President_X 概要

2016年5月7日 V2.0

天野 史斎

イントロダクション
①President_Xの目的

- コンピュータ将棋プログラム開発から調整の手間をなくす！未来永劫なくす！！

※ 完成時イメージです。

イントロダクション
②背景

- 最善手を求める計算はNP困難である。
  - たとえば…
    N手詰で相手を詰ませることだけがわかっているとき、次手mでN-1手詰になるか否か判定したい！
    →Nの指数関数オーダーの計算量がやっぱり必要…
    （故に最善手mの決定はNP完全よりﾑｽﾞい問題）
- P=NPの見込みはほとんど無い。
  人間はNP困難というお釈迦様の手のひらの上で勝負せねばならない。
  →完全解析未満の計算量で済む決定版アルゴリズムは存在が期待できない。
  → ヒューリスティクスの優劣と実装の完成度で勝敗が決まる。
  （完全解析せぬ限り正解がわからないのだから、これはもう近似ですらない。）

勝ち続けようとする限り、調整の無間地獄が続く、、

イントロダクション
③ソリューション

- 調整の自動化
- モデル生成の自動化 ←New!
- 精確で高効率な探索

イントロダクション
④拓かれる未来

次のようなプログラムができる。

- 放流しておけば勝手に強くなる。
- 気が向いたときバッチ学習させたらさらに強くなる。

次のような人にﾏｼﾞお勧め！

- 本業が忙しすぎてコンピュータ将棋開発に手が回らない開発者。
- コンピュータ将棋ソフトのアクティブなヘビーユーザー。

※ 完成時イメージです。

プログラム名

- 名前：President_X
- よみがな：だいとうりょうえっくす
- 略称：X
- 由来：次のどれか。

1. 大いに投了させる。
2. ト○ン○旋風にあやかった。
3. GALAXの超AI「総裁X」

ソースコード

- 公開予定は未定。
  1ヶ月ぐらい様子を見て冷静になって判断したい、

構成要素
①評価関数

Basicなヒューリスティクスを使用。

評価値＝駒得＋駒位置の利得①＋駒位置の利得②＋進行度利得

- 駒得は例の数値で固定値
- 駒位置の利得①は、隣接する駒との関係において、駒がそこに居ることの価値を得点化したもの
- 駒位置の利得②は、KPPを得点化したもの
- 進行度利得は、玉までの手数を得点化したもの
  行き着かない場合は適当に補正した値

※ Futility purningするのでそれに都合が良い構成とした。「お釈迦様の手のひらの上」（前出）においては正解かどうかは定説よりは結果で決まる…

ソースコード対応箇所：“sim_eval.h”のevaluate()

構成要素
②通常探索ルーチン

- 基本的にNegascout。
- 下記を特色とする。
  1. 分割差し手生成
  2. Null move purning開始条件の自動調整機能
  3. 探索窓幅を極力広げないケチケチ探索（Search Instability対策済み）
  4. 王手状況での探索延長を阻害しないfutility purningの実装
  5. 子ノード以下の王手状況をROOTに持ち帰る機能（置換表経由でも無問題）
     ※ 通常探索の評価値はPV先端の王手状況しか反映しない。

ソースコード対応箇所：

1, 2: “sim_search.cpp”のsim_search()　-- これが通常探索ルーチンの大元。

3: ”aspiration_search.cpp”内の2関数

4: “SimSearchState.cpp”のon_frontier_node()およびその呼び出し元と呼び出し先。

5: “SimPV_w_depth.h”のSimPV_w_depthクラス、および“SimMove.h”で定義されるノードステータスフラグ利用箇所

構成要素
③置換表

下記を全部区別する。

- 駒の配置
- 葉からの深さ（rdepth）
- 手番
- 同じ局面の初手以降の出現回数

これで巡回手順が探索に影響する余地は無くなった…ハズ…

ソースコード対応箇所：
“search_util_on_ttab.h”のget_hash_and_ent()

Search Instabilityとは…

次の2つに尽きる（と思う）。

1. 置換表に誤った情報が積まれて探索を誤る。
2. Null move purning成功時に返される当座の探索窓のβ値等、探索窓と結びついたad-hocなreturn valueを、異なる探索窓の下で評価値上下限の境界値として不用意に採択した結果探索を誤る。

どっちも対策した。

ソースコード対応箇所:

1: 前出sim_search() 内の置換表エントリ更新箇所、
　および前出get_hash_and_ent()　（エントリをrdepth別としている点）

2: “aspiration_search.cpp”内の2関数

その他

下記は普通の作り。

- Hash move
- Killer move
- Ordering
- Df-pn search
  ただし置換表は前々頁のやつ。
- 時間制御

下記は、差し手の先の王手状況を反映する工夫を行ったがそれ以外は普通。

- ROOTでのPV並べ替え
  ソースコード対応箇所：“PVRanking.h”のPVRankingクラスとその利用箇所。

で、肝心の学習はどうなの？

下記を実装。

- Null move purningのためのnull move発動条件のオンライン学習機能
  - 単純ベイズ学習器使用。
    - Null moveの成否をnull moveの都度記録。
      学習データは当座の駒得、β値に対するマージン、累積的な王手状況。
    - 置換表クリアタイミングでnull move成否確率更新。
    - 成功率が低い場合はnull moveしない。

  ソースコード対応箇所：“BayesGeneric.h”のBayesGenericクラスおよびその使用箇所。

- 駒位置の利得②（KPP）のオンライン学習機能
  - これも単純ベイズ学習器。
    - 当方が負けたとき、当方視点のKPPを負けパターン、相手視点のKPPを勝ちパターンとして学習する。（わかる範囲での敗着までの手順も重み付けして考慮。）
    - 当方が勝ったときは何もしない。（相手が不当に弱かっただけかもしれない。）

  ソースコード対応箇所：“BayesKPP.h”のBayesKPPクラスおよびその使用箇所。

！！！秘中の秘！！！
モデル生成の自動化

- 調整結果の入れ物としてはKPPでも実は十分？
  - 次元下げした結果もそうでない結果も問題なく収容可能
    （そうでないという根拠は目下見出されていない。）
- しかし、結果を得るまでの途中効率が悪い。
  - 動かせるパラメータが多すぎ、事実上ボナンザメソッドでないとうまく扱えない。
- そこで、サンプルを説明する最小パラメータ個数のモデルを自動生成させる
  （真の次元下げ）

クラスタブームの次に来るのはこれに違いないいやそうに決まっている。

President_Xは先取りして一応搭載。

作者のズボラな無調整ぶりのわりに

「駒得評価++」が曲りなりにも成立しているのはこれのおかげ。

TODO:

- 駒位置の利得②が目下全く効いていないので効かせる。
  - 実は勝ち局面と負け局面の判別までは行えているのだが、駒の数に比例した評価値にならないため駒得に隠れてしまっている。（局面全体で見た勝ち負け確率のlogの差なのが問題？（ソースコード対応箇所：BayesKPP::GetValue()））
  - 評価値化方法を変えるか学習データを大量にぶち込んでみるか学習方法自体を変える。
- もっと深く探索できるようにする。
  - 現状平均的に7～8手。
  - 通常探索ルーチン本体sim_move()の呼び出し回数で計測するNPSはCore-i7 6700Kで20万～30万NPS。
    Futility purningで刈られるノードも含めてよければ200万ノードぐらいは見切れている勘定だが…
    （NPSの正しい計算方法とか相場とかは知らない）
  - ﾁｮｰ安全で高効率なreduction手法を導入する。
  - 前向き枝刈手法の導入を前向きに検討する。

WCSC26 一次予選大反省会

- バグを直した（と思った）らNPSぐらいは確認しよう…

  ソースコード対応箇所：“SimSearchState.h”のoute_calling_test_ex()、、、

おわりに

すばらしい対局の機会を毎年与えてくださる関係者の方々と、
熱戦を繰り広げられた参加者の皆様、ならびに
大会を盛り上げてくださる将棋ファンの皆様に
この場を借りまして御礼申し上げます。

ありがとうございました。

第一部・完