Vol.2014-GI-31 No.7
2014/3/17

コンピュータ将棋における高次元組み合わせ評価のための評価項目自動抽出に関する研究

後藤　嵩幸[1,a)]　橋本　剛[1]

概要：コンピュータ将棋は，Bonanza メソッドの考案により非常にレベルが高くなった．現在では，プロ棋士との対局も盛んに行われており，トッププロレベルに迫っている．しかし，中盤以降に比べて序盤が弱いという弱点も抱えている．コンピュータ将棋の性能は評価関数の質に大きく影響される．正確に盤面評価を行うほどコンピュータ将棋は強くなる．評価関数のパラメータは機械学習により自動調整されているが，40 駒からなる将棋の盤面をそのまま評価するためには膨大な評価項目が必要となる．現状では部分局面に分割することにより，近似的に評価を行なっている．しかし，現在の多くのプログラムは駒組み合わせ全てを保持するオール（all）型の評価関数を用いているため，3 駒程度の少ない駒数にしか分割できない．そのため序盤で特に重要な盤面の細かな違いを認識することが出来ない．高次元組み合わせ評価を行うためには，重要な駒組み合わせのみを評価するフレック（frequency）型評価関数が必要である．フレック型評価関数実現のためには，重要な評価項目を抽出する必要がある．しかし，現在重要な評価項目を自動抽出するための有効な手法は存在しない．そこで本研究では，乱数を用いた盤上の駒組み合わせのランダムカウントにより，評価項目を自動抽出する手法を提案する．実際に提案手法により抽出した項目を用いてフレック型評価関数を設計し，Bonanza に組み込んだ．そして，オール型評価関数と対局実験を行い，性能向上を確認した．

1. はじめに

チェス・将棋・オセロ等の二人零和完全情報ゲームのプログラムでは，評価関数を使う minimax 法が一般的な手法である．チェスでは，1997 年に手動調整された評価関数を用いたプログラム「ディープブルー」が当時の世界チャンピオンに勝利した．しかし，同じ頃のコンピュータ将棋は弱く，アマチュアレベルであった．以前の評価関数は開発者により手動でパラメータ調整されることが多かった [1]．そのため，開発者にそのゲームの知識が必要で，評価項目数を増やすことが困難であった．その後，しばらくコンピュータ将棋は弱いままだったが，2006 年保木により機械学習による評価関数の自動学習法（Bonanza メソッド）が考案された [2]．この学習法を導入したプログラム「Bonanza」はその年に世界大会に出場し優勝した．これにより，Bonanza メソッドの有用性が認められた．Bonanza メソッドにより，評価関数を自動調整できるようになったため，開発者の将棋の知識にかかわらず強いプログラムが開発できるようになった．現在の強豪ソフトはほとんど Bonanza メソッドを導入しており，大会の上位を占めている [3][4][5]．また，コンピュータ将棋全体のレベルも格段に向上し，現在ではプロ棋士との対局で勝利するなど名人レベルに迫っている．このように，コンピュータ将棋は強くなっているが，まだ弱点も存在する．

2013 年に行われた将棋電王戦の第 1 局において「習甦」と阿部光瑠四段が対局した際，阿部四段が習甦の序盤の弱点を研究し対策を講じたため，習甦は完敗した．第 3 局の「ツツカナ」と船江恒平五段の対局においても，船江五段が同様に対策を講じ成功したが，ツツカナに最終盤で逆転され敗北した．その後のリターンマッチでは船江五段が序盤から圧倒し，勝利した．このように．コンピュータ将棋は序盤が弱点であると一般に認識されている [6]．

人間は，盤面を評価する際に駒をある程度の塊として見て評価を行っている [7]．それに対し，コンピュータは組み合わせパターンの多さから，大きな塊で直接評価することはできない．Bonanza メソッドが考案される以前は，評価関数を手動調整していたため，1 駒の位置等の少ない情報でしか評価が行えなかった．Bonanza メソッドにより，より多くの評価項目を扱えるようになったが，まだ多くのコンピュータ将棋は盤面を 3 駒以下の少ない駒による部分局面に分割して盤面評価を行っている．そのため，盤面の細

[1] 松江工業高等専門学
MCT, Nishiikuma, Matsue, Shimane , 14–4, Japan
[a)] maxmiu2000@gmail.com

かな違いを認識することが出来ない．この違いは序盤で特に重要である．

現在のコンピュータ将棋の多くは駒組み合わせ全てを評価している．ここでは上記の手法をオール（all）型と呼ぶ．この方法では，4 駒以上の高次元な駒組み合わせを行うことは，組み合わせ数が膨大になり不可能である．そのため，人間のように駒を大きな塊で評価することができないこの問題を解決するためには，無駄な組み合わせを排除し重要な組み合わせのみを評価する方法が必要である．ここでは，この手法をフレック（frequency）型と呼ぶ．現在では，フレック型評価項目を手動で設定して用いているプログラムはあるが，有効な手法は提案されていない．矢野らは駒組み合わせの重要度を用いて抽出する研究を行ったが，実用的ではなかった [8]．そこで，本研究では乱数を用いた駒組み合わせの出現頻度カウントによる評価項目抽出法を提案する．本稿では，提案手法でカウントを行い，抽出された駒組み合わせを示す．その後，実際に抽出された項目を用いてフレック型評価関数を Bonanza に実装し，オール型を組み合わせたプログラムとオール型のみのプログラムと対局実験を行い，有用性を示す．

2 章ではオール型とフレック型の評価関数について説明するとともに，フレック型実現のための問題点を述べる．

3 章では矢野等による，駒組み合わせの結合度を用いた重要な評価項目自動抽出のための研究について述べる．

4 章では，新しい評価項目抽出法についてのアイディアを提案し，抽出される駒組み合わせを確認した．

5 章では提案手法により抽出した評価項目を用いて実際に評価関数を設計を行った．

6 章では実装したフレック型評価関数をオール型評価関数と組み合わせて Bonanza に実装し，オール型評価関数のみのプログラムと対局させて有用性を検証した．

7 章ではまとめと今後の課題を述べる．

2. 評価関数

2.1 オール（all）型評価関数

現在の多くの将棋プログラムは，盤面評価に盤上の駒組み合わせの位置評価を用いている．その多くは評価項目として全ての組み合わせを保持しているオール型評価関数である．そのため項目数が膨大であり，高次元な組み合わせ評価を行うことが出来ない．例えば，Bonanza は王を含む 3 駒を評価しているが，それだけで 500 万項目にもなる．しかし，組み合わせ全てが評価に用いられるわけではない．

Bonanza メソッドでは評価関数の調整にプロ棋士の対局の棋譜を用いる．そのため，対局に現れない項目は学習することが出来ず評価値が設定されない．実際に，Bonanza の評価項目 500 万のうち殆どの項目は評価値 0 である．このようにオール型評価関数は非常に多くの無駄を含んでいるといえる．この無駄な項目を省き，重要な項目のみを評価出来れば，4 駒以上の高次元組み合わせを評価できる．

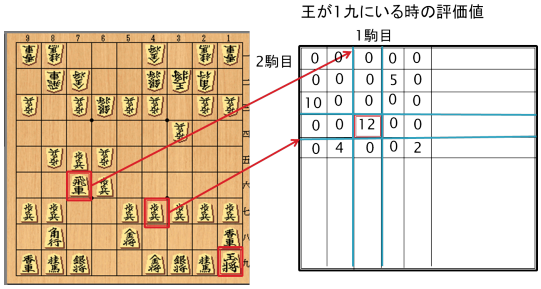

図 1 オール型評価関数
Fig. 1 All type evaluation function

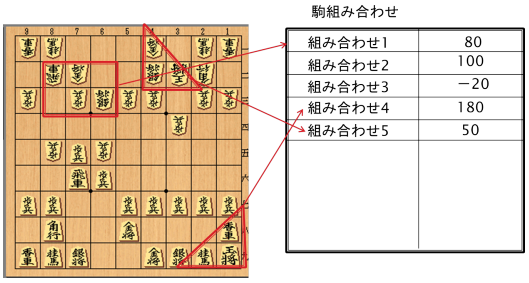

図 2 フレック型評価関数
Fig. 2 Frequency type evaluation function

2.2 フレック（frequency）型評価関数

上述したように，現在の多くのコンピュータ将棋はオール型を用いている．しかし，オール型の評価関数では，盤面評価を少ない駒組み合わせで行わないといけないため，重要な盤面の細かな違いを認識できない．特に序盤では細かな違いが大きく影響すると思われる．そこで，高次元な組み合わせ評価を可能とする新たな評価関数が必要である．

高次元な駒評価を行うためには，無駄な項目を省いて重要な組み合わせのみを評価する必要がある．重要な組み合わせとは，対局中に頻繁に現れる駒組み合わせといえる．これら対局中に現れる頻度の高い組み合わせのみを保持したフレック型の評価関数が実現できれば，重要な組み合わせのみを細かく評価できる．しかし，フレック型評価関数実現には後述するような課題がある．

2.3 評価関数の比較

図 1 と**図 2** にオール型とフレック型それぞれのイメージ図を示す．オール型は図 1 のように盤上の 2 駒の位置と種類を参照する巨大な行列になっており，それが王将の位置毎に用意されている．それらの駒組み合わせには使われない組み合わせも多数存在しており，評価値 00 ばかりのスパースなものになっている．一方，フレック型は図 2 のように重要な駒組み合わせのみ保持しているリスト状になっている．重要な組み合わせしか無いので無駄な項目はなく，組み合わせ数も一定ではなく可変に扱える．

2.4 フレック型評価関数の問題点

上述したようにフレック型評価関数ならば大きな塊での評価が可能である．しかし，膨大な組み合わせから重要な評価項目を抽出するのは困難である．現在でも「GPS 将棋」[3] のようにオール型とフレック型を組み合わせて用いているプログラムはあるが，フレック型の評価項目は手動で設定したものであり，自動で項目を抽出する有効な手法は提案されていない．

対局中によく現れる駒組み合わせを抽出するためには，出現頻度をカウントする必要がある．しかし，将棋の駒組み合わせは 2 駒で約 500 万，3 駒で約 100 億と 3 駒以上ではメモリにのりきらず扱うことが出来ない．そのため，高次元の組み合わせを単純にカウントすることは不可能である．このように，重要な評価項目自動抽出はコンピュータ将棋において，大きな課題の一つとなっている．

3. 関連研究

矢野等は盤面を駒やマスによって構成されるグラフと仮定し，各エッジに結合度を定義することにより組み合わせの絞り込みを行った．具体的には，将棋の盤面を駒の種類や位置，手番等をノード，組み合わせ特徴をパスとしてグラフ化する．このとき，オール型手法を用いると，ゲームの構成要素を全対全でつないだ完全グラフになり，グラフが非常に巨大となってしまう．そこで矢野等は，2 駒間の結合度を学習により定義し，それらの積をパスの重要度として枝刈りを行うことにより，重要な組み合わせを抽出した．この手法は，理論的にはこの特徴数の増加を防ぎつつ，高次元な組み合わせ評価を行うことが可能となる．しかし，実際は特徴の計算や盤面のグラフへの変換に時間がかかるため，既存の手法に比べ学習時間が爆発的に増加する．そのため，実用的とは言いがたい．

4. 評価項目の自動抽出

4.1 提案手法

前述したとおり，重要な組み合わせを抽出するために全通りの駒組み合わせを単純な方法でカウントすることは不可能である．そこで，本研究では全ての組み合わせをカウントするのではなく，ランダムに駒組み合わせを選んでカウントを行う手法を提案する．データをランダムに用いる手法は確率的勾配降下法（SGD 法）と呼ばれる最適化アルゴリズムでも用いられており，その実用性が実証されている [9]．SGD 法とは勾配法の一種であり，自然言語処理等の大規模データを扱う分野で使われる手法である．通常の勾配法では，パラメータの更新にデータ全体を用いるが，SGD 法では更新の度にデータをランダムに選んで用いる．少ないデータをランダムに用いるだけでは，良い学習結果は期待できない．そこで，パラメータ更新回数を十分多くすることで，良い学習結果を得ている．今回のカウントも，

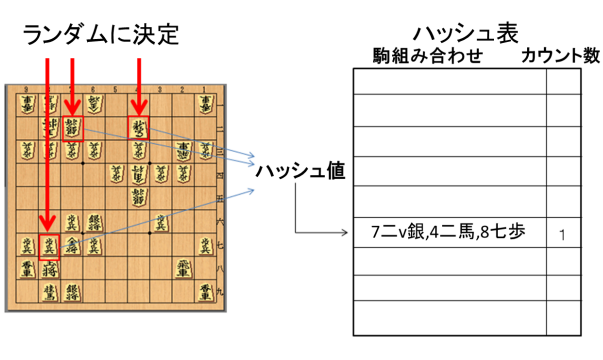

図 3 ランダムカウントイメージ
Fig. 3 A image of random count

十分にカウント回数を確保することにより，重要な項目を抽出出来ると言えそうである．

4.2 提案手法の確認

実際に提案手法を用いてカウントを行い，どのような組み合わせが抽出されるかを確認した．**図 3** にランダムカウントのイメージ図を示す．まず，対局中の盤面から駒をランダムに複数選択する．それらの駒をハッシュ値に変換してカウント用テーブルに保存する．その後盤面を 1 手進めてまた駒組み合わせを選択する．もしその組み合わせがすでにテーブルに登録されていればカウント数を増やし，なければ新しく登録する．以上を繰り返すことにより，出現頻度のカウントを行った．

今回，1 から 5 駒までの駒組み合わせのカウントを行った．3 駒以上はランダムカウントのみを，2 駒以下の組み合わせは全通りカウントが行えるため，ランダムカウントと全カウントの 2 通り行った．

4.3 カウント結果

4.3.1 1，2 駒のカウント結果

2 駒以下の組み合わせは，ランダムカウントと全通りカウントを行った．**表 1** と**表 2** に 1 駒のカウント結果を，**表 3** と**表 4** に 2 駒のカウント結果を示す．表中のインデックスはカウント数が多い順に並べた時の順位である．また，v 駒名は後手の駒を表す．表 1 と表 3 は全通りカウントした結果であり，表 2 と表 4 はランダムにカウントした結果である．

表 1 と表 2 を比較してみると，同じ駒が比較的近い順位に現れていることが分かる．この傾向は 1 駒カウントにかぎらず，表 3 と表 4 のように 2 駒組み合わせでも同じ組み合わせが近い順位に現れていることが分かる．このことから，ランダムにカウントを行っても出現頻度の高い組み合わせが得られると期待できる．

4.3.2 3 駒以上のカウント結果

3 駒以上の駒組み合わせは，全通りカウントするとカウント用テーブルがメモリに乗らないため，ランダムカウント

表 1 1 駒のカウント結果例（全カウント）
Table 1 A part of results with all one piece count.

インデックス	駒
101	3 八金
102	6 三 v 金
103	6 八角
104	4 四 v 銀
105	1 二 v 香
106	6 六銀
	（v は後手駒を示す）

表 2 1 駒のカウント結果（ランダムカウント）
Table 2 A part of results with random one piece count.

インデックス	駒
101	3 八金
102	6 三 v 金
103	6 八角
104	4 四 v 銀
105	1 二 v 香
106	6 六銀

表 3 2 駒のカウント結果（全カウント）
Table 3 A part of results with all two pieces count.

インデックス	1 駒目	2 駒目
998	6 七歩	7 一 v 銀
999	8 七歩	4 三 v 金
1000	1 三 v 歩	5 八金
1001	1 三 v 歩	8 五 v 歩
1002	3 一 v 銀	8 二 v 飛
1003	4 一 v 金	2 七歩

表 4 2 駒のカウント結果（ランダムカウント）
Table 4 A part of results with random two pieces count.

インデックス	1 駒目	2 駒目
1009	4 一 v 金	4 三 v 歩
1010	8 七歩	4 三 v 金
1011	7 一 v 銀	9 三 v 歩
1012	1 九香	4 五 v 歩
1013	7 一 v 銀	2 九桂
1014	5 六歩	8 二 v 飛

のみを行った．**表 5**〜**表 7** にカウント結果の一例を示す．

表 5 は，カウント数が多かった組み合わせである．現れている駒は初期位置の香車や桂馬などが多い．

表 6 はカウント数が中程度であり，対局中にある程度現れた組み合わせである．囲い等の戦術上で有用な形が現れている．

表 7 はカウント数が 1，2 回とほとんど対局中に現れない組み合わせである．めったに存在しないような位置に駒が現れている等，偶然現れたような組み合わせが見られた．

4，5 駒組み合わせに関しても同様にカウントを行った．その結果，3 駒と同様な傾向が見られた．

表 5 3 駒のカウント結果例（高頻度）
Table 5 A part of results with three pieces count. (High frequency)

1 駒目	2 駒目	3 駒目	カウント数
9 一 v 香	9 九香	1 九香	13003
1 一 v 香	9 九香	9 一 v 香	12668
2 一 v 桂	1 九香	1 一 v 香	10666
8 九桂	9 一 v 香	1 一 v 香	10383
1 一 v 香	2 一 v 桂	9 一 v 香	10246

表 6 3 駒のカウント結果例（中頻度）
Table 6 A part of results with three pieces count. (Middle frequency)

1 駒目	2 駒目	3 駒目	カウント数
9 九香	8 八角	3 九銀	1093
7 七銀	3 六歩	7 八金	1093
8 七歩	5 三 v 歩	4 四 v 歩	1093
6 七歩	7 九銀	2 八飛	1093
2 一 v 桂	8 五 v 歩	5 七歩	1092

表 7 3 駒のカウント結果例（低頻度）
Table 7 A part of results with three pieces count. (Low frequency)

1 駒目	2 駒目	3 駒目	カウント数
3 七 v 馬	4 六金	2 四 v 歩	2
8 二 v 銀	3 五飛	7 七銀	2
4 七歩	4 五 v 角	8 八銀	1
8 八銀	5 四 v 馬	3 四 v 歩	1

5. フレック型評価関数設計

3 駒組み合わせカウント結果を用いて，実際にフレック型評価関数を設計した．評価関数の実装には木構造リストを用いた．各ノードには駒の種類と位置を保持しており，抽出された組み合わせをエッジでつないでいる．末端ノードには駒ではなく評価値が格納されている（**図 4**）．評価の際には，ルートノードから辿り一番目のノードの駒が盤上にあるかどうかを調べる．この時，駒が存在していれば一つ下の子ノード，なければ次の兄弟ノードを調べる．これを繰り返し，末端ノードまで到達すれば辿ったノードの駒組み合わせが盤上に存在するということなので，末端ノードに保持している評価値を現評価値に加える．以上のようにフレック型評価関数を実装した．

図 4 の例で詳しく説明する．まずルートノードの 3 八金が盤上にあるかどうか調べる．盤上に存在するため，一つ下に行き子ノードの各駒の有無を調べる．これらの中で，

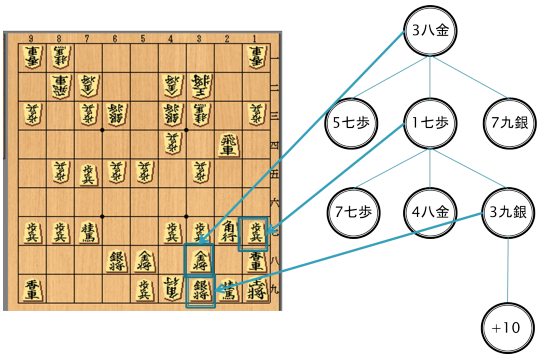

図 4 評価木
Fig. 4 A tree of evaluation function

盤上には 1 七歩しかないため，1 七歩の子ノードを調べる．同様に木を調べることにより，3 八金，1 七歩，3 九銀の組み合わせが盤上にあることがわかり，評価値 10 を得る．

実際に対局に用いる際に，抽出した項目だけでは評価関数として不十分だと考えられる．そこで，オール型の評価関数と組み合わせることにした．全通り扱える王と他の 1 駒の組み合わせはオール型評価関数で評価を行い，それ以上の組み合わせ評価をフレック型で行い足し合わせるようにした．

6. 対局実験

ランダムカウントにより抽出された項目の有用性を調べるため，設計したフレック型評価関数を Bonanza に実装し，オール型評価関数と対局を行った．フレック型評価項目は，Bonanza が学習用に用いている棋譜約 50000 局分から 4 章のようにカウントしてを行い，8192 項目抽出した．学習には Bonanza メソッドを用いた．学習用の棋譜は Bonanza が学習に用いている棋譜を利用し，棋譜数 1000，反復回数 6 回で行った．学習時間としては少ないが，時間の都合上短い学習時間での実験となった．対局条件を以下に示す．

- 対局数:先後入れ替えでそれぞれ 100 局
- 探索打ち切り設定:深さ 5
- 投了値:2000
- 最長定跡手数:無限

6.1 実験 1

まず，王と他の 1 駒を評価項目とするオール型評価関数と，それにランダムカウントで抽出した評価項目を組み込んだフレック型評価関数を追加したものを対局させた．対局結果を**表 8** に示す．プログラム 1 がオール型評価関数，プログラム 2 がオール型評価関数とフレック型評価関数を組合わせたものである．

表の通り，プログラム 1 に抽出した評価項目を加えたプログラム 2 の方が NPS が増加し，遅くなった．しかし，プログラム 2 の方が勝ち越し，性能向上していることが分かる．このことから，ランダムカウントにより抽出された項目は評価に有用であるといえる．

表 8 対局結果 1
Table 8 A result of games(1).

先手	勝ち数（平均 NPS（K））	後手
プログラム 1	36（228）:60（19）	プログラム 2
プログラム 2	57（27）:43（182）	プログラム 1

表 9 対局結果 2
Table 9 A result of games(2).

先手	勝ち数（平均 NPS（K））	後手
プログラム 2	64（15）:36（120）	プログラム 3
プログラム 3	64（180）:36（26）	プログラム 2

表 10 対局結果 3
Table 10 A result of games(3).

先手	勝ち数（平均 NPS（K））	後手
プログラム 2	49（24）:51（11）	プログラム 4
プログラム 5	32（3）:67（11）	プログラム 4

6.2 実験 2

実験 1 で用いたプログラム 2 と，王と 2 駒のオール型評価関数（プログラム 3）を対局させた．プログラム 3 は，Bonanza の評価関数から王 2 つと他の駒を除いたものである．対局結果を**表 10** に示す．表の通り NPS，勝利数ともにプログラム 2 が劣る結果となった．

6.3 実験 3

フレック型評価項目を 20000 と 100000 に増やして対局を行った．対局結果を表 8 に示す．プログラム 4 がフレック型評価項目数 20000，プログラム 5 がフレック型評価項目数 100000 である．表の通り評価項目数を増やすと NPS がさらに低下し，勝率も悪くなった．

7. まとめ

フレック型評価関数実現のために，ランダムカウントによる重要な駒組み合わせの抽出法を提案した．実際に提案手法を用いてカウントすることにより，どのような駒組合わせが抽出されるかを確認した．カウント結果には囲い等の重要な駒組み合わせが見られた．その後，抽出された組み合わせを用いてフレック型評価関数を設計し，従来のオール型評価関数組み込み対局実験を行った．その結果，オール型のみのプログラムに対してオール型とフレック型を組み合わせたプログラムが勝ち越す結果となった．このことから，ランダムカウントを用いて評価項目を抽出する本手法の有用性を示すことが出来た．しかし，評価項目数を増やした場合の性能低下や，フレック型に比べて評価時間が大幅に遅くなる等の問題点も見られた．これらを改善

することにより，更に性能向上が期待できそうである．

8. 今後の課題

今後の課題として評価項目の抽出法や評価関数の構造等を見直しが必要そうである．

評価項目の抽出法の改善案として，カウント中に駒組み合わせを上書きする対策があげられる．現在，駒組み合わせのカウントの際にカウント用ハッシュ表が埋まってしまった場合，それ以上新しい組み合わせを登録しないようになっている．そのため，1 回しか選ばれなかった組み合わせも保持しており，まだムダな組み合わせが残ってしまっている．学習後の評価値を見てみても評価値 0 の項目が現れてしまっている．この問題を解決するために，カウント回数 1 のまましばらくカウント数が増えなかった組み合わせは，新しい組み合わせで上書きする等の対策が必要であるといえる．

また，評価値の計算や学習にかかる時間が，オール型評価関数に比べてかなり増加してしまっている．これは，評価する際に評価木をたどる時間がかなりかかっているためである．現在は，Bonanza に実装されている差分評価を行っておらず，毎回全組み合わせを評価している．今後は評価木の実装方法の見直しや差分評価の実装等を行い高速化を図っていく．

参考文献

[1] 橋本剛, 飯田弘之：相対座標系から絶対座標系へ—将棋評価関数の設計思想—. 第 9 回ゲームプログラミングワークショップ. pp.88-91, (2004)
[2] 保木邦仁：局面評価の学習を目指した探索結果の最適制御. 第 11 回ゲームプログラミングワークショップ. pp.78-83, (2006)
[3] 金子知適, 山口和紀：将棋の棋譜を利用した大規模な評価関数の学習. 情報処理学会論文誌 51(12). pp.2141-2148, (2010)
[4] 下山晃：Blunder のアルゴリズム. http://hp.vector.co.jp/authors/VA039571/blunder/Blunder-20090507.pdf
[5] 鶴岡慶雅：選手権優勝記–激指の技術的改良の解説. 情報処理 51(8). pp.1001-1007, (2010)
[6] 古作登：コンピュータ将棋の弱点を探る. 人間に勝つコンピュータ将棋の作り方. pp.213-247, (2012)
[7] 伊藤毅志, 松原仁,Gbimbergen Reijer：将棋の認知科学的研究 (1):記憶実験からの考察. 情報処理学会論文誌 43(10), pp.2998-3011, (2002)
[8] 矢野友貴, 三輪誠, 横山大作, 近山隆：ゲーム構成要素を組み合わせた特徴の最適化. 第 15 回ゲームプログラミングワークショップ, pp.15-22, (2010)
[9] Tong Zhang：Solving Large Scale Linear Prediction Problems Using Stochastic Gradient Descent Algorithms. In ICML ’04: Proceedings of the twenty-first international conference on Machine learning, (2004)