二人ゲームプレイヤのPrior Knowledgeを用いたUCTによる個性の実現手法と評価

志水 翔[1,a)]　金子 知適[1]

概要：本研究は，個性を持つコンピュータプレイヤの実現を目的として， UCT と prior knowledge を用いる方法を提案する．現在までに囲碁や将棋などのコンピュータにとって難しいゲームでも，強いプログラムが作られるようになってきており，強さだけでなく個性を持つプレイヤが期待されている．本研究では UCT を対象とし，UCT は囲碁プログラムの思考方法として標準的に使われている探索手法である．また prior knowledge は UCT の探索を効率化するための技術である．これまで prior knowledge は局面の有利不利のヒューリスティックな判定のために用いられてきたが，本研究ではこれをプレイヤの個性がどの程度表れている局面かの判定に用いることを提案する．どうぶつしょうぎは完全データベースが利用可能であるので，どうぶつしょうぎを対象に実験を行い，提案手法の評価を行った．どうぶつしょうぎは他のゲームと比べると探索空間が小さく, 提案手法の性能を多角的に評価しやすいためである．実験結果から，指し手に特徴を持つコンピュータプレイヤを複数実現できたことが確認された．個性を持たせると，そうでない場合に比べて同じ条件では，探索効率が劣り弱くなる傾向にある．しかし，プレイアウト回数や前向き枝刈を組み合わせることで強さは調整可能であり，広い範囲の強さで提案手法は個性を実現できることも確認された．

Evaluation and Implementation of UCT with Prior Knowledge for Computer ' s Styles of Playing Two-Player Games

Sho Shimizu[1,a)]　Tomoyuki Kaneko[1]

Abstract: This paper presents a method for generating computer players with different playing styles, by incorporating customized prior knowledge in UCT. Recently, strong computer players have been made even in Go and Shogi, where ordinary human players had outperformed computer players for many years. Therefore, researches toward computer players with playing different styles become important for providing more fun in playing against computers. We adopted UCT which is a popular search algorithm with many Go programs, and prior knowledge which is an enhancement to improve the playing strength of UCT players. While original prior knowledge is to heuristically discriminate good positions from bad ones, it is adapted here to present whether a position is preferable in terms of its playing style. We conducted experiments in Dobutsu-shogi and evaluate the proposed method. Dobutsu-shogi is suitable for our evaluation because its state space is relatively small. Our experiments showed that computer players with playing different styles are successfully generated by our method. Basically, a player having its playing style tends to be weaker than normal players. However, it is also confirmed that the playing strength can be controlled by the number of playouts and by the forward pruning with oracles while keeping its playing style.

1. はじめに

現在までに様々な二人ゲームのコンピュータプレイヤに関する研究が進んできている．これらの研究は主に，コン

[1] 東京大学大学院総合文化研究科
Graduate School of Arts and Sciences, The University of Tokyo
[a)] shimizu@graco.c.u-tokyo.ac.jp

ピュータプレイヤを人間よりも強くする目的で行われてきた．チェスでは 1997 年に “ Deep Blue ” が当時の世界チャンピオンである Garry Kasparov に勝利した [2]．また，オセロでも 1997 年 “ Logistello ” が当時の世界チャンピオンである村上健に勝利している [1]．また，将棋では，第二回・第三回電王戦では，複数のコンピュータプレイヤがプロ棋士に勝利している．この様に，強いコンピュータプレイヤを生み出すことは様々なゲームで達成されつつある．こうした背景から，人間が楽しむためのコンピュータプレイヤの要素として、強さ以外の要素の需要が高まっている．その一つとして個性があり、個性に関する研究も行われている．生井らの研究 [7] では，将棋のプロ棋士の棋譜から，定跡データベースの作成，評価関数の学習を行っている．また，この手法は棋風を反映させたい棋士の棋譜が大量に必要である．池田らの研究 [9] では，不自然でない弱さと特徴付けを実現している．UCB1 最大の手と着手の遷移確率を利用して不自然でない弱さを実現している．これらの手法では特定の個性を実現することができる．しかし，既存手法のみでは，実現できない個性もある．本研究では既存手法では実現できていない個性を含めた，「個性を持つレベル調整可能なコンピュータプレイヤの汎用な実現手法」を目標として， prior knowledge [4], [5] による UCT でのコンピュータプレイヤの個性の実現手法を扱う．UCT と prior knowledge は特定のゲームに特化した手法では無いので，汎用な手法になることが期待される．また，本研究はモンテカルロ木探索での個性の実現手法の基礎研究であり，提案手法の応用範囲は更に広くなると期待される．

2. 関連研究

2.1 コンピュータプレイヤの個性

生井らの研究 [7] では，将棋のプロ棋士の棋譜から，序盤の定跡データベースの作成，中盤以降の評価関数の学習を行っている．序盤の定跡データベースの学習では，ある局面からの指し手とその選択確率をプロ棋士の全棋譜から学習している．著者らはこの定跡データベースにより序盤の定跡選択の模倣を行えたと評価している．また評価関数の学習については，Bonanza Method を用いている．プロ棋士の棋譜を教師データとして，教師データに一致するように静的評価関数を最適化する．この評価関数では，中盤以降の指し手の特徴の模倣はできていないと評価している．

また，池田らの研究 [9] では，モンテカルロ木探索を基にして，囲碁における，不自然でない弱さと特徴付けをそれぞれ異なる手法で実現している．不自然でない弱さについては，ノードの勝率と静的評価関数による選択確率を用いて実現されている．ノードの勝率により，着手の良し悪しを決定する．形勢に応じて，有利であれば悪い手を選び，不利であれば良い手を選ぶようにする．このように選ばれる手が不自然な手にならないように静的評価関数による選択確率が用いられる．パラメータを変えることによって手加減をすることが出来ている．また，著者らは複数人の目視によって，不自然な手の排除を確認できたと結論づけている．

特徴付けについては，プレイヤアウトの結果の計算方法を変えることによって行われている．例えば，プレイアウトの結果の計算の際に通常 1 つの交点を 1 目と数えるが，「隅・辺の交点を 1.5 目と扱う」などのような重み付けを行うことで実利派の特徴付けを行っている．しかし，終局時の情報から得られる特徴しか扱えないという課題が残されている．

滝瀬ら [8] は従来の将棋プログラムが入玉の絡む対局を苦手としていることを問題としている．入玉が絡む棋譜のみを用いて学習を行うことと入玉ステップ数という新しい特徴に関する重みの学習を行うことによって，入玉指向のプログラムの作成に成功している．入玉に関する学習を行う前と比べて，勝率を落とすことなく，入玉勝ちする割合が増えている．

また，登坂ら [6] は将棋の棋士の特徴を棋譜から抽出する実験を行っている．羽生義治棋士とその他のプロ棋士の棋譜を比較することにより，棋風の客観的な差異を見つけることを試みている．結果として，特定の駒を使う頻度，駒ごとの指す位置，終局までの手数，手の組み合わせごとの出現頻度に有意差が確認されている．

2.2 UCT とその拡張

UCT , prior knowledge ,Elo-rating による評価関数の学習を紹介する。UCT と prior knowledge については 4 節で詳細を説明する。

2.2.1 UCT

min-max 探索を基本とする手法では，局面の有利不利を推定する為に，評価関数を用いることが一般的である．しかし， UCT は木を成長させながら，局面の有利不利をランダムな着手 (プレイアウト）の結果を基に推定し，探索を行う手法である．

UCT はノードの選択に UCB1 値を用いる探索手法である．UCB1 値はノードのプレイアウトによる勝率と探索の目新しさを表す項からなる値である．UCB1 値を用いることによって探索と活用のバランスを取りながら探索を進めることができる．囲碁では， UCT は min-max 探索を基本とする手法よりも有用であると考えられている．実際に，有力な囲碁プログラムの多くは UCT を採用している．また，様々なゲームの探索に有効であることが分かっていて，将棋のコンピュータプレイヤや二人有限完全情報ゲーム以外のコンピュータプレイヤにも応用しようとする試みがある．

また， UCT は十分にプレイアウトを行えば，min-max 最善手を選択するという証明を持つ．しかし，実際のゲー

ム木においては，実時間では十分にプレイアウトを行うことは難しい．その為，できる限り有望なノードを選択し，プレイアウトを行なうことが UCT の性能向上に繋がると知られている．UCT の拡張の代表的な手法として， prior knowledge , Elo-rating による評価関数の学習がある．

2.2.2 Prior Knowledge

prior knowledge は，事前知識を用いたヒューリスティックな評価関数によって，探索を効率化する手法である．UCT でノードを生成する際に，評価関数によってノードの勝率と訪問回数を与える．prior knowledge は UCT のノード選択に影響を与える手法の一つである．適切な評価関数を与えれば，有望なノードに割くプレイアウト数が増え，有望でないノードに割くプレイアウト数が減ることで，少ないプレイアウト数で良い結果が得られると考えられる．囲碁の 9 路盤による実験では， prior knowledge を用いることによって勝率が向上している． また，プレイアウト数を増やすことによって prior knowledge の影響は薄れる性質がある．この性質から， prior knowledge による個性の実現は，強さをある程度保つことが期待される．

実際のゲーム木ではゲームの性質などから簡単にノードの良し悪しが分かることがある．例えば，将棋では，飛車や角をただで相手に取られる手の多くは悪い手である．この様にプレイアウトを行う前にノードの良し悪しが分かる場合に， prior knowledge による探索の効率化を行うことができる．

2.2.3 Elo-rating による評価関数の学習

提案手法 [3] では，Bradley-Terry モデルに基づいて，Elo-rating による特徴の選択確率を学習している．

本研究では前向き枝刈を完全データベースで行ったが，完全データベースが利用できないゲームでは Elo-rating を用いることが自然である．一方，Elo-rating は学習のために，熟達者の棋譜を必要とする．本研究はどうぶつしょうぎを対象に評価したので，棋譜の入手がむずかしく，代わりに完全データベースによる前向き枝刈りを用いた．

Elo-rating による評価関数の学習は本研究で提案する個性の実現にも有効と思われる。本研究では扱わなかったが，[8] のように，特定の特徴が表れやすくなる様に評価関数の学習を行い，その評価関数を prior knowledge の評価関数として用いることが考えられる．また，プレイアウトへの利用での個性の実現なども今後の研究課題として考えられる．

3. どうぶつしょうぎと完全データベース

本稿ではどうぶつしょうぎを対象に提案手法の評価を行うので、どうぶつしょうぎというゲームと、完全データベースを求めた研究について紹介する。

3.1 どうぶつしょうぎ

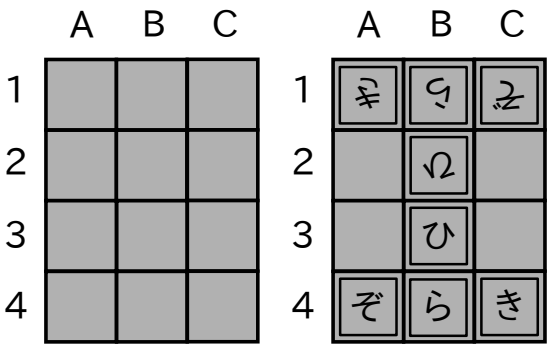


図 1 どうぶつしょうぎのボードと初期配置

どうぶつしょうぎは将棋に似た 2 人ゲームである．どうぶつしょうぎで使われる駒の説明は，以下に示す．

ひよこ 前に 1 マスのみ動ける．将棋の歩と同じ動き．

きりん 上下左右の 4 近傍に 1 マス動ける．

ぞう 斜めの 4 近傍に 1 マス動ける．

にわとり 左下，右下以外の周囲 8 近傍に 1 マス動ける．将棋の金と同じ動き．

らいおん 周囲 8 近傍に 1 マス動ける．将棋の玉と同じ動き．

どうぶつしょうぎは**図 1** の左図の 3 × 4 のボードを使用する．各列は上から A,B,C と示され，各行は左から 1,2,3,4 と示される．盤上の各マスは「A2」,「C4」の様に示される．初期配置は図 1 の右図に示す．盤上の駒は駒の名前の初めのひらがな 1 文字で表している．例えば，B3 にある「ひ」の駒は「ひよこ」の駒を表す．先手番は盤上の 4 行目から 1 行目に向かう方向を向いている駒を使い，4 行目を「先手のエリア」とする．後手番は盤上の 1 行目から 4 行目に向かう方向を向いている駒を使い，1 行目を「後手のエリア」とする．ひよこは相手のエリアに入るとにわとりに成る．

相手の駒がいるマスに駒を動かす場合は，その相手の駒を盤上から取り除き，自分の持ち駒にする．これをどうぶつしょうぎでは「つかまえる」と言う．にわとりをつかまえる場合，ひよことして持ち駒になる．

終局条件は以下の 3 つである．

キャッチ 相手のらいおんをつかまえる．捕まえた側のプレイヤーの勝利となる．

トライ 自分のらいおんが次の相手の手でつかまらない様に，相手のエリアに動かす．相手のエリアにらいおんを動かしたプレイヤーの勝利となる．本稿では，トライで勝つことを特に「トライ勝ち」と言う．

引き分け 同じ局面が 3 回目に現れた時に引き分けとなる．

3.1.1 どうぶつしょうぎの完全データベース

本研究で扱うどうぶつしょうぎは完全データベースを持つ [10]．初期局面から開始して「手番のプレイヤーが相手のライオンを捕まえられる時は必ず捕まえて勝つ」,「ライ

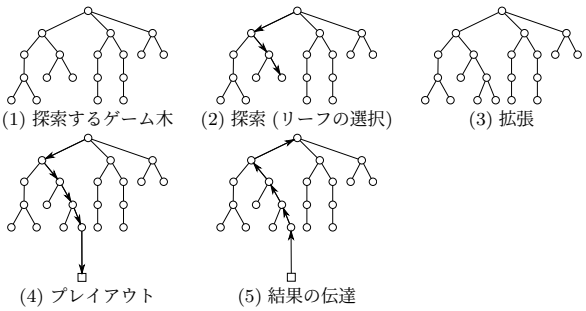
オンがトライした時はゲームを終了する」として到達可能な局面数は 246,803,167 と求まっている．末端局面を除いた局面数は 99,485,568 である．また，末端局面を除いた 99,485,568 局面のうち，手番の勝ちは 56,474,473 局面，引き分けは 2,682,700 局面，負けは 40,328,395 局面とである．初期局面からの勝敗は後手勝ちである [10].

本研究では，この完全データベースを**図 2** の (3) の手順でノードを生成する際の枝刈りに利用する．完全データベースは prior knowledge の実装には必要が無いので，完全データベースを持たないゲームにも本手法は用いることができると考えられる．

4. UCT での個性の実現

UCT で探索するプレイヤを対象に，個性を実現する方法を提案する．提案手法は，実現したい個性を prior knowledge により表現することと，プレイアウト数ならびに前向き枝刈により強さを制御することの組み合わせからなる．まず前提である UCT の動作を説明し，続いて個性の実現手法を強さの制御手法について提案する．

4.1 UCT



(1) 現時点で探索木がここまで成長していたとする．
(2) 情報を得るリーフを一つ選ぶ．リーフの選択はルートから有力なノードを選び続けることで行う．
(3) リーフの訪問回数が閾値に達していれば展開する．
(4) プレイアウトを行う．
(5) プレイアウトの結果をリーフからルートに順に反映される．

図 2 UCT の探索手順

UCT の動作の概略を図 2 に示す．プレイアウトを繰り返すにつれて木は成長する．成長した探索木からプレイアウトを行うリーフを選ぶ．実際のゲーム木においては，実時間では十分にプレイアウトを行うことは難しい．その為，できる限り有望なリーフを選択し，プレイアウトを行なうことが UCT の性能向上に繋がると知られている．その為，図 2 の (2) の手順で行われるリーフの選択の方策に様々な工夫が施されている．リーフの訪問回数が閾値に達していれば，合法手によって遷移するノードを展開する．その後，プレイアウトを行い，その結果をリーフからルートに順に反映する．図 2 の一連の手順の中で UCT の性能に大きく関わるのが，(2) の手順で行われるリーフの選択である．

4.1.1 UCB1

UCT では図 2 の手順 (2) で，以下のように定義される UCB1 値最大のノードを繰り返し選択することでリーフの選択を行う．

$$\mathrm{UCB1}(s,a) = Q(s,a) + c\sqrt{\frac{\log n(s)}{n(s,a)}} \tag{1}$$

$$n(s) = \sum_a n(s,a) \tag{2}$$

s は状態を表す．a はアクションを表す．UCB1(s,a) は状態 s のアクション a に対する UCB1 値を表す．Q(s,a) は状態 s でアクション a を選んだ後のプレイアウトで得られた平均報酬を表す．c は探索と活用の割合を表す定数である．n(s,a) は状態 s でアクション a を選んだ回数を表す．

式 (2) の第一項はノード s で a を着手したプレイアウトの勝率を表す．第二項はノード s での着手 a の目新しさを表す．プレイアウトの結果を反映させる際に，UCB1 値は探索で通ったノードと着手の組のみ更新される．

4.2 Prior Knowledge と特徴の設計

prior knowledge は，UCT でノードを生成する際に，式 (3), (4) の様に n(s,a)，Q(s,a) を初期化する．$n_{\mathrm{prior}}(s,a)$，$Q_{\mathrm{prior}}(s,a)$ は評価関数である．

$$n(s,a) \leftarrow n_{\mathrm{prior}}(s,a) \tag{3}$$

$$Q(s,a) \leftarrow Q_{\mathrm{prior}}(s,a) \tag{4}$$

本研究では UCT における prior knowledge による特徴付けの性質を測ることが目的である．個性を表すような要素を持つアクションを a^* とすると，$n_{\mathrm{prior}}(s,a)$，$Q_{\mathrm{prior}}(s,a)$ は式（5），（6）の式で定義される．

$$n_{\mathrm{prior}}(s,a) = v \tag{5}$$

$$Q_{\mathrm{prior}}(s,a) = \begin{cases} 0 & a = a^* \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

a^* の決め方には直接個性を表すアクションを a^* とする方法と個性に関連する要素を持つアクションを a^* とする方法の 2 種類が考えられる．例えば，直接個性を表すアクションとは，「銀をよく使う」という個性に対する「銀を使う手」ことである．また，個性に関連する要素を持つアクションとは，「実利派」という個性に対する「1 線から 4 線までに打つ手」のことである．特徴によっては複数の a^* の決め方があるが，この a^* の決め方により，提案手法の性能が変わると予想される．

4.3 強さの制御

実用的には，個性を実現するだけでなく，対戦相手にあわ

せた様々な強さの個性的なプレイヤを実現することが重要である．たとえば個性が実現できたとしても，弱すぎるなどでは実用にならないためである．UCT での強さの制御手法としては，プレイアウト数を増やすことと progressive widening のように前向き枝刈を部分的に導入することが行われている．そこで本稿では，プレイアウト数を増やすこととともに，完全データベースを用いた前向き枝刈の導入により強さを制御することを提案する．

level は UCT の強さを変える変数である．本研究で用いる UCT は完全データベースを利用し，強さを調節した．図 2 の（2）の手順でノードを生成する際に，完全データベースを用いて枝刈りをする．以下に示す（1）-（3）の手を次の手順にしたがって新たなノードとして生成する．（1）の手が生成できる場合は（1）の手のみを生成する．（1）の手が生成できず，（2）の手が生成できる場合は（2）の手のみを生成する．（1）（2）の手が生成できない場合は (3) の手を生成する．

(1) *level* 手以内に勝つことができる手
(2) *level* 手以内に負けない手
(3) （1）（2）以外の手

level の値を大きくすると UCT が強くなると期待できる．

5. 実験と評価

5.1 実験の設定

本研究では Simple，Kirin，Zo，Try，Nari，Capture という 5 つプログラムを用いた．実験に用いるプログラムには 3 つのパラメータ n，v，*level* を与える．n は UCT のプレイアウト回数を表す．v は特徴付けの重みを表す係数である．本研究で用いる UCT のプレイアウトは，「相手の王を取れる時は必ず取る」，「他に合法手がある場合，次の手で相手にライオンを取られる手を打たない」というヒューリスティックを用いている．このヒューリスティックが適用されない場合はランダムな指し手となる．以下，各プログラムにパラメータ n,v,*level* を指定する場合は，それぞれ Simple(n,*level*)，Try(n,v,*level*)，Capture(n,v,*level*)，Kirin(n,v,*level*)，Zo(n,v,*level*)，Nari(n,v,*level*) と表記する．また，一部のパラメータを設定していない場合は，設定していないパラメータを変数名とする．例えば，$n = 1000$ の Kirin をまとめて表現する場合，Kirin(1000,v,*level*) と表す．

Simple は UCT を用いるプログラムである．Kirin，Zo，Try，Nari，Capture はそれぞれ提案手法を実装した UCT を用いるプログラムである．Kirin はきりんを動かす手を a^* としている．a^* は持ち駒からきりんを打つ手を指す手も含む．Zo はぞうを動かす手を a^* としている．a^* は持ち駒からぞうを打つ手を指す手も含む．Nari はひよこをにわとりに成らせる手を a^* としている．つまり，ひよこを相手のエリアに打つ手である．Try はらいおんが手番のプレイヤから見て，左上，上，右上のいずれかに動く手を a^* としている．Capture は相手の駒を取る手を a^* としている．

Kirin, Zo は特定の駒を使いやすいという特徴を表すことが期待されるプログラムである．特定の駒の使用頻度が特徴に関連する可能性があると，参考文献 [6] で示されている．Try はトライをしやすくなるような特徴を表すことが期待される．トライをしやすくなるという特徴は将棋での入玉勝ちに関連するものであると考えられる．参考文献 [8] に関連する特徴であると考えられる．Nari，Capture はルールに基づく特徴である．ルールに基づく特徴は一目で分かりやすい為，初心者らしい特徴となることが期待される．

また，a^* の決め方には直接個性を表すアクションを a^* とする方法で a^* を決めたプログラムは Kirin, Zo, Nari, Capture である．個性に関連する要素を持つアクションを a^* とする方法でを決めたプログラムは Try である．トライ勝ちしやすくする a^* として他に「トライを確定する手」，「ライオンが相手のエリアに入る手」などが考えられる．トライを確定する手を a^* とすると，a^* は直接個性を表すアクションとなる．

実験では各プログラムを先手後手入れ替えて，Simple(1000,1) と 1,000 局対戦させ，勝率と特徴の表れる手の割合を計測した．勝率は，勝ちを 1 勝，引き分けを 0.5 勝，負けを 0 勝として計算した．提案手法を導入したプログラムの他に，ベースラインとして Simple(1000,1) も対戦させた．

5.2 特徴の表れる指し手と勝率: 駒を動かす特徴

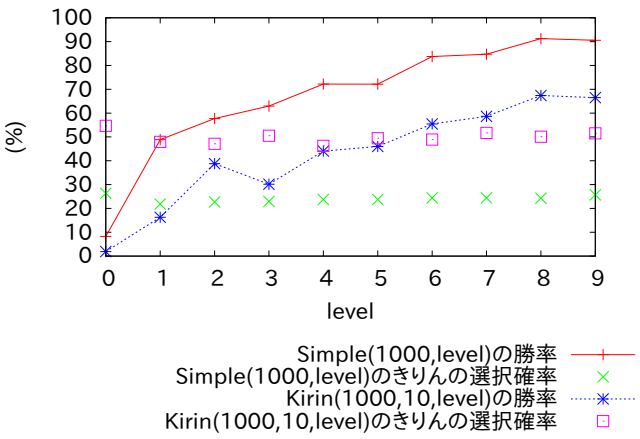


図 3 Kirin(1000,10,*level*) の勝率と特徴の選択確率

Kirin(1000,10,*level*) に関する結果は**図 3** に示す．縦軸は勝率ときりんの選択確率を表す．横軸は *level* の値を表す．きりんの選択確率は，（プログラムがきりんを指した回数）/（プログラムの総指し手数）で計算している．Kirin は Simple と同じプレイアウト回数では，Simple より勝率が低

いことが分かる．*num* と *level* の値が同じ Simple と Kirin では Kirin はきりんの選択確率が多く，Kirin は特徴を表すことができている．また，どちらのプログラムも *level* を増やしてもきりんの選択確率はあまり変化が無い．提案手法は，ノードの前向き枝刈りを行っても，特徴を表すことができていることが分かる．その為，他の UCT の枝刈りを行う手法と組み合わせることで，個性があり強いプログラムを作ることが期待される．また，Simple(1000,10,2) の勝率は 57.7%，Kirin(1000,10,6) の勝率は 55.45%であり，同程度の勝率であるが，きりんの選択確率は Simple(1000,10,2) は 22.73%，Kirin(1000,10,6) は 48.91%であり，大きく異なっている．勝率を保ったまま特徴を出すことに成功している．更に，提案手法を他の UCT を強くする手法と組み合わせることで，様々な強さで特徴を出せる可能性がある．今回の実験では，1 試合当たりの手数は 30 手程であった．Kirin の選択確率には十分な母数があるので，提案手法の効果が顕著に現れていることが分かる．

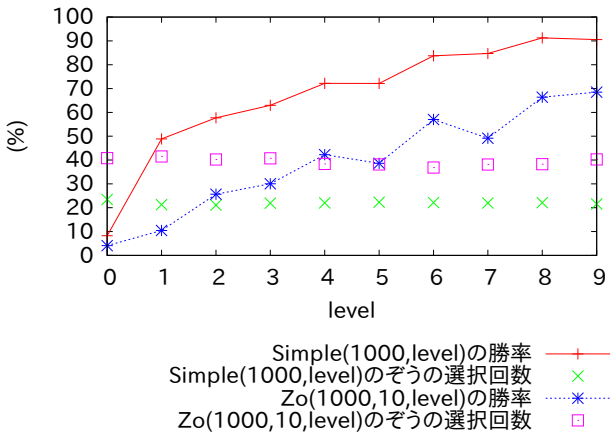


図 4 Zo(1000,10,*level*) の勝率とぞうの選択確率

Zo(1000,10,*level*) に関する結果は**図 4** に示す．縦軸は勝率とぞうの選択確率を表す．横軸は *level* の値を表す．ぞうの選択確率は，（プログラムがぞうを指した回数）/（プログラムの総指し手数）で計算している．図 3 の結果と同様に Zo でも Kirin と同様の性質があることが分かった．

Nari(1000,10,*level*) に関する結果は**図 5** に示す．左の縦軸は勝率を表す．右の縦軸はプログラムの成る確率を表す．横軸は *level* の値を表す．成る確率は，（プログラムがひよこをにわとりに成らせる手を指した回数）/（プログラムの総指し手数）で計算している．また，成る確率は右の縦軸を使っているので，範囲は 0 – 1% の値を表している．Nari は Simple と比較すると，勝率も悪く，成る確率も少ない．成る確率が Simple の方が高かった理由は確実ではないが，以下のような理由が有力であると考えている．

Kirin と Zo に関する結果から，特定の駒を使いやすいという個性は prior knowledge によって実現できている．

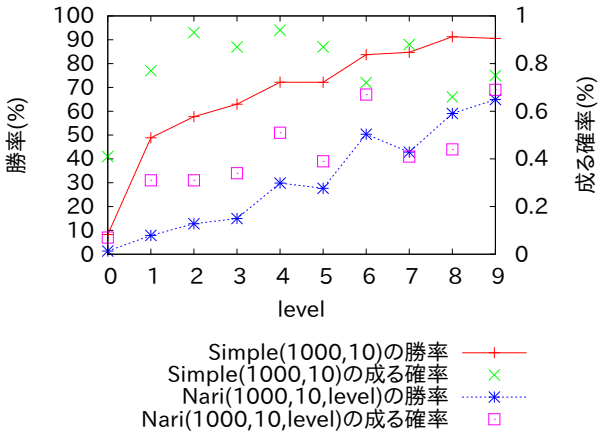


図 5 Nari(1000,10,*level*) の勝率と成る確率

これらの特徴は合法手の中で出現頻度が高く，特徴を持つ合法手が複数ある局面の出現頻度も高い．その為，特徴のある良い手を見つけやすく，提案手法によって個性を実現できたのではないかと考えられる．一方，「ひよこがにわとりに成る手」は「特定の駒を使う手」と比較して，合法手の中での出現頻度が少ない．ルール上，1 つの局面で最大 2 手しかひよこがにわとりに成る手は存在しない．これは，きりんやぞうを使う手と比較するとかなり少ない．その為，特徴のある良い手を見つけにくく，提案手法によって個性を実現できなかったのではないかと考えられる．

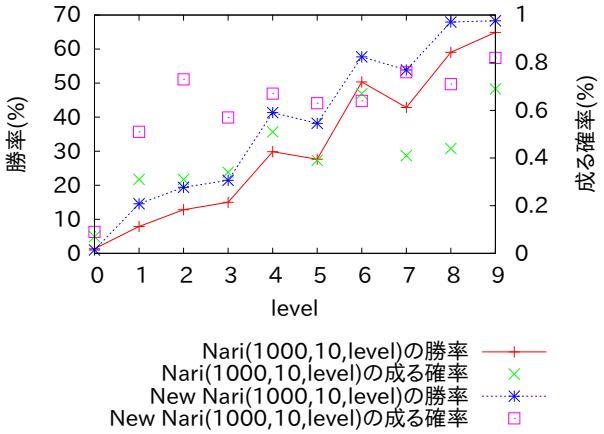


図 6 a^* を「ひよこが手番のプレイヤから見て，左上，上，右上のいずれかに動く手」と変更した Nari(1000,10,*level*) の勝率と成る確率

新たに a^* をひよこが手番のプレイヤから見て，左上，上，右上のいずれかに動く手とするプログラム New Nari を作成し，図 5 と同様の行った．勝率，成る確率共に Nari と比べて New Nari の方が高いという結果となった．「ひよこが手番のプレイヤから見て，左上，上，右上のいずれかに動く手」に「ひよこがにわとりに成る手」は含まれ，「ひよこが手番のプレイヤから見て，左上，上，右上のいずれかに動く手」の出現頻度は多い．その為，特徴のある良

い手を見つけやすく，New Nari の方が強くて特徴がでたのではないかと考えられる．このことからも，a^* とする手は出現頻度の多い手を選ぶと性能が上がると考えられる．

5.3 特徴の表れる指し手と勝率: 他の特徴

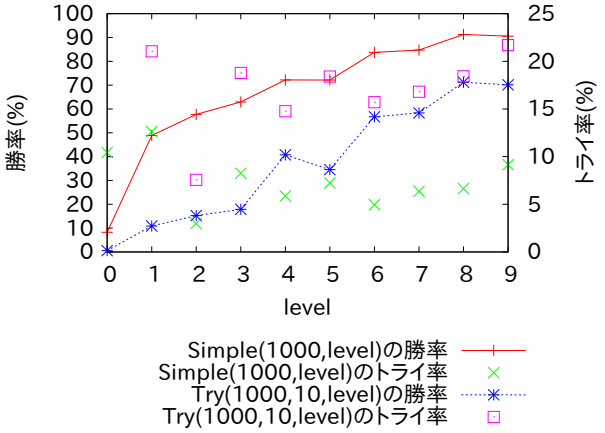


図 7 Try(1000,10,*level*) の勝率とトライ率

Try(1000,10,*level*) に関する結果は**図 7** に示す．左の縦軸は勝率を表す．右の縦軸はトライ率を表す．横軸は *level* の値を表す．トライ率は（プログラムのトライ勝ち数）/（プログラムの勝ち数）で計算される．また，トライ率は右の縦軸を使っているので，範囲は $0-25\%$ の値を表している．

Try はトライ勝ちしやすくなるように期待したプログラムである．そこで，「らいおんが手番のプレイヤから見て，左上，上，右上のいずれかに動く手」を a^* とした．Try は Simple よりも勝った試合の内トライ勝ち率が高く，特徴を表せている．Simple(1000,2) の勝率は 57.7%，Try(1000,10,6) の勝率は 56.7%，Try(1000,10,7) の勝率は 58.35%である．しかし，Simple(1000,2) のトライ勝ちの割合は 2.98%，Try(1000,10,6) のトライ勝ち率は 15.71%，Try(1000,10,7) のトライ勝ちの割合は 16.81%である．勝率が同じ程度でも特徴を表すことができている．Try は提案手法を導入した他のプログラムと違い，実現したい特徴を持つ手をそのまま a^* としていないが，特徴を表すことができた．

Capture(1000,10,*level*) に関する結果は**図 8** に示す．左の縦軸は勝率と捕獲率を表す．横軸は *level* の値を表す．捕獲率は（プログラムが相手の駒を取った確率）/（プログラムの総指し手数）と計算される．Capture は Simple と比較しても特徴が表れていない．

5.4 強さの調整手法と効果

これまでは前向き枝刈の強さで強さの影響を測定していたが，プレイアウト数の変更の影響を紹介して，総合的に

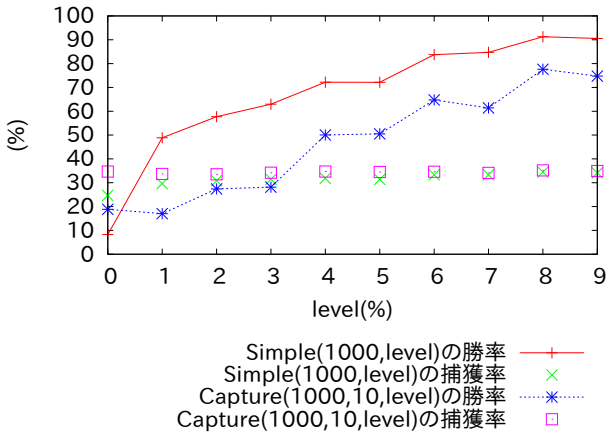


図 8 Capture(1000,10,*level*) の勝率と特徴の出現割合

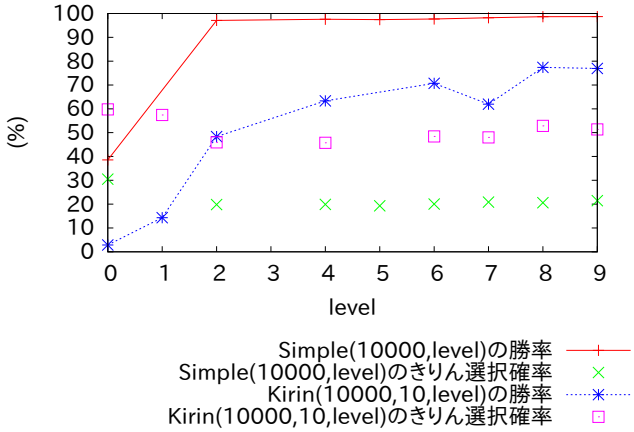


図 9 Kirin(10000,10,*level*) の勝率ときりんの選択回数

議論する．Simple(10000,*level*) と Kirin(10000,10,*level*) を比較した．結果は**図 9** に示す．図 3 の結果と比較すると，プレイアウト回数を増やすことによって勝率が向上していることが分かる．また，提案手法による特徴の実現も行えている．しかし，Kirin(1000,10,0) と Kirin(10000,10,0) の結果の比較より，プレイアウト数を増やすだけでは，勝率は向上していない．提案手法を導入したプログラムを強くするには，更にプレイアウト数を増やすか他の手法との組み合わせを行う必要がある．

5.5 Prior Knowledge の与え方による効果

prior knowledge を与えるノードの範囲を変えることによる，提案手法の効果への影響を評価する．新しいプログラム Root, Own を Kirin と比較する．Root は prior knowledge を UCT のツリーのルートでのノードの生成にしか用いないプログラムである．また，Own は UCT のツリーのルートの手番と同じノードでのノードの生成にしか用いないプログラムである．Root, Own は共にその他の設定は Kirin と同じである．結果を**図 10** に示す．結果から Root と Own は共に Kirin よりも勝率が良かった．Root と Own は自分だけ個性を持っていて，相手は普通のプレ

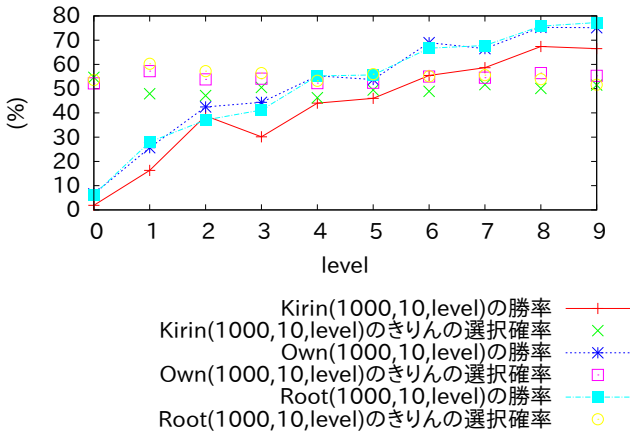


図 10 Root(1000,10,*level*), Own(1000,10,*level*), Kirin(1000,10,*level*) の勝率ときりんの選択確率

イヤーであると考えて思考を行うプログラムと考えることができる．今回は相手が Simple(1000,1) である為，この想定通りの相手と対戦したことになり，結果が良かったと考えられる．しかし，相手を特に想定しない場合は標準的な UCT と戦うものとして実装することが一般的であると考えられる．また，事前に相手に個性があることが分かっていれば，相手番のノードには別の prior knowledge を用いることで，勝率を向上させることができる可能性がある．

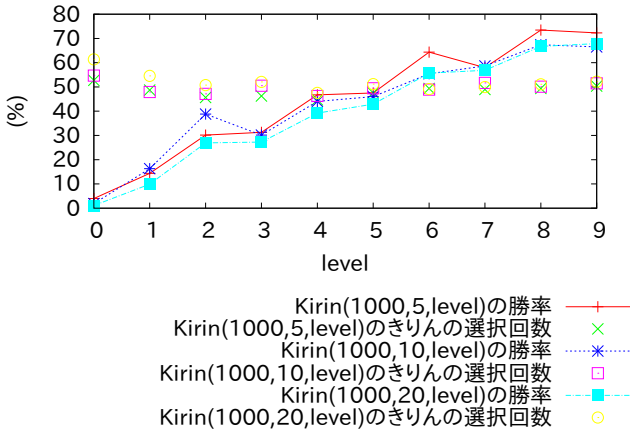


図 11 Kirin(1000,5,*level*), Kirin(1000,10,*level*), Kirin(1000,20,*level*) の勝率ときりんの選択確率

v を大きくすると，prior knowledge による影響が大きくなるので，v が小さいと UCT に近く，v が大きいとより特徴を持ち，探索に無駄が出やすいと考えられる．そこで，Kirin(1000,5,*level*), Kirin(1000,10,*level*), Kirin(1000,20,*level*) の比較を行った．結果を**図 11** に示す．しかし，図 11 からは、観察した範囲では v の違いによる影響は顕著でない．

6. おわりに

本研究では個性を実現することを目標として，UCT に prior knowledge を応用した．実験結果から prior knowledge により，個性を表すことができることが分かった。どのような個性が表せるかについては今後も研究していく必要がある．また同時に，prior knowledge による個性の実現を行う提案手法の様々な性質を確認することができた．個性に prior knowledge を用いる場合，探索効率が落ちることが多く，弱くなる傾向にある．しかし，プレイアウト回数を増やすことやツリーの前向き枝刈りを行うという，一般的な UCT を強くする手法と組み合わせても個性を実現できることが分かった．ただし，本稿の実験の設定ではプレイアウト回数を増やすだけでは，勝率はあまり上がらなかった．提案手法とプレイアウト回数の関係については今後も検討していく必要がある．今後は，Elo-rating などの UCT を強くする他の手法とも組み合わせることができるか検討していく必要がある．また，個性を出す可能性のある手の出現頻度が低いと提案手法は効果を発揮しずらいことも判明した．その為，個性を表し，かつできるだけ出現頻度の高い手を個性を持つ可能性のある手とするような prior knowledge を与える工夫が必要である．提案手法は，個性を表す手法として，今後も研究の予知がある．また，他のゲームへの応用も可能であると期待される．

参考文献

[1] M. Buro. Improving heuristic mini-max search by supervised learning. *Artificial Intelligence*, Vol. 134, No. 1–2, pp. 85–99, January 2002.

[2] Murray Campbell, A. Joseph Hoane, Jr., and Feng-hsiung Hsu. Deep Blue. *Artificial Intelligence*, Vol. 134, No. 1–2, pp. 57–83, January 2002.

[3] Rémi Coulom. Computing elo ratings of move patterns in the game of go. In *Computer games workshop*, 2007.

[4] Sylvain Gelly and David Silver. Combining online and offline knowledge in uct. In *Proceedings of the 24th International Conference on Machine Learning*, ICML'07, pp. 273–280. ACM, jun 2007.

[5] Sylvain Gelly and David Silver. Monte-carlo tree search and rapid action value estimation in computer go. In *Artificial Intelligence*, Vol. 175, pp. 1856–1875. Elsevier Science Publishers Ltd, jul 2011.

[6] 登坂紘介, 松原仁. 将棋における棋譜データベースからの棋士の特徴. 研究報告ゲーム情報学（GI）, 第 2006-GI-16 巻, pp. 9–16. 一般社団法人情報処理学会, jun 2006.

[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. 研究報告ゲーム情報学（GI）, 第 2010-GI-24 巻, pp. 1–7. 一般社団法人情報処理学会, June 2010.

[8] 竜司滝瀬, 哲朗田中. 入玉指向の将棋プログラムの作成. ゲームプログラミングワークショップ 2011 論文集, 第 2011 巻, pp. 25–31, oct 2011.

[9] 池田心, Simon Viennot. モンテカルロ碁における多様な戦略の演出と形勢の制御 ～ 接待碁 ai に向けて. ゲームプログラミングワークショップ 2012 論文集, 第 2012 巻, pp. 47–54, nov 2012.

[10] 田中哲朗. 「どうぶつしょうぎ」 の完全解析. 研究報告ゲーム情報学（GI）, 第 2009-GI-22 巻, pp. 1–8. 一般社団法人情報処理学会, June 2009.