# モンテカルロ碁におけるプレイアウトからの情報抽出

橋本千裕† 村松正和†

† 電気通信大学情報工学専攻

本稿では, モンテカルロ碁における新しいアプローチを提案する. 従来のモンテカルロ碁では勝率以外の情報は得ていない. それに対し我々は, プレイアウトから連の生存確率や 2 つの連間の結合確率などの情報を得た. 特に, プレイアウトをより意味のあるものにするためにのいくつかの確率的指標を提案する. GNU Go とのテスト対局から, 我々のアプローチは有望であるという結果を得た.

# Extracting Information from Playouts in Monte-Carlo Go

Chihiro HASHIMOTO† Masakazu MURAMATSU†

† Department of Computer Science, the University of Electro-Communications

This paper presents new approaches in Monte Carlo Go. The traditional Monte Carlo Go program extracts no information from playouts except winning rates. In contrast, we propose to derive more information from playouts, by calculating, e.g., probabilities that each string is alive and that each pair of strings is united at the end of the game. In particular, we propose an index based on some conditional probability to control playouts to have more reality. Numerical experiments against GNUGo shows that our approach is promising.

## 1 はじめに

コンピュータ囲碁の分野では, 従来の知識ベースのプログラム [1] に代わり, モンテカルロ法を用いたプログラムが流行している. これらのプログラムは UCT[4] やパターンなどを組み合わせることで, 飛躍的に進歩を遂げてきた ([5],[6]). 実装の容易さや囲碁の知識を必要としないことも手伝って, モンテカルロ法を用いたプログラムが徐々に増えてきている.

モンテカルロ法は, 乱数を用いることで近似値を得るアルゴリズムとして有名であるが, 最初に Brugmann[9] によってモンテカルロ法を用いた囲碁プログラムに関する研究が行われた. その有効性は Bouzy ら [2][3] の研究によって示されたが, 初期のモンテカルロ碁は今日注目されるほどの棋力はまだなかった. 近年 Crazy Stone[5] や MoGo[6] といった強い囲碁プログラムが出現している. 中でも Crazy Stone はアマチュアの 3 段程度だと言われている ([10]). モンテカルロ法は現状では最も有効な手法であると考えられている.

通常の UCT を用いたモンテカルロ法では, 木探索部分とランダムシミュレーション部分に分かれる. 木探索部分では UCB 値に従うため, 数多くのプレイアウトを行いながらも勝敗以外の結果は得てない.

本研究では, 勝率以外の情報の利用可能性に着目した. プレイアウトから連 [1] の生存確率と連間の結合確率を得て, それらの情報を基に囲碁プログラムに応用する手法を提案する. GNU Go[8] とのテスト対局を行った結果, 単純な手法よりも勝利数が向上する結果を得た. また, それ以外の手法として RAVE[6] と組み合わせた場合についてもテスト対局を行ったが勝利数の目立った向上は得られなかった.

本論文の構成は以下のようになっている. 2 章では抽出した情報について, 3 章では囲碁プログラムへの適用について説明する. 3 章でプレイアウトの制御とテスト対局を行い, 4 章でまとめと今後の課題を述べる.

## 2 プレイアウトからの情報抽出

囲碁において, 勝敗を大きく左右する要素として, 連の死活や連間の連結がある. ここで連とは「縦横に繋がった同じ色の石の集合」である ([1]). 本研究では, 従来のモンテカルロ法を用いたプログラムを使用し, 勝率以外の情報として連の生存確率と異なる 2 つの連間の結合確率について抽出を試みた. また, それらの情報を基に連の生死と勝敗への影響度を数値化, 連のグループ化を行った.

### 2.1 連の生存確率

連の生存確率を得るため, UCT を用いたモンテカルロ法における連の死活を次で定義する.

$$\begin{cases} K_{root} & : & \text{現在局面} \\ K_{end} & : & \text{プレイアウトの終了局面} \\ S_{root} & : & K_{root}\text{上にある連} \\ \text{生き} & : & S_{root}\text{の石が } K_{end}\text{上に存在する} \\ \text{死に} & : & \text{そうでない場合} \end{cases} \tag{1}$$

(1) は, 盤上に存在する連がプレイアウトの最後まで生き残ったかどうかに着目した. 1 プレイアウトの最後に生存回数をカウントし, 最後にプレイアウト数で割ることで生存確率とした.

### 2.2 連の生存確率分布

連の生存確率と勝敗への影響の度合いを知るために, 実際の局面に対して各連の生存確率分布を求めた.

図 1, 2, 3 の局面はプロの対局棋譜の序盤 20 手, 中盤 80 手, 終局まで打った局面である. これら局面に対して $50,000playout$ を行い, 連の生存分布を得たところ図 4, 5, 6 のような分布となった.

序盤では, 全体の 83%の連が生存確率 70%以上の範囲に分布している. 序盤は布石段階であり, 死活を争う場面がないことが少ないことが原因であると考えられる.

中盤では, 攻め合いが発生したり死活が明確になる部分が出現するため, 生存確率がより明確になり生存確率が分布する範囲が広くなった. 中盤においても生存確率が高い連が多く, 全体の 86%の連が生存確率 60%以上 90%未満の範囲に分布している.

終局した状況では, ほとんどの連の死活が明確になるため, 生存確率の高い範囲に分布する連と低い範囲に分布する連に分かれた.

また他の棋譜についても同様の傾向が得られた.

以上より, 多くの連は死ににくく, 生存確率が高い範囲に分布することがわかった. 従って, 生存確率及び生存確率分布のみから勝敗に大きく影響する連を知ることは困難であり, 異なる指標が必要であると考えられる.

### 2.3 連の勝敗影響度

2.2 より, 何らかの方法で勝敗への影響の度合いを数値化し, 連を差別することが考えられる. 本研究では連の生死が勝敗へ与える影響の度合いを連の勝敗影響度とし, 次で定義する.

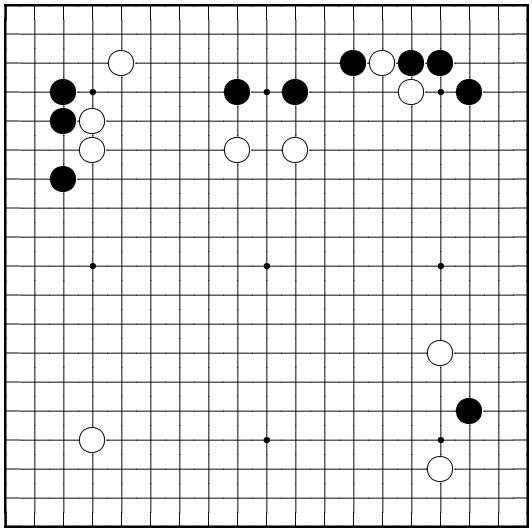

図 1: 20th 富士通杯最終予選 20 手まで

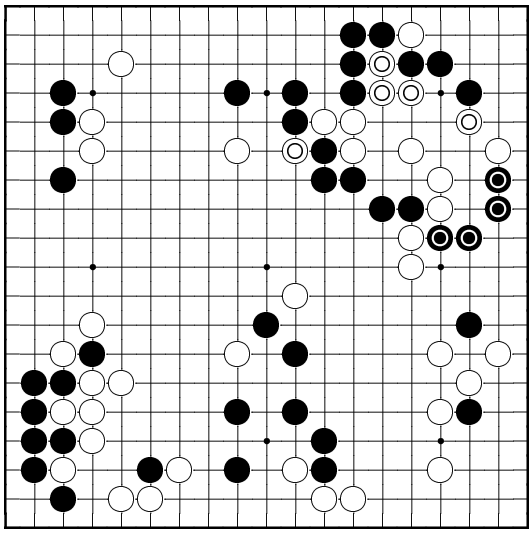

図 2: 80 手まで

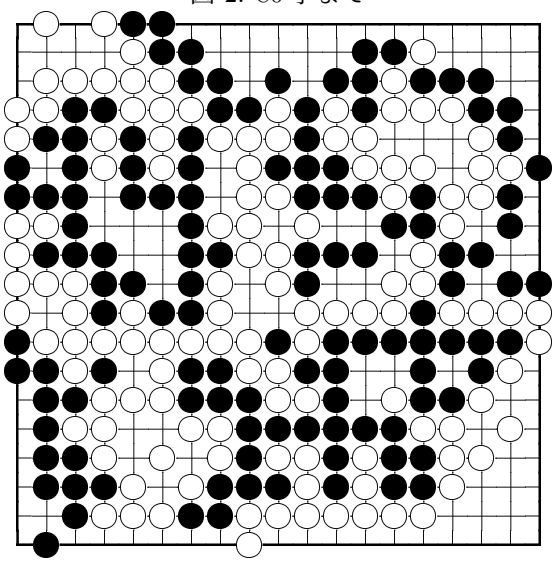

図 3: 終局

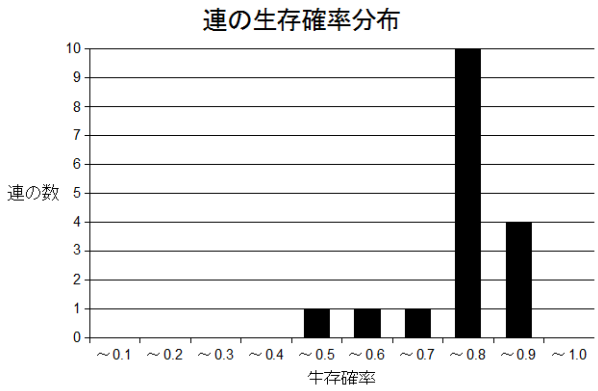


図 4: 生存確率分布　序盤 20 手

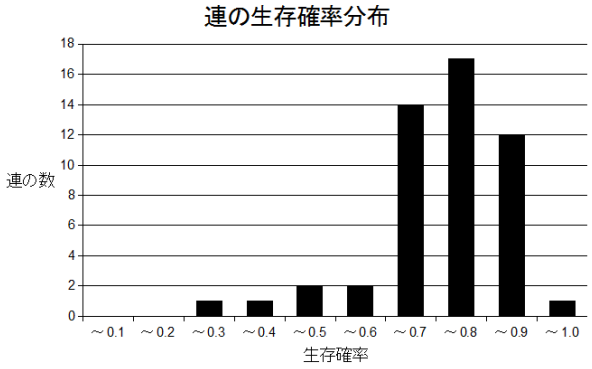


図 5: 生存確率分布　中盤 80 手

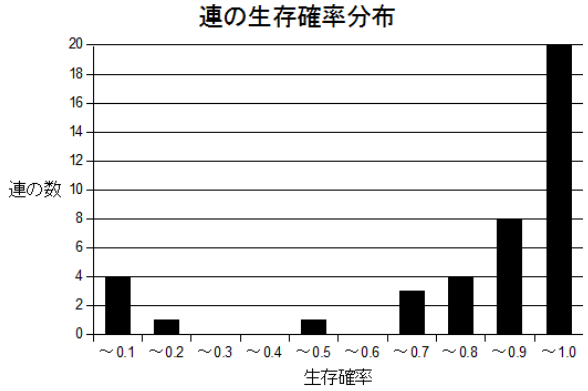


図 6: 生存確率分布　終局

$$\begin{cases} S & : & \text{ある連} \\ L_S & : & \text{連 } S \text{ が生きる事象} \\ W_S & : & \text{連 } S \text{ の色の側が勝つ事象} \\ I_S & : & \text{連 } S \text{ の勝敗影響度} \\ I_S & = & P(L_S) \times \{P(W_S|L_S) - P(W_S)\}. \end{cases} \tag{2}$$

式 (2) は, 勝率 $P(W_S)$ に対して, 連 $S$ が生きたときの勝率 $P(W_S|L_S)$ との差に着目した. $P(W_S|L_S) - P(W_S)$ は, すでに死んでいると考えられる連が生き返った場合に極端に大きくなるため, 連の生存確率との積をとり, 式 (2) とした.

図 2 の局面に対して勝敗影響度を計算した結果, 図 2 の◉及び◎で示されている連の勝敗影響度が高くなった. まだ眼のない黒の連や切った形になっている白の連など重要な連が差別化できていると言える.

### 2.4　連の連結確率

囲碁の対局において, 連同士のつながりは非常に重要である. 連同士のつながりは連の死活とも関係があるため, 重要な要素であると考えられる. 連の結合を次で定義する.

現在局面 $K_{root}$ 上にある同色の 2 つの連を $S_1$, $S_2$ とする時,

$$\begin{cases} \text{連結} & K_{end}\text{上で連 } S_1 \text{と } S_2\text{が一つの連} \\ \text{非連結} & \text{それ以外} \end{cases} \tag{3}$$

とする. (3) は, 局面上の二つの連がプレイアウトの終了局面で結合して一つの連になったかどうかに着目した. 連の生存確率と同様に, プレイアウトの終了局面で結合回数をカウントし, プレイアウト数で割ることで結合確率とした.

### 2.5　連のグループ化

死活は通常, 連単体で生きる場合は少なく, 複数の連でダメを共有することで生きることが多い. つまり, 死活の運命共同体である連の集合があると言える. 従って, 連を何らかのルールでグループ化することは自然な考え方である. 2.4 によって得られた結果を基に連を簡単にグループ化する.

連結定数を $C$ とおくとき, 以下を満たす場合に同じグループとする.

$$S_1\text{と } S_2\text{の結合確率} \geq C. \tag{4}$$

図 7 はプロの対局棋譜の中盤 80 手までである. この局面で $50,000playout$ を行い, グループ化を行った結果を図 7 に示す. グループ化の閾値として, 結合定数 $C = 0.7$(結合確率 70%) 以上の連同士を同じグループとした. グループ化の結果は, 囲碁知識のある人間の視点から見れば, 妥当なグループ

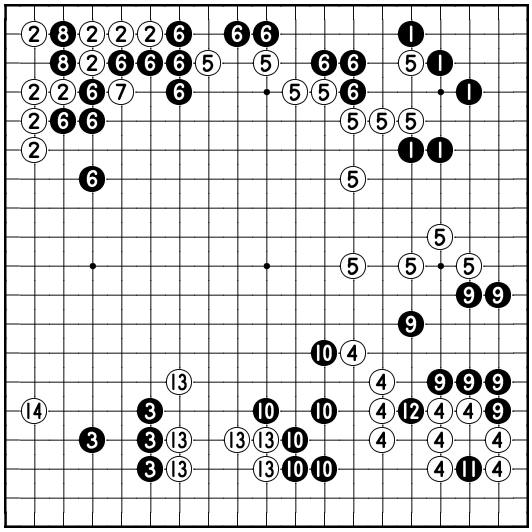

図 7: 20th NEC 杯第 1 局 80 手までをグループ化

化がなされたと言える．右下の❾と⓬の連の結合確率は 65%であったためグループ化されなかった．プレイアウト中に重要視されず, 結合されなかったものと考えられる．実際に, 勝敗影響度は 43 個の連のうち 34 位であった．

### 2.6 グループの勝敗影響度

2.4 では, 結合確率によってグループ化を行った．同じグループに属する連は生死を共にすると考えられるので, グループにも勝敗影響度を設定する．

あるグループ $G$ について, $G$ に属する連を $S_G$, 連 $S_G$ の勝敗影響度を $I_{S_G}$ とするとき, グループ $G$ の勝敗影響度 $I_G$ を以下で定義する．

$$I_G = max\{I_{S_G}\}. \tag{5}$$

グループの勝敗影響度は, グループを構成する連の勝敗影響度のうち, 最大のものとした．

図 7 においては, ❿, ⑤, ⑬, ❾の順にグループの勝敗影響度が高かった．

## 3 勝敗影響度を用いたプレイアウト制御

### 3.1 プレイアウト制御

2.6 を用いて木探索部分に手を加えることで, プレイアウトの改善を試みた．本手法は情報収集を行うフェーズとその結果を集計し, プレイアウトを制御する準備を行うフェーズ, 木探索を制御するフェーズの 3 フェーズに分かれている．

情報収集を行うフェーズでは UCT を用いた一般的なプレイアウトを行う．その間に連の生存回数及び結合回数を得る．

情報収集終了後に生存確率と結合確率からグループ化を行い, グループの勝敗影響度を設定する．

木探索を制御する部分では, 勝敗影響度の高いグループの近傍に打たせるような重みをつける．重み付けは探索木の根でのみ行う．

重み付けは以下の式に従って行う．

$$\begin{cases} P & : & \text{重み付けを行う候補手} \\ value & : & P \text{ の勝利数} \\ playout & : & P \text{ のプレイアウト数} \\ playout^* & : & \text{総プレイアウト数} \\ \alpha & : & \text{従来の } UCB \text{ 値を調整する定数} \\ \beta & : & P \text{ に対する重み付け定数} \\ UCB(P) & : & P \text{ の } UCB \text{ 値} \\ UCB(P)^* & : & \text{重み付けを行った } P \text{ の } UCB \text{ 値} \\ UCB(P) & = & \frac{value}{playout} + \alpha\sqrt{\frac{\log(playout^*)}{playout}} \\ UCB(P)^* & = & \frac{value}{playout} + (\alpha+\beta)\sqrt{\frac{\log(playout^*)}{playout}} \end{cases} \tag{6}$$

式 (6) の後半部分は, 従来の UCB 値の定数部分に重み付けを行うことで, 勝敗影響度の高い群の近傍に対してより多く探検させることを意図したものである．

### 3.2 数値実験

#### 3.2.1 実験で用いたプログラム

実験を行う際に利用したプログラムは, UCT 及びパターンを利用したモンテカルロ碁プログラムである．パターンはランダムシミュレーション中にのみ利用した．ランダムシミュレーション中はパターンにマッチした手からランダムで着手を行い, パターンにマッチする手がない場合はすべての候補手からランダムに着手を行う．パターンはハネやキリ, ツケなど囲碁用語に現れる形を回転, 対称などを含め約 700 種用意した．プログラムは Intel Core2Duo 2.4GHz(Linux, メモリ 2GB) 上に実装し, 並列化を行った．

#### 3.2.2 対局条件

3.1 で提案した手法を用いて GNU Go とテスト対局を行った．対局は中国ルールで, コミは 7 目半とした．

実験で用いるプログラムは 1 手ごとに $50,000 playout$ を行った．そのうち最初の $10,000 playout$ は情報収集フェーズとし, その間は重み付けを行わない通常のプレイアウトを行った．グループ化の閾値として, 結合定数 $C = 80\%$ 以上のものを同じグループとした．また, 重み付けは勝敗影響度が高い上位 2 位までのグループに属する連からマンハッタン距離 2 以内の候補手に対して行った．

#### 3.2.3 重み付けした場合

GNU Go の Level 10 と 100 局ずつ対局を行った．GNU Go との対局結果を表 1 に示す．

表 1: GNU Go Level 10 との対局結果

| $\alpha$ | 勝利数 |
|---|---|
| 0 | 15 |
| 0.1 | 51 |
| 0.2 | 36 |
| 0.3 | 41 |
| 0.4 | 51 |
| 0.5 | 40 |
| 0.6 | 37 |
| 0.7 | 38 |
| 0.8 | 39 |
| 0.9 | 41 |
| 1.0 | 41 |

$\alpha = 0$ の対局は式 (6) より, 重み付けされない単純な UCB 値に従って着手された結果である. まったく重み付けを行わなかった場合の勝利数が 15 であったのに対し, 重み付けを行った場合の勝利数はそれの約 2〜3 倍となった. 勝敗への影響が大きいと考えられるグループの周囲の着手に重み付けを行うことで, 勝利数を大きく伸ばすことができた.

#### 3.2.4 RAVE と併用した場合

本研究では, 3 以外の手法として, RAVE との組み合わせによってさらなる性能の向上を試みた. S. Gelly らによって考案された RAVE はプレイアウトの質を向上させる手法として高い評価を得ている. RAVE はヒューリスティックな手法でプレイアウトの学習効果を高める手法であり, その効果が連の生存確率や結合確率に良い方向に影響する可能性がある. 3.2.3 で使用したプログラムに RAVE を追加し, 3.2.3 と同様の実験を行った. その結果を表 2 に示す.

表 2: RAVE を組み合わせた場合の GNU Go Level 10 との対局結果

| $\alpha$ | 勝利数 |
|---|---|
| 0 | 51 |
| 0.1 | 49 |
| 0.2 | 57 |
| 0.3 | 45 |
| 0.4 | 44 |
| 0.5 | 44 |
| 0.6 | 40 |
| 0.7 | 34 |
| 0.8 | 30 |
| 0.9 | 27 |
| 1.0 | 38 |

$\alpha = 0$ の対局は, RAVE のみを用いた場合の結果である. $\alpha = 0.2$ であったときに, RAVE のみを用いた場合より勝利数を向上させることができた.

## 4 おわりに

本研究では, プレイアウトから勝率以外の情報として連の生存確率と異なる 2 つの連間の結合確率を得た. そこから連の生死の勝敗への影響度を数値化し, 連のグループ化を行った. 囲碁プログラムへの応用として, 勝敗への影響が大きいグループの周囲の候補手に対して, 重み付けを行うことでプレイアウトを制御する手法を提案した. GNU Go とのテスト対局を行った結果, 重み付けを行わない単純な手法よりも勝利数が 約 2〜3 倍 に向上する結果を得た. この結果から, 勝率以外の情報が強い囲碁プログラムを作る上で重要な要素になりうるという確証を得たと考えている. また, RAVE と併用した場合においては, RAVE のみの勝利数を上回るケースはあったが, 対局数が少ないため, 更なる対局実験が必要と思われる. しかし, 併用することでより強いプログラムを作ることができるという予感はある.

今後の課題として, 1 つは生存確率及び結合確率以外の情報の抽出が挙げられる. 例えば, 模様の争点などを思考過程の早い段階で発見することができれば, それを利用して戦略的なプレイアウトができるかもしれない.

もう 1 つは 19 路盤への適用である. 今回用いた手法だけでは 19 路盤で勝利をあげることができなかったことを報告しておく. 我々は 2.3 で提案した勝敗影響度は 19 路盤においても有効であろうと考えているが, 情報収集のためには 9 路の場合よりも多くのプレイアウトを必要とするだろう. また, 棋力を向上させるにはその他の手法と組み合わせる必要があると考えている.

## 参考文献


[1] 清愼一, 山下宏, 佐々木宣介. コンピュータ囲碁の入門. コンピュータ囲碁フォーラム (編). 共立出版, 2005.

[2] B.Bouzy. Associating domain-dependent knowledge and Monte Carlo approaches within a Go program. *Information Sciences,* Vol.175, No.4, pp247-257, November 2005.

[3] B.Bouzy and B.Helmstetter. Monte Carlo Go developments. *Advances in Computer Games. Many Games, Many Challenges,* pp.159-174. Kluwer, 2003.

[4] L. Kocsis and C. Szepesvári. Bandit based monte-calro planning. *European Conference on Machine Learning,* pp.282-293, 2006.

[5] Rémi Coulom. Efficient selectivity and backup oprators in Monte-Carlo tree search. In P.Ciancarini and H.J. van den Herik, editors, *Proceedings of the 5th International Conference on Computer and Games,* Turin, Italy, 2006.

[6] S.Gelly, Y.Wang, R.Munos and O. Teytaud. Modification of UCT with Patterns in Monte-Carlo Go, Technical Report RR-6062, INRIA, 2006.

[7] S. Gelly and D. Silver. Combining online and offline knowledge in UCT, *Zoubin Ghahramani (Ed.) Proceedings of the 24th International Conference on Machine Learning,* pp. 273.280, Omni Press, 2007.

[8] GNU Go
http://www.gnu.org/software/gnugo/

[9] B.Brugmann: Monte Carlo Go. Technical report, Physics Department, Syracuse University, 1993.

[10] メイエン事件簿:第 30 話 かんぱいモンテカルロ
http://taisen.mycom.co.jp/taisen/contents/igo/meien/meien_30.htm