入玉指向の将棋プログラムの作成

滝瀬竜司[†1] 田中哲朗[†2]

現在の将棋プログラムの多くは入玉が絡む局面の扱いを不得意としているが，トッププロに勝つためには，入玉が絡む局面を正しく扱い「自玉が入玉できそうな時は入玉を目指す」,「敵玉の入玉を正しく阻止する」将棋プログラムの作成が不可欠だと考えられる．本稿ではオープンソースの将棋プログラム Bonanza[1] の評価関数と定跡を変更することにより入玉志向の将棋プログラムを作成する試みをおこなう．その結果，元の Bonanza の評価関数に「入玉ステップ数」という特徴を加えて学習した評価関数を用いることにより，勝率を落とさずに入玉率を大幅に上げることができた

Development of entering-king oriented shogi programs

RYUJI TAKISE[†1] and TETSURO TANAKA[†2]

Most of recent shogi programs are thought not to treat entering-king positions correctly. In this paper, we tried to make a shogi progam that treats entering-king positions correctly; i.e. selects entering-king moves if possible and defends the opponent's entering-king moves. We have made a couple of experiments to make a entering-king oriented shogi program by means of modifying the evaluation function and the opening book of an open source shogi program Bonanza. We made an evaluation function which has a new feature "entering-king step" besides original features, and obtained the weight parameter of this feature by using machine learning. As a result, we successed to raise the entering-king rate without decreasing winning rate.

1. はじめに

近年のコンピュータ将棋の進歩は目覚ましく，この数年の間に行われた公開の場でのトップアマや女流棋士[2] 相手の対局では，大幅に勝ち越している．そのため，公式の場でトッププロを破る日も遠くないと言われている．その一方で，対コンピュータ将棋戦略を研究しているアマチュア強豪の多くが，コンピュータ将棋の苦手な展開があることを指摘している[3]．そこで指摘されている展開の一つが入玉絡みの展開である．

入玉とは，自分の玉を敵陣 (相手側の 3 段以内) に移動することを指す．入玉して成駒で周りを固めると，敵から容易に詰まされなくなるので有利になる．両方の玉が入玉すると，両者共に相手玉を詰ますことができなくなり，勝負が終わらなくなるので，駒の点数を計算して，それによって勝敗および持将棋 (引き分け) を判定する．

本稿では,「入玉勝ち」を「勝ちが決まった時点 (相手の投了，勝ち宣言) で勝ったプレイヤーの玉の位置が敵陣にある」ことであると定義する．人間の感覚と一致するとは限らないが機械的に判定が可能であり，人間の感覚と異なるケースはそれほど多くないと考え，この定義を用いることにした．

人間の将棋でも「入玉勝ち」の頻度はそれほど多くない．本稿で使った，プロの棋譜と将棋倶楽部 24 の棋譜集[4,5] とプロの棋譜合わせて約 52 万局中で入玉勝ちの出現頻度を調べたところ約 1.5 %という結果が得られた．一方，コンピュータ将棋同士の自動対局が行われている floodgate[⋆1] の 2010 年度の対局約 10 万局中では，入玉勝ちは約 2.6%なので，コンピュータ将棋の方が頻度が大きいと言える．

将棋は相手玉を詰ますことが目的のゲームであり，通常は自玉の周りに駒を集めて囲いを作り，また敵玉を攻撃するために駒得を目指す．しかし，入玉絡みの局面では，自玉の周りの駒が邪魔になることもあるし，片方が入玉し，他方が入玉できない状況では，入玉していない側がどれだけ駒得していても，不利になっていることが多い．両方が入玉して点数勝負になると，小駒 (飛車角以外の駒) の価値が変わらなくなってしまい，通常の将棋とは違うゲームのように見えることも

†1 東京大学総合文化研究科
Graduate School of Arts and Sciences, The University of Tokyo
takise@tanaka.ecc.u-tokyo.ac.jp
†2 東京大学情報基盤センター
Information Technology Center, The University of Tokyo
ktanaka@tanaka.ecc.u-tokyo.ac.jp

⋆1 http://wdoor.c.u-tokyo.ac.jp/shogi/floodgate.html

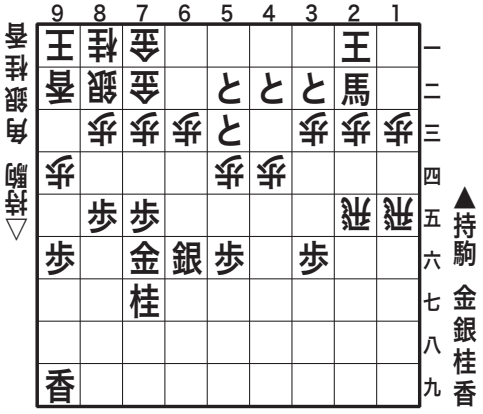

図 1 先手勝勢だが Bonanza が後手良しと判断する局面 (文献 6) より)

ある．

コンピュータ将棋においては評価関数と探索が重要だが，コンピュータ将棋が入玉絡みの局面を苦手としているのは，入玉絡みの局面の評価関数が不適切なのが原因ではないかという指摘がある[6]．図 1 は文献 6) で紹介されている局面で，先手勝勢だが Bonanza の評価関数は後手良しの評価を返す．

以前のコンピュータ将棋では，人間の経験則によって決定する駒の評価値をベースに，玉の安定度や大駒の mobility などに関連する特徴に対して，人手で重みをつけた評価関数を用いるのが一般的だったが，2006 年に浅い探索を組み合わせた兄弟ノードとの比較により評価関数のパラメータを学習させた Bonanza が第 16 回世界コンピュータ将棋選手権で優勝したのをきっかけに，多くの上位プログラムが評価関数のパラメータを学習により求める手法を用いるようになった．この方法により，教師とする上質の棋譜が十分あれば，多数のパラメータを持つ複雑な評価関数も扱えるようになった．

実際，Bonanza の最新版 (Bonanza version 6) では，評価関数の特徴として玉を含むすべての 3 駒の関係 (駒の種類と座標) を用いているが，これらの特徴に対応する重みパラメータを入れるファイル中のパラメータ数は約 9300 万個になっている[⋆1]．

これだけの複雑な評価関数を用いながら，Bonanza が入玉を扱うのが苦手な理由としては，以下の理由が考えられる．

(1) 学習する棋譜の中で入玉棋譜が不足している．

(2) 3 駒の関係を特徴とするだけでは，入玉絡みの局面を正しく扱えない．

Bonanza の最新版は，パラメータ学習のためのプログラムを含んだ形でソースが公開されているが，学習に用いた棋譜は公開されていないので (1) は直接確かめることはできない．ただし，Bonanza の最初のバージョンの学習に関して解説した文献 1) では，学習に用いた棋譜のうち，アマチュアの棋譜に関しては，どちらかの玉が相手陣の手前 (先手玉の場合は 4 段目，後手玉の場合は 6 段目) まで到達したものだけを用いているという記述があるので，Bonanza の最新版のパラメータを学習する際に，同じ方法を用いているとしたら，入玉棋譜の大部分が学習に使われたことになり，この仮説は退けられる．ただし，

- 学習に用いる棋譜数を増やす必要があったので，含まれる入玉棋譜の割合が少なくなった．
- 入玉棋譜を多く含むように選択をおこなうと棋力が低下したので，そのような選択をやめた．

という可能性は存在する．

(2) は，

- 入玉できるかできないかにより，盤面の評価値を大きく変える必要がある．
- 盤面の配置の微妙な違い (たとえば，盤上の駒一つと持駒との交換) により，入玉できるかできないかが決まることがある．
- この違いを 3 駒の関係の特徴に関するパラメータだけで表現しようとすると，入玉に関係しない局面の評価が適切に行われない可能性がある．

などを考えると有力な仮説であると考えられる．ただし，どのような特徴を加えれば，この問題を解決できるかは容易には決定できない．

本稿では，まず (1) の可能性を検証するために，Bonanza version 6 の特徴をそのまま用いて，教師を入玉勝ちの棋譜の勝った側の着手だけに限ってパラメータを学習させる試みをおこなった．

また，入玉絡みの局面の評価に有効であると考えられる特徴として，「入玉ステップ数」という特徴を提案し，入玉勝ちの棋譜を用いてその特徴のパラメータだけを学習する試みをおこなった．

それに加えて，入玉勝ちの棋譜を用いて定跡を作成する試みをおこなった．

以下，第 2 節では関連研究に関して述べ，第 3 節では提案する「入玉ステップ数」という特徴の詳細を述べる．第 4 節では，パラメータ学習および，得られた

⋆1 独立なパラメータ数は対称性の関係でこれより小さい

パラメータを用いた評価関数を用いた将棋プログラムを用いた対戦実験をおこない．第 5 節では結論と今後の課題を提示する．

2. 関 連 研 究

本研究ではオープンソースの将棋プログラム Bonanza[1] をベースに実験をおこなった．Bonanza はオープンソースの将棋プログラムの中でもトップクラスの棋力であり，これを用いたのは以下の理由による．

- ソースが小さく手を入れやすい⋆1．
- 評価関数に用いられている特徴が単純．

Bonanza が評価関数に用いる駒割以外の特徴は，基本的には玉を含む 3 駒の関係 (駒の種類，位置) のみである⋆2．テーブルとしては，以下の 2 つにわかれている．

kkp 両者の玉と，それ以外の駒一つの関係 (駒の種類，位置)

kpp 片側の玉と，相手玉以外の駒 2 つの関係 (駒の種類，位置)

左右を入れ替えた関係，手番を入れ替え盤面を上下に入れ替えた関係は同じものと見なし，持駒は個数に応じて別の位置にあると見なしている．これらの特徴と重みパラメータとの線形和を評価関数として用いる．他の多くの将棋プログラムが導入している大駒の mobility や利きの有無を含んだ特徴，ゲームの進行度を用いた非線形項などを使わないにも関わらず，比較的精度の高い評価関数が実現されている．

Bonanza では浅い探索を組み合わせた兄弟ノードとの比較により評価関数の重みパラメータを調整する手法を提案していて[1]，その部分のソースも公開されている．今回の実験でも，学習ルーチンに手を入れて学習対象の局面，調整するパラメータは変更したが，パラメータ調整の原理はそのまま用いた．

入玉指向の将棋プログラムを作るのは，単に強いだけでなく個性を持ったゲームプレイヤーを作る試みであると見なすこともできる．この視点から見た先行研究はいくつかあり，中でも文献 7) は Bonanza を用いて，プロ棋士の棋譜を用いたパラメータ調整をおこない，棋風を模倣させようというもので，本稿と関連が深い．

⋆1 著者の一人はオープンソース将棋プログラムである GPS 将棋の作者の一人でもあり，今回も GPS 将棋をベースに実験することも検討したが，教育的効果を考えて大学院生でもソースの全容を把握可能な Bonanza を選択した．

⋆2 Bonanza version 6 では，いわゆる丸山スペシャル (第 20 回世界コンピュータ将棋選手権で稲庭将棋が用いて二次予選に進出して注目された) への対策のための評価関数の補正が入っている．

3. 提案する特徴

入玉指向のプログラムを作成するために，最初に考えられるのは，玉が敵陣にある時のみボーナスを与える評価関数を使う方法であろう．

しかしこの方法は，探索木の末端で入玉済みの局面が現れるような，入玉直前の局面でしか有効ではない．一般に入玉を指向するか，囲いを強化するかの選択は自玉が自陣にいるうちに決定しなくてはいけないが，この場合，かなりの深さを読まないと探索木内に入玉済みの局面が出てこない．

一方で入玉前の状態，すなわち自玉が 4,5,6 段目にある時も距離に応じたボーナスを与えれば，浅い探索木でも入玉を指向する手を選択する確率が上がることが期待される．しかし，一般に中段 (4,5,6 段) の玉は危険であり，入玉できそうにない場合に中段に留まるのは賢明とは言えない．

詰みに関しても同様の議論が成り立つ．敵玉を詰めると勝ちになるが，詰みにいたる手数は実戦でも数十手になることもあるし，途中では大幅な駒損をすることもある．通常の探索では読み切れないし，詰手順の途中の評価関数の値を高くしようとすると，詰まないのに詰ませにいって駒損して負けてしまう危険性が増す．

そのため，多くの将棋プログラムは詰将棋ルーチンを使って，詰みの有無を通常探索とは別に求めている．

入玉に関しても同様の専用ルーチンで探索をさせることが考えられるが，以下の点が詰将棋と異なっているため，専用ルーチンを用いても探索空間がそれほど減らない可能性がある．

- 入玉を目指す手として，玉を移動する手，道を空ける手，敵の大駒の利きを止める手など多種類があり，詰将棋における王手と比べて多数の手を探索する必要がある．
- 入玉を阻止する手としても，敵玉を詰ませる手，逃げ道をふさぐ手など多数の手がある．
- 詰みがあるとその瞬間に勝ちが決定するので，詰むならどんな状況でも関係がないが，入玉はそこで勝負がつくわけではない．

これらの理由により，入玉可能性を探索により完全に求めるのは困難と思われる．そこで，以下のような探索によって入玉可能性の高い局面を検出する特徴を求め，その重みパラメータを学習によって決定した評価関数を用いる方法を提案する．

- 入玉側の指し手は玉を前，または斜め前に移動する手のみ生成する．

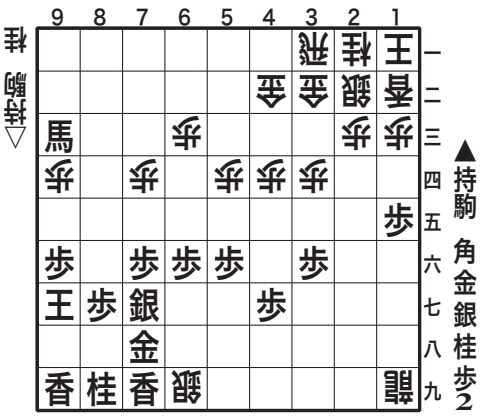

図 2 先手の「入玉ステップ数」が 4 手の局面

- 移動の際に相手の駒はとらない（移動できるのは空きマスのみ）.
- 敵側は何も指さない．ただし，敵の利きのある地点に玉は移動しない.

実際に入玉する場合では，入玉を阻止する側が道をふさぐ手を指し，入玉側の指し手も前進だけでなく，駒をとったり，玉以外の駒を動かして道をあけることもある．そのため，上記のルールでは完全ではないが，入玉可能かを調べるのは最大で 43 マスを調べればよく，効率がよいため，コストの問題を考えて，今回はこのルールを適用した．以下では「入玉ステップ数」としてこのルールで入玉可能な時はこの手数[⋆1]，このルールでは入玉不可の時は ∞ と定義する．容易にわかるように，現在の定義では「入玉可能」と判断された時は，玉が何段目にあるかだけでこの「入玉ステップ数」が決まる.

図 2 の局面の「入玉ステップ数」はこの定義に従うと，先手が 4 手 (86, 85, 84, 83 への移動)，後手が ∞ 手となる．実際にはこの局面では後手が飛車を回ったり，桂馬を打ったり，金や香車を取ってから打ったりして，入玉を防ぐ手があるので，入玉の可能性は自明ではないが，単に玉が 7 段目にいるという特徴よりは，より入玉しやすさを示す特徴になっていることは明らかだろう.

⋆1 既に入玉済みの時は 0

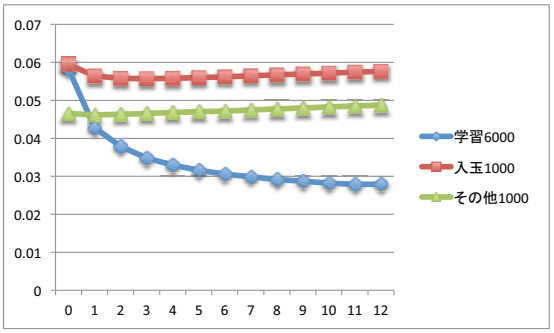

図 3 反復回数ごとの target の変化

4. 実　　験

4.1 特徴追加なしの入玉勝ち棋譜を用いた学習

まず，Bonanza で使われている特徴の追加をせずに，入玉勝ち棋譜だけを用いて，重みパラメータを学習させる実験をおこなった．学習にはプロの棋譜と将棋倶楽部 24 の棋譜集の棋譜計 525119 棋譜の中から入玉して勝った 7822 棋譜中の 6000 局を用いた．学習の際には，入玉勝ちした側の指し手のみを学習対象とするように Bonanza の learn コマンドを改造したものを用いた．重みパラメータの初期値は Bonanza 付属の重みパラメータ (fv.bin) を使い，一回の反復ごとの重みパラメータの更新回数 (step) は 32 回とした．学習時に探索する深さは標準値 (SEARCH_DEPTH=2) を用いた.

重みパラメータの数に比べて，学習に用いた 6000 局というのはかなり少ない．しかも，入玉勝ちした側の指し手のみを学習対象としたため，通常の学習の 3000 局分の局面しか学習に用いていないということになる．そのような場合では，過学習の可能性があるので，学習中の重みパラメータに対して，入玉して勝ったうちで学習に用いなかった 1000 棋譜と入玉していない 1000 棋譜に関する target 値 (文献 1) の学習の目的関数 J' を指し手の数で正規化したもの) を観察することにした．この値が小さいほど，棋譜の手と，その時点の重みパラメータで求めた評価値に従って浅い探索で求めた最善手との一致度が高いことを示している．反復回数 12 までの結果を図 3 に示す.

容易に予想されるように非入玉棋譜の目的関数は反復ごとに上昇しているが，学習に用いなかった入玉棋譜に対しても反復回数 3 回が最小となりそれ以降は上昇している．学習に用いた棋譜に対する目的関数が減少しているので，過学習に陥っている可能性が推測され，反復回数 3 回程度の重みパラメータを用いること

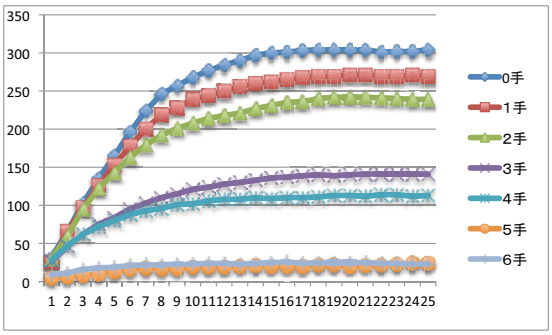

図 4 「入玉ステップ数」の重みパラメータの学習

表 1 得られた重みパラメータ

入玉ステップ数	0	1	2	3	4	5	6
重みパラメータ	304	269	239	141	113	24	23

が有効であると予想される．

4.2 「入玉ステップ数」の特徴を加えた学習

「入玉ステップ数」の特徴を加えた評価関数に関しても，さきほどの実験と同じ 6000 棋譜を用いておこなった．今回の実験では，Bonanza 本来の特徴に関する重みパラメータは変更せずに,「入玉ステップ数」の値の 0 ～ 6 に対応する 7 つの重みパラメータだけを変更するように Bonanza の学習プログラムを修正した．なお,「入玉ステップ数」が ∞ の時の重みパラメータは 0 に固定した．なお，Bonanza の評価関数では，3 駒間の関係の重みパラメータは，駒割の重みパラメータの 1/32 の値を用いているが,「入玉ステップ数」に対応する重みパラメータは駒割に匹敵するほど大きいことが予想されたため，駒割と同様に 1/32 せずに使うようにした．

収束の様子を調べるのに，前の実験では target の値を用いたが，今回は target の値の変化が小さいこと，また，学習する重みパラメータが 7 個と少ないため図 4 で，学習の反復回数ごとの 7 つの重みパラメータの変化を直接示す．反復回数 16 回までは単調に増加する傾向にあったが，それ以降は収束していると考えられたため，反復回数 25 回で学習を打ち切った．

得られた重みパラメータを表 1 に示す．駒割に換算すると，歩 1 枚の重みが 87 となっている．

この表をみると，ステップ数 0～2，3～4，5～6 の 3 段階にはっきりとした差があることがわかる．ステップ数 5～6 は自分の陣地の 1 段目と 2 段目であり，自玉がまだ囲いの中にいる状態である．囲いによって自玉が自分の陣地の 1 段目にいるか 2 段目にいるかが異なるので，一概に入玉まで近い歩数 5 の方がステップ数 6 より評価が高いとはいえず，そのために差がほとんどないと思われる．手数 3～4 は，自玉が囲いから出てきて危険な状態が多く，入玉が出来るならば目指すべき状態であるといえる．0～2 は入玉までのステップ数が短く，入玉出来る確率も大きいことが予想でき，そのために 3～4 よりも差が出ていると思われる．全体としては入玉までのステップ数が 0 のときが一番評価値が大きく，手数が大きくなるにつれて，評価値が小さくなる傾向がわかる，これは入玉をすると寄せにくくなり，勝ちやすいという人間の感覚と合致しており，学習は成功していると考えられ，それを示すために次節で対戦実験を行う．

4.3 対 戦 実 験

提案する手法の有効性を確認するために，対戦実験を行う．実験に用いる将棋プログラムは以下の通りである．

BonaznaStd 標準の Bonanza

LearnEking-3 特徴は増やさずに入玉勝ち棋譜 6000 局で重みパラメータを学習．反復回数 3 回後の重みパラメータを使用．

LearnEking-5 LearnEking-3 の反復回数を 5 にしたもの

LearnEking-12 LearnEking-3 の反復回数を 12 ににしたもの

LearnRandom LearnEking-12 の入玉勝ち棋譜 6000 局の代わりに 525119 局中から乱数で抽出した 6000 局 (LearnEking-12 と同様に勝ったプレイヤーの手だけを用いる) で学習させたもの．

EkingStep 「入玉ステップ数」の特徴を評価関数に加えたもの．重みパラメータは反復回数 25 回後のもの (表 1)．

オリジナルの Bonanza との差が重要なので，対戦相手はすべて標準の Bonanza(BonanzaStd) とする．その他の実験条件は以下の通りである．

- 相手時間を使用した先読み (ponder) は用いない．
- 思考ノード数を 250,000(用いた計算機では一秒程度の思考時間に相当) に制限．
- 評価値が手番のプレイヤーにとって，-5000 を下回った時に投了
- 総手数が 2,048 手になったら引き分けとする．
- 先手，後手それぞれ　250 局の計 500 局対戦

対戦実験の結果を表 2 に示す．BonanzaStd 同士の対戦も 500 局だが，先手後手共に同じプログラムなので，勝ち負けに関しては 2 倍にカウントされていることに注意が必要になる．定跡はランダムに選ばれるため，同じ対戦で全く同じ棋譜が生成される頻度は小さ

いが，実際に同じ棋譜が生成されなかったことは確かめた．

特徴を増やさずに入玉勝ち棋譜で学習させたプログラム LearnEking-{3,5,12} は共に標準を上回る入玉勝率を達成したが，「入玉ステップ数」の特徴を加えた EkingStep はこれらの中で最高の 5.8%の入玉率を達成した．LearnEking-{3,5,12}, EkingStep の入玉勝率が BonanzaStd の入玉勝率を上回ることは，フィッシャーの正確確率検定により有意性を確認できた．

勝率 (千日手および引き分けは 0.5 勝に換算) は，LearnEking-{3,5,12} 共に 5 割を下回った．BonanzaStd より勝率が下回っていることはカイ二乗検定により確認できた．乱数で抽出した 6000 局で反復回数 12 回学習させた LearnRandom-12 の勝率は 5 割を下回るものの，カイ二乗検定では有意性が確かめられない程度の落ち込みしか見せていないことから，Bonanza の特徴に追加せずに，入玉勝ちの棋譜だけを用いた学習をおこなうと，入玉が絡まない局面の評価が正確に行えない重みパラメータが得られる可能性が高いことが分かった．

一方，EkingStep の勝率は先手番，後手番共に 5 割を上回ったが，カイ二乗検定によって有意性が言えるほどの差は出ていない．

人間での対戦では入玉に適した戦形と入玉に向かない戦形があることが知られている．Bonanza には，棋譜を元に定跡ファイルを作成する機能があるので，入玉勝ちの棋譜 6000 局から作成した定跡ファイルを用いた場合に，入玉勝率が上昇するかを確かめる実験をおこなった．

Bonanza は棋譜から定跡ファイルを作成する機能があるが，これは，ある頻度を超えて出現した局面に関して，その局面における勝率を大きく下回らない展開になった手を勝率を元にした重みをつけて作成するものである．定跡を使う際は，対象局面が定跡に含まれている場合は，定跡ファイルに登録されている手の重みを正規化して乱数により手を選択する．

標準の定跡ファイルは 23620 局面 (28913 手) 登録されているのに対して，6000 局だけから作成した定跡ファイル (以下では book1) は 176 局面 (251 手) しかない．そこで，book1 に含まれている局面の手と重みはそのまま残し，book1 に含まれていない局面の手と重みは標準の定跡ファイルのものを用いる定跡ファイル (以下では book2) も作成した．

これを入玉勝率，勝率共に良かった EkingStep と組み合わせた以下の 2 つのプログラムを前の実験と同じ条件で BonanzaStd と対戦させた．

EkingStep-book1　EkingStep で定跡ファイルを book1 を使用

EkingStep-book2　EkingStep で定跡ファイルを book2 を使用

この対戦結果を表 3 に示す．参考のため，表 2 の EkingStep の結果も含めた表としている．

EkingStep-book1 に関しては入玉勝率が 6.6%と上昇しているが，EkingStep の 5.8%と比べて，有意に改善されているとは言えない．また，EkingStep-book2 は入玉勝率，勝率共に EkingStep を下回った．この方法で定跡ファイルを作成して，入玉勝率を改善することには失敗したといえる．

5. おわりに

本稿では，オープンソースの将棋プログラム Bonanza をベースに入玉指向の将棋プログラムを作成するために，元の評価関数の特徴を増やさずに，入玉勝ちの棋譜のみから重みパラメータを学習する方法，「入玉ステップ数」という新たな特徴を導入して，重みパラメータを学習する方法を提案し，実験をおこなった．どちらの方法でも入玉勝率は有意に上昇したが，特に「入玉ステップ数」を導入したものに関しては，勝率を落とすことなく入玉勝率を増やすことに成功した．

また，入玉勝ちの棋譜から作成した定跡ファイルと組み合わせることにより，より入玉勝率があがることを期待したが，今回用いた方法で作成した定跡ファイルでは有意な入玉勝率の上昇は確認できなかった．

なお，本稿では入玉するまでを目的として特徴を導入したが，相入玉した後で，宣言勝ちをするために味方の駒を敵陣に入れたり，大駒を捕獲するなどの戦略に対応した特徴は扱っていない．そもそも，現在公開されている Bonanza には宣言勝ちが組み込まれていないが，それも含めて入玉後に勝ち切るための改良は今後必要になると思われる．

また，今回は時間の関係で，学習対象の棋譜の数や学習時の探索深さが十分得られなかったため，新たな特徴の重みパラメータのみを学習して，既存パラメータは固定化した．しかし，文献 9) に提案されている既存パラメータを活かす形で学習する方法も適用可能であり，これを用いて十分な棋譜数，十分な探索深さで学習をおこなえば，更に良質の評価関数が得られる可能性も期待できる．

参考文献

1) 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

表 2 対戦結果 (標準定跡)

プログラム	先手番			後手番			勝率	入玉勝率 (%)
	勝ち (入玉)	負け	引き分け	勝ち (入玉)	負け	引き分け		
BonanzaStd	256(13)	235	9	235(7)	256	9	0.5	2
LearnEking-3	117(11)	133	0	87(14)	160	3	0.411	5
LearnEking-5	91(14)	158	1	115(13)	134	1	0.414	5.4
LearnEking-12	87(11)	160	3	79(15)	171	0	0.335	5.2
LearnRandom-12	122(5)	121	7	117(4)	127	6	0.478	2.6
EkingStep	130(16)	116	4	136(13)	108	6	0.532	5.8

表 3 対戦結果 (定跡変更)

プログラム	先手番			後手番			勝率	入玉勝率 (%)
	勝ち (入玉)	負け	引き分け	勝ち (入玉)	負け	引き分け		
EkingStep	130(16)	116	4	136(13)	108	6	0.532	5.8
EkingStep-book1	110(13)	132	8	121(20)	128	1	0.471	6.6
EkingStep-book2	121(10)	122	7	115(16)	133	2	0.481	5.2

ショップ 2006, pp. 78 – 83(2006).
2) 松原仁 編: あから 2010 勝利への道, 情報処理学会誌, Vol. 52, No. 2, pp. 152 – 190(2011).
3) 古作登: コンピュータ将棋の不遜な挑戦 : 6. 最強将棋ソフト「激指」との戦いに学ぶ – 対コンピュータ戦略の必要性, 情報処理学会誌, Vol. 51, No. 8, pp. 1018 – 1021(2010).
4) 久米宏: 将棋倶楽部 24 万局集, ナイタイ出版 (2002).
5) 久米宏: 最強の棋譜データベース, 成甲書房 (2004).
6) 片上大輔, 山本一成: コンピュータは七冠の夢を見るか?, 将棋世界 2010 年 3 月号, pp. 160 – 165(2010).
7) 生井智司, 伊藤毅志: 将棋における棋風を感じさせる AI の試作, 情報処理学会研究報告, Vol. 2010-GI-24, No.3, pp. 1 – 7(2010).
8) 保木邦仁: Bonanza version 6.0, http://www.geocities.jp/bonanza_shogi/ .
9) 矢野友貴, 三輪誠, 横山大作, 近山隆: 既存評価関数のパラメタを活かした適応学習, 第 14 回ゲームプログラミングワークショップ 2009, pp. 1 – 8(2009).