九路盤囲碁プログラム”Moskit”について

電気通信大学 情報理工学研究科 情報通信工学専攻 伊藤研究室所属
森 大道

1. はじめに

ご覧頂きありがとうございます. こちらでは簡単ではありますが, 九路盤囲碁プログラム”Moskit”について説明をさせて頂きます. その前に諸注意としまして, 本プログラムでは盤の情報の取得, パターンの判定方法などに関しては, 別の参加プログラム”blast”のものを参考にさせて頂いている形となっております. 作成者の下川氏からの承認は勿論得ておりますが, そのため私が作成したアルゴリズムは殆ど囲碁の思考ルーチン部分のみとなっておりますで御容赦下さい。

2. アルゴリズム

モンテカルロ法

今や囲碁プログラムの基本的な概念となっているモンテカルロ法ですが, 本プログラムでもそれを用いております. 囲碁におけるモンテカルロ法を簡単に説明すると, 現在の盤面の状態を保持した状態で, 乱数を用いて適当に着手箇所を決め, 黒白交互 (即ち相手の手も適当な乱数によって生成します) に打ち, 終局まで導いた後勝ちか負けかを判断するというシミュレーションを多数行うことによって, 各着手箇所の勝率を割り出し, 次の着手箇所を決めるというものです.

本来ならば下の図 1 のような木構造を用いて, 一回のシミュレーション中に着手可能な手 (合法手と呼びます) を自分の手, 相手の手と次々と枝分かれさせ, 保持することが一般的なのですが, 私が今回用いたアルゴリズムではその枝分かれを行っていません. つまり, 現在の盤面から終局まで, 適当な合法手を打ち続けた結果を 1 つのシミュレーションの結果として利用しています. そのため, 一つ一つのシミュレーションの結果の信頼度は低くなっていますが, 一方で多くのシミュレーションを行えるという利点があります.

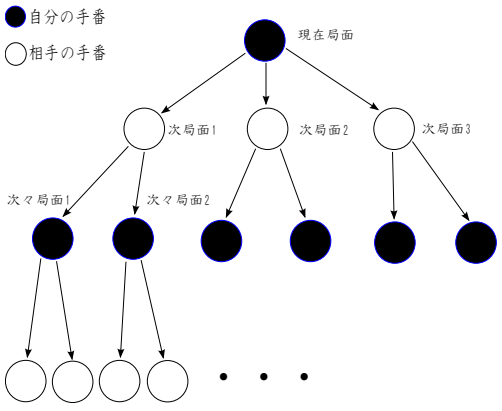

図 1: 一般的なゲーム木

UCB 値

上記で説明したモンテカルロ法をそのまま用いると, 現在の合法手全てに対し均等にシミュレーションを行い, その中から最も勝率の高いものを次の着手としますが, 勝率の低い手へのシミュレーションは無駄なシミュレーションとなってしまいます. ならば勝率の低い着手へのシミュレーションを抑えればよいという話になりますが, 十分な試行回数もないままに勝率を決め, その中から低い勝率の着手を外すというのは本来の勝率との誤差が大きくなりやすいです.

したがって, ここでは勝率とは別の, UCB(upper confidence bound) 値という値を用います. 一般的な UCB 値は以下の計算式で求められます.

$$UCB(i) = W_i + \sqrt{\frac{2\log N}{n_i}} \tag{1}$$

ここで, W_i は着手 i を指したときの勝率, N はその局面で試行したシミュレーションの総数, n_i は着手 i をシミュレーションした回数となっています. つまり, 平方根で示された値が本来の勝率との誤差になっているわけです. 一般的にはこの式の通りに値を求めるのですが, 私はルート内の係数 2 を排除し, 代わりに係数 0.4 を平方根の外にかける形としました. 理由はこのほうがよい結果となりやすいから, としか言えませんが, 恐らくアルゴリズムによってもどの値がよいかは変わってくると思われます.

また本プログラムでは, シミュレーション回数の上限を 100000 回と設定しています. 初めは各マスに着手と, そこから終局までモンテカルロシミュレーションを順に試行していくことでそのマスにおける勝率, 並びに UCB 値を求めていますが, 一巡した後は, その中で UCB 値の高いものから優先して着手するようにしています. そうして UCB 値を更新しながら各点に着手していった後, 最終的に最も訪れた頻度の高い着手点を次の着手として決定します.

3. 終わりに

簡単ではありますが説明は以上となります. 実際のプログラムコードについてですが, モンテカルロ法については Moskit 内 random.c, 及び playout.c を. UCB 値については ucb.c をご参照頂ければと思います.

最後に簡単ではありますが, このような場を提供して頂いた三宅様をはじめとする CEDEC 運営委員の皆様方, 並びに, 囲碁プログラムについて手取足取り教えて頂いた下川氏をはじめとする同大学 村松研究室の皆様に謝辞を申し上げたいと思います.