コンピュータ将棋における MC Softmax 探索のための探索深さの指標

岩本裕大 [1]　五十嵐治一 [1]

概要：チェスや将棋などの知的ゲームでは，指し手の評価値の信頼性の目安として探索深さを用いることが考えられる．αβ探索のような全幅探索では，最善応手手順の深さを用いることが一般的である．しかし，選択探索においては，探索木中の末端ノードの深さがまちまちで，最善応手手順の深さだけで探索木全体の深さを評価するのは最善とは言えない．本論文では，選択探索の一種であるモンテカルロソフトマックス探索において，累積バックアップ確率を利用した「探索深さ期待値」を提案する．さらに，この探索深さの指標を，ノード選択方策や探索終了後の着手決定に利用する方法を考えて，それぞれ評価実験を行った．

キーワード：コンピュータ将棋，選択探索，MC Softmax 探索，探索深さ

Expection of Search Depth for Monte Carlo Softmax Search in Computer Shogi

HIROMASA IWAMOTO[†1]　HARUKAZU IGARASHI[†1]

Abstract: The search depth is used as a guideline for the reliability of the evaluation of a piece's move in intellectual games like chess and shogi. In a full-width search such as the alpha-beta search, it is common to use the depth of the principal variation for this purpose. However, in the selective search, the depth of the end nodes in the search tree varies, and it is not the best way to evaluate the depth of the entire search tree based on the depth of the principal variation. In this paper, we propose a "expectation of search depth" based on the cumulative backup probability in the Monte Carlo softmax search, witch is a kind of selective search. In addition, we used the index of search depth in a node selection policy and in a move decision after the completion of the search. The results of our evaluation experiments show the effectiveness of the search depth.

Keywords: Computer Shogi, Selective Search, Monte Carlo Softmax Search, Search Depth

1. はじめに

コンピュータ将棋などの探索アルゴリズムにおいて，探索深さは探索結果の信頼性に関する重要な情報である．一般的な探索法であるαβ探索は全幅探索を基にした探索法であるため，最善応手手順（PV）の深さを探索深さの指標として用いられるのが一般的である．しかし，モンテカルロ木探索[1]のような選択探索においては，探索木中の末端ノードの深さがまちまちで，PV の深さだけで探索木全体の深さを評価するのは最善とは言えない．特に，桐井・原らが提案している MC Softmax 探索[2]は選択探索であり，探索量の指標として PV 深さよりも適切なものが存在する可能性がある．

MC Softmax 探索では，任意のノードからそれ以下の子孫ノードへの累積した選択確率（実現確率）や，その逆の累積したバックアップ確率（累積バックアップ確率）を計算することができるので，これらを利用した探索の深さを考えることができる．本論文では，累積バックアップ確率を利用した「探索深さ期待値」を提案する．実際にこの探索深さの指標を，ノード選択方策や探索終了後の着手決定に利用する方法を考えて，それぞれ評価実験を行った．最終的には MC Softmax 探索を利用した将棋対局プログラムの棋力向上となることを目指す．

本論文の構成は以下の通りである．2.では MC Softmax 探索についての概要と具体的な探索アルゴリズムについて述べる．3.では選択確率における探索深さの問題点を指摘する．4.では探索深さ期待値の定義と再帰的な計算法を示した．5.ではこの探索深さ期待値のノード選択方策への利用法を，6.では探索終了後の着手決定へ利用する方法について述べる．5.と 6.ではそれぞれの評価実験の結果も報告する．

2. MC Softmax 探索

2.1 MC Softmax 探索とは

コンピュータ将棋の評価関数を大量の棋譜データを用いて学習する方式は， Bonanza が採用した「Bonanza メソッド」[3]が代表的である．この学習方式は，プロ棋士などの棋譜データから局面と着手を訓練データとする教師あり学習であった．そこでは，指し手の評価値として，その指し手以下の最善応手手順で得られた末端局面（Principal leaf）

1 芝浦工業大学 工学部 情報工学科
Shibaura Institute of Technology

に出現する特徴量パラメータを学習することができた．すなわち，出現局面での各合法手以下の principal leaf だけが学習対象であった．

一方，強化学習による pricipal leaf のパラメータ学習への取り組みとしては，TDLeaf(λ)[4] 法や PGLeaf 法[5]などの方式が提案されてきた．前者は強化学習の手法として価値ベースの TD 学習法を，後者は方策ベースの方策勾配法である REINFORCE[6]を用いている．

教師あり学習の Bonanza メソッドとこれら２つの強化学習法に共通するのは， principal leaf の局面に出現する特徴パラメータだけを学習の対象としていた点である．探索木に出現するそれ以外の末端局面や，内部ノードの局面については学習対象外であった．また，実装時の探索手法としては，どちらも minimax 探索（α β探索）のような全幅探索を用いていた．

上記のように出現局面の合法手の principal leaf だけではなく，探索木の leaf すべてを学習対象とするために，五十嵐らは minimax 探索ではなく，softmax 探索に基づくモンテカルロ・ソフトマックス探索（MC Softmax 探索，MCSS）を提案した[7]．MCSS では出現局面の合法手の評価値の勾配ベクトルを厳密に計算することができる．最近では，探索木内でのサンプリングによりこの勾配ベクトルを数値的に求める手法が考案され，TD 法や PG 法などの強化学習や棋譜からの教師あり学習，回帰や bootstrapping による学習への適用が提案されている[8]．この学習法では，複数の学習を同時に行うことや，内部ノードを学習対象とすることも可能である．

また，MCSS の探索は，兄弟ノードを同時に評価することが出来れば効率が良い．したがって，評価関数として深層ニューラルネットワークモデルのような計算コストの高い関数を用いる場合でも，GPU を用いることにより複数の局面の計算を並列実行することが可能である．

2.2 探索アルゴリズム

MC Softmax 探索の流れは次の通りである．

1) 初期設定：根ノードをノードuとする．
2) ノード選択：ノードuからノード選択方策による確率に従って子ノードcを選択，cが展開済みであればuを c に置き換えて繰り返す．
3) ノード展開：ノードcの局面における全ての合法手をノード c の子ノードとして生成．
4) 局面評価：3)で生成した子ノードそれぞれの局面を局面評価関数で評価．
5) バックアップ：根ノードからノード c への経路を逆に辿り，経路上のノード評価値をバックアップ方策に従い更新．

1)から 5)の処理を繰り返し行うことで，探索木を徐々に成長させていく．探索木を十分成長させた後，ノード評価値に基づき指し手を決定する．

ノード選択では，Boltzmann 分布により次のように定義されるノード選択方策$\pi_s(s'|s;T_s)$に従い確率的にノードを選択する．

$$\pi_s(s'|s;T_s) = \exp(E(s')/T_s) \Big/ \sum_{x\in C(s)} \exp(E(x)/T_s) \qquad (1)$$

ただし，E(s)はノードsの評価値，$C(s)$はsの子ノードの集合，T_sはノード選択温度である．

バックアップ操作では，以下の式によりノード評価値を更新する．

$$E(s) = \sum_{x\in C(s)} E(x)\,\pi_b(x|s;T_b) \qquad (2)$$

ただし，T_bはバックアップ温度である．$\pi_b(x|s;T_b)$はバックアップ方策で，(1)のノード選択方策π_sと同じ関数形であるが温度T_bが異なる．sが末端ノードのときは，子ノードが存在しないため局面評価関数$H(s)$を用いて$E(s)=H(s)$とする．

バックアップ方策$\pi_b(x|s;T_b)$は，子ノードxが親ノードsに対して，どの程度の重みを持たせてバックアップするかを決定する方策であり，(1)のようなボルツマン分布の関数値で与えられるので「バックアップ確率」と呼ぶ．これに対応して，ノード選択方策$\pi_s(s'|s;T_s)$は「ノード選択確率」と呼ぶ．さらに本論文では，あるノードからその祖先ノードまでのバックアップ確率を掛け合わせた確率を「累積バックアップ確率」，逆にあるノードからその子孫ノードまでの選択確率を掛け合わせた確率を「実現確率」と呼ぶことにする．

なお，$T_s = T_b$の時は，ノード選択確率はバックアップ確率に，実現確率は累積バックップ確率に一致する．

3. 選択探索における探索の深さとは

α β探索は，全幅探索である minimax 探索の高速化手法である．枝刈りは行うが，PV は minimax 探索と同一となるように探索を行っており，PV の末端ノードまでの深さ（以下，「PV 深さ」）で探索木全体の深さを評価することが一般的である．

MCSS のような選択探索の場合，探索木上での読みの深さは均一でなく，探索していないノードの先に有力な手順が含まれている可能性もある．そのため，PV 深さでは探索量を十分に評価できない．具体的な例を図 1 に示す．

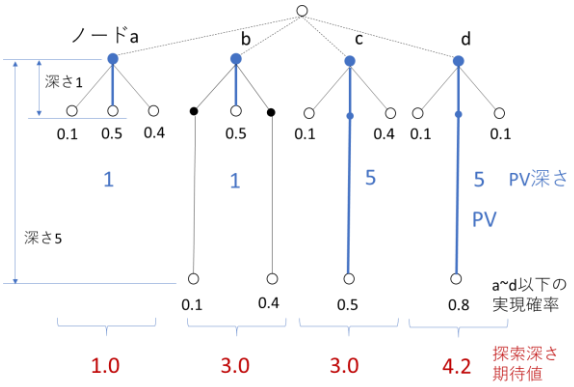

図 1　探索の深さを表す指標

図 1 では根ノードの下に 4 つの指し手があり，対応する子ノードを a~d とする．今，MCSS を実行し，図 1 のような探索木が得られたとする．ただし，a~d にバックアップされてきた評価値は同じとする．図では，白丸は末端局面，青色の太線は PV を表している．また，各末端ノードには，a～d からの実現確率と探索深さ期待値（次章で定義）が示されている．

a~d の評価値が同じときには，PV 深さを用いると，a と b は深さ 1 で同等の信頼度，c と d は深さ 5 で同等の信頼度という結果になる．しかし，b は a よりも PV 以外の有力な変化手順を深さ 5 まで読んでいるので，こちらの方が探索の信頼度は高いと考えるのが合理的である．

また，c は深さは 1 で浅いが実現確率が 0.4 と PV に対して強力な有力な変化手順が存在している．もう少しこの変化手順を読む必要がある．それに対して，d では有力な変化手順がないので，c よりも探索結果の信頼度は大きいと考えるべきである．

このように，本論文で提案する「探索深さの期待値」は，PV や変化手順の実現確率を考慮している．したがって，どの程度有力手順を深く探索した結果であるかという点に関する詳しい情報を，「PV 深さ」よりはきめ細かく与えることができる．

4. 探索深さ期待値の提案

4.1 探索深さ期待値の定義

探索深さ期待値は次のような式で定義する．

$$D(s) \equiv \sum_{l \in leaf(s)} depth(s,l)\, P(l|s;T_d) \qquad (3)$$

$$P(l|s;T_d) = \prod_{i=1}^{depth(s,l)} \pi_b(s_i|s_{i-1};T_d) \qquad (4)$$

ただし，$leaf(s)$はノードsから到達可能な末端ノードの集合，$depth(s,l)$はノードsからノードlまでの深さ，T_dは深さバックアップ温度である．

また，3 章では，実現確率を用いた説明を行ったが，探索木生成後の末端ノード局面の評価値のバックアップにはバックアップ確率を用いている．そこで，(4)の定義ではノード選択確率の代わりにバックアップ確率を用いた．したがって，(4)は実現確率ではなく累積バックアップ確率である．また，T_bを「深さバックアップ温度」と呼ぶ．

4.2 探索深さ期待値の再帰的な計算

(3)の定義では，基準となるノードから末端ノードまでの深さと累積バックアップ確率を深さ期待値を調べるノード毎に計算し直す必要があり，計算効率が非常に悪い．そこで，次のような再帰的な形に変形し，効率的に計算できるようにする．

$$D(s) = \sum_{c \in C(s)} \{D(c)+1\}\, \pi_b(c|s;T_d) \qquad (5)$$

ただし，ノードsが末端ノードのときは$D(s)=0$とする．この導出については，付録 A.1 に記す．

この変形により，探索深さ期待値が子ノードへの参照のみで計算可能になり，計算時間が大幅に短縮される．実装の労力についても，ノード評価値のバックアップ処理の際にバックアップ処理と類似した計算法を実装すればよく，手間はそれほど大きくない．そのため，探索深さ期待値を実用的な指標として用いることが出来るようになった．

5. 探索深さ期待値のノード選択方策への利用

5.1 ノード選択方策への組み込み

探索深さ期待値の利用法の一案として，ノード選択方策への利用を提案する．

MCSS の通常のノード選択方策(1)は，バックアップされた評価値と探索温度によってのみ決定される．評価値の高いノードが選択されやすく，評価値の低いノードは探索されないため見落としが発生する可能性があり，評価関数に高い精度が求められる．モンテカルロ木探索[1]では，UCT によって探索回数の少ないノードを優先的に探索する．それを参考にして，MCSS にも同様の仕組みを取り入れることが出来ないかを考えた．

そこで，ノード選択方策の目的関数として，ノードの評価値$E(s)$ではなく，探索深さ期待値などにより補正した式$E_s(s)$を用いる．すなわち，(1)の代わりに次の式(1')を考える．

$$\pi_s(s'|s;T_s) = \exp(E_s(s')/T_s) \Big/ \sum_{x \in C(s)} \exp(E_s(x)/T_s) \qquad (1')$$

今回は，$E_s(s)$として，UCT や AlphaZero の PUCT[9]を参考に 18 種類の関数$E_s^i(i=1,2,\ldots,18)$を候補とした．次節では，評価実験を行い，この中で最も棋力を高めた関数$E_s(s)$を決定した．なお，候補の具体的な定義は付録 A.2 に掲載した．

5.2 ノード選択方策の決定のための評価実験

ノード選択方策の目的関数を決定するために, (1')の目的関数の 18 候補$E_s^i(i = 1,2,\dots,18)$に従来の(1)を用いたプログラムを加えた 19 プログラムによる将棋の総当たりリーグ戦を行わせた．最終的な勝率から，最良の目的関数を決定する．各関数のハイパーパラメータcは，予備実験により勝率の高いものを事前に決定している．なお，本研究で用いたプログラムは，世界コンピュータ将棋オンライン大会（2020 年 5 月, 以下 WCSOC）に出場した「芝浦将棋 Softmax (WCSOC)」をベースにしている．以下，実験で用いたプログラムを単に「芝浦将棋 Softmax」と呼ぶ．

実験条件と実験結果を以下に示す．なお，互角局面については，やねうら王[10]の互角局面集を利用している．

- CPU：Intel Xeon E5-2695v4
- RAM：64GB
- 使用ソフト：芝浦将棋 Softmax
- 選択温度T_s：120
- バックアップ温度T_b：40
- 深さバックアップ温度T_d：100
- 静止探索：なし
- 持ち時間：1 手 1 秒
- 対局数：組み合わせ毎に 4 局（計 684 局）
- 開始局面：ランダムな互角局面から開始

表 1　目的関数E_s^iごとの総当たりによる勝率

i	c	順位	勝-負-分	勝率
1	0.001	5	40-32-0	.556
2	40	17	23-49-0	.319
3	20	10	37-35-0	.514
4	0.01	11	35-37-0	.486
5	0.01	7	38-34-0	.528
6	0.01	13	32-39-1	.444
7	0.01	7	38-34-0	.528
8	60	6	39-33-0	.542
9	0.005	11	35-37-0	.486
10	0.01	18	22-50-0	.306
11	0.001	19	5-67-0	.069
12	40	14	31-40-1	.431
13	0.01	16	28-44-0	.389
14	40	4	42-30-0	.583
15	40	7	38-34-0	.528
16	1.0	2	61-11-0	.847
17	0.3	3	46-26-0	.639
18	0.5	1	62-10-0	**.861**
従来法	-	15	31-41-0	.431

結果として, E_s^{18}が目的関数の候補の中での勝率が高くなった．その式を以下に示す．

$$E_s^{18}(s) = E(s)p(s) + H(s)\big(1 - p(s)\big) \tag{6}$$

$$p(s) = \frac{1}{2\,D(s)} \tag{7}$$

ただし，$D(s) = 0$ のとき(sが末端ノードであるとき)は$E_s(s) = H(s)$とする．$H(s)$はノードsの局面の静的評価の値である．説明と分かりやすさのため，上式は付録 A.2 の定義式(34)から少し変形を行っている．

(7)の$p(s)$は 0 から 1 の範囲の実数をとり，(6)では$p(s)$でノード評価値$E(s)$と局面評価値$H(s)$の利用割合を制御している．探索深さ期待値$D(s)$が増加すると$p(s)$が減少するため，探索が深くなるほど$E(s)$の比重が大きくなる．探索が浅いうちは局面の第一印象にあたる局面評価$H(s)$を用いることで，ノード評価値の変動の影響を和らげる意図がある．

5.3 PV 深さとの比較と棋力評価

(6)(7)を用いた新たなノード選択方策(1')について，より詳しく棋力評価実験を行う．対局相手を固定して勝率を比較する実験と，棋力を調べるプログラム同士で直接対戦させる実験を行う．また，探索深さ期待値の有用性を検証するため，(7)の$D(s)$に探索深さ期待値の代わりに PV 深さを用いた場合との比較も行う．

勝率比較のための対局相手としては，第 29 回世界コンピュータ将棋選手権（2019 年 5 月，以下 WCSC29）に出場したバージョンの「芝浦将棋 Softmax (WCSC29)」を用いる．今回の提案手法を組み込んだ芝浦将棋 Softmax は, WCSC29 出場後に一から開発し直したもので，MCSS を用いている点は共通しているものの，WCSC29 版とは別のプログラムである．また，WCSC29 版の評価関数は Bonanza ベースの KPP 型評価関数であるのに対し, 本プログラムは Apery[11] ベースの KPPT 型評価関数である．WCSC29 版は静止探索も行っているが，本プログラムでは行っていない点も異なっている．

棋力を比較するプログラムは，ノード選択方策として従来の(1)を用いる「従来法」, 提案手法であるノード選択方策に(1')(6)(7)を用いる「条件 A」，条件 A の探索深さ期待値を PV 深さに置き換えた「条件 B」の 3 つである．

同一ソフトに対する勝率比較実験の実験条件と対局結果を以下に示す． 使用した PC やソフトは前実験と同一である．

- 選択温度T_s：120
- バックアップ温度T_b：40
- 深さバックアップ温度T_d：100
- 静止探索：なし
- 持ち時間：1 手 10 秒

- 対局数：100 局
- 開始局面：互角局面 50 局を先後入れ替えて利用
- 対局相手：芝浦将棋 Softmax (WCSC29)

表 2　芝浦将棋 Softmax(WCSC29)との対局結果

対局ソフト	勝-負-分	勝率
従来法（深さ情報無し）	58-39-3	58%
条件 A（本提案）	83-13-4	83%
条件 B（PV 深さ）	81-19-0	81%

次に直接対局実験の実験条件と結果を以下に示す.

- 選択温度T_s：120
- バックアップ温度T_b：40
- 深さバックアップ温度T_d：100
- 静止探索：なし
- 持ち時間：1 手 1 秒
- 対局数：1000 局
- 開始局面：ランダムな互角局面から開始

表 3　直接対決の対局結果

組み合わせ	勝-負-分	勝率
条件 A vs 従来法	875-124-1	87.5%
条件 A vs 条件 B	587-413-0	58.7%

まず提案手法によるノード選択方策の導入の効果について考察する.表 2 の同一ソフトに対する勝率の比較では，従来手法では勝率 58%だったのに対し，提案手法である条件 A の勝率は 83%と勝率が 25 ポイント上昇している．しかも，表 3 の従来法と条件 A の直接対決では条件 A が大幅に勝っている．これらの結果から，ノード選択方策に探索の深さに応じた補正を加えることで将棋の棋力向上に役立っていると言える.

次に，条件 A と条件 B の勝率を比較することで PV 深さと比較した探索深さ期待値の優位性を考察する．同一ソフトに対する勝率は条件 A と条件 B で勝率は殆ど同じか条件 A が若干高い程度だが，直接対決では条件 A が条件 B に対して 58.7%と勝ち越している．直接対決の勝率について検定を行うと 5 割より有意に高く，探索深さ期待値の優位性が示された.

6. 着手決定への利用

6.1 着手決定関数について

探索深さ期待値の第 2 の利用案として，探索終了後の着手決定に利用する方法を提案する．通常では盤上のルート局面に対して評価値が最大となる子ノードの指し手を選択する．これに対し，探索深さ期待値から，その指し手以下の探索が十分に行われているかを考慮した着手決定を行う.アルゴリズムを以下に示す.

1) ルートノードsの探索深さ期待値$D(s)$と定数$r \in [0,1]$を参照し，$r(D(s)-1)$を基準値bとする.
2) sの子ノードの探索深さ期待値が基準値bよりも大きいノードの内，ノード評価値が最大のノードを選択する.

基準値bの中の$D(s)-1$は，(5)の再帰式から，子ノードの探索深さ指標の深さバックアップ方策で重み付けした期待値平均と等しくなっている.

6.2 評価実験

提案手法によるノード選択方策を組み込んだプログラムに，着手決定の提案手法を実装した，その上で探索深さの情報として，探索深さ期待値と PV 深さをそれぞれ用いた場合の同一ソフトに対する勝率比較実験を行う.

棋力を比較するプログラムは，5.3 の条件 A に着手決定関数を導入した「条件 C」と．条件 B に探索深さ期待値の代わりに PV 深さを用いた着手決定関数を導入した「条件 D」の 2 つである.

実験条件と結果を以下に示す.

- 選択温度T_s：120
- バックアップ温度T_b：40
- 深さバックアップ温度T_d：100
- 静止探索：なし
- 提案手法の定数 $r = 0.5$
- 持ち時間：1 手 10 秒
- 対局数：100 局
- 開始局面：互角局面 50 局を先後入れ替えて利用
- 対局相手：芝浦将棋 Softmax (WCSC29 版)

表 4　芝浦将棋 Softmax(WCSC29 版)との対局結果

対局ソフト	勝-負-分	勝率
条件 C（本提案）	90-6-4	90%
条件 D（PV 深さ）	78-19-3	78%

表 4 と 5.3 の表 2 の結果を比較すると，探索深さ期待値を用いた場合は勝率が 83%から 90%と上昇したのに対し，PV 深さを用いた場合では勝率は上昇しなかった.これは，PV 深さの場合は評価値最大のノードがそのまま基準値に用いられてしまうため，候補手のふるい落としが機能しなかったためと考えられる．この結果から，探索深さ期待値を用いた着手決定が有効であると言える.

5.3 の結果も併せ,探索深さ期待値を用いた探索木の状態評価の有用性，探索深さ期待値の PV 深さに対する優位性を示すことができた.

7. その他の利用法

探索深さ期待値の他の利用法の案として，思考時間の制御やノード選択温度の制御，定跡生成時などでの探索結果の評価などが挙げられる．それぞれについて説明する．

7.1 思考時間の制御

実際の対局において，持ち時間を有効に利用することは重要な課題の一つである．持ち時間を有効に活用するためには，有力な候補手の多い難しい局面に時間をかけ，有力な手が一つしか無いような局面では時間を節約する必要がある．手の狭い局面では，探索木は深さ方向に成長しやすく，探索深さ期待値の増加速度が速くなると考えられる．そのため盤上のルートノードの探索深さ期待値を観測することで，時間を適切に配分できるのではないかと考えられる．

7.2 ノード選択温度

本論文ではノード選択方策の目的関数を変更する形でノード選択方策を制御したが，探索量の増加に合わせて選択温度を変化させることも考えられる．一例として次のような式を示す．

$$T_s(s) = \alpha^{D(s)} T_{s0} \tag{8}$$

ここで，αは 1 付近の定数，T_{s0}は温度の基準となる定数である．αを 1 未満とすると探索が深くなるほど温度が低くなり，反対にαを 1 より大きくすると探索が深くなるほど温度が高くなる．

7.3 定跡生成

定跡に新しく棋譜を登録する際や，複数の定跡を統合する際などに，同じ局面についての探索結果が衝突する可能性がある．この優先順位付けに探索深さ期待値を用いることが出来ると考えられる．

8. おわりに

本論文では，選択探索である MC Softmax 探索において探索木全体の深さを評価するための新たな指標,「探索深さ期待値」を提案した．探索深さ期待値はルートノードから末端ノードまでの深さを，累積バックアップ確率を重みとした期待値として定義している．この値は再帰的に効率良く計算することが可能である．

探索深さ期待値の利用例として，ノード選択方策への組み込みと探索後の着手決定方法への利用を提案した．これら 2 種類の利用については棋力の測定実験を行った．その結果から探索深さ期待値を用いることにより棋力が向上していること，さらに PV 深さを用いることよりは有意に有効であることが確認できた．

今後は，探索深さ期待値の上記以外の利用法の具体化と実装を行っていきたい．さらに，将棋以外のゲームにも探索深さ期待値を用いた手法が適用可能かどうか検証するのも今後の課題である．

参考文献

[1] Coulom, R.. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. Computers and Games, 2006, p. 72-83.

[2] 桐井杏樹, 原悠一, 五十嵐治一, 森岡祐一, 山本一将. 確率的選択探索の将棋への適用. ゲーム・プログラミング・ワークショップ GPW2017 論文集, 2017, p. 26-33.

[3] 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. ゲームプログラミングワークショップ 2006 論文集, 2006, p. 78-83.

[4] Baxter, J., Tridgell, A., and Weaver, L.. KnightCap: A chess program that learns by combining TD(λ) with game-tree search. Proceedings of the Fifteenth International Conference (ICML '98), 1998, p. 28-36.

[5] 森岡祐一, 五十嵐治一. 方策勾配法と α β 探索を組み合わせた強化学習アルゴリズムの提案. ゲーム・プログラミング・ワークショップ GPW2012 論文集, 2012, p. 122-125.

[6] Williams, R. J. Simple Statistical Gradient- Following Algorithms for Connectionist Reinforcement Learning. Machine Learning, 1992, vol.8, p. 229-256.

[7] 五十嵐治一, 森岡祐一, 山本一将. 方策勾配法による静的局面評価関数の強化学習についての一考察. ゲーム・プログラミング・ワークショップ GPW2012 論文集, 2012, p.118-121.

[8] 五十嵐治一, 森岡祐一, 山本一将. MC Softmax 探索における局面評価関数の学習. ゲーム・プログラミング・ワークショップ 2018 論文集, 2018, p. 212-219.

[9] David Silver et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science, 2018, vol. 362, p. 1140-1144.

[10] “やねうら王”. http://yaneuraou.yaneu.com/, (参照 2020-10-09).

[11] “Apery”. https://hiraokatakuya.github.io/apery/, (参照 2020-10-06).

付録

付録 A.1 探索深さ期待値の再帰的計算の導出

(i)ノードsが末端であるときは，ノードnを根とする探索木の末端はノードn自身のみであるので，$depth(s,s) = 0$より$D(s) = 0$である．

(ii)ノードsが展開済みノードのとき，その子ノードの集合を$C(s)$，その内の 1 つを$x \in C(s)$とする．ノードの親子関係より $depth(s,l) = depth(x,l) + 1$ が成り立つので，

$$D(x) = \sum_{l \in leaf(x)} \{depth(s,l) - 1\} P(l|x;T_d) \tag{9}$$

$$= \sum_{l \in leaf(x)} depth(s,l) P(l|x;T_d) - \sum_{l \in leaf(x)} P(l|x;T_d) \tag{10}$$

となる．この第二項は全ての末端の累積バックアップ確率の総和であるため値は 1 である．また，(10)の第一項のバックアップ方策について，ノードの親子関係より，

$$P(l|x;T_d) = \frac{\pi_b(x|s;T_d)}{\pi_b(x|s;T_d)}\,P(l|x;T_d) \tag{11}$$

$$= \frac{1}{\pi_b(x|s;T_d)}P(l|x;T_d) \tag{12}$$

が成り立つ．これを(10)の第一項に代入すると，

$$D(x) = \frac{1}{\pi_b(x|s;T_d)}\sum_{l\in leaf(x)} depth(s,l)\,P(l|s;T_d) - 1 \tag{13}$$

$$\{D(x)+1\}\,\pi_b(x|s;T_d) = \sum_{l\in leaf(x)} depth(s,l)\,P(l|s;T_d) \tag{14}$$

となる．木グラフの性質から，全ての子ノードの末端ノードの集合の総和は親ノードの末端ノードの集合と等しい（$leaf(s) = \coprod_{x\in C(s)} leaf(x)$）ため，

$$D(x) = \sum_{x\in C(s)}\sum_{l\in leaf(x)} depth(s,l)\,P(l|x;T_d) \tag{15}$$

$$= \sum_{x\in C(s)} \{D(x)+1\}\,\pi_b(x|s;T_d) \quad (\because (14)) \tag{16}$$

が成り立ち，再帰的な形が導出できる．

付録 A.2 ノード選択方策の目的関数の候補

$$E_s^1(s) = E(s) + c\frac{\ln(N(s_p))}{N(s)+1} \tag{17}$$

$$E_s^2(s) = E(s) + cE_0(s)\frac{\ln(N(s_p))}{N(s)^2+1} \tag{18}$$

$$E_s^3(s) = E(s) + c\frac{D(s_p)}{\exp(D(s))} \tag{19}$$

$$E_s^4(s) = E(s) + c\frac{\ln\left(N(s_p)\right)}{N(s)^3+1} \tag{20}$$

$$E_s^5(s) = E(s) + cE_0(s)\frac{\ln\left(N(s_p)\right)}{N(s)^3+1} \tag{21}$$

$$E_s^6(s) = E(s)\left\{1 + c\frac{\ln\left(N(s_p)\right)}{N(s)^3+1}\right\} \tag{22}$$

$$E_s^7(s) = E(s) + cE_0(s)\exp\left(D(s_p)/2 - D(s)\right) \tag{23}$$

$$E_s^8(s) = E(s) + c\exp\left(D(s_p)/2 - D(s)\right) \tag{24}$$

$$E_s^9(s) = E(s) + cE_0(s) \tag{25}$$

$$E_s^{10}(s) = E(s)\left\{1 + c\exp\left(D(s_p)/2 - D(s)\right)\right\} \tag{26}$$

$$E_s^{11}(s) = E(s)(1 + c\frac{\mathrm{N(s_p)}}{N(s)^2+1}) \tag{27}$$

$$E_s^{12}(s) = \begin{cases} E(s) - cD(s) & if\ E(s) \geq 0 \\ E(s) + cD(s) & if\ E(s) < 0 \end{cases} \tag{28}$$

$$E_s^{13}(s) = E(s)\left(1 + cD(s)\right) \tag{29}$$

$$E_s^{14}(s) = E(s) + c\frac{D(s_p)}{D(s)^2+1} \tag{30}$$

$$E_s^{15}(s) = E(s) + cD(s) \tag{31}$$

$$E_s^{16}(s) = E(s)\left\{1 - \frac{c}{D(s)+1}\right\} + E_0(s)\frac{c}{D(s)+1} \tag{32}$$

$$E_s^{17}(s) = E(s)\left\{1 - \frac{c}{\sqrt{D(s)}}\right\} + E_0(s)\frac{c}{\sqrt{D(s)}} \tag{33}$$

$$E_s^{18}(s) = \begin{cases} E(s)\left\{1 - \frac{c}{D(s)}\right\} + E_0(s)\frac{c}{D(s)} & if\ D(s) > 0 \\ E_0(s) & if\ D(s) = 0 \end{cases} \tag{34}$$

ただし，s_pはノードsの親ノード，$N(s)$はノードsの訪問回数，cはそれぞれの関数で設定されるハイパーパラメータを表す．本文中の(6),(7)は，E_s^{18}を$c = 1/2$としたときの式である．