4Q-3

モンテカルロ碁におけるプレイアウトの高速化

高橋 大介† 中村 克彦‡

東京電機大学大学院理工学研究科† 東京電機大学理工学部‡

1 まえがき

近年, コンピュータ囲碁においてモンテカルロ・シミュレーションと木探索にもとづく着手決定方式の有効性が注目され, UCT アルゴリズムが多くの囲碁プログラムで採用されるようになった. UCT アルゴリズムではプレイアウト（終局までのランダムシミュレーション）の結果から探索木が作り出される. 探索木を作る過程で木の各ノードに対してそのノードを通過したプレイアウトの数, および勝率がカウントされる. これらの値はプレイアウトの結果を末端ノードからルートノードに向かって繰上げていくことによって更新される. プレイアウト中に通過した探索木のノードはすべては保持されず, 既に一度通過したノードを再び通過する時に限り，子ノードが生成される. この一連のステップを多数回繰り返し, 探索木を成長させていくことによって有望な着手を探し出していく. 現在, プレイアウトに囲碁のパタン知識を組み合わせ, その精度を向上させる研究や探索木の枝狩りの研究などが盛んに行われている.

一方で, UCT アルゴリズムで求められる着手の精度はプレイアウトの質と回数に依存している. このため, プレイアウトを高速に行うことが重要である.

本研究ではプレイアウトにおける空点のランダムな選択法, 木構造を用いたブロック（連：縦横に連結した同色の石の集合）の新しい方式のデータ構造によるプレイアウトの高速化手法について述べる.

2 プレイアウトにおける空点の選択法

プレイアウトにおいて盤上の空点の座標データは着手点が選ばれる度に 1 つ削除されていく. 石が盤上から取り除かれたときは石が置かれていた座標が新たな空点となる. 一般に 1 つの候補点を決定するためにいくつかの空点を調べる必要がある. このため, 空点を調べる回数は N 路盤の $N \times N$ よりも大きい.

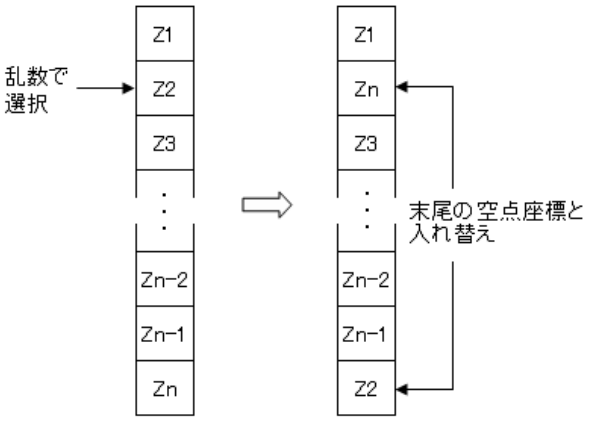

図 1: プレイアウト時の座標データの管理

空点の集合から乱数で候補手を選ぶ方法として盤上の空点のリストを作り, その中からランダムに選ぶ方法がある. これは候補手 1 度選ぶ度に空点のリストを線形探索する必要がある. また, N 路盤を横と縦に分け, 横の着手確率の合計を記憶しておき, 横方向に N 回走査, 縦方向に N 回走査して着手点を決定する方法もある. この方法では平均 N 回の走査で着手点を決定できるが, その計算量は盤面の広さに依存したものとなる.

着手点の決定を定数時間で行う方式を示す. 各要素が盤面の座標 Z_1〜Z_N で初期化されている長さ N の 1 次元配列 $A[N]$ を用意し, 以下の手順で空点の選択を行う.

1. 1 から K の範囲の乱数 R を発生させ, 配列 $A[R]$ の座標データを取り出す
2. 図 1 に示すように乱数で得たデータと K 番目のデータを入れ替える
3. 配列上の空点の集合と石の集合の境界部分の添え字を保存する
4. K の値を 1 小さくする

以上の操作を繰り返すことで, 配列の先頭の方には盤上の空点の座標が, 末尾の方には石が置かれている

Speeding up playouts in Monte Carlo Go

†Daisuke Takahashi, ‡Katsuhiko Nakamura

†Graduate School of Science and Engineering, Tokyo Denki University

‡School of Science and Engineering, Tokyo Denki University

†honyarara@ms.dendai.ac.jp, ‡nakamura@rd.dendai.ac.jp

Copyright ©2011 Information Processing Society of Japan.
All Rights Reserved.

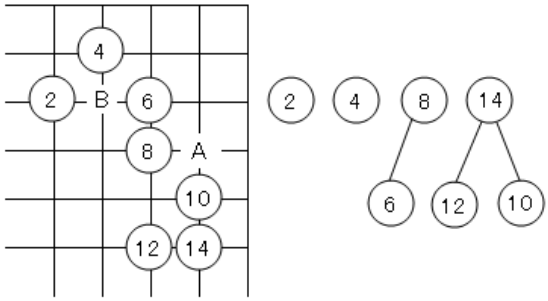

図 2: 盤上のブロックと木構造の関係

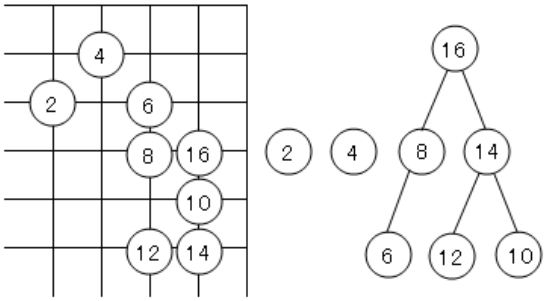

図 3: ブロックの統合時の木構造

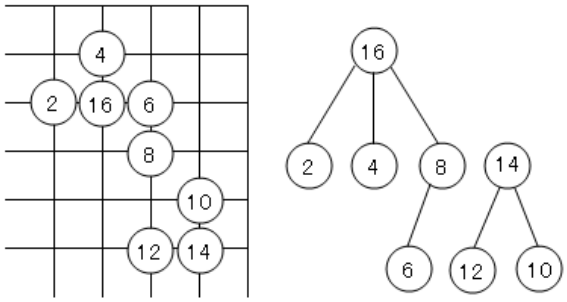

図 4: 複数のブロックの統合時の木構造

座標が集められる．石の打ち上げに伴う空点座標の追加も容易である．

3　木構造を用いたブロックの管理

効率的にプレイアウトを行うためにはブロックの管理方式が重要である．一般に, 盤上に置かれた石は, 座標, ブロック番号など, 各ブロックは色, 構成する石の数, ダメの数などの属性をもつ．ブロック同士の統合はブロックの番号を統一することで実現される．しかし, 盤上の石 1 つ 1 つにこれらの情報を付加すると大きな石の統合や打ち上げの際には, ブロックを構成する石 1 つ 1 つをたどって値を更新しなくてはならないため, 計算コストが大きくなる．

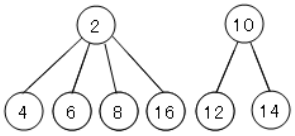

図 5: ブロック番号を判断するための木構造

これら問題点を解決するために, われわれはブロックの管理のために次のような木構造を用いる．

1. 盤上の石は着手番号をもつ．黒石には奇数, 白石には偶数の着手番号が与えられる．
2. 盤上のそれぞれのブロックは図 2 のような木をもつ．
3. 新たに置かれた石によってブロックの統合が起こったときは新たに置かれた石を木の根として既存の木を統合する．例えば, 図 2 の A 点に石 16 が置かれたときは図 3 に示すように木を更新する．また, 石 16 が図 2 の B 点に置かれ, 複数のブロックの統合が起こったときは図 4 に示すように木を更新する．

各石からそのブロック番号を判別するために, 上記の木に重ねて次のような木構造を作る．

1. 木を構成する各節点はその木の中で最小の石番号をもつ節点へのポインタをもつ．
2. ブロックが図 4 のように統合した場合, 図 5 のように大きなブロック番号をもつ節点から小さなブロック番号をもつ節点へのポインタが作られる．

このようにブロックを 2 重の木構造で管理することでブロックの統合を定数時間で行うことができる．

4　むすび

本研究では，プレイアウトにおける候補手と連の効率的な管理方法を示した．今後の課題として，本研究で示した候補手とブロックの管理手法を用いたモンテカルロ碁プログラムの評価を行うことが挙げられる．

参考文献

[1] Remi Coulom，Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search, pp.2-3, 2006

[2] 中村貞吾，コンピュータ囲碁，人工知能学会誌 24 巻 3 号 pp.341-343，2009

Copyright ©2011 Information Processing Society of Japan.
All Rights Reserved.