「技巧」アピール文書

出村 洋介

2012 年 3 月 31 日

1 「技巧」とは

「技巧」は、2012 年 2 月から開発が始められた将棋プログラムです。「3 か月という限られた開発期間で可能な限り強いプログラムを作る」を目標に、現在も開発が続けられています。技巧という名前は、巧みな技を見せてほしいという作者の願いから付けられました。

開発期間の制約が極めて厳しいため、強豪プログラムが採用する標準的な技法を踏襲している部分も多いですが、コーディングの工夫によって高速に動作するようにしています。残りの期間は主に評価関数と ProbCut まわりに注力するつもりです。

以下は、技巧が採用する主要アルゴリズム等についての簡単な説明です。

2 将棋盤

- Redundant Bitboard[1]
- 各種高速化の工夫
  - C++ のテンプレートの活用により、実行時の条件分岐を可能な限り排除しています [4]
  - Rotated Bitboard, Magic Bitboard の不使用
    - 飛び駒のブロックパターンを計算で求めると、レジスタ上で演算されるため少しだけ速いと思います。
  - 指手の逐次生成

3 通常探索

- 全幅探索 + 静止探索
  - 現在のところ、2 駒関係 $+\alpha$ の評価関数を利用した場合、探索速度は序盤 80 万局面/秒、終盤 150 万局面/秒程度です[*1]。
- NegaScout
- 反復深化
- トランスポジションテーブル

[*1] 測定環境: Core i7 2700K 3.5GHz, シングルスレッド, Ubuntu 64bit

- 手の並び替え
  - Hash Move
  - Killer Move
  - SEE ないし MVV-LVA (Most Valuable Victim - Least Valuable Aggressor)[3]
    - あまり見かけませんが、MVV-LVA は計算量が極めて小さいため、時と場所によっては SEE よりも効率的ではないかと思います。
  - 将棋依存の手（取る手・成る手・王手）
- 前向き枝刈り
  - Null Move Pruning
    - 極めて安全に枝刈りができるとの評価があったため [8]、採用を決めました。
  - ProbCut (予定)
    - 学習によって高精度な評価関数を作成できるようになったため、評価関数によって将来局面の近似が比較的正確にできるようになっています。そこで、前向き枝刈り法として ProbCut に着目しています。ProbCut を採用する場合、マージンをどの程度にとればよいのかが問題になりますが、局面の実現確率と組み合わせるとうまくいくのではないかと考えています。

4 評価関数

- 平均化パーセプトロンを用いた学習（激指メソッド）[6]
  - 学習にかかる時間が短いのがメリットです。
- 特徴ベクトルの工夫[*2]
  - 逆転の発想 (?) で、「駒と駒」の関係だけではなく、「駒と空きマス」の関係も考慮しています。実際に学習してみると、例えば飛車の周りの空きマスにはプラスの評価が与えられることがわかりました。
- 高速化の工夫
  - 差分計算（局面を進めるとき）
  - キャッシュの利用（局面を戻すとき）

5 詰め探索

- df-pn 探索 [7]
- サイクル問題対策 [5]
  - 岸本・Muller の GHI 解決策
  - 最小距離法

[*2] 現在も調整が続いているため、変更が生じる場合があります。

参考文献

[1] Bonanza ソース完全解析ブログ, http://d.hatena.ne.jp/LS3600/20091225

[2] Chess Programming Wiki, Flipping Mirroring and Rotating, http://chessprogramming.wikispaces.com/Flipping+Mirroring+and+Rotating

[3] Chess Programming Wiki, MVV-LVA, http://chessprogramming.wikispaces.com/MVV-LVA

[4] 金子知適, An Introduction to OSL, http://www.graco.c.u-tokyo.ac.jp/~kaneko/papers/csa200903.pdf

[5] 岸本章宏, サイクル空間における AND/OR 木探索, ゲーム計算アルゴリズム, pp.127–144 (2010)

[6] 鶴岡慶雅, 激指の技術的改良の解説, 情報処理, Vol.51, No.8, pp.1001–1007 (2010)

[7] 長井歩, df-pn アルゴリズムと詰将棋を解くプログラムへの応用, コンピュータ将棋の進歩 (4), pp.96–114 (2003)

[8] 保木邦仁, Crafty と比較した Bonanza の有効分岐因子, 人工知能学会誌, Vol. 26, No.3, pp.295–300 (2011)