夏のプログラミング･シンポジウム「プログラム詠み会」2015.9.4-6

2048 プレイヤの実装

山口 文彦[1,a)]

概要：夏のプログラミング・シンポジウム 2015 のプロソン (Programming Marathon) に参加し，寺田先生ご提案の「対戦型 2048」のプレイヤを実装した。このゲームは二人完全情報ゲームなので，α-β 探索を実装した。自手番・相手番のそれぞれを 1 手と数えての 7 手読みとした。実装に用いた言語は C 言語で，コードの行数は 600 行ほどである。盤面の評価は，守備側の評価として，2048 ゲームのスコアが高いことと，大きな数字を盤面の端に寄せることを高い評価とした。攻撃側は，この評価が低くなるように手を選ぶ。結果として，夏プロ期間中に行われた対戦では好成績を収めることができた。

キーワード：2048, $\alpha\beta$ 探索, 評価値

1. はじめに

2048 はパズルゲームである [1]。4×4 の盤面の空いているマスに，ランダムに 2 または 4 が配置される。プレイヤは数を上下左右の一方の端の寄せる (数の記されたタイルが滑り落ちていくイメージである)。このとき同じ数が二つ隣り合うと，くっついて 2 倍の数のマス一つに変わる。したがって盤面の各マスには，2 のべき乗数が入るか，または空となる。数がくっついて大きな (2 倍の) 数になった際に，出来た数がスコアに加算される。4 方向のどの方向に寄せても盤面が変化しなくなったらゲームオーバーである。

2015 年 GPCC 課題として，この 2048 を対戦型にする提案が寺田先生からなされた。対戦型では，盤上に 2 または 4 を配置する側を攻撃，通常のパズルゲームと同様に盤上の数を寄せる操作をする側を守備と呼ぶ。二人のプレイヤは攻守を交替してプレイし，それぞれが守備側で得たスコアによって勝敗を決める。詳しくは GPCC の報告ページを参照されたい。

[1] 長崎県立大学
[a)] yamagu@sun.ac.jp

2. 実装

プロソン期間中に対戦型 2048 のプレイヤを実装した。寺田先生ご提供のゲームサーバと TCP を介して通信する。アルゴリズムは α-β 法によるゲーム木探索である。言語は C を用いた。コード量は 600 行ほどであり，そのうち 70 行ほどは TCP 接続のためのコードである。

盤面の表現は，4×4 の `unsigned char` の配列とした。空マスを 0 で表し，2^n の値が入るマスを n で表した。これによって 2^{255} までの数が表現できるが，現実的にはゲーム中にそこまで大きな数を作ることができないため，$2^{13} = 8192$ 程度まで表せれば十分なようである。

当初，サーバから盤面の情報を渡されてから手 (守備側は寄せる向き，攻撃側は置く数と位置) を返すまでに，100ms 以内という仕様だった。その後，この制限時間は緩和されて，事実上無制限となった。実行環境は，CORE i7 2.4GHz，メモ

リ 8GB，Windows 7 上の VMplayer で Ubuntu 14.04 (64bit) を動かしている。コンパイラは gcc 4.8.4 である。この環境で現実的な時間内に可能な深さの探索として，7 手読みを行なった。

序盤 (特に最初) は選択肢が多いので探索に時間がかかり，終盤は選択肢が減って同じ深さの探索にかかる時間が短かくなるが，ゲームの進捗による読みの深さの調整は行なっていない。

2.1 評価関数

守備側のスコアによって勝敗が決定するので，スコアを評価値の基準とした。また，2048 のパズル (守備側) では大きな数を角や辺に置くことが，少なくとも人間にとっては，ゲームを進めやすいという知見がある。そこで，4 箇所の角にある数 (の 2 を底とする対数) を 10 倍，角以外の辺にある数 (同じく対数) を 5 倍し，スコアの 1000 倍に加えて評価値とした。スコアの重みが大きいので，同じスコアの盤面を比較したときのみ，大きな数が周に配置されていることが意味を持つ。

3. まとめとその後

夏プロ 2 日目の夜に行われた対戦では，三好さん・八木原さんのプログラムに勝つことができた。ただし，このうち三好さんのプログラムは 4 を置かない仕様だったので，公平な戦いだったとは言えない。両者とも 4 を置かない仕様で事前に行なった対戦では三好さんのプログラムに敗けている。三好さんのプログラムはモンテカルロ木探索と聞いている。

プロソンに際しての個人的な目標として，GPGPU で遊んでみることを掲げていたが，CUDA ライブラリのインストールに手間取り，GPU を使った実装には至らなかった。高性能なグラフィックボードを備えた大きくて重いノート PC を会場に持ち込んだものの無駄になってしまった。

表 1 評価関数の異なるプレイヤの対戦結果

	M.M.P.4	評価関数 1	評価関数 2
M.M.P.4	1044	1316	1320
評価関数 1	5052	2800	2592
評価関数 2	6608	5172	2284

表 2 評価関数ごとの実行時間と呼出し回数

		実行時間 (ms)	評価関数の呼出回数 (回)
評価関数 1	カット有	383	19,628,228
	無	44,636	2,802,218,688
評価関数 2	カット有	487	24,953,572
	無	41,663	2,802,218,688

夏プロ終了後にも評価関数について考察したので報告する。隣合う数の差が小さいことは (守備側にとって) 有利である。そこで，隣合うマスの両方ともが空でない場合に数 (対数) の差の絶対値を，スコアの 1000 倍から減算する評価関数を作った。盤面の周のマスには隣のマスが少ないため，周に大きな数を置いても評価値が小さくなりにくい。そのため，この評価関数でも大きな数を周に置く指向が表現される。

2.1 節で説明したものを評価関数 1，隣合う数の差を減算する評価関数を評価関数 2 とし，寺田先生の MinMaxPlayer4 を対戦相手に加えて対戦した結果を表 1 に示す。MinMaxPlayer4(表 1 では M.M.P.4 と表記) は，守備側の手番で数えて 2 回まで読む (自手番・相手番の両方を数えると，守備側で 4 手読み，攻撃側で 5 手読みに相当する)。評価関数はスコアのみとなっている。表 1 の表側は守備側のプレイヤ，表頭は攻撃側のプレイヤで，守備側が獲得したスコアを示している。表 1 から，おおむね評価関数 2 が評価関数 1 よりも高いスコアを得ていることが分かる。

また，評価関数ごとに，初期盤面 (すべてのマスが空) の攻撃側の探索に掛かる時間と評価関数の呼出し回数を，表 2 に示す。比較のため，α-β カットをしない場合についても計測した。この表から，実行時間に大きな違いはないものと考えられる。

謝辞 サーバだけでなく，テストに適切な対戦相手となった MinMaxPlayer を提供して下さった寺田先生に感謝します。

参考文献

[1] `http://gabrielecirulli.github.io/2048/`, accessed Oct.20, 2015