CSA 例会資料

鶴岡 慶雅

平成 15 年 9 月 16 日

1　片側を制限した探索

1.1　動機

類似ハッシュを利用した簡易詰みチェックでは、攻め側の指し手を類似ハッシュから取ってきた 1 手だけに限定して探索を行なうことで、きわめて効率的に詰みチェックを行なうことができる。その理由は、片側の指し手を 1 手に限定することで、実質的な探索深さが半分になるからである。そのような、片側を制限した探索を通常の探索に応用できないだろうか？

探索アルゴリズムの中では、null-window search のように、探索結果がある評価値を超えるかどうかだけがわかればいい、という場合が多い。その場合、「ある評価値を超えない」ということに関しては、相手側の指し手を限定して探索してもかまわない。なぜなら、相手側の指し手を限定するということは、自分にとって有利な探索をしているのだから、仮に限定しないで探索しても「ある評価値を超えない」という結論が変わることはないからである。

PVS アルゴリズムでは、principal variation 以外は、null-window search を利用して、その指し手が今までの best value を更新する可能性があるかどうかをチェックするということを行なう。そこで、片側を制限した探索を PVS アルゴリズムに適用してその効果を検証する。

1.2　プログラムリスト

以下に PVS に実装した場合の探索ルーチンのリストを示す。通常の PVS と異なる点は、手番 (turn) が限定されている側 (rside) の場合は、類似ハッシュから取り出してきた 1 手のみしか探索しないという点である。類似局面としては、兄弟局面のうち「パス」に対する局面を利用する。

注意しなければならないのは、transposition table に探索結果を保存するときで、例えば、相手側の指し手を限定した探索を行なっている場合は、ベータカットの情報を保存してはいけない。その他の場合も同様の注意が必要。

```
val_t
Searcher::pvs(Shogiban *ban,
              const int depth, const int leaf_depth,
              val_t alpha, val_t beta,
              ShogibanHashKey akey, int rside,
              List<Move>& best_ml)
{
  int turn = ban->turn();
  if (ban->ou_masu(enemy_side(turn))->has_kiki(turn))
    return 100000 - depth;   // 相手の玉を取れる

  if (depth >= leaf_depth) return p_eval(ban); // 末端

  // ハッシュルックアップ
  int t_depth, t_eval, t_type;
  Move t_move;
  if (p_lookup_hash(ban, leaf_depth - depth, &alpha, &beta, &t_eval, &t_move)) {
    best_ml.clear();
    best_ml.push_back(t_move);
    return t_eval;
  }

  list<Move> sml;
  list<Move> ml;  // 指し手リスト

  // 類似局面での最前手を transposition table から探す
  if (tptable->retrieve_search_info(akey, &t_depth, &t_eval, &t_type, &t_move)) {
    if (is_legal_move(ban, t_move)) {
      ml.push_back(t_move);  // 類似局面での最善手
      if (rside == ban->turn()) {
        goto done_move_generation; // 他の手は生成しない
      }
    }
    rside = NONE;
  }

  pvs(tid, ban, depth, leaf_depth - 1, alpha, beta, akey, rside, sml); // 浅い探索
  ml.push_back(sml.front());                    // 浅い探索での最善手
  ml.push_back(Pass);                           // パス
  generate_moves(ban, leaf_depth - depth, ml);  // その他の手

 done_move_generation:

  int best = alpha;
  ShogibanHashKey ckey = ban->key();
  for (List<Move>::const_iterator i = ml.begin(); i != ml.end(); i++) {
    ShogibanHashKey akey1 = akey;
    akey1.update(ban, *i); // 類似局面のハッシュキーを更新
    ban->move(*i);
    list<Move> eml;
    int val;

    if (best <= alpha) {
      val = -pvs(tid, ban, depth + 1, leaf_depth, -beta, -best, akey1, rside, eml);
```

```
    } else {
      ShogibanHashKey akey = ckey;
      akey.update(ban, Pass);
      val = -pvs(tid, ban, depth + 1, leaf_depth, -(best+1), -best, akey, enemy_side(turn), eml);
      if (val > best) {
        val = -pvs(tid, ban, depth + 1, leaf_depth, -beta, -best, akey1, rside, eml);
      }
    }

    ban->reverse();
    if (val > best) {
      if (val >= beta) {
        if (rside != enemy_side(turn)) {
          store_search_info(ban->key(), leaf_depth - depth, val, SearchInfo::LOWER, *i);
        }
        return val;
      }
      best_ml = eml;
      best_ml.push_front(*i);
      best = val;
    }
  }

  if (best > alpha) {
    if (rside == NONE)
      store_search_info(ban->key(), leaf_depth - depth, best, SearchInfo::EXACT, best_ml.front());
  }
  else {
    if (rside != turn)
      store_search_info(ban->key(), leaf_depth - depth, best, SearchInfo::UPPER, Pass);
  }

  return best;
}
```

2　実験結果

表 1, 2 に問題集による実験結果を示す。探索時間は、限定探索をすることにより相当高速化されている。探索深さ 6 だと 2.5 倍 (!)。探索結果もほとんど変わっていない (まれに微妙にかわることがある。バグ？あるいは実装上の問題か)。

探索深さ	限定なし	限定あり
4	9	5
5	114	71
6	2133	866

表 1: 「コンピュータ将棋の進歩２」問題集４ ８問にかかった時間 (秒)

問題番号	限定なし	限定あり
1	(2959) ▲４三角成	(2959) ▲４三角成
2	(81) ▲５四金	(81) ▲５四金
3	(-239) ▲９五歩打	(-239) ▲９五歩打
4	(162) ▲２三歩成	(162) ▲２三歩成
5	(469) ▲２二角打	(469) ▲２二角打
6	(3597) ▲４一竜	(3540) ▲４一竜
7	(2405) ▲１一飛成	(2405) ▲１一飛成
8	(220) ▲７三歩打	(220) ▲７三歩打
9	(3017) ▲４二と	(3017) ▲４二と
10	(3998) ▲２五桂打	(3998) ▲２五桂打
11	(971) ▲５一角成	(971) ▲５一角成
12	(1757) ▲２二桂成	(1757) ▲２二桂成
13	(5248) ▲６一角成	(5248) ▲６一角成
14	(3195) ▲１一角成	(3195) ▲１一角成
15	(1925) ▲３二金	(1925) ▲３二金
16	(2477) ▲５九玉	(2477) ▲５九玉
17	(1885) ▲４四歩打	(1885) ▲４四歩打
18	(89) ▲２四歩打	(89) ▲２四歩打
19	(994) ▲７四桂打	(994) ▲７四桂打
20	(1880) ▲５四歩	(1880) ▲５四歩
:	:	:

表 2: 「コンピュータ将棋の進歩２」問題集の探索結果 (深さ 5)

3　おわりに

PVS アルゴリズムで全幅探索に近い場合はかなり有効かもしれない。激指での実装はまだうまくいっていない。