2005－GI－14 (2)
2005／9／5

将棋における ProbCut の静止探索への応用

竹内聖悟† 金子知適† 川合慧†

本稿では、将棋において ProbCut を静止探索へと応用し、その有効性を示した。ProbCut は浅い探索と深い探索の結果間の相関に基づく前向き枝刈手法で、はじめにオセロにおいて導入され[7]、チェスにおいても有効に応用された。しかし、将棋における先行研究では、有効性が限定的にしか示されていない。強いプログラムを作ることに成功していないという意味で原因は主に、将棋において局面の評価が難しいから他のゲームに比べ相関が低いこともこの問題を解決するために、ProbCut を通常探索にではなく静止探索に応用した。問題集を解く実験から、ProbCut を用いたプログラムがより短い時間でより多くの問題を解けること、さらに自己対戦の結果 (50 勝 30 敗) から強さが確認された。

Application of ProbCut into QuiescenceSearch in Shogi

SHOGO TAKEUCHI ,† TOMOYUKI KANEKO † and SATORU KAWAI †

In this paper, we propose to apply ProbCut into quiescence search in Shogi, and show its effectiveness. ProbCut is a sophisticated technique for selective search, based on correlation between results of shallow search and deep search. It was originally intrudeced in Othello[7], and already successfully applied to Chess. However, in previous work on Shogi, the effectiveness of ProbCut has been very limited in a sense that ProbCut has not succeeded to make significantly strong program. It is mainly because evaluation of positions are much more difficult in Shogi and the correlation is not so strong compared to other games. In order to resolve this difficulty, we introduced ProbCut into quiescence search instead of normal (toplevel) search. In experiments on solving standard test set of Shogi, we show that our program with ProbCut solves more problems in less time compared to original version. Its strongness is also confirmed by self play results (50 win, 30 lose).

† 東京大学大学院総合文化研究科
Department of General Systems Studies, Graduate School of Arts and Sciences, The University of Tokyo
{takeuchi,kaneko,kawai}@graco.c.u-tokyo.ac.jp

1. はじめに

ProbCutは浅い探索と深い探索の結果間の相関に基づく、前向き枝刈手法であり、オセロにおいて初めて応用された[5]、チェスにおいても有効に応用された。しかし、将棋における先行研究ではその有効性は限定的にしか示されておらず、市販ソフトを含む強い将棋プログラムはProbCutを採用していない。全幅探索においてはProbCutを用いた方が強くなることが示された[16]が、一般的なプログラムで用いられるような前向き枝刈手法は採用されておらず、前向き枝刈なしの将棋プログラムは非常に弱いため、ProbCutが一般的な将棋プログラムにおいても有効であるかは一概に言えない。他の研究では探索ノード数がProbCutにより減少したことを示したが、棋力の向上ははっきりとは示されていない[13]。

ProbCutの将棋への応用における問題点は、局面の評価が困難であることと、探索結果間の相関が他のゲームと比べると強くないことが主である。この問題を解決するため、本研究ではProbCutを通常探索ではなく静止探索へと応用する。

本稿では、将棋においてProbCutを静止探索へと応用し、その有効性を示す。まず、浅い探索の結果と深い探索の結果の間の相関が通常探索のそれよりも静止探索の方が高いことを示し、静止探索がProbCutを応用するのによりふさわしいことを示した。次に、ProbCutを使ったプログラムがオリジナルよりも良い性能となることを示した。問題集を解かせた結果からProbCutを用いたプログラムがオリジナルよりも良い性能を示し、さらに、自己対戦の結果(50勝30敗)からも棋力の向上を示すことが出来た。

2. 関連研究

2.1 前向き枝刈手法

アルファベータ法の成功の後、前向き枝刈手法は、適切な変化を深く集中的に探索するような手法が特に研究されてきた。ある分野に特化した枝刈は力強いが、null move pruning, 静止探索[3]を含むような分野に依存しない手法が発展、成功してきた。最近では、実現確率探索という手法が提案され[15]、多くの将棋プログラムで採用されている。この手法は半自動的に分野依存の知識を取り入れ、各指し手に対して適切な打ち切り深さをプロ棋士の棋譜から統計的に決定するというものである。

展開するノードを選ぶために、評価値の信頼性を考えるのは有用であり、いくつかの手法では評価関数を拡張し[2,4,11]、また他の手法では共謀数[8]や、pn探索[1]の証明数と反証数のような追加の数を導入した。Df-pnは深さ打ち切りのpn探索で、将棋[9,10]と囲碁[7]の終盤において決定的な役割を果した。

2.2 ProbCut

ProbCutはオセロ,チェス,将棋に応用された。オセロ,チェスともに良い結果を出しているが、将棋ではその成果は、全幅探索の時だけなど限定的である。

3. ProbCut

ProbCutはBuroによって提案された選択的探索深化の手法で、オセロにおいて成功を収めた。ProbCutは浅い探索の結果と深い探索の結果の間に強い相関があると考え、浅い探索の結果を元に深い探索の結果を確率的に予測し、探索の枝刈りに利用しようというものである。浅い探索の結果から深い探索の結果を予測し、それが探索ウィンドウの外に具体的には、浅い探索の結果が探索ウィンドウの外にあると高確率で言える、探索ウィンドウからあるマージン以上に外にある、時にその枝を刈ってしまうものである。

4. 将棋について

将棋はチェスに良く似たゲームであるが、取った駒を再利用できるという点で大きくチェスと異なる。このルールによりゲームの終盤になっても駒の数が減らず、合法手数は大きなままで、ゲームとしてより複雑なものとなっている。

5. GPS将棋

本稿での実験で使用した将棋プログラムGPS将棋はOSL[14]を利用しており、市販ソフトに劣らない実力を持

表 1 対象局面

	手数	棋譜数	全局面数	対象局面数	割合	除外局面数	割合
PieceEval	40 - 60	100	2,000	2,000	100.0	0	0.0
	60 - 80		1,983	1,977	99.7	6	0.3
	80 -		3,613	3,349	92.7	264	7.3
	Total		7,596	7,326	96.4	270	3.6
ProgressEval	40 - 60	100	2,000	2,000	100.0	0	0.0
	60 - 80		1,983	1,976	99.6	7	0.4
	80 -		3,613	3,328	92.1	285	7.9
	Total		7,596	7,304	96.2	292	3.8

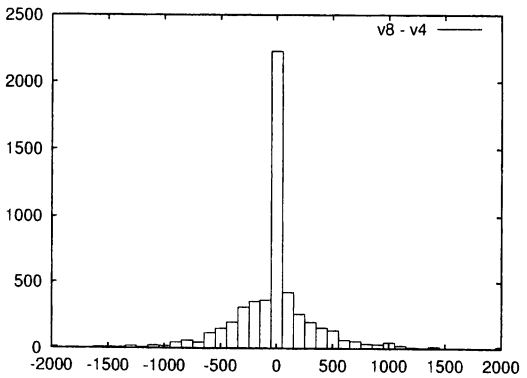

図 1 深さ 4,8 の探索結果の差のヒストグラム (PieceEval)

ち、2005 年の世界コンピュータ将棋選手権では 8 位に入賞した。

5.1 実現確率探索

GPS 将棋の通常探索は、実現確率探索と MTD(f)-best[12] の組み合わせで行われている。

5.2 静止探索

GPS 将棋では、通常探索の葉ノードにおいて静止探索を行っている。駒を取る手や王手など指手を限定し、打切り深さ 8 としてアルファベータ探索を行っている。

5.3 評価関数

駒得と玉の危険度を考慮した評価関数 (ProgressEval) を用いている。実験では、駒得の部分の評価関数 (PieceEval) も用いた。

6. 実験結果

6.1 異なる深さの探索結果間の関係

浅い探索の結果と深い探索の結果の間の相関を測るため実験を行った。ある局面から深さ 4,8 の静止探索を行い、その差を測った。将棋 24 万局中の棋譜 100 局分を用いた。指し手、局面については、中終盤を対象とするために 60 手目以降を対象とした。さらに、詰めろなどで値の間に差が 160,000 以上生じた 101 局面を除き、最終的な実験対象 5496 局面を得た。結果は表??のようになった。

6.1.1 静止探索

本実験で対象としている値に関しては、図 1, 図 2 のようになっており、$[v8' - n * \sigma, v8' + n * \sigma]$ の間にある確率で v8 が存在すると言える。よって、$v8' - n * \sigma > \beta$, $v8' + n * \sigma < \alpha$ を満たした時に、α カット, β カットが実際の値 v8 でも起きていると推定し、カットを行う。以降、$w = n * \sigma$ としてマージンと呼び使用する。

6.1.2 Comparison with Other Games

先行研究で得られた相関関係のデータと本研究で得られたデータとを比較した。結果は、表 2 のようになった。将棋の先行研究と比較すると、深さの関係もあってか分散はやや大きくなったが、相関は高くなっており、

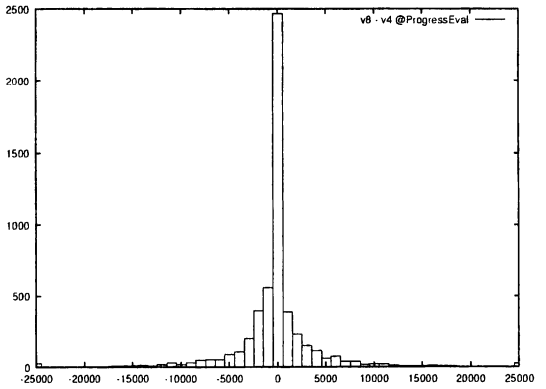

図 2 深さ 4,8 の探索結果の差のヒストグラム (ProgressEval)

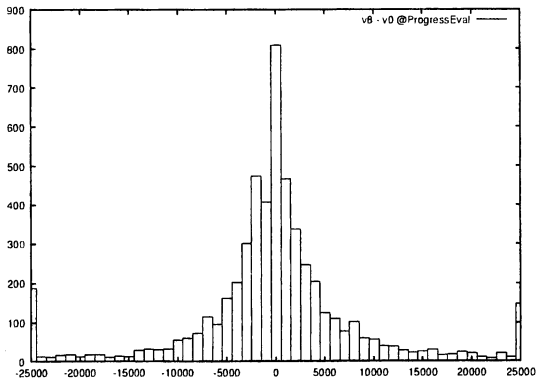

図 3 深さ 0,8 の探索結果の差のヒストグラム (ProgressEval)

静止探索への応用が有効であることを示しているといえる。また、通常探索を用いたもの (Realization Probability Search) と比較すると、相関が高く、分散が低くなっており、やはり静止探索への応用が有効であると言える。

6.2 効率化とリスクの推定

探索の効率化と ProbCut によるリスクを測るための実験を行った。この場合のリスクとは、ProbCut を使わない場合にカットしてはいけないノードを ProbCut を用いることでカットしてしまうことである。

対象局面に対して探索深さを 8,4 として探索を行った。また、ProbCut と実際の探索コストを比較するため、探索深さ 8 の $\alpha\beta$ 探索を行った。

結果は、**表 4**, **表 5** のようになった。なお、ノード数比は (深さ 4 の $\alpha\beta$ 探索ノード数) / (深さ 8 の $\alpha\beta$ 探索ノード数) であり、A,B,C,D は**表 3** と対応している。さらに、探索効率を次のように求めた。なお、ALL は A,B,C,D 全てである。

$$(\text{探索効率}) = \frac{(\text{深さ } 4 \text{ の } \alpha\beta \text{ 探索ノード数 (ALL)} + \text{深さ } 8 \text{ の } \alpha\beta \text{ 探索ノード数 (B, D)})}{(\text{深さ } 8 \text{ の } \alpha\beta \text{ 探索ノード数 (ALL)})}$$

結果は、**表 6** のようになった。マージン w を歩 4 枚分以上とすると ProbCut よりも効率が悪くなってしまうため、それ以下の数字を用いるのが適切と考えられる。リスクも多くて 3%程度であるが、これがどの程度利いてくるのかは問題集や自己対戦などの実際に探索をさせてみないと評価はできない。

6.3 問 題 集

静止探索への ProbCut による能力の向上を見るため、問題集 (ラクラク次の一手) を解かせた。結果は**表 7** で、本研究で作成したプログラムの方が正答数が多く、解答に要するノード数は少なくなった。結果から ProbCUt に

表 2 本研究と関連研究のデータの比較

ゲーム	r^2	σ	d	d'	局面	備考
オセロ	0.97	.	8	4	28 discs	5)
	0.96	.	8	4	44 discs	
チェス	0.81	55.8	5	3	middle-game	6)
	0.88	51.8	5	3	endgame	
	0.67	82.	8	4	middle-game	
	0.81	75.	8	4	endgame	
将棋	.	137.4	5	3		16)
	.	155.5	4	2		
	.	213.0	3	1		
将棋	0.74	263.0	4	2	opening	13)
	0.87	302.8	4	2	middle-game	
	0.88	406.9	4	2	end-game	
将棋	0.951	357.8	8	4	all	(Ours: Quiescence Search, PieceEval)
	0.921	155.1	8	4	40-60	(Ours: Quiescence Search, PieceEval)
	0.944	258.6	8	4	60-80	(Ours: Quiescence Search, PieceEval)
	0.954	401.5	8	4	80-	(Ours: Quiescence Search, PieceEval)
	0.947	308.7	8	4	all	(Ours: Quiescence Search, ProgressEval)
	0.946	91.36	8	4	40-60	(Ours: Quiescence Search, ProgressEval)
	0.946	186.2	8	4	60-80	(Ours: Quiescence Search, ProgressEval)
	0.948	362.5	8	4	80-	(Ours: Quiescence Search, ProgressEval)
	0.880	466.1	8	4	all	(Ours: Realization Probability Search, PieceEval)
	0.750	282.6	8	4	40-60	(Ours: Realization Probability Search, PieceEval)
	0.855	424.9	8	4	60-80	(Ours: Realization Probability Search, PieceEval)
	0.897	576.1	8	4	80-	(Ours: Realization Probability Search, PieceEval)
	0.849	428.5	8	4	all	(Ours: Realization Probability Search, ProgressEval)
	0.729	237.0	8	4	40-60	(Ours: Realization Probability Search, ProgressEval)
	0.801	369.0	8	4	60-80	(Ours: Realization Probability Search, ProgressEval)
	0.867	561.5	8	4	80-	(Ours: Realization Probability Search, ProgressEval)
	0.798	636.6	8	0	all	(Ours: Quiescence Search, PieceEval)
	0.526	379.1	8	0	40-60	(Ours: Quiescence Search, PieceEval)
	0.739	558.5	8	0	60-80	(Ours: Quiescence Search, PieceEval)
	0.831	766.9	8	0	80-	(Ours: Quiescence Search, PieceEval)
	0.761	569.8	8	0	all	(Ours: Quiescence Search, ProgressEval)
	0.514	274.5	8	0	40-60	(Ours: Quiescence Search, ProgressEval)
	0.667	460.0	8	0	60-80	(Ours: Quiescence Search, ProgressEval)
	0.792	722.2	8	0	80-	(Ours: Quiescence Search, ProgressEval)

表 3 ProbCut の利用による探索結果の組み合わせ

		ProbCut を利用しない	
		T	F
ProbCut を利用	T	A	C (false positive)
	F	B (false negative)	D

表 4 ProbCut の効果 (PieceEval)

	w = 0.5 pawn				w = 1.0 pawn			
	#positions	rate(%)	node w/o PC	node w PC	#positions	rate(%)	node w/o PC	node w PC
A	5,458	74.59	1.546	0.256	4,937	67.38	1.361	0.252
B	468	6.36	0.625	0.723	1,014	13.84	0.871	1.026
C	289	3.93	0.597	0.046	208	2.84	0.456	0.037
D	1,112	15.12	0.653	0.746	1,168	15.94	0.734	0.840
	w = 4.0 pawns				w = 10 pawns			
	#positions	rate(%)	node w/o PC	node w PC	#positions	rate(%)	node w/o PC	node w PC
A	4,624	63.18	1.174	0.381	2,411	32.91	0.395	0.176
B	1,346	18.37	1.119	1.373	3,605	49.20	1.979	2.804
C	129	1.76	0.278	0.039	7	0.10	0.011	0.001
D	1,223	16.69	0.852	0.984	1,304	17.80	1.038	1.207

表 5 ProbCut の効果 (ProgressEval)

	w = 0.5 pawn				w = 1.0 pawn			
	#positions	rate(%)	node w/o PC	node w PC	#positions	rate(%)	node w/o PC	node w PC
A	5,262	72.03	1.441	0.259	4,911	62.23	1.251	0.249
B	592	8.10	0.594	0.700	963	13.18	0.823	0.980
C	218	2.98	0.529	0.035	152	2.08	0.378	0.025
D	1,233	16.88	0.554	0.646	1,279	17.51	0.667	0.773
	w = 4.0 pawns				w = 10 pawns			
	#positions	rate(%)	node w/o PC	node w PC	#positions	rate(%)	node w/o PC	node w PC
A	4,365	59.75	0.989	0.325	1,853	25.37	0.313	0.101
B	1,531	20.96	1.111	1.412	4,074	55.77	1.838	2.689
C	80	1.10	0.235	0.022	2	0.03	0.021	0.000
D	1,329	18.19	0.783	0.912	1,376	18.84	0.947	1.097

表 6 ProbCut の探索効率

		w = 0.5 pawn	w = 1.0 pawn	w = 4.0 pawns	w = 10 pawns
PiceEval	40 - 60	0.862	1.057	1.251	1.590
	60 - 80	0.649	0.791	1.034	1.512
(quiescence-search)	80 -	0.478	0.548	0.689	1.182
	Total	0.518	0.630	0.812	1.224
ProgressEval	40 - 60	1.031	1.120	1.335	1.623
	60 - 80	0.708	0.837	1.079	1.506
(quiescence-search)	80 -	0.425	0.564	0.749	1.186
	Total	0.526	0.650	0.856	1.247
PiceEval	40 - 60	0.511	0.659	0.702	0.968
	60 - 80	0.289	0.386	0.461	0.843
(realization probability search)	80 -	0.249	0.309	0.379	0.740
	Total	0.328	0.422	0.486	0.829
ProgressEval	40 - 60	0.677	0.741	0.913	1.015
	60 - 80	0.407	0.503	0.768	1.956
(realization probability search)	80 -	0.246	0.306	0.570	0.812
	Total	0.438	0.510	0.744	0.923

表 7 問題集の結果

	正答数	node-rate
noPC8	169	1.0
noPC4	162	0.498
0pawns	172	0.620
1pawn	174	0.743
2pawns	173	0.837
3pawns	180	0.947

表 8 自己対戦の結果

margin	win	lose	draw
1pawn	42	35	3
3pawns	50	30	0

より能力が向上したといえる。また、誤ってカットしてしまうリスクもこの結果からは、大きな影響はないと考えられる。

6.4 自己対戦

ProbCut を用いたプログラムと用いないプログラムとの対戦を行った。なお、マージン w は歩 1 枚分,3 枚分とした。

結果は表 8 のようになり、ProbCut を用いることによる棋力の向上が明らかに見られた。

7. おわりに

本稿では、将棋において ProbCut を静止探索へと応用し、その有効性を示した。ProbCut とは浅い探索と深い

探索の結果間の相関に基づいた前向き枝刈手法である。しかし、将棋においては結果間の相関がはっきりと強くないため、その有効性は限定的にしか示されてこなかった。本研究では通常探索ではなく静止探索へと ProbCut を応用することで、将棋へと ProbCut を効果的に取り込むことができた。問題集を ProbCut を用いたプログラムの方がオリジナルよりも短い時間でより多く正答し、さらに 50 勝 30 敗という自己対戦の結果からも ProbCut による棋力の向上を示すことができた。

今後の課題としては、適切なマージンの値を求めることで、これは問題集や自己対戦などの実験を多く行うことで求められる。

参 考 文 献

1) L. Victor Allis, Maarten van der Meulen, and H. Jaap van den Herik. Proof-number search. *Artificial Intelligence*, 66:91–124, 1994.
2) Eric B. Baum and Warren D. Smith. A bayesian approach to relevence in game playing. *Artificial Intelligence*, 97:195–242, 1997.
3) Don F. Beal. A generalised quiescence search algorithm. *Artificial Intelligence*, 43:85–98, 1990.
4) H. Berliner. The B* tree search algorithm: A best-first proof procedure. *Artificial Intelligence*, 12:23–40, 1979.
5) Michael Buro. Probcut : An effective selective extension of the α - β algorithm. *ICCA Journal*, 18(2):71–76, 1995.
6) A.X. Jiang and M. Buro. First experimental results of probcut applied to chess. In *Proceedings of the Advances in Computer Games Conference 10*, pages 19–32, Graz, 2003. Kluwer Academic Publishers.
7) Akihiro Kishimoto and Martin Mueller. Df-pn in go: An application to the one-eye problem. In *Advances in Computer Games 10*, pages 125–141. Kluwer Academic Publishers, 2003.
8) D. A. McAllester. Conspiracy numbers for min-max search. *Artificial Intelligence*, 35:287–310, 1988.
9) A. Nagai and H. Imai. Application of df-pn$^+$ to othello endgames. In *Game Programming Workshop in Japan '99*, pages 16–23, October 1999.
10) Ayumu Nagai. A new depth-first-search algorithm for and/or trees. Master's thesis, Department of Information Science, University of Tokyo, Japan, 1999.
11) A. J. Palay. The B* tree search algorithm – new results. *Artificial Intelligence*, 19:145–163, 1982.
12) Aske Plaat, Jonathan Schaeffer, Wim Pijls, and Arie de Bruin. Best-first fixed depth minimax algorithms. *Artificial Intelligence*, 87:255–293, 1996.
13) Shibahara, Inui, and Kotani. Effect of probcut in shogi – by changing parameters according to position category –. In *Game Programming Worshop 2002*, number 17 in IPSJ Symposium Series, pages 73–79, 2002. (in Japanese).
14) Testsurou Tanaka, Shunske Soeda, and Tomoyuki Kaneko. OpenShogiLib – construction of an efficient shogi library. In *Game Programming Workshop in Japan '96*, November 2003. (in Japanese).
15) Y. Tsuruoka, D. Yokoyama, and T. Chikayama. Game-tree search algorithm based on realization probability. *ICGA Journal*, 25(3):145–153, 2002.
16) Kazuki Yoshihara and Takashi Chikayama. Applying probcut to shogi. *Computer Software*, 19(3):73–79, 2002. (in Japanese).