不動碁のアルゴリズム

[概要]

不動碁は，その本体部分には MoGo の論文にある決定的なシミュレーション，UCT アルゴリズム，および RAVE を用い，これに独自のネットワーク並列アルゴリズムを加え，PS3 などのゲーム・コンソールと PC が混在したパーソナル・クラスタ上で動作します．
＃この大会では，ハードウェアの制限のためネットワーク並列部分は全く動いていません．

[基礎]

モンテカルロ碁プログラムは，現在の局面から合法手を一つ選び，そこから終局まで内部で擬似的な対局（シミュレーション）を非常に多数回実行し，その勝率を選んだ手の評価値とします．これによって，これまで評価が難しかった「厚み」を最終的なスコアとして数値化することに成功しました．また，これもこれまで処理が難しかった「捨石」や「劫」なども自然に扱うことができます．

しかし，初期のモンテカルロ碁プログラムは，木探索機能がないために局面を深く読むことができず，あまり強くありませんでした．これは，手の評価値（勝率）が確率的な誤差を含んでいて，ミニマックス法という広く使われている木探索アルゴリズムが使えないためです．その後，2006年に UCT（Upper Confidence bounds applied to Trees）とう確率変数向きの木探索アルゴリズムが開発されてから急激に強くなり，現在最も強いプログラムはアマ六段ぐらいで，プロ棋士に五子で勝っています．

[MCTS アルゴリズム]

MCTS（Monte Carlo tree search）という名前は，必ずしも全てのモンテカルロ碁プログラムが UCT アルゴリズムを採用しているわけではないので，それに代わる一般的な名称として考案されました．MCTS アルゴリズムは，与えられた局面をルート・ノードとし，各ノードで合法手の中から最適な（評価値が最も大きい）手を選びながら探索木を降り，末端（リーフ・ノード）に到達したら，そこからシミュレーションを開始します．この際，この手を通った回数がある値を超えていたら（例えば MoGo はゼロ，つまり毎回），新しいノードを作って探索木を成長させます．結果が得られたら，そこから探索木をルート・ノードまで遡りながら，通り道の各ノードの着手の評価値を更新します．これを（一手 30秒など）指定された時間繰り返し，最後にルート・ノードで最も多く選ばれた手を最善手とします．なお，評価値の計算に UCB1（Upper Confidence Bounds 1）アルゴリズムを使うのが UCT アルゴリズムです．

[シミュレーション]

モンテカルロ碁プログラムの強さを決める第一の要因はシミュレーションの質です（二番目は速度，つまり一手当たりのシミュレーション回数）．どの手が最善かは（当然）分かりませんから，複数の良さそうな手をその「良さ」に応じた確率でランダムに選んで打ちます．この際，どれだけ良い手に集中できるか，どれだけ悪手を打たないかが非常に重要です．例えば三目中手の形があったらその真ん中に打つ手の確率を上げるのが有効です（いわゆる「形」の急所）．

初期の MoGo は，3x3 のパターンをいくつか用意し，直前の相手の手の周囲の状況がそ

のどれかに一致したら「必ず」そのパターンに指定された手を打つという方針で，近接戦に強くなりました．他方，Crazy Stone は，同じようなパターンを使いますが，それにはある確率が割り当てられており，それに応じて手を選びます．これにより，シミュレーションがより手広くなり，序盤のような広い局面に強くなっていると考えられています．不動碁が用いているのは前者です．

[決定的と非決定的]
モンテカルロ碁のプログラムは大きく二つ，シミュレーションに決定性を導入しているものといないものに分けられます．一般に，前者は攻め合い（近接戦）に強く，後者は布石が上手いという特徴があります．前者の創始者は MoGo，後者は Crazy Stone と言うことができ，不動碁は前者に属します．他に，Zen や Pachi も前者で，Aya や Erica は後者に属します．もちろんどのプログラムにも独自の改良や拡張が加えられていますが，両方の特徴を兼ね備えた，つまり序盤も上手く攻め合いも強いプログラムは未だ現れていません．

[RAVE]
MoGo の作者による RAVE（Rapid Action Value Estimate）アルゴリズムは，強化学習をその理論的基盤とし，最初のモンテカルロ碁プログラムである Gobble に使われていた All Moves As First（AMAF）ヒューリスティックを UCT に適用したもので，シミュレーションの結果をとことん利用することができる優れた(副作用もありますが)アルゴリズムです．今の強いプログラムは，何らかの形で AMAF のアイデアを利用しています．不動碁は RAVE を（MoGo 以外で）最初に実装したプログラムです．

[並列化]
シミュレーションは各々独立(無関係)なので，複数個同時に実行することができます．メモリを共有するマルチプロセッサ・システムの場合，シミュレーションを複数のスレッドで並列に実行し，共有している探索木中の情報にアクセスする時だけ排他制御を行います．この手法は MCTS の非常に自然な並列化でかつ実装も容易なので，不動碁はもちろん，ほぼ全てのモンテカルロ碁プログラムで使われています．

コンピュータ・クラスタの様なメモリを共有しない環境では，リーフ並列とルート並列の二種類のネットワーク並列が代表的ですが，他にもいくつか研究されています．ネットワーク並列の場合，通信がずっと遅いので，メモリ共有システムの様に探索木を共有するのは事実上不可能です．

リーフ並列の場合，マスター・コンピュータは（最適な手を選びながら）探索木を降り，探索木のリーフ・ノードに達したら，局面を全てのスレーブ・コンピュータに対してブロードキャストし，(以前に発行したものも含めて)何か一つ結果が返ってくるまで待ち，到着した結果を全て処理した後，ルート・ノードに戻って以上の処理を繰り返します．スレーブ・コンピュータは，(暇な時に)新しい局面を受け取り，それを終局までシミュレートし，結果をマスター・コンピュータに返します．この方法は，探索木がメモリに収まらない様なゲーム・コンソールをスレーブ・コンピュータとして使うのに適しています．

この非同期型の実装は不動碁のオリジナルで，それまでリーフ並列は同期型（送信した局面のシミュレーションが全て終わるまで待つ）で実装されて性能が良くないと報告され

ていたのですが，これによりリーフ並列もルート並列と同等の性能であることが実証されました．

ルート並列は，スーパーコンピュータの様な大規模な並列システムで使われている方法で，各コンピュータは独立に探索を行いながら，一定の周期でお互いのルート・ノードの情報を交換することで，ルート・ノードを擬似的に共有します．

このどちらも，並列化によって（探索木の深さは増えませんが）ルート・ノードの各合法手の評価値の分散が減少することで，プログラムが強くなると考えられています．

以上．