航空宇宙システム学第二
最終課題レポート

03-170355 航空宇宙工学科 3 年 眞木俊弥 *

2017 年 7 月 30 日

1 概要

今回の最終課題では，世界的に有名なボードゲームであるリバーシをプレイできるソフトウエアを開発した。宇宙工学入門や空気力学で科学計算を行い，その結果をグラフなどで描画するソフトウェアはいくつか開発してきたので，今回は科学的なシミュレーションとは関係のないプログラムを作ろうと思ったことが開発のきっかけである。ただ，オリジナルのゲームを製作するのはアイデアを考えることに時間を取られるため，シンプルだが奥が深く，なおかつ CUI で表示が可能なリバーシを選んだ。また，リバーシの AI の研究は歴史が深く，アルゴリズムをインターネット上で探しやすかったことも理由にあげられる。

2 プログラムの解説

提出するプログラムの概要を解説する。なお，全てのクラス，関数には docstring をつけているので，Sphinx を用いてドキュメンテーションも作成した。詳細はそちらを参照していただけると幸いです。LaTeX を使用した PDF バージョンは，

./reversi_documentation.pdf

に掲載しました。なお，もともと html ドキュメント用に書いたため，少々体裁が崩れてしまっています。正式な html 形式のドキュメントは，ファイルパス，

./_build/html/index.html

にあります。こちらの方が見やすく，検索機能も備わっています。

2.1 play_reversi.py

これを実行すると，mac の terminal.app のようなシェル上でインタラクティブにリバーシをプレイすることができる。黒石，白石の種類を人間/AI の中から番号で選ぶとゲームが始まり，人間は次に打つ場所のマスの位置をタイプすることでゲームを楽しむことができる。

* email: t.maki.0402@gmail.com

使用する AI は一定のインターフェースを持つ Python モジュールを追加することで拡張可能である。(後述)

2.2 reversi.py

リバーシのゲームの進行に必要なクラスを収めたプログラム。

- reversi.StoneColor
- reversi.Board
- reversi.Reversi

の 3 つのクラスが実装されている。

ヒューマンインターフェースや AI の実装とは切り離されているので，リバーシのゲームをプレイする目的だけではなく，強化学習などのために AI 同士を対戦させる用途など，リバーシに関連した他の用途で使用されることも想定している。そのため，今回は使用していないが，オプションとして棋譜を出力することも可能である。

また，Reversi クラスにおいて，ゲームの進行において次の手を選ぶプログラムは，プレーヤーが人間である場合も含めて別に用意する必要がある。詳細は Reversi クラスの docstring の説明である以下の引用を参照のこと。

> リバーシのゲームを実行するクラス。
> プレーヤを担当するプログラムは別にモジュールとして与える必要がある。そのモジュールは，putStone という引数を一つ受け取る関数を含む必要があり，その関数は引数として reversi.Board クラスオブジェクトを受け取り，盤面を評価し，次の手を決め，返り値として返す必要がある。返り値は，置いた位置の座標を表すタプル (例えば，E6 を選んだならば，(5, 4) のように 0 7 までの int 型の整数二つの組) を返し，パスを選んだ時には，None とする必要がある。(Tips: 位置を表す整数値のタプルは，reversi.Board.posName2Position メソッドに位置の名前の str 型文字列 (例”E6”) を渡すことで得ることができる)
> Atrributes:
> - board (Board) 盤面のオブジェクト
> - black_player_type (module) 黒石のプレーヤーの種類
> - white_player_type (module) 白石のプレーヤーの種類

2.3 プレーヤのプログラム

2.2 節で解説した，プレーヤーのプログラムを説明する。なお，プレーヤはルールに反する手は打たないことを前提としている。

2.3.1 human_interface.py

人間がキーボードで次の手を選ぶことによって，次の手を選ぶ。対話的インターフェースを提供する。

2.3.2 ai_random.py

盤面上の打つことができる位置にランダムに石を置く AI。

2.3.3 ai_static_weighting.py

盤面の位置に点数を割り当て，その時打てる位置の中で最も点数が高かったものを返す AI。重み付けについては，[1] を参考にした。先読みはしない。

2.3.4 ai_minimax.py

ミニマックス法で先読みし，その時打てる位置の中で最も評価が高い位置に石を置く AI。アルゴリズムは，[2] と，[3] を参考にした。評価方法としては， 2.3.3 節でも使用している重み付けと，四隅に石が固まって置かれていると評価を高くする方法を併用している。

2.4 roundRobin.py

前節で紹介した AI を ai_random.py 相手に複数回対戦させ，強さを見るためのプログラム。進捗を表示するために，tqdm[4] を用いている。100 回対戦させた結果を表 1 に示す。

この結果から，ミニマックス法は，ランダムに対しては圧倒的に有利になっていることがわかる。また，全体の傾向として，白の方が有利であることがうかがえる。

表 1　AI とランダム AI の対戦結果

AI の種類	手番	勝利数	敗北数	引分数
ai_static_weighting.py	黒	79	18	3
ai_static_weighting.py	白	96	4	0
ai_minimax.py	黒	95	5	0
ai_minimax.py	白	97	3	0

3 アピールポイント

今回は，題材としてはリバーシという典型的なものを扱ったが，拡張性を持たせることを強く意識して開発した。特に，リバーシの根幹部分の reversi.py は，オブジェクト指向プログラミングを徹底し，汎用的なモジュールとして使いまわせるようにした。また，Python のインタープリター言語としての特徴を生かしたモジュール化も行った。

また, 可読性や可搬性を意識して, 全てのオブジェクトに Google Style Guide に沿った docstring をつけた。そこで，Python 用のドキュメントジェネレータである Sphinx[5] を利用して，html ドキュメントを自動生成させたものもソースコードと共に提出する。

参考文献

[1] オセロ（リバーシ）の作り方（アルゴリズム） ～石の位置による評価～ Retrieved July 30, 2017, http://uguisu.skr.jp/othello/5-1.html

[2] オセロゲーム開発 ～ミニマックス探索法～ Retrieved July 30, 2017, http://uguisu.skr.jp/othello/minimax.html

[3] リバーシの評価関数について Retrieved July 30, 2017, http://www.info.kindai.ac.jp/ takasi-i/thesis/2012_09-1-037-0133_S_Shiota_resume.pdf

[4] tqdm Retrieved July 30, 2017, from https://pypi.python.org/pypi/tqdm

[5] Sphinx Retrieved July 30, 2017, from http://www.sphinx-doc.org/ja/stable