モンテカルロ法の彩について

山下 宏

モンテカルロ法の衝撃

1996年 囲碁プログラム「彩」を作りはじめる
2007年 GnuGoがKGS6級、彩は8級
2007年 10月にモンテカルロ法に移行
2008年　2月、モンテカルロ法の彩がKGSで3級に

12年かけてGnuGoにすら届かなかったのが
4ヶ月でGnuGoを遥かに追い越した！

従来の人間の思考の真似をしていたプログラマには大ショック

モンテカルロ法を囲碁に適用すると？

1. 全ての着手可能な場所に乱数で石を置く。
2. 1.を白黒交互に繰り返す。打つ場所がなくなってパスが2回続けば終局。
3. 点数を計算する。
4. 1. - 3. を何度も繰り返して点数の平均値を求める。

本当に乱数で打っていくと終わらないため、「眼」には打たない、というルールを
付け加える。

目数差ではなく、勝率、を評価基準に使う

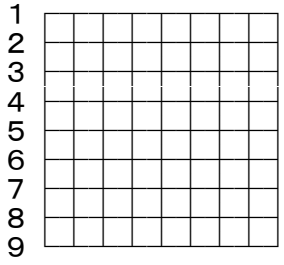
```
  1回試して 15目勝った、(+15)
  2回目は    20目負けた、(-20)
  3回目は    81目勝った  (+81)  ... 大きく勝ちすぎている！
-----------------------------------
                  合計 +76   平均 +76/3 = +25.3目勝ち
```

といった目数差を平均するより

```
  1回試して 15目勝った、勝ち +1
  2回目は    20目負けた、負け +0
  3回目は    81目勝った  勝ち +1
-----------------------------------
                       合計 +2   平均 +2/3 = 0.66、勝率 66%
```

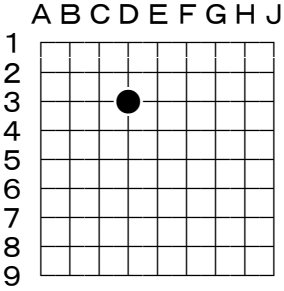
というように、勝率で評価する方が安定した評価ができる。

どのくらい正しく局面を評価できるか？



九路盤で初期局面で10000回モンテカルロ法を繰り返した場合、
先手(黒)の勝率は？　（コミ６目半で）

純粋乱数　　勝率43.6% ... 先手の利を生かしきれてない
パターン利用 勝率47.9% ... 50%に近づく



ハンデとして黒石をＤ３に置いて、黒から打ち始めると？

純粋乱数　　勝率49.1% ...　上がってはいるが　　+5.5% と小さい
パターン利用 勝率59.1% ...　かなり勝率が上がる。+11.2%
　　（もっと強くなれば100%に近づくはず）

Ｄ３に置いた場合の、地になる確率
（+100で黒地になる確率が100%、-100で白地になる確率が100%）

39	45	54	55	50	39	28	23	21
35	43	55	63	54	38	28	21	19
27	31	46	79	45	37	23	19	20
19	25	31	39	29	28	26	21	24
11	13	21	17	15	21	39	23	17
5	6	9	6	7	10	16	11	13
-1	1	2	-5	-2	4	11	9	10
-3	-2	-5	-4	-2	0	1	5	7
-5	-7	-2	-2	-3	-2	1	2	3

D3の左上は黒地になる確率が高い

遠くなると影響が小さくなる

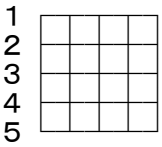
実際に、純粋乱数とパターン利用を対戦させると
100回対戦させて99回パターン利用が勝つ。（九路盤、1000playoutで）

3x3のパターン

・プロの棋譜1万局から収集
・何回その形が出現したか、と実際に次に打たれた回数で、着手確率を求める。

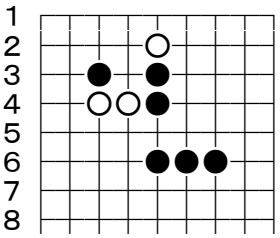
例：＋＋○ この形はで中央に●を置く確率は
　　●＋● （ 62 / 71 = 0.873 ） 87%の確率
　　○○● この形は1万局の中で71回出てきて、そのうち62回はすぐに打たれた。

彩では対称性を無視すると、 3x3 で 3^8 = 6561 通り、に
盤端の状態を含めて x25、 さらに8マスで直前に石が打たれたかどうか、で x9
これに白黒用で x2 、合計 6561 x25 x9 x2 = 295万パターン、を保持してます。



周りが全部中央(C3)。上だけ盤端(C2)、上と右が盤端(D2)
という感じで5路盤のどこを中心に3x3を取るか、
という感じで25通りを保持してます。

3x3のパターンの部分更新は、 1手ごとに
・自分が打った位置の周り8箇所（確率が高くなる）
・相手が直前に打った場所の周り8箇所（高い確率を戻す）
と最大16箇所更新しています。（他にも連が取れる場所などを高得点に）



黒石を置く確率（合計しても100%にはならない）

4	4	4	14	22	14	4	4	4
4	100	30	2709	0	1760	42	42	4
4	835	0	3565	0	108	228	42	4
4	1627	0	0	0	4	228	42	4
4	127	27	803	108	15	4	96	4
4	42	228	32	0	0	0	30	4
4	42	228	195	4	4	4	100	4
4	42	42	42	42	42	42	42	4

playoutでの工夫

playout ... 乱数で手を選ぶ、を繰り返して終局まで1回打ち切ること

3x3のパターンですべての手に確率をつける
当たりにされた石が逃げる確率を上げる
石を打ち上げる手の確率を上げる
簡単なシチョウで石を殺せる場合、確率を上げる
石を打った直後にダメが1になって4子以上取られるなら別候補を探す
パスは打つところが完全になくなった場合だけ選ぶ。
自殺手はいったん、確率を0にして別の手を捜す。

UCB (Upper Confidence Bound. 信頼上限)

UCB = その手の勝率 + α * sqrt(log(すべての手を試した回数) / その手を試した回数)

その手を100回試して67回勝った場合、「その手の勝率」は0.67
すべての手を試した回数が1200回ならば

UCB = 0.67 + α * sqrt(log (1200) / 100)

がその手のUCBの点数になる。UCBが一番高い手を選ぶ。
試した回数が少ないうちは誤差？が大きいので選ばれやすい。
回数が増えると勝率が高い手が選ばれる。
浅い読みで良さそうな手を優先的に深く読む最良優先探索のような感じになる。

α : 0.3-1.2 ぐらい。実験で決める。
log の底は自然対数 e

UCT (Upper Confidence bound for Tree)
UCBを各深さで行う。最良優先のような探索になる。

UCTのサンプルコード
http://www32.ocn.ne.jp/~yss/uct_aya.cpp

9路盤と19路盤での違い

9路盤では枝刈しなくてもそこそこ強いものが作れる。
19路盤では可能な手の数が多いため「手をしぼる」作業を行わないと
あまり強くできない。（深く読めない）

19路盤では以下の方法で確率の高い手から上位20手程度だけを探索している。

・3x3のパターン
・盤の端からの距離での確率
・直前の相手の手からの距離
・2手前の自分の手との距離
・連の死活
個々の連に対して深さ7手の死活探索を行って生死を判断する
「死んだ」石が動き出す確率を極端に下げる
---> 19路では効果的。9路では効果なし。
---> 19路では部分問題に切り分けがモンテカルロでも有効か？
・5x5や7x7のパターン

筋のいい手を打ちやすいがあまり効果がなかった。

候補手の数は読みが深くなるにつれ少しずつ増やしていく

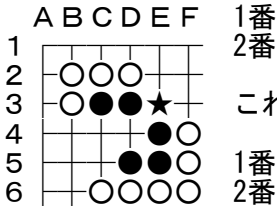
playoutを正確にしていく事が重要か？
Crazy Stone、彩
UCTの木をどんどん深くしていく方法が重要か？
MoGo

私はplayoutを強くすることが重要ではないかと思ってます

データ構造

彩のデータ構造は下のような感じです。
・ダメの情報（位置、数）を常に正確に保持する。
・石の数
・連同士が融合したとき、小さい方の連は大きい連番号を参照するようにする

特徴としては連構造体の中に「仮の連番号」というのを持っていて
たとえば下図で、★に黒が打った場合に、



1番の連（C3）仮の連番号 1
2番の連（E4）仮の連番号 2

これが★に打った場合、

1番の連（C3）仮の連番号 2
2番の連（E4）仮の連番号 2

というようにしてC3にある石の連番号を求める時に、仮連番号を
たどっていく、という構造にしています。

ただ、ダメの位置、を正確に持つために大きな石同士が融合するときは
かなり重いです。

Core Duo1.83GHz(シングルスレッド)で初手で打つ速さは
9路盤　40000po/秒　平均 77.5手 パターンなし。データ構造のみ(ルール無視で81回石を置くだけ)
19路盤　9300po/秒　平均354.4手 パターンなし。データ構造のみ
9路盤　7890po/秒　平均 99.8手 playoutのみ。6270po/秒 UCTあり
19路盤　1830po/秒　平均422.4手 playoutのみ。1660po/秒 UCTあり

連のデータ構造

```
#define KOKYU_MAX 229  // この数以上ならば呼吸点位置は無視（計算上は19路で229個）
#define ADJACENT_MAX 10  // 19路で理論上は132個？隣接する敵の連

typedef struct {
 int kari_ren;  // 仮の連番号
 int ishi_num;  // 連の石の数
 int col;   // 連の色（ある方が絶対に便利なので）
 int start;   // 開始石番号
 int end;   // 終了石番号
 int kokyu_num;  // 呼吸点の数
```

```
 int exist;   // 盤上に存在していれば1、取られたら0
 short kokyu_iti[KOKYU_MAX]; // 呼吸点の位置
 int   adjacent_num;    // 隣接する敵の連の数
 int   adjacent[ADJACENT_MAX]; // 隣接する敵の連の番号(重複を許可。石が取られたときに削
ってない) 不正確！
} STRING;
```

連の石の位置は int ishi_list[BOARD_SIZE] という配列を一つ持って、
C3の黒石の隣、C4に黒を打った場合、ishi_list[C3] = C4, ishi_list[C4] = NULL
というようなリスト構造にして連の構造体には最初の座標だけを持つようにしています。
石をたどるときは、このリストをたどっています。

後は彩も3x3のパターンを使って、1手ごとに白と黒の確率分布を
更新しているのですが、これもかなり重いです。
後はplayoutでは石を戻さずに、最後まで打ったら次回にはコピーしなおしています。

現在の棋力

KGS(無料の囲碁サーバ、GTPが使えれば簡単に接続できる)で3級
日本の棋力になおすと初段程度？

AyaMC 3k (2.9k) 160000playout (Xeon   2.66GHz 8core、1手8秒ほど)
AyaMC2 6k (5.5k) 10000playout (Opteron 2.2 GHz 2core、1手2秒ほど)
対局条件：持ち時間10分切れたら30秒。ただし30秒x5回の考慮時間

16倍時間をかけると約2.6kほど棋力が上がる。

GnuGo   5k
彩(クラシック) 8k

銀星囲碁7との対戦成績
28勝5敗 勝率 0.84 (30000playout x8 = 240000 po)
http://www.yss-aya.com/bbs_log/bbs2008.html#bbs91

GnuGoとの成績(gnugo 3.7.10 Level 10)
19路
10000playout 44勝107敗 勝率 0.291
9路
 1000playout 587勝413敗 勝率 0.587

--- 連続 GNU Go 3.7.10 - Aya 6.19d, mt=1, 2007-10-13
playoutを増やした時の勝率の変化

0.613   10000po
0.723   20000po
0.731   40000po
0.813   80000po
0.813 160000po
0.852 320000po
0.898 640000po
0.888 1280000po

ランダム（少しアタリ逃げ確率アップあり）playout ＋ UCT
plyaout数 約 10000 playout/秒

5-yamashita
123勝177敗 勝率0.41 思考時間 1局 11分 (Opteorn 2.2GHz)
147勝153敗 勝率0.49 思考時間 1局 17分
168勝132敗 勝率0.56 思考時間 1局 40分
114勝 56敗 勝率0.67 思考時間 1局100分
http://blog.livedoor.jp/yss_fpga/archives/50287760.html

気づいた事
・モンテカルロ法とはいえ、死活を間違うと致命的に間違う。
・隅の死活はダメ。
・連絡や石の強さ、中で生きられない厚み、などの微妙な判断は優れてる。
・playoutで死活を判断した手を打つ必要がある。
　例：次に三目中手で活きますよ、と打たれたら中手を打って殺さないとダメ

サンプルコード

19路盤で乱数分布から1手を選ぶ方法
http://www.yss-aya.com/bbs_log/bbs2007.html#bbs337
マルチスレッドでの乱数の失敗（乱数の変数はスレッドごとに持たせる）
http://www.yss-aya.com/bbs_log/bbs2007.html#bbs336

モンテカルロ法で囲碁、将棋（こちらを最初に読むと分かりやすいと思います）
http://www32.ocn.ne.jp/~yss/monte.html
モンテカルロ法の彩について（今回の内容そのままです）
http://www32.ocn.ne.jp/~yss/cgf20080412.html
