情報量に基づく探索制御手法 – チェスにおける Singular Extension への応用 –

竹内聖悟[†] 金子知適[†] 山口和紀[†]

効率的な探索を行うために探索制御手法の改善を行った。評価値の差を利用した従来の手法では評価値自体は利用してこなかった。本稿では、評価値から得られる情報量を利用したマージンの制御手法を提案した。具体的には、評価値から勝率を推測し、その勝率から得られる情報量に応じてマージンを変更するという手法である。この手法により、効率的な探索制御が可能となる。提案手法をチェスにおいて Singular Extension や Futility Pruning などの探索制御手法に対して応用し、問題集や自己対戦の実験から提案手法の有効性を示した。

Method for Search Control Based on Information Content – Application for Singular Extension in Chess –

SHOGO TAKEUCHI,[†] TOMOYUKI KANEKO[†] and KAZUNORI YAMAGUCHI [†]

In order to perform efficient search, we have improved methods for search control. Existing methods that use difference of evaluation values but do not use the evaluation value itself. In this paper, we present a method for margin control based on information content. We estimate a win probability by an evaluation value, and estimate information content by the win probability, then change the margin according to the information content. We applied this method to Singular Extension and Futility Pruning in Chess. Improvement is confirmed by solving problems and by self-play.

1. はじめに

強いゲームプレイヤを作成するためには、探索と評価関数が重要である。効率的な探索を行うことができれば、より強いゲームプレイヤを作成することができる。効率的な探索を行うために、有効な手をより深く読み、有効でない手を浅く読むという探索の制御が研究されてきて成功を収めている。

これまでの探索制御手法には、評価値の差から有効さを判断して探索の制御を行うものが多い。例えば、探索延長の手法である Singular Extension[1] は、ある指し手の評価値が兄弟ノードの評価値にマージンを取った値よりも大きければ、その指し手を延長する。他にも、枝刈手法である Futility Pruning[6], ProbCut[3] などが評価値の差を比較する手法として挙げられる。

しかし、これらの手法では評価値そのものは考慮されておらず、評価値の差の比較による判断が適切でないケースが起こりうる。例えば、Singular Extension において、評価値に 100 以上の差が生じた場合に探索延長を行うとする。評価関数は最大最小で +/−1500 程度の値を取るものとして、ある局面に対して評価値 (0, 100) の候補手ペア、また別の局面に対して評価値 (1000, 1100) の候補手ペアがあるとする。この場合、どちらも singular だと判断し、探索延長が行われることになる。しかし、明らかに前者と後者とでは評価値差 100 の価値が異なり、同じ扱いとするのは適切ではない。

評価値自身の値によって、同じ評価値差でもその価値が異なることは、評価値と勝率の関係を図示した Evaluation curve[13][8] からも明らかである。Evaluation curve によれば、評価値 0 付近では評価値の増減に伴い勝率が大きく増減するが、評価値がある程度高くなるか低くなると勝率はほぼ横ばいとなっている。このように、勝率がほぼ同じ局面が他と比べ Singular であるとは考えにくく、評価値の差を基準として探索制御を行うのは問題があると考えられる。

この問題を解決するために、本研究では評価値から得られる情報量に基づいたマージン制御手法を提案する。具体的には、評価値と Evaluation curve から勝率を求め、勝率から得た情報量に応じてマージンを変化

† 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,yamaguch}@graco.c.u-tokyo.ac.jp

させるという手法である。

Singular Extension を例とすると、形勢が一方に偏ったような局面においては、評価値の増減に対して勝率の増減が少ないため、Singular と判定された手が有効であることは少ないと考えられる。情報量が少ない局面はマージンを増やすことで、形勢が偏った局面での延長が減り、有効な探索延長だけが行われるようになり、探索量の削減につながる。また、形勢が互角な局面では評価値の差による手法とほぼ同じように判定を行うことが可能である。

本研究では、Chess を題材として探索延長手法の Singular Extension, 枝刈手法の Futility Pruning へと提案手法を応用した。問題集および自己対戦の実験結果から本手法の有効性を示すことができた。

本論文の構成は以下の通りである。まず 2 章で関連研究について述べ、3 章で提案手法について詳しく説明し、4 章で手法の評価実験を行い、5 章で結論を述べる。

2. 関 連 研 究

ここでは探索制御手法である枝刈手法と探索延長の関連研究と、評価値と勝率の関係を表した Evaluation Curve について述べる。

これまでの探索制御手法の多くは、評価値の差から指し手を探索することの有効さを判断し、探索の制御を行っている。これらの手法に共通する特徴は、2 つかそれ以上の評価値の差を比較していることである。また、将棋で成功した実現確率探索[10] も探索制御手法の 1 つである。探索を深さで制御するのではなく、指し手の実現確率で制御する手法で、実現確率は指し手のカテゴリー毎に棋譜から求められる。

2.1 枝 刈 手 法

枝刈手法とは、指し手や局面の情報、浅い探索の結果などを利用し、指し手を読まずにすませて探索コストを減らす手法である。指し手がゲーム木における枝にあたるため、枝刈手法と呼ばれる。

2.1.1 Futility Pruning

Futility Pruning[6] はチェスで広く用いられ、将棋でも成功している[14] 手法である。

探索中、末端の 1 手前のノード (Frontier nodes) での評価値がマージンを取っても探索窓から外れているならば、探索した値も探索窓から外れているだろうとして、その枝を刈る手法である。マージンについては、1 手で更新可能な評価値の最大値を用いることで、Min-Max 法と同じ結果を得ることができる。しかし、、より効率的な枝刈のために最大値よりも小さな値を用いることがある。

また、Frontier node の 1 つ上のノード (Pre-frontier nodes) においても同様の枝刈を行う Extended futility pruning、さらに 1 つ上のノード (Pre-pre-frontier nodes) において枝刈ではなく探索深さを減少させる Limited razoring という手法もある。

2.1.2 ProbCut

ProbCut[3] はオセロで成功した手法で、浅い探索での評価値から深い探索の評価値を予測しようというものであり、浅い探索の評価値が現在の探索窓から十分外れている場合に枝刈を行う。

ProbCut を複数の深さのペアに対して行う Multi-ProbCut という手法もある[4]。

2.1.3 Multi-cut alpha-beta-pruning

Multi-cut alpha-beta-pruning[2] はチェスで提案された手法。n 手について浅い探索を行い、その内の k 手 (あらかじめ決めた閾値) についてカットが起きたなら、深い探索においてもカットが起きるだろうと考え、枝刈を行う手法である。この手法では、評価値にマージンを取るのではなく、カットの回数にマージンを取っている。

2.1.4 Null move pruning

Null move pruning[5] はチェスで成功し、広く用いられている手法である。探索を行う前に、パスした場合の浅い探索を行い、その結果が β カットを起こすなら、他の指し手でも β カットが起こるだろうと考え、枝刈を行う手法である。

2.1.5 RankCut

RankCut[7] はチェスにおいて提案された手法である。Move Ordering が十分よければ、後半の指し手は前半の指し手よりも悪いと期待できる。枝刈手法は最善手を更新する確率が低い手を刈っていると考え、ここでは、Move Ordering で後半に並べられた指し手は最善手を更新する確率が低くなっていると考え、指し手が最善手を更新する確率が十分低ければ枝刈するという手法である。指し手が最善手を更新する確率を事前に求めておき、各手を探索していき、更新する確率があらかじめ決めた閾値を下回った時に探索を終えるというものである。

2.2 探 索 延 長

探索延長の手法には、ゲームの知識に依存した探索延長が多く見られる。例えば、チェスや将棋において王手や駒を取る手などを延長することが行われている。

次から知識に依存しない延長手法を説明する。

2.2.1 0.5 手延長アルゴリズム

0.5 手延長アルゴリズム[11][12] は、反復深化法におい

て前回の最善手を 0.5 手延長して読むという手法である。

この手法を用いた将棋プログラム YSS は 2007 年の世界コンピュータ将棋選手権において優勝している。

2.2.2 Singular Extension

Singular Extension[1] はチェスで実験が行われ、有効性が示された。手法は、あるノードの評価値が兄弟ノードの評価値よりも十分に高ければ、そのノードが”Singular” なので探索の延長を行うというものである。つまり、1 ノードだけ評価値が離れているのは不自然であると考え、探索を延長するのである。

この手法は PV ノードと Fail-high ノードとで区別して行われる。PV ノードでは最善手の値が兄弟ノードの値よりも十分高いならば延長する。Fail-high ノードでは、探索深さを減らした探索を行い、現在の PV の値よりも兄弟ノードの値が十分に小さければ延長を行うというものである。

この手法は将棋へも応用され、成功を収めている[9]。

2.3 Evaluation Curve

Evaluation curve[13][8] は竹内らが提案した評価関数の評価手法である。

棋譜の各局面を評価関数により評価し、棋譜の勝敗とを比較することで、評価関数における評価値と勝率との関係を図示する。同じ評価値ならば同じ勝率のはずと考え、局面の勝率ではなく、評価値に対する勝率を求めた。

ある評価項目について特徴のある棋譜を用いた場合の Evaluation curve と通常の Evaluation curve とを比較することで、評価関数がその評価項目を適切に評価できているかを評価できることを示した。

本研究では、評価関数の評価は行わず、評価値から勝率を得るために Evaluation curve を用いた。

3. 情報量に基づく閾値の調整

本稿では、情報量に基づいて閾値の調整を行う手法を提案する。

探索制御手法では、探索コストと探索結果の正確性との間にトレードオフの関係がある。探索延長では、探索結果を正確にするため、探索コストをかけて延長を行い、枝刈では、カットにより探索結果が不正確になるが、その探索コストが減少する。

既存の手法は、評価値の差を利用しているが、評価値自体は考慮されていない。

評価値自体を利用する方法として、評価値から得られる情報量に基づいた探索制御を提案する。具体的には、評価値と Evaluation Curve から勝率を求め、その勝率から情報量を求め、マージンをその情報量に応じて変更するという手法である。探索量を情報量に応じて変更するというもので、評価値によらず一定のマージンを取る従来の手法よりも、トレードオフの関係を効率的に扱うことができる。また、ゲームの知識は Evaluation Curve だけであり、チェスや将棋、オセロ、囲碁ではほぼ同じ形を取ることがわかっているので、さまざまなゲームへの応用が可能である。

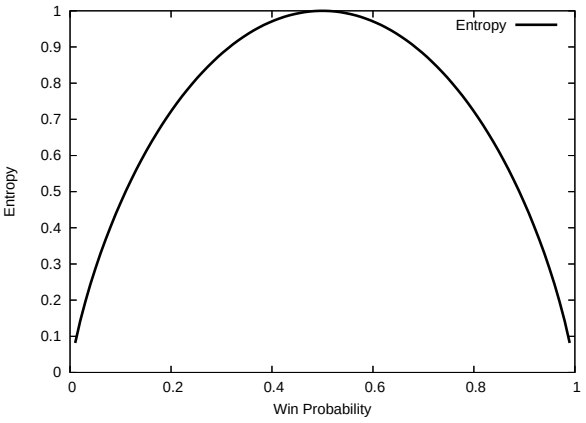

図 1 情報量

なお、情報量 (I) は、先手の勝率を p として、引き分けを考えないとすると $I = -p\log(p) - (1-p)\log(1-p)$ となる。$p = 0.5$ で最大値 $I = 1$ を取り、$p = 0, 1$ で最小値 $I = 0$ を取る。

本稿で実験の対象とする探索延長手法である Singular Extension、枝刈手法の Futility Pruning のマージンについて以下に述べる。

Singular Extension のような探索延長では、見込みのあるところを延長し、見込みのないところは延長しないようにしたいので、情報量の少ないところでは延長が起きにくいようにマージンを増やすようにすることが考えられる。勝率 0.5 の時に、元のマージンと同じになるように、マージンを $(2 - I) \times M$ をとした。M は元のマージンを表している。

Futility Pruning のように枝刈手法では、見込みのないところを枝刈したいので情報量の少ないところではカットが起きやすいようにマージンを少なくするということが考えられる。今回は、単純にマージン M を情報量 I 倍に、つまり $I \times M$ とした。こちらも勝率 0.5 の時には元のマージンと同じ値を取る。

情報量と、情報量を利用した Singular Extension, Futility Pruning でのマージンを図にしたものが図 1, 2, 3 である。X 軸は勝率を、Y 軸はマージンの係数を表している。

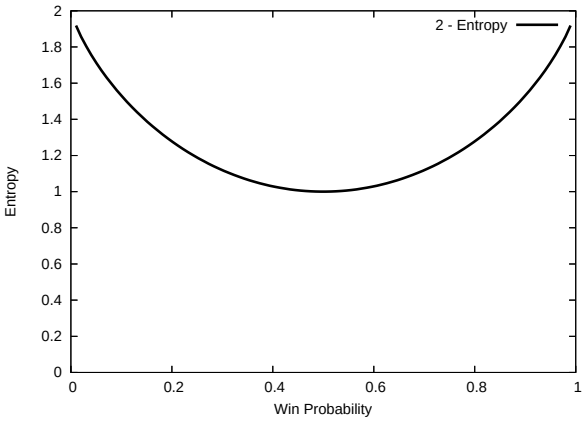

図 2 Singular Extension でのマージン

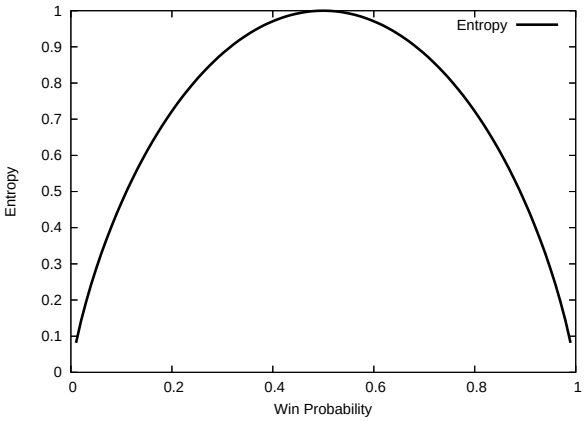

図 3 Futility Pruning でのマージン

4. 実　　験

提案手法の有効性を示すため、問題集と自己対戦の実験を行った。

まず、各プログラムに問題集を解かせて、その正解数と探索ノード数の増減から有効性を測った。従来手法より正答数の減少なしに、探索ノード数を減少していれば改善したと言える。

その後、各プログラム同士での自己対戦を行い、棋力の改善を確認した。

なお、探索制御手法の Singular Extension と Futility Pruning とを実験の対象とした。

実験には、オープンソースのチェスプログラム Crafty☆ (version. 20.14, 19.9) を利用した。Singular Extension(SE) を利用した実験では Singular Extension が実装されている version 19.9 を、Futility Pruning(FP) を利用した実験では最新版である version 20.14 を利用した。また、Singular Extension と Futility Pruning とを組み合わせて行った実験 (SE + FP) では、Futility Pruning の実装されている version 20.14 に Singular Extension を実装して実験を行った。

実験に利用したマシンの性能は CPU opteron 2.4GHz, メモリ 8GB である。

4.1 Crafty における実装

Crafty における Singular extension, Futility pruning, 提案手法の実装について述べる。

Singular extension は、現在の α 値を基準として、$\alpha - M$ について深さを減らした null-window 探索を行い、$\alpha - M$ を上回る手が 1 手の時を singular として探索延長しており、singular の判定が論文のものとは少し異なる。また、マージンは 75 点となっている。

一方、Futility pruning については、Extended futility pruning, Limited razoring を行っている。Crafty では Futility pruning, Extended futility pruning のマージンは Bishop の半分の点数 150 点となっており、Extended futility pruning のマージンとしてはだいぶ小さくなっている。また、Limited razoring のマージンは Queen の半分の点数 450 点で、やはりマージンが小さく取られている。

なお、Crafty の実装では Futility Pruning と Extended Futility Pruning とが一くくりに扱われているので、以降の実験において Futility Pruning を利用するときは Extended Futility Pruning も利用している。

提案手法を用いるにあたっては、あらかじめ求めた値を表として持つことでプログラムの実行速度の低下を防いだ。実際、Futility Pruning を用いたプログラムは WAC-08 を 29.81 秒で解くのに対し、提案手法のプログラムは 30.21 秒で解いており、大きな速度低下は見られなかった。

4.2 Evaluation Curve

提案手法で利用する勝率を求めるため、Evaluation curve を描いた。

Crafty の評価関数を用いて描いた Evaluation curve が図 4 である。棋譜には International Correspondence Chess Federation (ICCF☆☆) の棋譜 45,955 局を利用した。なお、評価の際には静止探索を行い、引き分けの棋譜を除いた。また、図の評価値 100 点は Pawn 1 枚の価値に相当する。

図からも分かるように評価値が 200 以上または、−200 以下では勝率がほぼ横ばいとなっており、従来

☆ ftp://ftp.cis.uab.edu/pub/hyatt/

☆☆ http:/iccf.com/

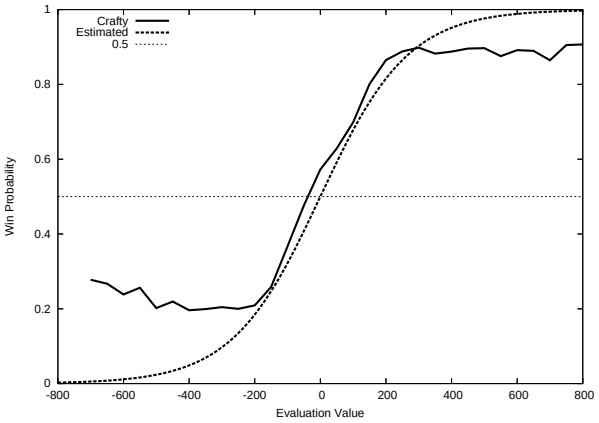

図 4 Crafty の Evaluation curve, X 軸が評価値, Y 軸が勝率を表す（それぞれ先手からみたもの）

手法のように評価値によらずにマージンを一定とすることに問題があることが確認できる。

実験にあたっては、近似値を利用する。近似式は評価値 V に対して $1/(1+exp(-a \times V))$ である。なお、$a = 0.007434035$ である。ただし、Evaluation Curve とは一致しておらず、より正確な近似式が必要と考えられる。

4.3 実験方法

提案手法の有効性を評価する実験として行った問題集と自己対戦について説明する。

4.3.1 問題集

探索ノード数と正解数の増減から本手法の有効性を示すため、各プログラムに問題集を解かせた。各プログラムに問題集を解かせて、その正解数と探索ノード数の増減から有効性を測った。提案手法のプログラムが従来手法のプログラムと比較して、正答数の減少なしに、探索ノード数の減少を実現していれば改善したと言える。

実験には、

- Encyclopedia of Chess Middlegames (ECM, 879 問)
- Win At Chess (WAC, 300 問)
- 1001 Winning Chess Sacrifices (WCS, 1001 問)

の 3 つの問題集を利用し、探索深さを 8, 10, 12 と変えて問題集を解かせた。なお、この実験手法は Extended Futility Pruning, RankCut での評価実験と同じ手法である。

w, w/o は with, without の略で、各探索制御手法を利用したプログラムかどうかを表している。また、Entropy は前章で説明した情報量を利用して探索制御を行う提案手法のプログラムを表している

表 1 問題集の解答結果: Singular Extension (w/o SE - SE)

Test Suite	w/o SE #Nodes	#Solved	w SE ΔNodes	ΔSolved
ECM-08	1,390,338,123	475 / 879	+11.60%	0.00%
ECM-10	11,340,788,040	529 / 879	+20.71%	+0.95%
ECM-12	37,575,128,546	560 / 879	+11.86%	-0.71%
WAC-08	175,954,826	288 / 300	+14.74%	0.00%
WAC-10	1,352,623,681	294 / 300	+23.42%	-0.34%
WAC-12	6,906,672,553	297 / 300	+22.90%	-1.01%
WCS-08	1,021,967,599	698 / 1001	+11.55%	+0.14%
WCS-10	8,663,084,038	712 / 1001	+18.80%	0.00%
WCS-12	31,959,667,685	719 / 1001	+14.71%	-0.56%
Sum-08	2,588,260,548	1461 / 218	+11.79%	+0.07%
Sum-10	21,356,495,759	1535 / 218	+20.11%	+0.26%
Sum-12	76,441,468,784	1576 / 218	+14.05%	-0.70%

表 2 問題集の解答結果: Singular Extension(w SE - Entropy)

Test Suite	w SE #Nodes	#Solved	Entropy ΔNodes	ΔSolved
ECM-08	1,551,587,534	475 / 879	-6.38%	0.00%
ECM-10	13,689,686,672	534 / 879	-12.57%	-1.12%
ECM-12	42,032,576,831	556 / 879	-5.58%	+0.36%
WAC-08	201,896,436	288 / 300	-5.77%	0.00%
WAC-10	1,669,473,552	293 / 300	-11.32%	+0.34%
WAC-12	8,488,615,843	294 / 300	-12.14%	+0.68%
WCS-08	1,139,972,611	699 / 1001	-6.78%	-0.29%
WCS-10	10,291,408,740	712 / 1001	-11.16%	-0.14%
WCS-12	36,661,180,592	715 / 1001	-7.36%	+0.42%
Sum-08	2,893,456,581	1462 / 2180	-6.50%	-0.14%
Sum-10	25,650,568,964	1539 / 2180	-11.92%	-0.39%
Sum-12	87,182,373,266	1565 / 2180	-6.97%	+0.45%

4.3.2 自己対戦

棋力の改善を確認するため、各プログラム同士で自己対戦を行った。

各プレイヤの持ち時間は 10 分とし、先後を入れ替え 100 ゲーム行った。なお、先読み機能はオフとし、オープニングブックは使っていない。

4.4 Singular Extension

Singular Extension に対して実験を行った。

4.4.1 問題集

Singular Extension を利用しないプログラム (w/o SE) と、利用したプログラム (w SE)、提案手法を利用したプログラム (Entropy) に問題集を解かせた。

結果を表 1, 2 にまとめた。Singular Extension を利用することで探索ノード数が 10% から 20%増加するが、正答数はほぼ変化しておらず、Singular Extension の有効性は示せなかった。一方、提案手法では従来の Singular Extension よりも探索ノード数を約 10% 減少させ、正答数にもほぼ変化がないなど、従来の手法よりもよい結果が得られ、有効性を示すことができた。

表 3　対戦結果: Singular Extension

	w/o SE	w SE
w SE	24 - 17 - 59	-
Entropy	36 - 7 - 57	29 - 9 - 62

表 6　対戦結果: Futility Pruning

	w FP
Entropy	19 - 26 - 55

表 4　問題集の解答結果: Futility Pruning (w/o FP - FP)

Test Suite	w/o FP #Nodes	#Solved	w FP ΔNodes	ΔSolved
ECM-08	632,190,700	501 / 879	-29.78%	-1.42%
ECM-10	4,500,660,999	596 / 879	-29.04%	-0.51%
ECM-12	22,895,014,278	651 / 879	-21.97%	+0.61%
WAC-08	109,534,425	280 / 300	-26.11%	-0.36%
WAC-10	835,304,414	292 / 300	-39.16%	+0.34%
WAC-12	3,786,819,695	293 / 300	-17.85%	0.00%
WCS-08	558,172,051	840 / 1001	-24.03%	-0.24%
WCS-10	4,459,338,290	854 / 1001	-28.37%	+0.12%
WCS-12	18,900,396,508	862 / 1001	-16.90%	+0.46%
Sum-08	1,299,897,176	1621 / 2180	-26.94%	-0.62%
Sum-10	9,795,303,703	1742 / 2180	-29.53%	-0.06%
Sum-12	45,582,230,481	1806 / 2180	-19.48%	+0.44%

表 7　問題集の解答結果: SE + FP (w/o SE + FP - w SE + FP)

Test Suite	w/o SE + FP #Nodes	#Solved	w SE + FP ΔNodes	ΔSolved
ECM-08	632,190,700	501 / 879	-3.86%	-0.80%
ECM-10	4,500,660,999	596 / 879	+5.58%	-0.50%
ECM-12	22,895,014,278	651 / 879	+13.90%	-0.31%
WAC-08	109,534,425	280 / 300	-0.72%	+0.71%
WAC-10	835,304,414	292 / 300	-2.67%	+0.34%
WAC-12	3,786,819,695	293 / 300	+29.87%	-0.34%
WCS-08	558,172,051	840 / 1001	+9.32%	0.00%
WCS-10	4,459,338,290	854 / 1001	+2.33%	+0.70%
WCS-12	18,900,396,508	862 / 1001	+14.98%	+0.81%
Sum-08	1,299,897,176	1621 / 2180	+2.06%	-0.12%
Sum-10	9,795,303,703	1742 / 2180	+3.40%	+0.23%
Sum-12	45,582,230,481	1806 / 2180	+15.68%	+0.22%

表 5　問題集の解答結果: Futility Pruning (FP - Entropy)

Test Suite	w FP #Nodes	#Solved	Entropy ΔNodes	ΔSolved
ECM-08	487,116,160	494 / 879	-0.53%	+0.61%
ECM-10	3,487,887,652	593 / 879	-1.64%	-1.01%
ECM-12	18,770,555,896	655 / 879	-9.73%	-1.07%
WAC-08	86,857,841	279 / 300	-0.83%	0.00%
WAC-10	600,246,973	293 / 300	-1.27%	0.00%
WAC-12	3,213,249,623	293 / 300	-10.86%	0.00%
WCS-08	450,039,990	838 / 1001	-0.72%	-0.12%
WCS-10	3,473,919,274	855 / 1001	-2.48%	+0.23%
WCS-12	16,167,663,022	866 / 1001	-12.87%	-0.35%
Sum-08	1,024,013,991	1611 / 2180	-0.64%	+0.12%
Sum-10	7,562,053,899	1741 / 2180	-1.99%	-0.23%
Sum-12	38,151,468,541	1814 / 2180	-11.15%	-0.55%

4.4.2 自 己 対 戦

Singular Extension を利用したプログラム (w SE) と利用しないプログラム (w/o SE)、提案手法を利用したプログラム (Entropy) との間で自己対戦を行った。

結果を表 3 にまとめた。Entropy モデルは Singular Extension を利用したプログラムにも利用しないプログラムにも有意に勝ち越しており、本手法が有効であることが示せた。

4.5 Futility Pruning

Futility Pruning に対して実験を行った。

4.5.1 問　題　集

Futility Pruning を利用しないプログラム (w/o FP) と、利用したプログラム (w FP)、提案手法を利用したプログラム (Entropy) に問題集を解かせた。

結果を表 4, 5 にまとめた。Futility Pruning によって正答数の大きな増減なしに 20% 以上の探索ノード数が減少している。また、探索深さ 12 の場合、提案手法によって従来の Futility Pruning よりも正答数の大きな減少なしに、約 10% の探索ノード数の削減が達成されており、提案手法の有効性を示すことができた。

4.5.2 自 己 対 戦

結果は表 6 である。有意に勝ち越すことはできなかったが、問題集の結果から考えると、探索深さが深くなるほど本手法が有効であるので、自己対戦の持ち時間を長くすることで結果が変わる可能性がある。

4.6 SE + FP

最後に、これまで実験してきた Singular Extension と Futility Pruning を組み合わせて実験を行った。

4.6.1 問　題　集

Crafty version 20.14 の Futility Pruning をオフにしたプログラム (w/o SE + FP) と、Singular Extension と Futility Pruning を組み合わせたプログラム (w SE + FP)、提案手法を利用したプログラム (Entropy) に問題集を解かせた。

結果を表 7, 8 にまとめた。Singular Extension と Futility Pruning とを組み合わせることで探索ノード数が探索深さ 12 において 15% 増加した。提案手法では、探索深さ 12 において正答数の大きな増減なしに探索ノード数が 15% 減少しており、提案手法の有効性を示すことができた。しかし、これは Singular Extension と Futility Pruning とを組み合わせる前の探索ノード数とほぼ同じである。

4.6.2 自 己 対 戦

本手法を利用したプログラムと、FP だけ利用したプログラム、SE と FP とを利用したプログラムの自己

表 8 問題集の解答結果: SE + FP (w SE + FP - Entropy)

Test Suite	w SE + FP #Nodes	#Solved	Entropy ΔNodes	ΔSolved
ECM-08	607,806,892	497 / 879	-2.69%	+1.41%
ECM-10	4,751,758,956	593 / 879	-3.63%	-0.84%
ECM-12	26,078,173,367	649 / 879	-16.24%	-0.92%
WAC-08	108,743,461	282 / 300	-1.08%	0.00%
WAC-10	813,012,471	293 / 300	-1.65%	-0.68%
WAC-12	4,917,978,022	292 / 300	-17.02%	-0.34%
WCS-08	610,173,898	840 / 1001	-5.44%	+0.36%
WCS-10	4,563,162,991	860 / 1001	-8.64%	-0.12%
WCS-12	21,731,433,547	869 / 1001	-14.78%	-0.46%
Sum-08	1,326,724,251	1619 / 2180	-3.83%	+0.62%
Sum-10	10,127,934,418	1746 / 2180	-5.73%	-0.46%
Sum-12	52,727,584,936	1810 / 2180	-15.71%	-0.61%

表 9 対戦結果: SE + FP

	w/o SE w FP	w SE+ FP
Entropy	26 - 35 - 39	22 - 26 - 52

対戦を行った。

結果を表 9 にまとめた。提案手法を用いたプログラムでは、有意に勝ち越すことはできなかった。問題集の実験を考えると、探索ノードの削減が多いのは探索深さ 12 の時であり、Futility Pruning の実験同様、持ち時間の多い自己対戦では結果が変わる可能性がある。

5. おわりに

これまでの評価値の比較を行う探索制御手法では、評価値そのものの情報は利用されてこなかった。

本稿では評価値から得られる情報量に基づく探索制御手法を提案した。具体的には、Evaluation Curve と評価値から勝率を求め、その勝率から求まる情報量に応じて探索制御のマージンを変化させる手法である。この手法をチェスにおいて、Singular Extension, Futility Pruning へと応用し、問題集と自己対戦による評価実験の結果から、提案手法の有効性を示すことができた。

提案した手法は、ゲームの知識としては Evaluation Curve を利用しているが、Evaluation Curve はチェスや将棋、オセロ、囲碁においてもほぼ同じような形をとることがわかっており[8]、多くのゲームでの応用が可能である。

今後の課題として、マージンを利用しない探索制御手法における評価値の情報を利用した探索制御の改善、引き分けを考慮した情報量の利用、Futility Pruning での探索深さを増やした実験、Evaluation Curve のより正確な近似式 (図 5 参照) を用いての実験などが挙げられる。

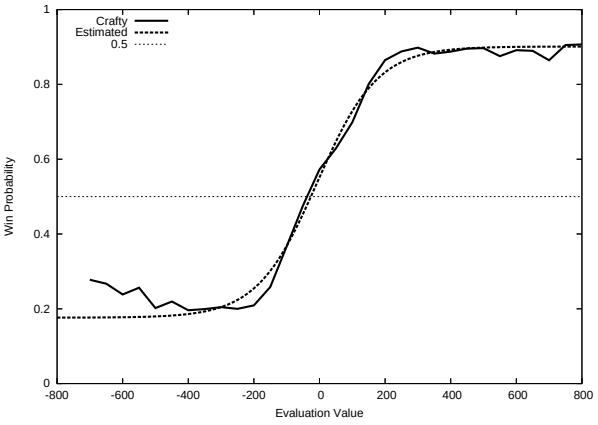

図 5 Crafty の Evaluation curve とその近似, X 軸が評価値, Y 軸が勝率を表す（それぞれ先手からみたもの）

参 考 文 献

1) T. Anantharaman, M. S. Campbell, and F. hsiung Hsu. Singular extensions : Adding selectivity to brute-force searching. *Artificial Intelligence*, 43(1):99–109, 1990.
2) Y. Björnsson and T. A. Marsland. Multi-cut alpha-beta-pruning in game-tree search. *Theoretical Computer Science*, 252(1-2):177–196, 2001.
3) M. Buro. Probcut : An effective selective extension of the α - β algorithm. *ICCA Journal*, 18(2):71–76, 1995.
4) M. Buro. Experiments with multi-probcut and a new high-quality evaluation function for othello. In H. I. H. J. van den Herik ed., *Games in AI Research*, pp. 77–96. Van Spijk, 2000.
5) C. Donninger. Null move and deep search. *ICCA Journal*, 16(3):137–143, 1993.
6) E. A. Heinz. Extended futility pruning. *ICCA Journal*, 21(2):75–83, 1999.
7) Y. J. Lim and W. S. Lee. Rankcut – a domain independent forward pruning method for games. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-2006)*, 2006.
8) S. Takeuchi, T. Kaneko, K. Yamaguchi, and S.Kawai. Visualization and adjustment of evaluation functions based on evaluation values and win probability. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI-2007)*, pp. 858–863, 2007.
9) 中山, 小谷. singular extension の着手の性質. 第 4 回ゲームプログラミング ワークショップ, pp. 38–53, 1997.
10) 鶴岡, 横山, 丸山, 近山. 局面の実現確率に基づくゲーム木探索アルゴリズム. 第 7 回ゲームプログラミング ワークショップ, pp. –, 2001.

11) 山下. 0.5 手延長アルゴリズム. 第 4 回ゲームプログラミング ワークショップ, pp. 46–54, 1997.
12) 山下. Yss-「コンピュータ将棋の進歩 2」以降の改良. 松原（編）, コンピュータ将棋の進歩 5, 第 1 章, pp. 1–32. 共立出版, 2005.
13) 竹内, 林, 金子, 川合. 勝率と評価値の歪みに基づく評価関数調整法 – 将棋における進行度差の評価 –. 第 11 回ゲームプログラミング ワークショップ, pp. 56–63, Nov. 2006.
14) 保木. コンピュータ将棋における全幅探索と futility pruning の応用. 情報処理, 47(8):884–889, 2006. ミニ小特集 コンピュータ将棋の新しい動き.