将棋AIにおける評価項目自動抽出法の高速化手法

野津　佑太[1,a)]　後藤　嵩幸[2]　橋本　剛[1]

概要：コンピュータ将棋の評価関数は，Bonanza メソッドの考案により機械学習で値を決めることが一般になり，格段にレベルが向上した．現在では，プロ棋士との対局で勝利するなど，トッププロの実力を追い越そうとしている．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．将棋プログラムの強さは評価関数の精度で決まる．40 駒からなる将棋の盤面を評価する際，一般には局面を駒数個からなる部分局面に分割し，足し合わせて評価を行う．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評価関数が必要である．後藤は，重要な駒組み合わせのみを抽出する評価項目自動抽出法を提案し，高次元組み合わせ評価が可能となることを示唆した．しかし，学習や対局中の探索時間が大幅に増加し，まだ実用レベルには至っていない．そこで本研究では，評価項目自動抽出法の高速化手法を提案する．高速化手法を実装し，高速化前のプログラムと速度比較実験・対局実験を行い，性能向上を確認した．

Acceleration method for Evaluation Factors Automatic Extraction Method on Shogi AI

Yuta Notsu[1,a)]　Goto Takayuki[2]　Hashimoto Tsuyoshi[1]

Abstract: Appearance of Bonanza method then made determining values by machine learning become common and the level of Computer shogi has been dramatically raised. As Computer shogi won against professional shogi players, its ability is getting ahead of the top professional players. Meanwhile, it still has a weak point in the opening game. Performance of shogi program is determined by the accuracy of evaluation function. When a shogi position is evaluated, the position is generally divided into partial positions composed of a few pieces and is evaluated by sum of the all partial positions. A number of current computer shogi programs use evaluation function that keeps all combinations of a few pieces. Consequently, it cannot recognize small difference especially in the opening game. To achieve high-dimensional combination evaluation, it is necessary to use evaluation functions that evaluate only high-frequency evaluation items. Goto proposed ‘evaluation factors automatic extraction method’ to extract only important combinations of pieces, and suggested that high-dimensional combination evaluation was possible. However, this method caused increase in search time in learning and playing game and didn’t become a practical level. We therefore propose an acceleration method for evaluation factors automatic extraction method and implemented it. Speed comparison experiments and self-play experiments against previous program are performed and it shows the superiority.

1. はじめに

チェス・将棋・囲碁等の二人零和完全情報ゲームのプログラムでは，評価関数を使った Minimax 法が一般的な手法となっている．以前は，評価関数の値は開発者が手動で設定していたため，強いプログラムを作るには開発者にもゲー

[1] 松江工業高等専門学校
MIT, Matsue College,
Nishiikuma, Matsue, Shimane, 14–4, Japan
[2] ソニーデジタルネットワークアプリケーションズ
Sony Digital Network Applications, Inc.
[a)] s1514@matsue-ct.jp

ムに関する知識と実力が必要であった [1]．チェスにおいては，1997 年に手動調整の評価関数を使用するコンピュータ“ディープ・ブルー”がトッププロに勝利した．コンピュータ将棋では，保木により機械学習を用いた評価関数の自動学習法 (Bonanza メソッド) が考案され [2]，将棋が弱い人でも強いプログラムを作れるようになった．2010 年 10 月にはあから 2010 が清水市代女流王将に勝利 [3][4][5]，2016 年に行われた第 1 期電王戦では Ponanza が山崎隆之叡王に二番勝負で勝ち越すなど [6]，コンピュータ将棋が人間の実力を追い越そうとしている．また囲碁においては，2016 年に DeepLearning [7] による学習の成功が AlphaGo とイ・セドル九段との対戦によって大きく注目された．DeepLearning による学習は盤面情報を与えるだけで評価出来ることが画期的だが，膨大な計算が必要で，将棋で実用化された報告はない．

将棋プログラムの強さは評価関数の精度で決まる．より精確に盤面を評価するには多くの項目を評価する必要がある．しかし，評価項目数が増えるにつれて学習に費やす時間とメモリのコストは高くなる．例えば，将棋の駒の位置関係を用いた組み合わせ特徴の場合，2 駒間で評価する項目数は約 500 万にもなる．そのため普通に学習をした場合膨大な時間がかかる．現在，より効率的に学習を行うために様々な手法が考えられている．後藤は，対局中に高い頻度で現れる駒組み合わせを自動で抽出するのに成功した [8]．しかし，学習や対局中の探索時間が大幅に増加しており，まだ実用的ではない．これは，駒組み合わせのデータがリストを用いた木構造に保存されており，評価値参照に非常に時間がかかるためである．本研究では，後藤が提案したこの手法，評価項目自動抽出法が将棋において有効であるかを検証するため，高速化手法を提案しプログラムの性能評価を行う．また，高速化のために盤面の差分計算を提案する．その実装のため駒組み合わせ参照の新しい仕組みを提案する．その後，高速化後のプログラムと高速化前のプログラムの速度比較，対局実験を行い，有用性を検証する．

2 章では評価項目自動抽出法の内容について説明するとともに，その問題点を述べる．3 章では高速化の方法と，その実現のための新しいデータ構造について述べる．4 章では速度比較実験の方法とその結果について述べる．5 章では対局実験の方法とその結果について述べる．6 章では速度比較実験，対局実験の結果について考察する．7 章では本研究で提案した手法および実験結果についてまとめる．

2. 評価項目自動抽出法

2.1 目的

現在のコンピュータ将棋の多くは，3 駒の組み合わせ全てを評価する等，網羅的な手法を採用している．ここでは上記の手法をオール（all）型と呼ぶ．**図 1** にそのイメージを示す．図の右の配列に，王と他の 2 駒の全組み合わせの評価値が保持されている．例えば，Bonanza は王を含む 3 駒を評価しているが，それだけで約 9000 万項目にもなる [9]．この方法では，評価する駒の組み合わせ数を増やせばより正確に盤面を評価することができるが，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難しい．そのため，評価が微妙な序盤が弱点であるといわれている [10]．

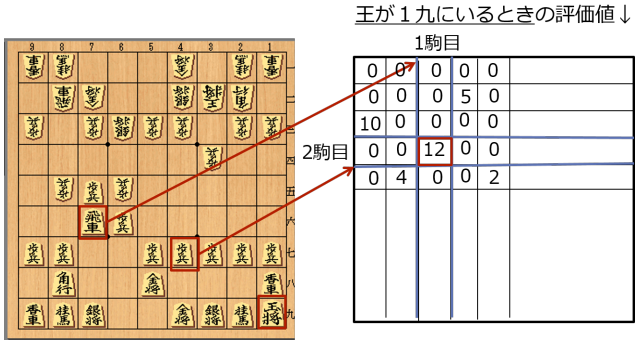


図 1 オール型評価関数
Fig. 1 All type evaluation function.

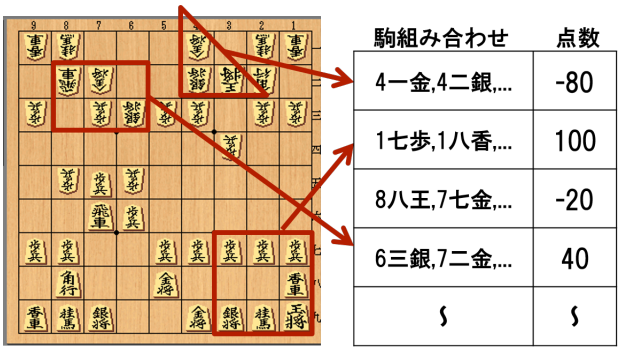


図 2 フレック型評価関数
Fig. 2 Frequency type evaluation function.

表 1 オール・フレックの特徴
Table 1 A characteristics of all and freq.

	長所	短所
オール型	設計が容易	項目数が膨大
フレック型	項目数が抑えられる	組み合わせの抽出が困難

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．文献 [8] に倣って，この手法をフレック（frequency）型と呼ぶ．フレック型は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．**図 2** にフレック型評価関数のイメージを示す．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．オール型とフレック型の特徴をまとめたものを**表 1** に示す．

膨大な駒の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．よって実用的な手法はまだ存在しない．評価項目自動抽出法は，それらの項目を対局中に現れる頻度を基に自動で抽出するのを目的とする．

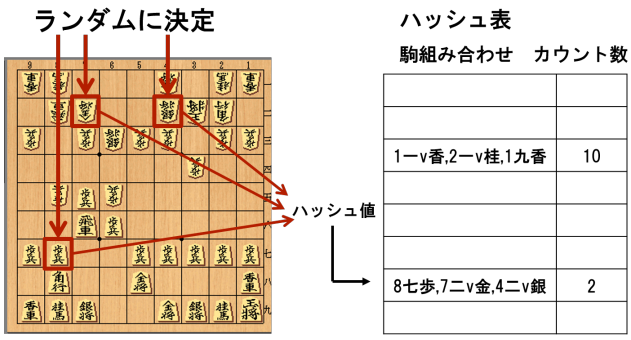


図 3　ランダムカウントの手法
Fig. 3　A random count.

2.2　手法

よく現れる組み合わせを調べるためには，対局中に現れる駒組み合わせの出現頻度をカウントすれば良い．出現頻度が高い組み合わせほどプロが対局中によく作る形だといえるため，評価する価値が高いといえる．しかし，全ての組み合わせをカウントしようとすると，従来の評価手法と同様に項目数が膨大となりカウントすることは不可能になる．そこで，カウントする組み合わせを盤上からランダムに選択する．**図 3** にランダムカウントのイメージを示す．

ランダムカウントの手順は，

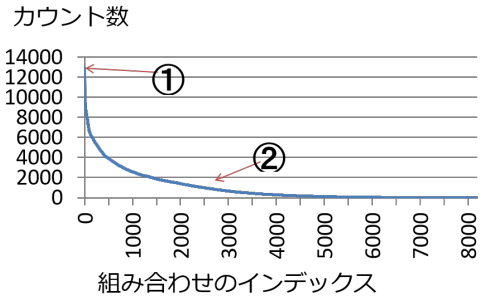
(1) 現在の局面からランダムに駒を選ぶ
(2) 選んだ駒組み合わせをカウントする
(3) 局面を進め，1 から繰り返す

である．

検証の結果，プロがよく作る形は対局中に中程度の頻度で現れていることが分かった．抽出した駒組み合わせとそのカウント数のグラフを**図 4** に示す．カウント数の多いグループ①ではあまり動かない駒の組み合わせが抽出されているが，グループ②には囲いなどの重要な組み合わせが抽出されている．また，数万局に一回しか選ばれない組み合わせも存在していることが分かった．現在の評価関数はこのようなめったに現れない組み合わせも評価項目として保持しているため，無駄が多いことが分かる．

2.3　問題点

評価項目自動抽出法には，学習や対局中の探索時間が大幅に増加するという問題がある．原因は評価値参照のためのアクセスに非常に時間がかかることである．**図 5** にそのイメージを示す．抽出した駒組み合わせはリストを用いた木構造に保持されている．図 5 の簡略化された盤面を評価する場合について考える．結果として 3 つの駒組み合わせが一致したが，そのために本来関係のない灰色のノードも辿って，抽出した駒組み合わせと盤面の比較を行う必要がある．これが探索時間の増加の原因である．



グループ①

1 駒目	2 駒目	3 駒目	カウント数
91 v香	99 香	19 香	13003
11 v香	99 香	91 v香	12668
21 v桂	19 香	11 v香	10666

グループ②

1 駒目	2 駒目	3 駒目	カウント数
99 香	88 角	39 銀	1093
77 銀	36 歩	78 金	1093
87 歩	53 v歩	44 v歩	1093

図 4　自動抽出グラフ
Fig. 4　A graph of automatic extraction.

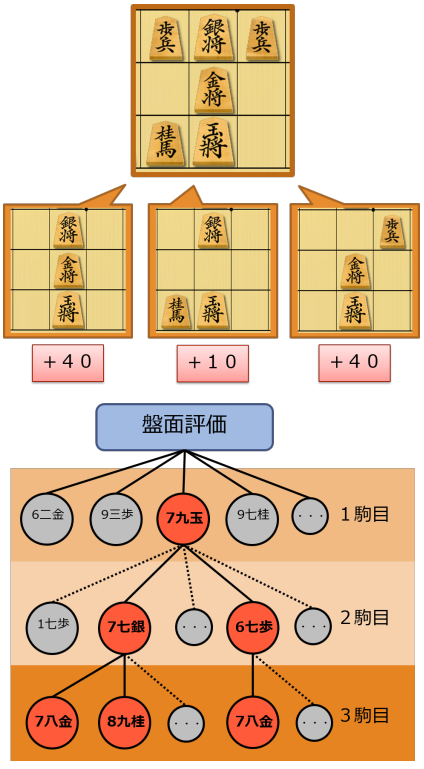


図 5　木構造の探索の問題点
Fig. 5　A problem of tree structure search.

3.　高速化手法

3.1　差分計算

対局中などに次の 1 手を選ぶ際，毎回盤面全体の評価を

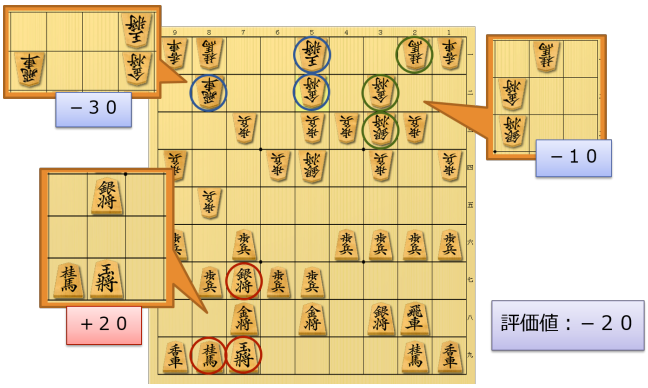


図 6　全体の評価
Fig. 6　Whole Evaluation.

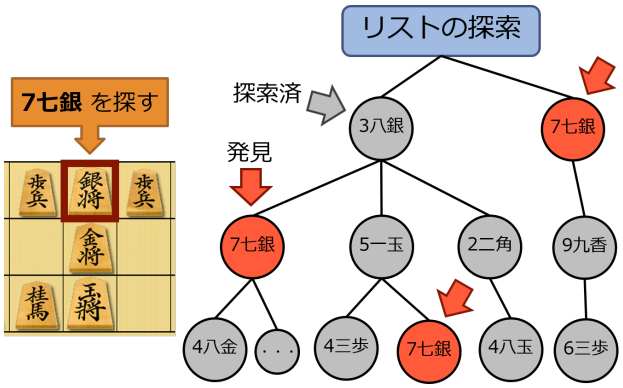


図 8　評価木の探索
Fig. 8　Evaluation tree search.

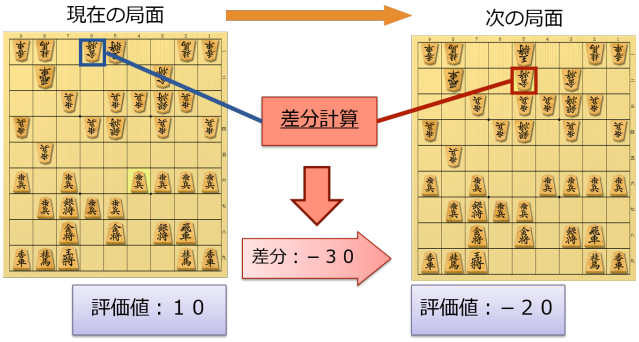


図 7　差分計算
Fig. 7　Calculation difference.

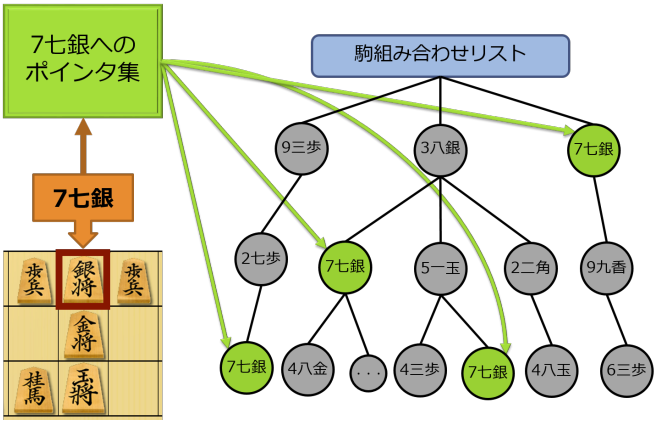


図 9　ポインタリスト
Fig. 9　Pointer list.

行う．全体評価では，盤面と抽出した評価項目全てを比較して評価値を算出する．**図 6** にイメージを示す．図 6 では，全体評価の結果 3 つの駒組み合わせが評価項目と一致している．それらの評価値を合計することで，現在の局面の評価値-20 が算出される．

ここで，現在の局面と次の局面との差は指した 1 手のみである．その 1 手に関する評価値のみ計算しなおすことで，盤面評価の高速化が可能である．この手法を差分計算と呼ぶ．**図 7** にイメージを示す．図 7 では，四角で囲まれた金が差分計算の対象となる駒である．現在の局面の評価値から，現在の局面の 6 一金に関する評価値を減算し，次の局面の 5 二金に関する評価値を加算すると，次の局面の評価値を求めることができる．尚，評価関数の差分計算は Bonanza などの将棋プログラムではすでに行われている手法である．

3.2　問題点

前述したとおり，差分計算では動かした駒のみに絞って評価をする．つまり，該当する駒が含まれる駒組み合わせをすぐに参照できる仕組みが必要である．評価項目自動抽出法で取得した駒組み合わせの項目は木構造で保持されている．そのため，木の全てのノードを辿って参照しなければ該当する駒を見つけることができない．**図 8** にそのイメージを示す．図 8 では 7 七銀に関する駒組み合わせを探索している．探索した駒が盤面と一致しているかを比較し，また組み合わせの中に 7 七銀が含まれるかどうかも調べる必要がある．これでは差分計算をしても通常の探索と同様の時間がかかってしまう．

これを回避するためには，高速な参照手法が必要である．ここでいう参照とは，目的の駒が含まれる駒組み合わせを全て取得することである．これを実現するため，全ての駒に対してリスト内の位置を保存したポインタのリストをあらかじめ作成する．この手法のイメージを**図 9** に示す．図 9 では 7 七銀に関する駒組み合わせを取得している．この手法を用いることで 7 七銀のリスト内での位置を高速で得ることができる．しかし，この手法では一番左の枝でいう 9 三歩や 2 七歩など，組み合わせの他の駒が何なのかが分からないので，結局は探索をしなければならない．よって，木構造ではない新しいデータ構造で実装をしなおす必要がある．

3.3　新データ構造

新しいデータ構造に必要な条件は，

(1) 1 つの駒の種類と位置から，それが含まれる駒組み合

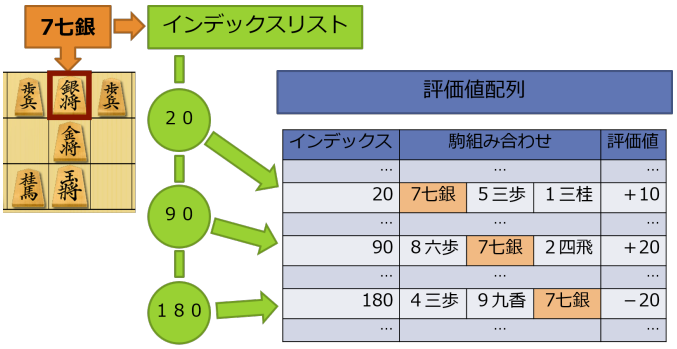


図 10　インデックスリストでのアクセス
Fig. 10　An access by index list.

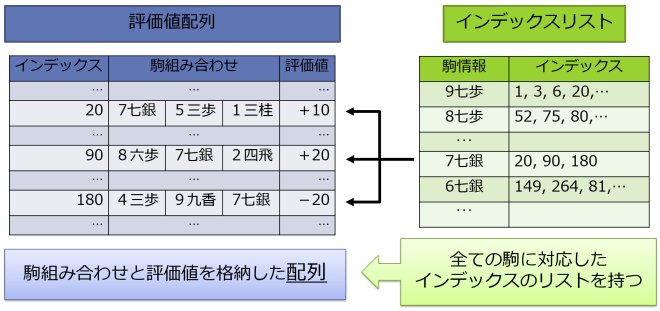


図 11　新しいデータ構造
Fig. 11　New data structure.

わせが高速に参照できること
(2) 一度に駒組み合わせ全てを取得できること
である．

現在の木構造では，(2) の条件を満たすことができない．そこで，駒組み合わせとその評価値を格納する評価値配列を作成する．更に，1 の条件を満たすために評価値配列のインデックスをすべての駒に対応してまとめたインデックスリストを作成する．**図 10** にその手法のイメージ図を示す．

図 10 では，7 七銀を含む駒組み合わせを取得したい．インデックスリストから 7 七銀に対応した 20，90，180 という 3 つのインデックスを取得し，それを用いて評価値配列にアクセスする．そして，参照した駒組み合わせが盤面と一致しているかを確認し，評価値の差分を求めることが出来る．

このデータ構造の全体のイメージを**図 11** に示す．すべての駒に対してインデックスリストをあらかじめ作成しておく．そのインデックスリストを用いて，駒情報から評価値配列に高速にアクセスできる．

4.　速度比較実験

4.1　概要

高速化手法による速度の差を調べるため，プログラムに棋譜を読み込ませ，ある局面での次の一手の思考時間の長さを比較した．実験は**表 2** のプログラムを用いて行った．

表 2　実験用プログラム
Table 2　Programs for experiment.

プログラム	概要
プログラム 1	Bonanza 6.0
プログラム 2	Bonanza ベースの後藤のプログラム
プログラム 3	プログラム 2 に高速化手法を実装したもの

表 3　速度比較結果
Table 3　The result of speed comparison.

手	プログラム 1[s]	プログラム 2[s]	プログラム 3[s]
11	0.04	5.92	9.76
15	0.04	20.59	11.88
71	0.22	173.0	3.67
73	0.24	83.0	1.73
155	0.21	29.01	5.48
157	1.48	628.0	26.74

プログラム 2 とプログラム 3 は，Bonanza の盤面評価・学習をする部分に評価関数が追加されており，そこで抽出した駒組み合わせ 10,924 項目の評価を行っている．実験に用いたコンピュータのスペックを以下に示す．

- CPU：Intel(R) Core(TM) i7 X980 @ 3.33GHz
- メモリ：24.0 GB
- OS：Windows 7 Professional 64bit

実験に使用した棋譜は第 27 期竜王戦七番勝負第 5 局，森内俊之竜王対糸谷哲郎七段のものである．また，探索深さは 5 である．

4.2　結果

実験結果を**表 3** に示す．ほとんどの場合でプログラム 3 の思考速度がプログラム 2 より速くなっていることが分かる．また，プログラム 1 よりは思考が低速である．プログラム 3 は，プログラム 1 の評価関数に抽出した駒組み合わせを足しているのでプログラム 1 より早くなることはない．

5.　対局実験

5.1　概要

高速化後のプログラムの有用性を調べるため，表 2 に示したプログラムで対局を行った．実験に用いたコンピュータのスペックは 4 章に示したものと同一である．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
- 時間設定：持ち時間 0 分，秒読み 5 秒
- 投了値：2000
- 最長定跡手数：無限

5.2　対局実験 1

まず，高速化されたプログラム（プログラム 3）の性能を確認するため，高速化前のプログラム（プログラム 2）と対

表 4　対局結果 1
Table 4　The result of games 1.

先手	勝ち数（平均 NPS（K））	後手
プログラム 2	98（50）: 2（2）	プログラム 3
プログラム 3	0（2）: 100（103）	プログラム 2

表 5　対局結果 2
Table 5　The result of games 2.

先手	勝ち数（平均 NPS（K））	後手
プログラム 1	93（336）: 7（90）	プログラム 3
プログラム 3	2（45）: 98（367）	プログラム 1

局させた．その結果を**表 4** に示す．勝ち数の左側は先手の勝利数，右側は後手の勝利数である．括弧内の数字は各プログラムの 1 秒あたり探索ノード数（Nodes Per Second：NPS）の 100 局平均である．表のとおり，プログラム 3 がプログラム 2 に対して勝ち越す結果となった．このことから，高速化による性能の向上が確認できる．

5.3　対局実験 2

次に，高速化されたプログラム（プログラム 3）の性能を確認するため，オリジナルのプログラム（プログラム 1）と対局させた．その結果を表 4 に示す．表の通り，プログラム 3 がプログラム 1 に対して負け越す結果となった．プログラム 3 の平均 NPS がプログラム 1 と比べ低く，更なる高速化が必要である．

6.　考察

速度比較実験と対局実験 1 の結果より，プログラム 3 はプログラム 2 より高速に評価を行うことが可能で，それにより性能が向上したといえる．よって高速化手法が有効であることが分かる．速度比較実験と対局実験 2 の結果より，プログラム 3 はプログラム 1 より低速であり，性能も低くなった．またプログラム 3 に追加した評価項目による性能の向上では，速度低下による性能の低下を補うことは出来なかった．

更なる高速化のためには，評価項目の見直しか，追加した駒組み合わせと盤面の比較方法の改良が必要である．今回，自動抽出法で抽出した 10,924 項目の駒組み合わせを実験に用いたが，評価値が 0 に近いものを削除することなどで計算時間を短縮することが出来る．また，追加した駒組み合わせを評価する際の，駒組み合わせと盤面との比較処理が速度低下の大きな要因であると考えられる．通常の比較より処理を軽くするため，ビット演算などを用いた手法を開発し，更なる高速化を実現させたい．

7.　まとめ

評価項目自動抽出法の実用化のために，差分計算を用いた高速化手法の実装を目的とした．その前段階として新しいデータ構造の実装を行った．高速化手法を実装後，その有効性を検証するため速度比較実験を行った．その結果，速度の向上が確認できた．また高速化による性能の向上を検証するため，高速化後のプログラム・高速化前のプログラム・オリジナルのプログラムで対局実験を行った．その結果，高速化後のプログラムは高速化前のプログラムに勝ち越したが，オリジナルのプログラムには負け越してしまった．高速化手法の有用性を示すことは出来たが，実用化のためには更なる高速化が必要である．

参考文献

[1] 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，橋本剛, 飯田弘之，2004.

[2] Large-Scale Optimization for Evaluation Functions with Minmax Search, Journal of Artificial Intelligence Research, Vol.49, pp.527-568，Hoki and Kaneko，2014.

[3] あから 2010 のシステム設計と操作概要, 情報処理 52(2), pp.162-169，保木邦仁, 金子知適, 横山大作, 小幡拓弥, 山下宏，2011.

[4] 清水女流王将対策と序盤戦術, 情報処理 52(2), pp.170-174，橋本剛，2011.

[5] 清水女流王将 vs あから 2010：コンピュータの思考過程を追う, 情報処理 52(2), pp.162-169，鶴岡慶雅, 金子知適, 山下宏, 保木邦仁，2011.

[6] 第 1 期 電王戦, http://denou.jp/2016/，DWANGO，参照 2016-09-24.

[7] Mastering the game of Go with deep neural networks and tree search, Nature 529, 484-489，David Silver et al.，2016.

[8] 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, ゲーム情報学, Vol.31, No.7, pp.1-6，後藤嵩幸, 橋本剛，2014.

[9] 「Bonanza4.1.3」ソースコード:コンピュータ将棋の進歩〈6〉, 第 1 章 pp.1-24, 保木邦仁，2012.

[10] 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, 古作登，2012.