人工知能 課題番号 17「ゲームのモンテカルロ木 (UCT) 探索を用いたゲームの探索プログラム」

工学部電子情報工学科 03-175001 浅井明里

2017 年 11 月 30 日

1　3 次元 Tit-for-tat におけるモンテカルロ探索法と minimax 探索法の対戦

本課題では 4 x 4 x 4 という三次元における Tic-for-tat ゲームでモンテカルロ探索法と minimax 探索法を対戦させ、探索の効率 (展開されたノードの数) 及び勝敗数を元に比較を行なった。プログラムについては授業及び講義サイトで紹介されている、3 次元 Tit-for-tat（4x4x4）のプログラム（その 2）の評価関数を拡張するなどの実装を行なった。

1.1　minmax 法の評価関数の改善

今回は以下の二つの評価手法によりある手を売った場合の盤面の評価値を算出し、$\alpha\beta$ を導入した minimax 法により最善手を求めた。

- 評価関数 1：盤面のそれぞれのマスを重み付けし、その時点で自身が取得しているマスのスコアの合計を評価値とする。
- 評価関数 2：現時点の盤面から自身が勝利する、実現可能な path の数から相手にとって実現可能な勝利 path 数をひき、定数倍したものを評価値とする。
- 評価関数 3：盤面のそれぞれの行、列に対して、いくつ自身で埋められているかに応じてスコアを与え、合計を評価値とする。

1.1.1　評価関数 1

この評価関数ではそれぞれの局面の盤面のそれぞれのセルを順に見ていき、そのセルが敵の駒で埋められていればそのセルにつけられたスコアを -1 倍したものを、自分の駒で埋められていればそのセルに割り当てられたスコアを足していき、その合計値を最終的な評価値とする。

評価関数は minimax.c の get_evaluation_value() という関数で定義されている。

1.1.2　評価関数 2

この評価関数では「あといくつ勝利するためのパスが残されているか」を数え上げ、その合計に 100 をかけて評価値とする。具体的にどう「勝利可能なパス」を求め、また評価値を算出するかを図 1 に示す。例えば「O」の勝利するために実現可能な path を考える。図 1 左上の盤面では、中央に「O」があるのみであるため、まだ全ての行、列及び対角線の path のいずれかで「O」が埋め、勝利をする可能性がある。よって図 1 左上の盤面での「O」にとって勝利可能なパスの数は 8 通り、同様にして、右上の盤面では「O」のすぐ上を「X」が埋めてしまっているため、勝利可能なパスの数は 6 通りに減少している。評価値はこれと同様の計算を相手側についても行い、

$$\text{評価値} = (\text{(自分の勝利可能なパスの数)} - \text{(相手の勝利可能なパスの数)}) \times C,\ C \text{ は自然数}$$

で算出する。例えば図 1 の右下の盤面では、「O」は 4 通りの勝利パスが、「X」は 6 通りの勝利パスが存在するため、この盤面の評価値は

$$(4 - 6) \times 100 = -200$$

となる。

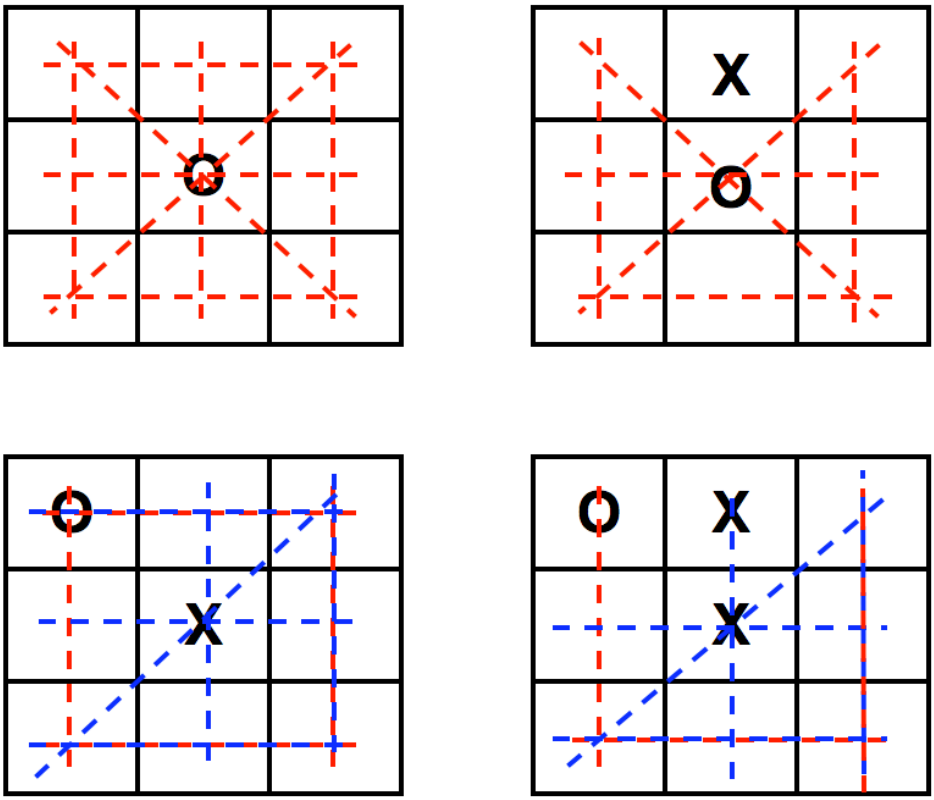

図 1　勝利 Path の計算

評価関数は minimax.c の get_evaluation_value_heuristics() という関数で定義されており、この関数の内部でヘルパー関数として、いくつまだ実現可能なパスが残されているか計算する get_possible_win(koma_type mycolor, const koma_type board[HEIGHT][AREA]) を呼ぶ。この関数ではあるパスについて相手が駒を置いていないかチェックし、まだ相手側が駒を置いていなければ勝利可能パスとして登録する。

```
// 4. Check the vertical among different hights. MAX = AREA(16)
for (j = 0; j < AREA; j++) {
  int tmp = 0;
  for (i = 0; i < HEIGHT; i++) {
    tmp = (board[i][j] != REVERSE(mycolor)) ? tmp + 1 : tmp;
  }
  if (tmp == 4) {
    possible_wins += 1;
  }
}
```

三次元 Tic Tac Tow では、ある一つの状態につき複数の盤面が存在するため、それぞれの盤面に登場する勝利パスを足し合わせ、最後に相手側の勝利パス数の合計をひいいた上で定数項をかける。

1.1.3 評価関数 3

この評価関数では、それぞれの局面の盤面の行、列の埋まり具合から「全て埋めることができた」「自身がすでに 3 つ埋めていて、残り一つのセルが空白である」「自身が 2 つ埋めている」「相手が 3 つ埋めていて残り一つが空白である」「相手が全て埋めた」などのケースそれぞれに一定のスコアを付与し、その合計を評価値とする。スコアは以下のように、どちらかが全ての列もしくは行を埋めた場合の評価値をより絶対値を大きくなるように設定している。

```
int convert_cell_num_to_score(int enemy_cell_num, int my_cell_num,
                              int empty_cell_num) {
  if (my_cell_num == 4) {
    return 2000;
  } else if (my_cell_num == 3) {
    return 600 + 50 * empty_cell_num;
  } else if (my_cell_num == 2) {
    return 100 - 100 * enemy_cell_num + 50 * empty_cell_num;
  } else if (my_cell_num == 1) {
    return -600 + empty_cell_num * 50;
  } else {
    return -2000;
  }
}
```

評価関数は minimax.c の get_evaluation_value_dynamic() という関数で定義されている。

2 結果の分析

2.1 探索効率の比較

探索効率については、UTC 法とそれぞれの評価関数を用いた minimax 法を対戦させ、一つの手を求めるためにかかった時間を計測した。図 2, 3, 4, 5 はそれぞれ UTC 法、評価関数 2 を用いた MiniMax 法 ($\alpha\beta$枝刈るを行なったもの), 評価関数 3 を用いた MiniMax 法 ($\alpha\beta$枝刈るを行なったもの), 評価関数 4 を用いた MiniMax 法 ($\alpha\beta$枝刈るを行なったもの) のそれぞれの手を求めるためにかかった時間をゲームごとに平均したものをプロットしている。UTC 方についてはステップアウト数の増加とともに計算時間が増加する傾向があるが、MiniMax 法では必ずしもこの傾向が観測できるわけではない。また評価関数 3 については他の二つの評価関数より計算時間が長くなっている傾向にある。これは評価関数 3 が他の二つの関数より評価値のばらつきが小さく、枝刈りの効率が悪いからではないかと推測する。また、UTC 法は MiniMax 法の手法よりも計算時間が短く済んでおり、効率的に探索できていると言える。

2.2 勝率の推移

図 6 に示した二つの表は、評価関数をそれぞれ変えた MiniMax 方に対する UTC 法の勝率ををステップアウト数ごとに一覧にしたものである。表の単位は % である。UTC 法はステップアウト数が 10000 程度までの時には MiniMax 法に全体として劣っているが、ステップアウト数が 100000 では UTC 法が圧倒的な勝率を見せている。評価関数 2 が先攻の際と後攻の際で勝率が大きく異なっているが、これは「勝てるパスを探す」という評価の仕方が、先攻の法が残り勝利パスが大きくなりやすいために、有利に働くからではないかと

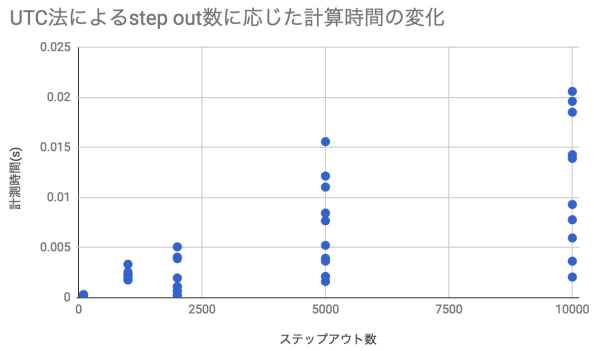

図 2　UTC 法での計算時間

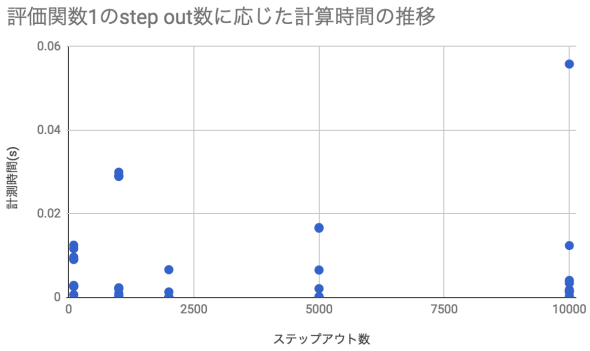

図 3　評価関数 1 を用いた MiniMax 法での計算時間

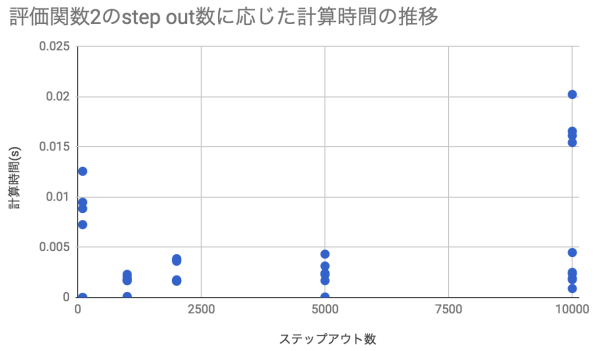

図 4　評価関数 2 を用いた MiniMax 法での計算時間

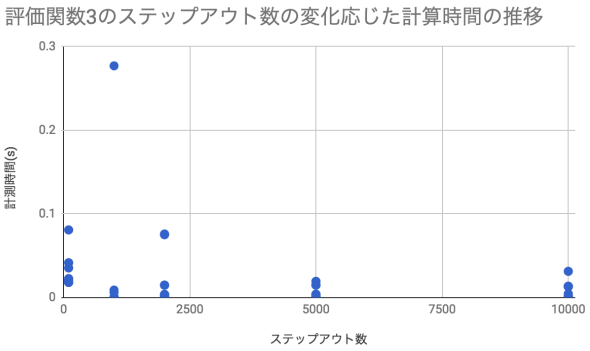

図 5　評価関数 3 を用いた MiniMax 法での計算時間

推測する。一方で、そもそもステップアウト数が 10000 の時点でのモンテカルロ法の強さは初期値に依存するために、実験時のモンテカルロ法のプレーヤーが「良くない」初期解を引いたという可能性も否定はできない。

参考文献

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. ”Introduction to Algorithms”, MIT Press, 2009.

[2] 伊庭斉志, 『人工知能と人工生命の基礎』, オーム社 , 2013.

UTCが先攻の場合の各評価関数を利用したMiniMax法との勝率

評価関数	100	1000	10000	100000
1	0	5	18	90
2	0	0	7	90
3	0	5	70	100

UTCが後攻の場合の各評価関数を利用したMiniMax法との勝率

評価関数	100	1000	10000	100000
1	0	6	16	80
2	0	12	56	100
3	0	7	61	100

図 6　UTC 法の勝率の変化