# 将棋における Dual Credit 探索

鈴木 豪　　小谷 善行
東京農工大学
go@fairy.ei.tuat.ac.jp　kotani@cc.tuat.ac.jp

概　　要

チェスや将棋において、強制手とよばれる相手に特定の指し手を強要する指し手がある。これらをすべて探索することは理想であるが、実際にこれを行うと探索爆発が起こり、現実的には計算が行き詰まる。Dual Credit アルゴリズムはこの問題を解決しようとする方法の一つである。Dual Credit 探索は、先手と後手のそれぞれの指し手にクレジットを与え、どちらか片方のクレジットが貯まった時点で深さに変換して、探索延長を行う。本稿では Dual Credit 探索を将棋に適用した実験について述べる。実験の結果、Dual Credit 探索による探索ノード数の増加は多い局面でも 20%以下であり、重要な指し手をうまく延長していると考えられる。

# Dual Credit search in Shogi

Tsuyoshi SUZUKI　　Yoshiyuki KOTANI
Tokyo University of Agriculture and Technology
go@fairy.ei.tuat.ac.jp　kotani@cc.tuat.ac.jp

Abstract

There are moves that are called "forced move" in Chess and Shogi. That is the move that force a specific move. Ideally one would extend the search one ply for each forced moves. This almost always led to a search explosion. Dual Credit algorithm is one of the methods which try to solve this problem. Credit is given to each move of the first player and the second player in the Dual Credit search. When the credit of either one increases, it is changed into the depth, and it does search extension. We describe the experiments that used the Dual Credit algorithm in Shogi. As a result, increase in the number of searched nodes by the Dual Credit search is less than 20% as to many positions, and it can guess it important move well extend.

## １ はじめに

チェスや将棋において、強制手とよばれる相手に特定の指し手を強要する指し手がある。例えば、王手、詰めろ、飛車取りなどである。これらは手を抜くことはできず必ずそれに対応する指し手を指さなければならず、強制手と呼ばれる。これは様々な状況で発生する。例えば、後手王には詰めろがかかっている。後手は様々な捨て駒の王手などで、自王の詰みを遅らせようとするが、結局は先手の勝ちに終わるときなどである。また、次のような場合もある。先手は後手からの脅威にさらされている（例えば飛車取りなど)。先手はその脅威から逃れようとするが、逃れる手段がないときはその脅威を避けるためにその脅威以上の価値に見える捨て駒を続け、結局のところ大きな損をするなどである。これらをすべて探索することは理想であるが、実際にこれを行うと探索爆発が起こり、現実的には計算が行き詰まる。Dual Credit アルゴリズムはこの問題を解決しようとする方法の一つであり、Deep Blue のソフトウェア探索で使用された[1]。Dual Credit 探索は、先手と後手のそれぞれの指し手にクレジットを与え、どちらか片方のクレジットが貯まった時点で深さに変換して、探索延長を行うというものである。本稿では Deep Blue のソフトウェア探索において使われた Dual Credit 探索を将棋に適用した実験について述べる。

## ２ Dual Credit アルゴリズム

Dual Credit 探索は次の特徴をもつ[1]。

- 強制手の列があるとき、それらをその都度延長していては探索量が増えすぎる。これに対応するためクレジットを導入する。延長したほうがよいと思われる指し手に対してはクレジットを蓄えておき、クレジットがある閾値を超えたところで実際の探索延長を行う。この閾値を設定することにより、探索延長は強制手の列が起こるまで遅らせることができる。
- 先手･後手の両方のクレジットを使って探索をすると、延長されすぎて探索量が多くなりすぎる。これを避けるためクレジットを先手･後手の二つに分けて加算していく。このとき片方の側に十分なクレジットがたまった場合には、探索延長を行うが、このとき他方のクレジットは捨てる。例えば、ある取り合いにおいて先手にとっては良い手が続くかもしれないが、後手にとってはそうでないとき、後手は深く探索することは無駄である。
- 評価値を更新するノードは、先手･後手の両方で、少なくともその時点における、最善のプレーを表しているため、これらはクレジットを貯める。

Dual Credit 探索を C ライクなコードで図１に示す。基本的にはαβ法に基づいている。1 行目ではクレジットを表す二つのパラメータ myCredit と hisCredit が追加されている。10～15 行目においてクレジットの更新を行っている。10 行目の CREDIT_LIMIT はクレジットを実際の探索の延長数に変えるための閾値である。11 行目は探索の深さを計算している。これは hisCredit から CREDIT_LIMIT を引いた値である。例えば、CREDIT_LIMIT=2、hisCredit=2.5 のときは１手探索延長となり、hisCredit=3.4 のときは２手延長となる。12,13 行目で対応するクレジットを引いている。23 行目には探索している指し手がそれまでの評価値を更新したときに実行される。ここの GenerateCredit()ではクレジットの計算

を行う。遷移確率による探索延長アルゴリズム[2]と比較した場合、遷移確率による方法では再探索される値は0.5と固定されているが、Dual CreditはGenerateCredit()関数の導入によりより柔軟である。もし、探索する指し手においてクレジットが加算されたなら、探索延長が発生するかをチェックし、その場合には再探索を行う。

```
 1  int DC(position p, int α, int β, int depth, float myCredit, float his Credit)
 2  {
 3   int count;
 4   int score;
 5   int i;
 6   int sc;
 7   float newCredit;
 8   int extension;
 9
10   if( hisCredit >= CREDIT_LIMIT){
11     extension = ceiling(hisCredit - CREDIT_LIMIT);
12     hisCredit = hisCredit - extension;
13     myCredit = max(myCredit - extension, 0);
14     depth = depth + extension;
15   }
16
17   if( depth == 0 ){ return Evaluate(p); }
18   score = α;
19   count = GenerateSuccessors(p);
20   for( i = 1 ; i <= count ; i++ ){
21     sc = -DC(p.i, -β, -α, depth - 1, hisCredit, myCredit);
22     if( sc > score ){
23        newCredit = GenerateCredit();
24        if( newCredit > 0 ){
25         sc = -DC(p.i, -β, -α, depth - 1, hisCredit, myCredit + newCredit);
26        }
27        if( sc > score ){ score = sc; }
28      }
29      if(score >= β ){ return score; }
30    }
31    return score;
32  }
```

図１ Dual Credit 探索

## ３ クレジットの計算

クレジットは探索延長をすべき指し手に対して加算される。Deep Blueでは、非凡な指し手、合法手がただ一つしかない場合の指し手、脅威を避ける指し手、前の指し手により可能となった指し手、局面依存の指し手などにクレジットが加算されている[1]。将棋においてもこれらと同様の指し手に対してクレジットを導入することができる。本稿では将棋の指し手の分類[3]を参考に表１のような指し手に対してクレジットを与える。この分類は大きくは駒の種類と、指し手の意味に分けられ、その組み合わせによりそれぞれクレジットが与えられる。ただし、組み合わせが存在しないものもある。これらの指し手に対するクレジットの値は手で調整したものを使っている。

表１　指し手の分類

| 駒の種類による分類 | 指し手の意味による分類 |
|---|---|
| 王 | 直前に動いた駒を取る手 |
| 飛 | 直前に動いた駒以外を取る手 |
| 角 | 取られそうな駒が逃げる手 |
| 金 | 相手の利きのあるところへの移動 |
| 銀 | 相手の利きのないところへの移動 |
| 桂 | 王手 |
| 香 | 王手の逃れ手 |
| 歩 | 価値の高い駒を取る手 |
| 龍 | 価値の低い駒を取る手 |
| 馬 | 駒が成る手 |
| 成銀 | 相手の利きがあるところへ駒を打つ手 |
| 成桂 | 相手の利きがないところへ駒を打つ手 |
| 成香 | 仮評価が高い手 |
| と金 | それ以外の手 |

## ４ 実験

将棋プログラムを使い Dual Credit 探索に関する実験を行った。実験は Dual Credit 探索と通常のαβ探索による対戦による比較、クレジットを計算する指し手の比較、クレジットを分離する場合としない場合の比較を行っている。探索の基本となる深さは４または６とし、プログラムは取り合い延長、王手延長、詰めの探索などを行っている。一手あたりの探索の制限時間は 10 秒とし､反復深化とトランスポジションテーブルを使用している。実験に使用したマシンは Pentium4-1.9GHz である。

### 4.1 Dual Credit 探索とαβ探索の比較

Dual Credit を使った探索と、Dual Credit を使わない通常のαβ探索の対戦による比較を行った。Dual Credit 探索には通常のαβ探索[4]よりも延長処理が入っていることになり、探索に時間を要している。対戦結果を表２に示す。Dual Credit 探索を使ったものが 62 勝 38 敗となった。対戦数が少ないためこれだけから強さの判断はできないが幾分優位ではないかと推測できる。

実験に現れた一局を通して Dual Credit により増えた探索ノード数を図２に示す。探索ノード数が 0～5%増えた局面は全体の 64%、6～10%増えた局面は全体の 32%、11～15%、16～20%増えた局面はそれぞれ全体の 11%あった。Dual Credit により増えた探索ノード数は多くはないことが分かる。

表２　Dual Credit あり対 Dual Credit なし

| 勝ち | 負け | 勝率 |
|---|---|---|
| 62 | 38 | 0.62 |

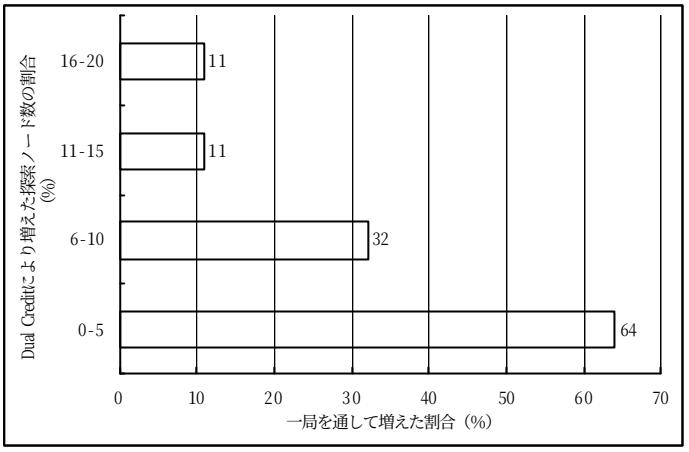


図２ Dual Credit により増えた探索ノード数

### 4.2 クレジットを計算する指し手による比較

図１の Dual Credit 探索では評価値を更新したノードに対してのみクレジットを計算して探索延長を行っていた。ここでは、全てのノードに対してクレジットを計算して探索延長を行うプログラムとの比較を行った。これは指し手の種類による探索延長[5]を、手番毎に分けて延長するかどうかを決めるようにしたものである。その対戦結果を表３に示す。Dual Credit 探索を使ったものが 47 勝 53 敗となった。対戦数からこの結果だけからどちらが強いとは言えないが、全ての指し手を対象にクレジットを導入したものと、評価値を更新したノードのみに Dual Credit を導入したものでは同程度の強さが得られている。

表３　評価値を更新したノードのみ対全てのノード

| 勝ち | 負け | 勝率 |
|---|---|---|
| 47 | 53 | 0.47 |

### 4.3 クレジットを分離しない場合の比較

図２において、クレジットを手番毎に分離しない場合は、指し手の種類の延長とほぼ同様なものとなる。ここでは探索時間が同程度になるように調整した手番を区別せずにクレジットを加算していくプログラムとの対戦による比較を行った。ただし、手番を区別しない場合にはクレジットが貯まりすぎるため、探索延長に変換する閾値は手番を分けた場合の２倍よりも若干大きな値にしてある。この場合の対戦結果を表４に示す。結果は手番を分けた場合の 66 勝 34 敗となった。また、手番を分けない場合の閾値を、手番を分けた場合の閾値の２倍にした場合には、探索延長が発生しすぎて探索の時間制限に引っかかり、表４の結果よりも負け越している。

表４　Dual Credit あり対クレジットのみ

| 勝ち | 負け | 勝率 |
|---|---|---|
| 66 | 34 | 0.66 |

## 5 考察

本実験で使用した Dual Credit 探索を使ったプログラムは、それを実装しない通常のαβ探索のプログラムと比較して探索ノード数が多い局面でも２割程度増加であり、多くの局面では 5%以下となった。強さに関しては探索延長の分だけ探索頂点数が増えたが勝率にして 62%という結果となった。これから、探索ノード数の増加は多くはないが強さに関しては幾分優位ではないかと推測できる。また、Dual Credit による探索延長を評価値を更新したノードに限らず、それ以外の指し手について適用した場合、対戦による比較では 47 勝 53 敗とほぼ同程度の結果が得られた。このことから評価値を更新したノードを中心に探索延長を行えばよいと考えられる。手番毎にクレジットを分離しない場合と、手番毎に分離した場合の対戦では、手番を分けたほうの 66 勝 34 敗となった。これは手番を分けたほうが広く浅く探索しているのに対し、手番を分けたほうが部分的に深く探索しているためと考えられる。また、別の実験においては取り合い延長などの処理をより軽くした場合には Dual Credit 探索を使ったものの勝率が高くなる結果も得られた。末端に近い部分では Dual Credit 探索を使うと有効ではないかと考えられる。

## 6 おわりに

将棋プログラムに Dual Credit 探索を実装し、実験を行った。手番に無関係にクレジットを加算していく延長よりも、手番を考慮した延長を行うほうが全体的に広く浅い延長にならず、重要な指し手を延長することができる。また、評価値を更新するノードを探索延長の対象としても、それ以外のノードも均等に延長しているものと同程度の勝率を上げることができた。Dual Credit 探索は末端に近い部分などにおいて有効でないかと推測する。コンピュータ将棋のおいては遷移確率を導入した探索に期待がもたれているが、Dual Credit 探索で使われた手番ごとのクレジットの加算、再探索時のクレジットの設定などを導入することが考えられる。これらは今後の課題である。

## 参考文献


[1] Campbell, M. Hoane Jr., A.J., Hsu, F.-h. (2002). Deep Blue. Artificial Intelligence, Vol. 134, pp.57-83.

[2] 鶴岡慶雄,横山大作,丸山孝志,近山隆. (2001). 局面の実現確立の基づくゲーム木探索アルゴリズム, Game Programming Workshop 2001, pp.17-24.

[3] 小谷善行,飯田弘之 (1995). 何を刈るべきか－指し手の分類と指した手の割合, Game Programming Workshop 1995, pp.148-156.

[4] Knuth, D.E., Moore, R.W. (1975). An analysis of alpha-beta pruning, Artificial Intelligence 6(4). pp.210-229.

[5] 岩井麻子,鈴木豪,小谷善行,堤正義 (1999).将棋における指し手の種類別探索深さの調査. 情報処理学会ゲーム情報学研究会報告 99-GI-1, pp.85-89.

[6] Junghanns, A. (1998). Are there practical alternatives to alpha-beta in computer chess?, ICCA Journal, 12(1), pp.14-32.