特集　ゲーム情報学

応用

6 コンピュータ囲碁の現状

村松正和（電気通信大学情報理工学研究科情報・通信工学専攻）

コンピュータ囲碁の強さ

電気通信大学では2007年から毎年UEC杯コンピュータ囲碁大会[2]を開催している．UEC杯は国内外のプログラムが20以上集まり，技を競い合う，日本で最大の国際コンピュータ囲碁大会である．

2008年より毎年，このUEC杯の最後に，優秀な成績を収めたプログラムとプロとの置碁対局が行われている．2008年にはコンピュータがプロに7子のハンディキャップをもらって勝っている．2009年は6子でコンピュータ側が0勝2敗であったが，2010年は6子で1勝1敗と引き分けた．2011年のUEC杯は12月3・4日に行われたはずである．原稿執筆時点では結果は分からないが，どうなったであろうか．興味のある読者はWebサイト[2]をチェックされたい．

表-1はKGS☆1というインターネット碁会所での，2011年10月31日におけるコンピュータ囲碁のランクを掲げたものである．インターネット碁会所のランク付けに関しては諸説あるが，大雑把にいってdが（アマチュアの）段を表していると考えてよい．ZenとCrazy Stoneの5dを筆頭として，段位を持つプログラムがたくさんあることがお分かりだろう．アマチュア5段といえば，碁会所でも相当の打ち手である．今，コンピュータ囲碁はここまで来ているのである．

本稿では，近年急激に発展してきたコンピュータ囲碁の現状について述べるとともに，基本となる考え方，モンテカルロ木探索に焦点を当てて解説する．

なお，2008年の「情報処理」に美添氏がモンテカルロ木探索に関する解説[1]を寄稿されている．本稿もこれと重なる部分も多いが，美添氏の解説はより基礎的な部分に多くの記述が割かれているので，本稿はその後の発展部分に重心を置いて解説するつもりである．

KGSのID	rank	プログラム名	作者
Zen19D	5d	Zen	Yamato
CrazyStone	5d	Crazy Stone	Rémi Coulom
Ginseiigo	2d	KCC囲碁	KCC
ManyFaces	1d	Many Faces of Go	David Fotland
AyaMC	1d	彩	山下宏

表-1　KGS（インターネット碁会所）におけるコンピュータ囲碁のレベル

また，モンテカルロ木探索の理論的な側面に関しては，美添氏と私でやはり「数学セミナー」[3]にまとめている．こちらに興味のある方は合わせてご一読願いたい．

特に紙数の都合上，囲碁のルールに関してはここでは触れない．囲碁に関する基本的なことを知りたい方は，さまざまな入門書やWebサイトなどを参考願いたい．

min-max探索の枠組み

将棋・囲碁はゲーム情報学的には2人完全情報確定零和ゲーム☆2と呼ばれるカテゴリに属する．このようなゲームの進行は，局面（状態）をノードとし，一手打つたびに局面が変化する木（**ゲーム木**と呼ばれる）として表現できることはよく知られている．

原理的には，初期局面から最終局面を葉とするゲー

☆1 http://www.gokgs.com/
このサイトでは人間同士のみならず，コンピュータと人間やコンピュータ同士が囲碁を打ち合っている．

☆2 2人でプレイし，お互いにゲームの情報が完全に分かっており，確率的な要素がなく，結果が勝ち／負け／引き分けとなるゲームのことである．

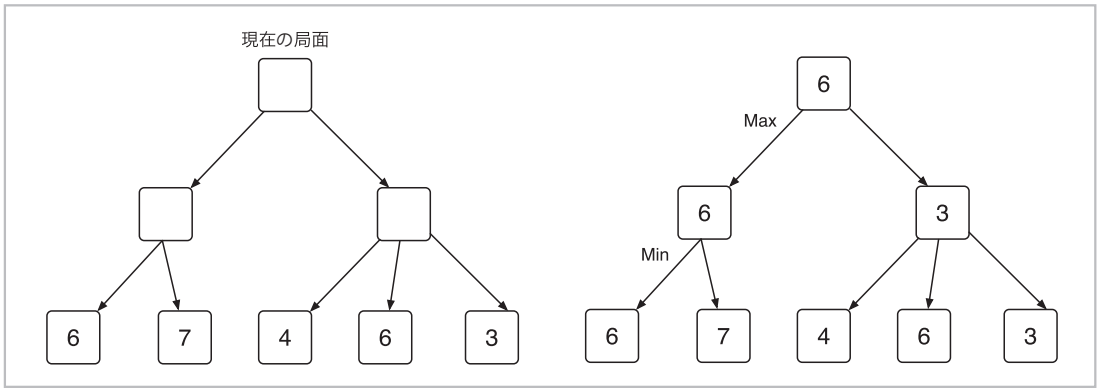

図 -1　深さ 2 のゲーム木探索

ム木をすべて探索すれば，その初期局面における最善手と勝敗を決定できる．しかし，オセロ以上に難しいボードゲームにおいては，ゲーム木が巨大になるため，現在の計算機の能力では完全解析は不可能である．

そのようなゲームにおいて，1 つの大きな目標は「人間より強いプログラムを作る」ことである．特にチェス，将棋，囲碁などは，これまで人間が英知を傾けて技術を磨いてきたものであり，それをプレイすることを職業とする人間もいる．そこまで努力してきた人間の，最も強いプレイヤよりも強いプログラムを作ることは，情報工学に携わる者の大きな夢である．その目標はチェスやオセロではすでに成功し，将棋でも実現が視野に入りつつある．

さて，チェスや将棋における成功はすべて，以下の「min-max 探索の枠組み」を基本としている．

1. 盤面から静的に形勢を（ある程度正確に）評価できる**盤面評価関数**を用意する．
2. 探索する深さを決めたゲーム木を生成し，その葉の値は盤面評価関数の値であるとして，最も評価関数値の高い手を探索する．

図 -1 に簡単な例を示す．左はゲーム木を深さ 2 まで展開した図であり，葉（末端のノード）の数字はその局面における盤面評価関数の値を表している．ここで深さ 1 のノードは相手の手番なので，その直下にあるノードの中で最小（つまり，相手にとっては最も得）となるノードを選んでプレイすると考えられる．次に根について考えると，自分の利益を最大にしたいので，直下になるノードの中で最大のノードを選んでプレイする．このような探索方法は **min-max 探索**と呼ばれる．結果として現在の局面からの最善手は左のノードをたどることであることが分かる（図 -1 右）．

この「盤面評価関数」＋「木探索」の組合せは，チェスにおいても将棋においても常に効果を発揮してきた．以下，この組合せを **min-max 探索の枠組み**と言うことにする．

min-max 探索の枠組みでは，高速な木探索アルゴリズムの開発と盤面評価関数の正確さが決定的な役割を果たす．ところが囲碁においては，次の点でこの min-max 探索の枠組みが有効でない．それは以下のような理由による．

1. 適切な盤面評価関数を作ることが非常に困難
   困難な理由はいろいろあるが，1 つは将棋のように駒に個性がないため，駒得のような分かりやすい指標がないことが挙げられる．また，局所的な成功が大域的な成功と結びつかないことが多々あることも一因である．ウッテガエシ，ナカデなどの捨て石テクニックは級位者でもよく使う基本的なものであるが，いずれも一時的に損をして後で得をするものであり，静的な盤面評価が困難である．
2. 分岐因子が大きい
   囲碁は平均的におよそ 250 手くらい候補手がある．これはチェス，将棋に比べてはるかに大きく，ゲーム木を深く探索するのを妨げている．

端的に言えば，囲碁という牙城に対して，min-max 探索という武器では歯が立たない，ということである．このような状態が長く続いていたところへ，彗星のように現れたまったく新しい手法が次章で述べるモンテカルロ木探索である．

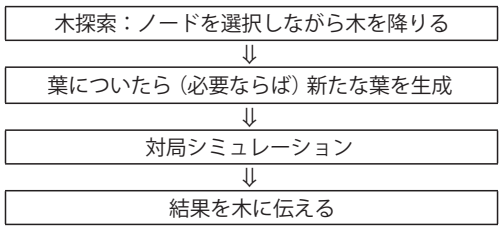

図 -2　スレッドの動き

モンテカルロ木探索の枠組み

モンテカルロ木探索は，一言で言うと，盤面評価と木探索を同時に行う手法と言える．これは min-max 探索が，最初にゲーム木はあるレベルまで展開してあり，その葉（末端）に静的盤面評価関数の値が入っている，と仮定するのとまったく異なる．モンテカルロ木探索は次のような考え方を基本としている．

1. 葉における盤面評価は対局をシミュレーションすることにより行う．この対局シミュレーションの勝率を見て盤面を評価する．
2. 1 において勝率が高い葉は，もっと深く探索するため，展開して木を成長させる．

囲碁において盤面評価関数を作ることは困難と述べたが，例外はある．それはゲームが終わったとき（終局時）で，このときにはお互いの地を数えるだけで勝敗が判定できる☆3．1 の心は，ある局面において，もしどちらかが非常に優勢であれば，（たとえ弱くても）同じ棋力のプレイヤがその局面から打てば優勢な方が勝つ確率は高かろう，ということである．

図 -2 にモンテカルロ木探索のアルゴリズムを示した．モンテカルロ木探索はマルチスレッド環境に相性が良く，図 -2 の一連の動きを 1 つのスレッドが担当する場合が多い．そこで本稿では，図 -2 を担当する一連の動きを「スレッド」と呼ぶことにする☆4．モンテカルロ木探索では，このスレッドをたくさん動かし，だんだんと木を成長させるとともに，盤面評価を行っていく．1 つのスレッドは「木探索」と「対局シミュレーション」という大きな 2 つの部分を含んでいる☆5．木探索も対局シミュレーションも，着手を繰り返すことに対応していることは注意されたい．

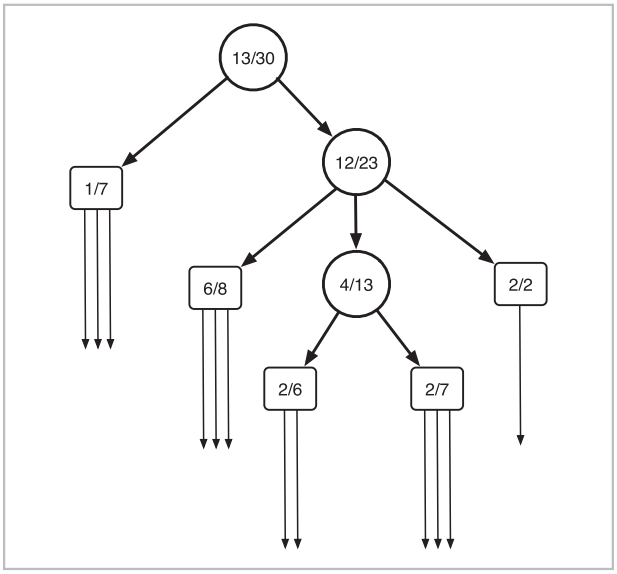

図 -3　モンテカルロ木探索

図 -3 はモンテカルロ木探索で生成されたゲーム木の一例である．生成された木の深さが揃っていないことが分かる．四角いノードは葉であり，そこから対局シミュレーションが始まる．細矢印はシミュレーションを表現している．各ノードの数字は，右がそのノードを通ったスレッドの回数，左がその中で勝利した数☆6 を表している．

実際にモンテカルロ木探索を行うには，「木探索」と「対局シミュレーション」のやり方を具体的に決めなければならない．ここにまたモンテカルロ木探索ならではの工夫が施されている．以下では，これらの詳細について見ていこう．

木探索

木探索では，なるべく有望なノードに降りていくことが重要であるが，その「有望なノード」を推定

☆3 詳しくは述べないが，コンピュータ囲碁では勝敗判定には中国ルールを使うのが一般的である．日本ルールが「地」の大小を争うのに対し，中国ルールは「地＋生き石」の大小を争うので，自分の地の中に石を置いていっても損にならない．人間がプレイした場合は，自分の地の中に石を置く人はあまりいないので，両方のルールはほぼ同じ結果を生む．

☆4 この一連の動作をまとめて一語で表した表現が過去の論文にないので，本稿ではこのようにした．

☆5 関連して「プレイアウト」という用語もあるが，これはここで言う「対局シミュレーション」を指すことも，「スレッド」を指すこともあるようである．混乱の元となるので，この解説では「プレイアウト」という用語は使わないこととする．

☆6 勝ち負けは根における手番で考えている．

するのに，シミュレーション結果をどのように用いればよいのだろうか．

まず最初に考えられるのは，そのノードを通過したスレッドの勝率，

$$\mathrm{MC}(v) := W(v)/N(v)$$

が最も高いノードに降りることである．ただし，$N(v)$ はノード v を通過したスレッドの個数，$W(v)$ はノード v を通過したスレッドのうち，シミュレーションで勝利[☆7] した回数である．しかし，このナイーブな方法はあまり良くない．もう一度図 -3 を見てほしい．ここで，真に一番良いのは左のノードであると仮定しよう．たまたま運悪く最初の 7 回のスレッドのうち 1 回しか勝っていない状態である．この場合，$\mathrm{MC}(v)$ の高いところを選択するばかりだと，左のノードは永遠に採択されなくなってしまうことが分かるだろう．目先に見える「有望な部分」(= 右側のノード) のみに計算資源を集中すると，「より有望な部分」あるいは「真に有望な部分」(= 左側のノード) を探索しない可能性が出てくるのである．

あるノードに関して，そこを通るスレッドの数が増えればそれだけそのノードにおける盤面評価は信頼性が高まる．しかし，計算資源は限られているため，無限にスレッドを生成するわけにはいかない．モンテカルロ木探索の成功の鍵となったのは，スレッドがどのノードを通過するかについて，綿密な設計を行ったことである．「有望な部分」に対する探索と，「まだ探索していない部分」に対する探索とのバランスをうまくとるのである．

木の各ノード v に対して定義される UCB (Upper Confidence Bound) 値 $\mathrm{UCB}(v)$ はこの目的のためによく使われる：

$$\mathrm{UCB}(v) := \mathrm{MC}(v) + C_{\mathrm{UCB}}\sqrt{\frac{\log N}{N(v)}}$$

ただし，N は v の親ノードを通過したスレッド数，C_{UCB} は集中と分散の割合を決めるパラメタである．UCB 値の第 2 項は N が大きくなるにつれおだやかに大きくなるので，たくさんのスレッドが発行されるといつかは「真に有望な部分」にも計算資源が割り振られることになる．理論的にも，UCB を用いた計算資源の割り当て方はある種の最適性を持つことが知られている (文献 3) 参照).

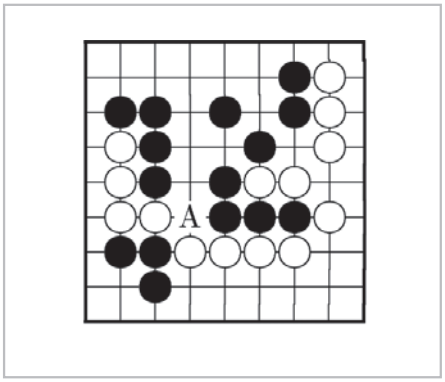

図 -4　A に打った方が勝つ確率が高い

木探索において，UCB 値が一番高い子ノードに降りていく手法を **UCT (Upper Confidence bound applied for Tree)** と言う．この手法は Mogo や Crazy Stone といった有力プログラムで早いうちから用いられ，特に 9 路盤での棋力の向上に劇的な成果を見せた．

UCT はそれだけでかなり有効な方法であるが，19 路盤という巨大なゲーム木を相手にした場合，1 つのスレッドから得られる情報が少なすぎることが問題となる．スレッドは木探索・対局シミュレーションを含めて数百手の盤面進行を行うのに，それから木に返される情報は「勝ち (1)」「負け (0)」だけなのである．

そこで考えられたのが **RAVE (Rapid Action Value Estimation)** という手法である．これは，囲碁において，正確な手順よりも，ある点を占めることが重要である場合が多いことを利用している．たとえば図 -4 を見てほしい．次が白番の図であり，もし白が次に A 点に着手すれば，(正しい手順であれば) 左下の黒石が取れる (逆に白がこの点を逃し，黒が A 点に着手すると黒が左辺の白石を取れる)．モンテカルロ木探索でこの A を探し当てたいわけであるが，スレッドの中で，たとえ初手でなくても，たまたま白が A を打てた場合，その勝率はかなり上がることになると想像できる．RAVE では，木の

☆7 以下では勝利はそのノードで手番の方に関してカウントする．

中のあるノードから打たれたすべての手を，1手目に打たれたものとして評価する．毎回のスレッドでAの点を占めた方が勝率が高いため，すぐにAの点が浮き彫りになってくる．

具体的には次のようにRAVE値は計算される．木探索の途中，あるノードvにおいてaという手を打ちv'へ遷移したとする．このとき，(v, a)に対してRAVE値を

$$\mathrm{RAVE}(v,a) := \frac{W_{\mathrm{RAVE}}(v,a)}{N_{\mathrm{RAVE}}(v,a)} + C_{\mathrm{RAVE}}\sqrt{\frac{\log N_{\mathrm{RAVE}}(v)}{N_{\mathrm{RAVE}}(v,a)}}$$

と定義する．ただし$N_{\mathrm{RAVE}}(v, a)$はvを通ったスレッドのうちでaが現れたスレッドの数，$W_{\mathrm{RAVE}}(v, a)$はその中で勝った数，$N_{\mathrm{RAVE}}(v) := \sum_a N_{\mathrm{RAVE}}(v, a)$，$C_{\mathrm{RAVE}}$は分散と集中の割合を決めるパラメタである．

UCTではシミュレーションが終わったとき，勝ち(1)か負け(0)を返し，これによりスレッドが通り過ぎたノードの情報をアップデートするが，RAVEではスレッドが通り過ぎたノードの子ノードの情報も同時にアップデートすることになる．このようにしてRAVEはスレッドの中で打たれた数百手の情報を木の構造に反映させる．スレッド中の一手一手がRAVE値に寄与するので，たとえスレッド回数が少なくても$N_{\mathrm{RAVE}}(v, a)$，$W_{\mathrm{RAVE}}(v, a)$といった値はすぐに大きくなり，安定する．これをUCB値の推定値として用いるのである．

実際に木を降りるときにはこれをUCBと混合して

$$\mathrm{UCT\text{-}RAVE}(v, a) := C_{\mathrm{UR}}\mathrm{RAVE}(v, a) + (1 - C_{\mathrm{UR}})\mathrm{UCB}(v)$$

を用い，この値が最大のノードを降りるように設計することが多い．C_{UR}はどのくらい2つを混ぜるのかを調節するパラメタである．また，スレッド数が増えるに従いUCBの信頼性が増すのでC_{UR}を変化させるなどの工夫も行われている．

対局シミュレーション

対局シミュレーションにおいて純粋にランダムに着手を選択すると，やはりあまり強くはならない．このようなプログラム同士が対戦すると，コミ分の有利さを維持できず，白番の勝率が高くなりがちである．

そこで，対局シミュレーションでの着手選択に人間の持つ知識を導入することが行われ，成果を挙げている．

現在，対局シミュレーションの考え方には次の2つの潮流がある．

1. Crazy Stone流の考え方
2. Mogo流の考え方

Crazy Stoneでは，対局シミュレーションにおいて一手ごとに乱数を振り，着手を選択する．その際，打たれやすい特徴を持った点に打たれる確率を増やすようにしている．

具体的にはまず，局面の各候補手に対して，周りのパターン，前の手との距離などのさまざまな特徴を抽出する．各特徴iに対し，その「強さ」として正の数γ_iを割り振る．ある候補手vの重みはvが持っているすべての特徴の「強さ」の積で決まるとモデル化する．次の着手は，すべての候補手に対し重みに比例した確率で選ばれることになる．

Crazy Stoneの作者，Rémi Coulomの論文[4]では，この数理モデルを提案し，さらに人間が打った棋譜における着手に最も適合するようなパラメタγ_iを学習するアルゴリズムを提案した．この技術により，シミュレーションの中で高い一致率（実際の棋譜と同じ場面で，人間と同じ手を選択する率）が実現されるようになり，またその結果としてプログラムは劇的に強くなった．

最近台湾で開発されたEricaもこの手法を応用しており，候補手周りの非常に大きなパターンを切り出し，その特徴に対して重み付けを行う手法をとっている．この手法でEricaは2010年のComputer Olympiad 19路盤部門で優勝した．

Mogo流の考え方は，直前に打たれた手の周りだけを考慮し，ある種の特徴やパターンに一致する手を打っていくものである．一致する手がないときに初めて，ランダムに着手を選択する．CrazyStone流の考え方では，一致率は高いと言っても半分程度なので，シミュレーションにおいてあちこちに着手が飛び，結果としてあまり本物の囲碁らしくなっていない場合が多い．これにくらべ，後者の方法では連

続した応手が続くので，シミュレーションの出来上がり図がより本物の囲碁らしくなると言われている．Crazy Stone の場合におけるような理論的な背景はあまり感じられないが，さまざまなヒューリスティックを導入しやすいこともあり，結構使われている．

現在の多くの強豪プログラムでは，この 2 つの良いところを混合する形で使われていることが多いようである．

この対局シミュレーションの部分は，プログラム作成者が最も工夫を凝らすところである．特に，オープンソースでないプログラムは，その詳細は秘密になっており，そこが強さを分けているとも言われている．

これからの展望

各スレッドはほとんど並列に動かすことができるため，モンテカルロ木探索は並列性に優れていると言える．では何百コアというクラスタを用いれば圧倒的に強くなるのではないか，と思われる方もいるかもしれないが，これがそう簡単ではない．モンテカルロ木探索はゲーム木探索であるので，たくさんのコアを用いてもなかなか深くヨムことは難しい．このあたりの状況は min-max 探索を基本とする将棋プログラムが抱える問題と同様である．

ここまで解説を読み，「死活や攻め合いはどのように判定しているのか」という疑問を持たれた方もいるだろう．囲碁を嗜む者にとって，死活や攻め合いは基本事項で，常日頃その能力を磨いておかなければならないものである．それなのに，この解説においてここに至るまで何も死活や攻め合いといったテーマが出てこなかった．

実は，モンテカルロ木探索を用いてもなかなか死活や攻め合いはうまく処理できず，単純な攻め合いさえ混乱して間違えるプログラムが多い．モンテカルロ木探索では確率的な要素に負う部分が多いため，深くヨムことは苦手なのである（それでこのくらい強いというのは，逆に言うと驚きであり，囲碁の奥深さを感じさせる）．

どの石が攻め合っているか，また，どの石の死活が問題か，という情報は，スレッドを重ねて木を成長させるうちにある程度分かってくる．その情報を対局シミュレーションに的確に反映させる手法が開発されれば，コンピュータ囲碁はまたもう一段強くなるかもしれない．

実は死活や攻め合いに関しては，Zen が何かうまいことをやってかなり正確に判定している．そこが Zen と他のプログラムとの差となっているように見受けられる．残念ながら Zen は商用ソフトであるため，このあたりの技術の詳細はベールに包まれているのが現状である．

コウも難しい問題で，特に二段コウやヨセコウといった形は，やはりコンピュータも理解し難いようである．コンピュータと打つと，そういう難しい形はわざと避けているのかもしれないと感じることがある．しかし高段者を目指すならばこれは避けて通れない道であるので，今後の解決を期待したい．

アマチュア 5 段には誰でも努力すればなれるが，それ以上になるには才能が必要，とよく言われている．コンピュータ囲碁も，ここまでともかくアマチュア 5 段までは来たし，現在の状況を見ているとまだしばらく伸びると思われる．

その先の領域までこのまま至るのか，あるいはやはり人間のプロレベルには到達できず再び停滞の時期に入るのか．これから数年はコンピュータ囲碁から目が離せない．

参考文献

1) 美添一樹：モンテカルロ木探索—コンピュータ囲碁に革命を起こした新手法，情報処理，Vol.49，No.6，pp.686-693 (June 2008).
2) UEC 杯コンピュータ囲碁大会 http://jsb.cs.uec.ac.jp/~igo/
3) 美添一樹，村松正和：コンピュータ囲碁の飛躍の背景／モンテカルロ木探索，数学セミナー 2010 年 10 月号.
4) Coulom, R. : Computing Elo Ratings of Move Patterns in the Game of Go, ICGA Journal, Vol.30, No.4, pp.198-208 (2007).

(2011 年 10 月 31 日受付)

村松正和（正会員）
muramatu@cs.uec.ac.jp

2009 年電気通信大学教授．専門は数値的最適化とオペレーションズ・リサーチ．コンピュータ囲碁にも興味を持つ．ゲーム情報学研究会幹事，Computer Go Forum（CGF）副会長.