x][y-1]==0)
                RenSet(RenNum,x,y-1);
```

```
        if(List[x-1][y]==List[x][y] && RenList[x-1][y]==0)
                RenSet(RenNum,x-1,y);
        if(List[x+1][y]==List[x][y] && RenList[x+1][y]==0)
                RenSet(RenNum,x+1,y);
}

public void RenSet() {
        int RenNum = 1;
        for(int x=1;x<5;x++) {
                for(int y=1;y<5;y++) {
                        if(RenList[x][y]==0 && List[x][y]!=0) {
                                RenSet(RenNum,x,y);
                                RenNum+=1;
                        }
                }
        }
}

public int Steel(int RenNum,int turn) {
        int a=0;
        for(int i=1;i<5;i++) {
                for(int j=1;j<5;j++) {
                        if(RenList[i][j]==RenNum) {
                                if(List[i][j]!=turn) a+=1;
                                else a-=1;
                                List[i][j]=0;
                        }
                }
        }
        return a;
}

public int SteelJudge(int renNum) {
        int steel=-1;
        for(int i=1;i<5;i++) {
                for(int j=1;j<5;j++) {
                        if(RenList[i][j]==renNum) {
                                if(List[i][j+1]!=List[i][j] && List[i][j+1]!=0 && steel!=0)
                                        steel = List[i][j];
```

```
                                        else if(List[i][j+1]==0) {
                                                steel = 0;
                                                break;
                                        }
                                        if(List[i][j-1]!=List[i][j] && List[i][j-1]!=0 && steel!=0)
                                                steel = List[i][j];
                                        else if(List[i][j-1]==0) {
                                                steel = 0;
                                                break;
                                        }
                                        if(List[i+1][j]!=List[i][j] && List[i+1][j]!=0 && steel!=0)
                                                steel = List[i][j];
                                        else if(List[i+1][j]==0) {
                                                steel = 0;
                                                break;
                                        }
                                        if(List[i-1][j]!=List[i][j] && List[i-1][j]!=0 && steel!=0)
                                                steel = List[i][j];
                                        else if(List[i-1][j]==0) {
                                                steel = 0;
                                                break;
                                        }
                                }
                        }
                }
                return steel;
        }

        public int SteelJudge(int x,int y,int turn) {
                int a=0;
                int steel=0;
                int b = RenList[x][y];
                for(int i=1;i<5;i++) {
                        for(int j=1;j<5;j++) {
                                int renNum=0;
                                if(RenList[i][j]!=0 && JudgeList[i][j]==0) {
                                        renNum=RenList[i][j];
                                        if(b!=renNum) {
                                                steel=SteelJudge(renNum);
```

```
                                        if(steel!=0)
                                                a+=Steel(renNum,turn);
                                }
                        }
                }
        }
        steel=SteelJudge(b);
        if(steel!=0)
                a+=Steel(b,turn);
        return a;
}

public void putJudge(int turn) {

        for(int i=1;i<5;i++) {
                for(int j=1;j<5;j++) {
                        if(List[i][j]==0) {
                                List[i][j]=turn;
                                RenSet();
                                int renNum = RenList[i][j];
                                int maxRen=0;
                                int othersteel=0;
                                for(int k=1;k<5;k++) {
                                        for(int l=1;l<5;l++) {
                                                if(RenList[k][l]>maxRen)
                                                        maxRen=RenList[k][l];
                                        }
                                }

                                for(int m=1;m<=maxRen;m++) {
                                        if(m!=renNum) {
                                                othersteel=SteelJudge(m);
                                        }
                                        if(othersteel!=0)
                                                break;
                                }
                                int a=SteelJudge(renNum);

                                if((a!=turn || othersteel!=0) && beforeList[i][j]!=turn ) {
```

```
                                        PutJudgeList[i][j]=turn; //置ける場所にその数値が入る
                                }
                                List[i][j]=0;
                                renListReset();
                        }
                }
        }
}

public void before(int x,int y,int turn) {
        for(int i=1;i<5;i++) {
                for(int j=1;j<5;j++) {
                        if(beforeList[i][j]==turn)
                                beforeList[i][j]=0;
                }
        }
        beforeList[x][y]=turn;
}

public int input(int x,int y,int turn) {
        int a=0;
        if(PutJudgeList[x][y]==turn) {
                List[x][y]=turn;
                before(x,y,turn);
                RenSet();
                SteelJudge(x,y,turn);
                jiSet();
                a=1;
        }
        return a;
}

public void print() {
        System.out.println("print");
        for(int i=1;i<5;i++){
                for(int j=1;j<5;j++){
                        System.out.printf("%d", List[i][j]);
                }
                System.out.println();
```

```
                }
                System.out.println();
        }

        public void jprint(){
                System.out.println("jprint");
                for(int i=1;i<5;i++){
                        for(int j=1;j<5;j++){
                                System.out.printf("%d", PutJudgeList[i][j]);
                        }
                        System.out.println();
                }
                System.out.println();
        }
}
```

CPU クラス

```
package YonroNoGo;

import java.util.Random;


public class CPU {
        int[][] 地の評価値= new int[6][6];
        int[][] 地の評価値2= new int[6][6];
        int[][] 地の評価値3= new int[6][6];
        int[][] buList= new int[6][6];
        int[][] buList1= new int[6][6];
        int[][] buList2= new int[6][6];
        int[][] buList3= new int[6][6];
        int[][] buList4= new int[6][6];
        int[][] buList5= new int[6][6];
        int[][] buList_1= new int[6][6];
        int[][] buList_2= new int[6][6];
        int[][] BuList= new int[6][6];
        int[][] BuList1= new int[6][6];
        int[][] BuList2= new int[6][6];
        int[][] BuList3= new int[6][6];
        int[][] BuList4= new int[6][6];
        int[][] BuList5= new int[6][6];
        int[][] BuList_1= new int[6][6];
```

```
	int[][] BuList_2= new int[6][6];
	int[][] inputcpu=new int[6][6];
	int Loop=1,x,y,n,m,o,p,e,f;

	List list = new List();

	public CPU() {
		for(int i=0;i<6;i++) {
			for(int j=0;j<6;j++) {
				地の評価値[i][j]=0;
			}
		}
	}

	public void buckup(int a){
		for(int i=1;i<5;i++){
			for(int j=1;j<5;j++){
				if(a==0)
					buList[i][j]=List.List[i][j];
				else if(a==1)
					buList1[i][j]=List.List[i][j];
				else if(a==2)
					buList2[i][j]=List.List[i][j];
				else if(a==3)
					buList3[i][j]=List.List[i][j];
				else if(a==4)
					buList4[i][j]=List.List[i][j];
				else if(a==5)
					buList5[i][j]=List.List[i][j];
				else if(a==-1)
					buList_1[i][j]=List.List[i][j];
				else
					buList_2[i][j]=List.List[i][j];
			}
		}
	}

public void Buckup(int a){
		for(int i=1;i<5;i++){
```

```
                for(int j=1;j<5;j++){
                        if(a==0)
                                BuList[i][j]=List.List[i][j];
                        else if(a==1)
                                BuList1[i][j]=List.List[i][j];
                        else if(a==2)
                                BuList2[i][j]=List.List[i][j];
                        else if(a==3)
                                BuList3[i][j]=List.List[i][j];
                        else if(a==4)
                                BuList4[i][j]=List.List[i][j];
                        else if(a==5)
                                BuList5[i][j]=List.List[i][j];
                        else if(a==-1)
                                BuList_1[i][j]=List.List[i][j];
                        else
                                BuList_2[i][j]=List.List[i][j];
                }
        }
}

public void buckupR(int a){
        for(int i=1;i<5;i++){
                for(int j=1;j<5;j++){
                        if(a==0)
                                List.List[i][j]=buList[i][j];
                        else if(a==1)
                                List.List[i][j]=buList1[i][j];
                        else if(a==2)
                                List.List[i][j]=buList2[i][j];
                        else if(a==3)
                                List.List[i][j]=buList3[i][j];
                        else if(a==4)
                                List.List[i][j]=buList4[i][j];
                        else if(a==5)
                                List.List[i][j]=buList5[i][j];
                        else if(a==-1)
                                List.List[i][j]=buList_1[i][j];
                        else
```

```
                                        List.List[i][j]=buList_2[i][j];
                        }
                }
        }

public void BuckupR(int a){
                for(int i=1;i<5;i++){
                        for(int j=1;j<5;j++){
                                if(a==0)
                                        List.List[i][j]=BuList[i][j];
                                else if(a==1)
                                        List.List[i][j]=BuList1[i][j];
                                else if(a==2)
                                        List.List[i][j]=BuList2[i][j];
                                else if(a==3)
                                        List.List[i][j]=BuList3[i][j];
                                else if(a==4)
                                        List.List[i][j]=BuList4[i][j];
                                else if(a==5)
                                        List.List[i][j]=BuList5[i][j];
                                else if(a==-1)
                                        List.List[i][j]=BuList_1[i][j];
                                else
                                        List.List[i][j]=BuList_2[i][j];
                        }
                }
        }

        public int rmcpu(int turn) {
                Random rnd = new Random();

                list.putJudge(turn);
                int J = 0;
                for(int i=1;i<5;i++){
                        for(int j=1;j<5;j++){
                                if(List.PutJudgeList[i][j]==turn)
                                        J++;
                        }
                }
```

```
		int ran1=0;
		if(J!=0){
			ran1 = rnd.nextInt(turn);
			ran1+=1;
		}
		return ran1;
	}

	public int cpujudge(int turn) {
		int kitaichi=0;
		list.putJudge(turn);
		for(int i=1;i<5;i++) {
			for(int j=1;j<5;j++) {
				buckup(-1);
				if(List.List[i][j]==0 && List.PutJudgeList[i][j]==turn) {
					List.List[i][j]=turn;
					list.RenSet();
					地の評価値2[i][j]=list.SteelJudge(i,j,turn);
					list.renListReset();
					buckupR(-1);
					if(kitaichi<地の評価値2[i][j])
						kitaichi=地の評価値2[i][j];
				}
			}
		}
		return kitaichi;
	}

	public void judge1(int turn,int loop) {
		loop++;
		list.putJudge(turn);
		int max=cpujudge(turn);
		int[][] steel_able = new int[6][6];
		for(int i=1;i<5;i++){
			for(int j=1;j<5;j++){
				if(max==地の評価値2[i][j])
					steel_able[i][j]=max;
				else
					steel_able[i][j]=-1;
```

```
                }
        }
        if(loop<=Loop){
                for(int i=1;i<5;i++){
                        for(int j=1;j<5;j++){
                                if(steel_able[i][j]==max){
                                        if(loop==1){
                                                x=i;y=j;
                                        }
                                        buckup(loop);
                                        List.List[i][j]=turn;
                                        list.RenSet();
                                        list.SteelJudge(i, j, turn);
                                        list.jiSet();
                                        player_turn(turn);
                                        judge1(turn,loop);
                                        buckupR(loop);
                                }
                        }
                }
        } else{
                if(地の評価値[x][y]<max)
                        地の評価値[x][y]=max;
        }
}

public void judge(int turn){
        list.listReset();
        buckup(-2);
        int loop=0;
        judge1(turn,loop);
        buckupR(-2);
        int i,j;
        for(i=1;i<5;i++){
                for(j=1;j<5;j++){
                        System.out.printf("%d",List.List[i][j]);
                }
                System.out.printf("¥n");
        }
```

```
            System.out.printf("¥n");


            list.RenSet();
            list.jiSet();
            list.listReset();
            list.putJudge(turn);
            int max=0;
            for(i=1;i<5;i++){
                    for(j=1;j<5;j++){
                            if(max<地の評価値[i][j] && List.PutJudgeList[i][j]==turn){
                                    max=地の評価値[i][j];
                                    n=i;
                                    m=j;
                            }
                    }
            }
            if(max==0){
                    int a=rmcpu(turn);
                    int b=0;
                    for(i=1;i<5;i++){
                            for(j=1;j<5;j++){
                                    if(List.PutJudgeList[i][j]==turn){
                                            b++;
                                    }
                                    if(a==b){
                                            n=i;
                                            m=j;
                                    }
                            }
                    }
            }
            System.out.println(turn);
    }

    public void player_turn1(int Turn,int loop){
            loop++;
            int turn;
            if(Turn==1) turn=2; else turn=1;
            list.putJudge(Turn);
```

```
		int max=cpujudge(Turn);
		int[][] steel_able = new int[6][6];
		for(int i=1;i<5;i++){
			for(int j=1;j<5;j++){
				if(max==地の評価値3[i][j])
					steel_able[i][j]=max;
				else
					steel_able[i][j]=-1;
			}
		}
		if(loop<=Loop){
			for(int i=1;i<5;i++){
				for(int j=1;j<5;j++){
					if(steel_able[i][j]==max){
						if(loop==1){
							e=i;f=j;
						}
						Buckup(loop);
						List.List[i][j]=Turn;
						list.RenSet();
						list.SteelJudge(i, j, Turn);
						list.jiSet();
						judge(turn);
						/*judge2(turn);*/
						player_turn1(Turn,loop);
						BuckupR(loop);
					}
				}
			}
		} else{
			if(地の評価値2[e][f]<max)
				地の評価値2[e][f]=max;
		}
	}

	public void player_turn(int turn){
		int a;
		if(turn==1) a=2; else a=1;
```

```
                cpujudge(a);
                list.putJudge(a);
                int max = 0;
                for(int i=1;i<5;i++){
                        for(int j=1;j<5;j++){
                                if(List.PutJudgeList[i][j]==a){
                                        if(地の評価値2[i][j]>max)
                                                max = 地の評価値2[i][j];
                                }
                        }
                }
                int set=0;
                for(int i=1;i<5;i++){
                        for(int j=1;j<5;j++) {
                                if(地の評価値2[i][j]==max && set==0){
                                        List.List[i][j]=a;
                                        list.RenSet();
                                        list.SteelJudge(i, j, a);
                                        list.jiSet();
                                        set=1;
                                }
                        }
                }
        }
}
```