リバーシ AI の自己学習

＜1.活動の概要 ＞

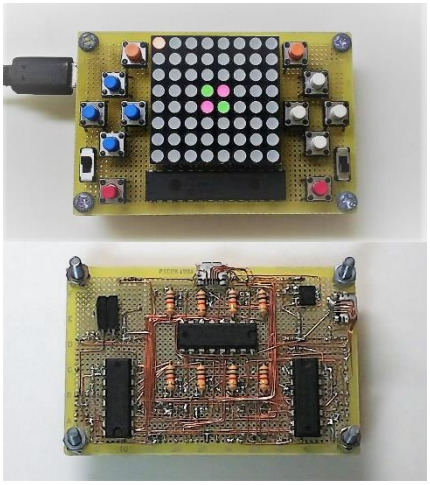
1-1.製作したもの

・A 自己学習プログラム(C 言語)(Android 上で動作)

・B リバーシ実行機及びその回路を制御するプログラム(C 言語)(マイクロコントローラとして PIC を使用)

共に単独で製作した。

自己学習プログラム、リバーシ実行機の順に活動内容を記す。



※写真　B リバーシ実行機

1-2.使用した外部プログラム

・Android アプリケーション「モバイル C」(Lee Jeong Seop)
(C コンパイラ。自己学習プログラム作成に使用。)

・Windows アプリケーション「MPLAB X IDE v3.50」(Microchip 社)
(PIC の統合開発環境ソフトウェア。PIC 用プログラムの書き込みに使用。)

・Windows アプリケーション「MPLAB X XC8 v1.38」(Microchip 社)
(PIC 用 C コンパイラ。PIC 用プログラム作成に使用。)

1-3.活動の動機

AI や人工知能が話題となるなか、AI の入門であろうゲーム対戦 AI について学びたいと思っていた。そんな中、2017 年夏、日常的に行っている電子回路の製作の中で 8×8 ドットマトリクスを入手する機会があった。その時、自由度の高いグラフィック LCD を用いずに何かを作成することに意義を感じ、ドットマトリクスから連想したリバーシの対戦 AI を製作したいと思ったため。

1-4.活動の目的

・AI を自作することで自己学習や AI の思考ルーチンについて学ぶ。

・AI に自己学習をさせてリバーシの定石である「角を取る」「序盤中盤は取りすぎない」の有効性を確認する。

・持ち運びを意識した作品の製作を通じて、回路の小型化を学び、回路の製作技術の向上を図る。

1	2	3	4	4	3	2	1
2	5	6	7	7	6	5	2
3	6	8	9	9	8	6	3
4	7	9	10	10	9	7	4
4	7	9	10	10	9	7	4
3	6	8	9	9	8	6	3
2	5	6	7	7	6	5	2
1	2	3	4	4	3	2	1

※図 1　対称性

＜2. A 自己学習プログラムについて ＞

2-1.概要

・C 言語を用いて書いたリバーシ AI の自己学習プログラム。

・AI の自己学習を目的としてはいるが、プレイヤーと対戦することも可能にした。

・ライブラリについては、標準ライブラリの stdio.h(計算課程・結果を表示するため)time.h(計算時間を測るため)のみを使用した。

2-2.構造と特徴

AI の評価関数については、各マスに点数を設定して、(自分の駒が置いてあるマスの点数の合計)-(相手の駒が置いてあるマスの点数の合計)を計算する方式をとった。以下、各マスの点数のことを評価ボードと呼ぶ。角や辺の有利な配置を教えておく方式や、置ける数に着目して計算する方式も検討した。だがその方式は、角や辺、置ける数が勝敗に重要であることを初めから AI に教えていることになると考えたため、用いなかった。評価ボードは一つの AI につき三つ用意し、ゲームを三分割して序盤中盤終盤で使い分けることとした。これにより、

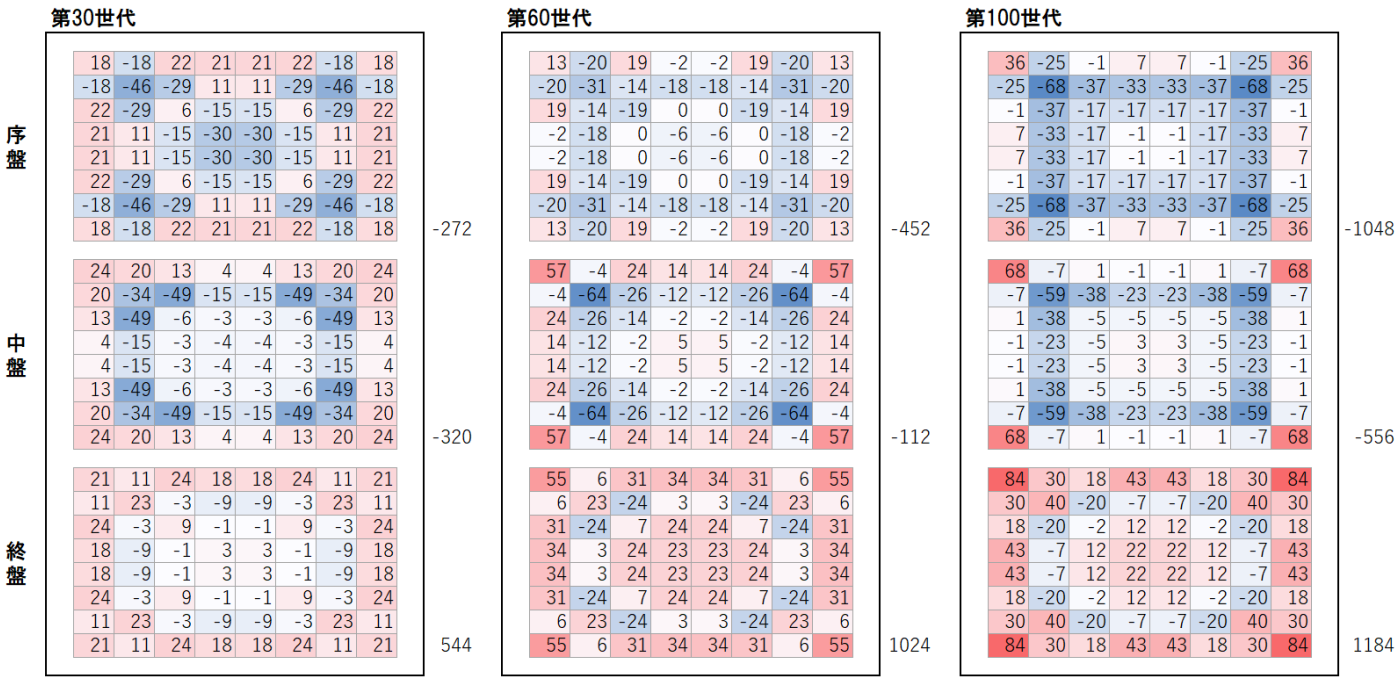
ゲームが進行するにつれてマスの重要性がどのように変化するのかを確認できる。また、リバーシにおける対称性に着目し、評価ボードのデータを 64 個から 10 個に減らす方法を用いた(図 1 参照)。これにより、評価ボードの自由度を減らすことなく収束速度を上げることが期待できる。探索の深さについては 4 手とした。探索の方法は、ネガマックス法とその改良版である α-β 法の両方を用いて、実行時間を比較した。

自己学習の方法として遺伝的アルゴリズムを用いた。勝者のみが子孫を残せるものとし、今回は一世代の数を 20、勝者の数を 2、突然変異種の出現はランダムとし、確率は 80 分の 1 とした。勝者の決定方法は先攻後攻区別ありの総当たりを 2 回行うものとした。全試合の(自分の駒数)-(相手の駒数)の和をその AI のスコアとして、順位を決めた。今後もさまざまな条件下で自己学習ができるよう、探索の深さだけでなく、ゲームの分割数、遺伝的アルゴリズムの各種値や子の類似性の高さ、総当たりの回数、さらには盤面の大きさもマクロ定義を用いて変更、設定できるようプログラムした。

このプログラムではさまざまな場面で乱数を必要とする。乱数の生成方法として、ライブラリの rand を用いる方法があるが、線形合同法が使われていて過去には疑似乱数としての質の低さが指摘されていたこともある。また、できる限り自分で作ってみることで様々なことを学びたいという意図もあり、結局どちらも質が低いのであれば自分で作る方が良いと考え、乱数には自分で作成した 16bit の線形帰還シフトレジスタ(LFSR)を用いることにした。

AI 同士での対戦だけでなく人間と戦うこともできるようにし、人間と対戦して学習する機能も製作した。AI が負けた場合、そのゲームのプレイヤーの手から相手の評価ボードを推測し、学習する。しかしこの学習方法は、人間が対戦するため実行速度も収束速度も遅く非常に非効率的である事、継続的に AI に勝つことが出来る人が必要な事、プレイヤーの実力を数値化できない為客観的に学習を評価できない事が問題点として上げられたため、性能評価ができなかった。

2-3.動作結果



※図 2 評価ボードの世代変化

・100 世代まで計算した。上の図は第 30 世代、第 60 世代、第 100 世代の評価ボードを示し、高い値ほど赤く、低い値ほど青く色付けをしたものである。また、右下に評価ボードの総和を記載した。

・100 世代の計算にかかった時間は手元の環境で、ネガマックス法 931 分、α-β 法 168 分であり、約 5.5 倍の効率であった。

2-4.考察

まず、第 100 世代に着目する。全ての評価ボードで角の点数が高くなっており、「角の重要性」を学習していることが分かる。特に序盤中盤では、角の周辺の値が低くなっており、角を取られる配置にならないようにしていることがうかがえる。また、序盤中盤は全体的に負の値となっていて、終盤は正の値となっている。つまり、終盤になるまでできるだけ少なく駒を取り、終盤で一気に駒を取るようになっており、「序盤中盤は取りすぎない」という定石を学習していることがわかる。他にも、リバーシには「中割り」という戦法がある。自分の石が相手の石に囲まれるように打つことで、相手が打てる場所を少なくするという戦法だ。この戦法を知った上で評価ボードを見ると、低い値がドーナツ状に分布していて、中央は比較的高い値になっていることに気が付く。このことから、自分は中央に、相手はその周りにあるような局面を作ろうとしていることが分かる。つまり、中割りの戦法を学習したと考えられる。

次に、世代の流れに着目する。第 30 世代序盤では、角の点数が正であることから角の重要性を学んでいる。しかし、序盤にも高い数値があり、全体を通じて中央の値は低くなっているため、序盤中盤は取りすぎない戦法と中割りはそこまで学んでいないようである。しかし 60 世代になると、序盤の正の値は 12 個のみとなっていて全体的に低い値となっており、序盤中盤は取りすぎない戦法を学習していることがわかる。そして 100 世代になり、序盤中盤の低い値がよりはっきりとドーナツ状に分布し、中割りの戦法を習得している。

この考察を踏まえて私はリバーシの定石について知る順番が人間と似ていると思った。人間もコンピューターも、分かりやすく結果の出やすい戦法から順に学び、強くなっていくのだと思うと不思議と親近感を覚えた。

2-5.問題と改善点

今回のプログラムはそこまで高いランダム性を求めていないこともあり、学習の面から LFSR を作成したが、他の疑似乱数と同様に LFSR にも欠点は多くある。今後、疑似乱数により高い質を求めるプログラムを作ることも想定されるので、メルセンヌツイスタなどの高品質な疑似乱数を扱えるようになりたいと考えている。

＜3. B リバーシ実行機について ＞

3-1.概要

・「盤がない時でも気軽にリバーシを楽しめるポータブルゲーム機を作る」を目的とし、小さいがユーザビリティの高いものを作ることを意識して製作した。
・赤と緑の二色ドットマトリクスで三色を用いてリバーシを表現。(写真参照)

3-2.プログラムの構造・特徴

AI とプレイヤーの対戦の実装のみでなく、プレイヤー同士の対戦や AI 同士の対戦も可能とし、非常に遅いが PIC 上で自己学習を行うこともできる。学習している際、盤面を常に表示し、AI 同士で打ち合っている様子を観察することもできるようにした。また、プレイヤーとの対戦中、AI が探索をしている様子、打った手の評価関数の値を表示することもできる。

8×8 ドットマトリクスの性質上文字を表示することは難しいが、パスが発生した際、通知がなければ不親切である。そのため、8×5 サイズの文字を右から左に流して表示する機能を付けた。ドットマトリクスは約 61Hz のダイナミック点灯で表示している。

自己学習プログラムでは、AI にゲームのルールのみを教えてどこまでゲームの攻略法を学習するかを調べることが目的であるため、最終盤の読み切りは行わなかったが、実行機ではより強い AI を作ることが目的であるため、最後の 12 手を読み切る機能をつけた。また、自己学習プログラムでは探索に再帰関数を用いたが、PIC のコンパイラでは再帰関数を用いることができなかったため、goto 文とスタックの考え方を用いて再帰関数を再現した。コードが煩雑になることを防ぐため、普段は goto 文は用いていないが、今回はリバーシ AI に再帰関数の再現は必要不可欠と判断した。

3-3.回路の構造・特徴

小型化を図るため、配線には線径 0.29ｍｍのポリウレタン導線を、基盤はハーフピッチを、チップ抵抗(１ｋΩを除く)とチップコンデンサーを用いた。基盤は C タイプ(72 x 48ｍｍ)で IC カードより一回り小さいサイズに収めることができた。

この回路はマイクロ USB コネクタを二つ持っており、共に電源を接続することができる。そのほかの用途として、側面のものはプログラムを書き込む際パソコンとつなぐ役割を、上部のものは PIC の空きポートと接続してあるため拡張の役割を持っている。上部のコネクタにもう一つのドットマトリクスを載せた回路を接続すれば、テトリスなどの縦長の画面で操作するものも開発できる。また、ピンソケットが裏面にあり、そこに電源を繋ぐこともできる。

8×8 ドットマトリクスの表示にはダイナミック点灯で 16 本のピンが必要だがシフトレジスタを用いて 6 本に減らすことで、より多くのスイッチや EEPROM や拡張用ピンを PIC に接続できるようにした。その反面、最大表示変更速度は遅くなったが、リバーシの表示切替の頻繁は低いので許容範囲内であり、良しとした。スイッチのチャタリングには、RC 回路を用いてハード的に対策をしている。EEPROM は I2C で接続されていて、評価ボードや対戦履歴などを保存し、対戦中に何らかの原因で停止した場合や電源の切替えをした場合にも、途中から再開することができる。

3-4.問題と改善点

全ての配線を 0.29ｍｍのポリウレタン導線で行ったため、GND の引き回しが悪く、各点での電圧レベルが不安定になってしまった。作成当初はスイッチを押したときに稀に PIC がリセットされてしまうほどだったので、GND ラインを増やすことで強化し安定したが、GND の引き回しの重要性を再度確認させられる結果となった。

＜4.今回の活動を通じて ＞

今回 AI を作成して、より AI に対する興味が湧いた。今回作成した α-β 法にランダム探索を加えた、モンテカルロ木探索によるリバーシ AI の作成を始めている。また、ルールが比較的複雑なゲームであるチェスや将棋の AI 作成に挑むために、リバーシ以外のゲーム AI としてコリドールの AI を製作しようと考えている。

また、製作をするにおいて、プログラムエラーや配線のミス、原因不明の不可解な挙動など、たくさんの障害があった。これは回路やプログラムを作るにあたって、毎回ぶつかる壁であるが、それを乗り越える度に大きな達成感があり、毎回、諦めないことの大切さを学ばせてくれる。今回も様々な場面で悩まされたが、結果的に自分を成長させることができたと思う。

＜5.その他 ＞

5-1.動作・開発環境

Android：AQUOS R

PC：Windows10 Intel(R)Core(TM)m5-6Y54@ 1.10GHz 8.00GB

PIC：PIC18F26K22 64MHz(16MHz4 逓倍)

5-2.参考とした資料・サイト

http://uguisu.skr.jp/othello/5-1.html

http://bassy84.net/othello.nakawari.html

https://ja.m.wikipedia.org/wiki/%E9%81%BA%E4%BC%9D%E7%9A%84%E3%82%A2%E3%83%AB%E3%82%B4%E3%83%AA%E3%82%BA%E3%83%A0(遺伝的アルゴリズム)