手抜きについて

手抜きチーム

2018 年 2 月 7 日

手抜きは CSA プロトコルで通信する簡単な将棋プログラムです。手を抜いて作っています。開発の目的は私が将棋のプログラムの仕組みを理解することです。今年の目標はルール通りに指すことと，投了すること（去年は投了しませんでした）と，KP で評価することです。KP の評価ベクターになのは mini の fv_nano.bin を使います。選定の理由は fv_nano.bin が KP だからです。

手抜きは前大会では C++ で実装していましたが，C++ がスタックトレースを吐かないのと.h を書かなければならないのと STL と Boost が嫌になって今大会では C#で実装しました。これにより 3 倍くらい遅くなりました。開発が落ち着いたらまた C++ にします。

（2018 年 3 月 19 日追記）ところが D 言語にしました。D は.h を書かなくていいところや前方宣言がいらないところや C++ と同じ速度で動くところがいいです。リポジトリ：https://github.com/hikaen2/tenuki-d

特長

- コンパクトなコードを目指しています
- 盤面のデータ構造に升の 1 次元配列を採用しています
- $\alpha\beta$ 法で全幅探索します
- 利きを持っていません
- 詰めルーチンを搭載していません
- 定跡を組み込んでいません
- 持ち時間を管理していません

開発者の紹介

鈴木太朗

鈴木はソフトウェア会社の会社員です。仕事のかたわら趣味の自転車と合唱に打ち込んでいます。趣味のかたわら手抜きの開発に打ち込んでいます。将棋は駒の動かし方が分かる程度です。Twitter: @hikaen2

玉川直樹

玉川は P 兼召喚士です。KPP とレーティングシステムを疑っています。将棋ウォーズ 2 段。プログラミングはまったくできません。Twitter: @Neakih_kick

去年からの変更点

1. ルートノードの探索を $\alpha\ \beta$ にした

去年はルートノードは minimax で探索していました（ルートでないノードは $\alpha\ \beta$ で探索していました)。今年はルートノードも $\alpha\ \beta$ 探索するようにしました。これによって探索時間はこうなりました（指し手生成祭りの局面, Core2 SL9400, 深さ 6 までの反復深化）:

変更前： 探索時間 784 秒, 探索ノード数 1,803,710,077

変更後： 探索時間 85 秒, 探索ノード数 138,771,448

2. 反復深化のときに前回の最善手から探索するようにした

たとえば深さ 5 の探索のときに，深さ 4 の探索で得た最善手から探索します。

変更前： 探索時間 85 秒, 探索ノード数 138,771,448

変更後： 探索時間 81 秒, 探索ノード数 130,748,443

3. 合法手を生成するとき，駒を取る手を先に生成するようにした

変更前： 探索時間 81 秒, 探索ノード数 130,748,443

変更後： 探索時間 68 秒, 探索ノード数 59,871,223

4. 静止探索を入れた

深さ 4 の静止探索 (Quiescence Search) を入れました。

変更前： 探索時間 68 秒, 探索ノード数 59,871,223

変更後： 探索時間 92 秒, 探索ノード数 107,209,671

5. KP で評価するようにした

KP で評価するようにしました。評価ベクターになのは mini の fv_nano.bin を使っています。

変更前： 探索時間 92 秒, 探索ノード数 107,209,671

変更後： 探索時間 287 秒, 探索ノード数 197,226,589

6. 置換表を実装した

move のみの置換表 (Transposition Table) を実装しました。

変更前： 探索時間 287 秒, 探索ノード数 197,226,589

変更後： 探索時間 86 秒, 探索ノード数 60,318,785