2003－GI－10　(1)
2003／8／4

MTD(f)の改良と評価

柴原一友　乾伸雄　小谷善行
東京農工大学工学研究科情報コミュニケーション工学専攻

概要

記憶容量を多大に必要とし、実際に実行することが難しいとされた最良優先探索を、深さ優先探索に置き換えて実行可能にした MTD という手法がある。MTD は null-window 探索を複数回使用することによって探索を行う手法であり、状況によってはαβ探索木よりも小さな探索木でゲーム木探索を実行する事が可能である。MTD は、初期値設定の違いによっていくつかの手法に分かれており、あらかじめ近似値 f を用意して探索を行う MTD-f が有望な方法である事が示されている。本研究では、ランダム木の探索を行う事で、MTD-f の性質について調査した。また、MTD-f は近似値 f の設定がミニマックス値から離れている場合、αβ探索よりも悪くなるという欠点がある。これを解決するため、MTD-f に対し、MTD-step を組み合わせた MTD-f-step と、ウィンドウ探索を組み合わせた MTD-f-αβを提案する。これらの手法を本研究室で開発した将棋プログラムに対して適用し、その性能を調査した。その結果、通常の MTD-f に比べ、約 3%速くなることが示された。

Improvement and Estimate of MTD-f

Kazutomo Shibahara　Nobuo Inui　Yoshiyuki Kotani
(Tokyo University of Agriculture and Technology)

Abstract

The best-first search is an ideal algorithm but is practically difficult to use because of the problem of strage. MTD is an advanced best-first search which explores a game-tree in a depth-first manner. MTD uses a NULL-window search at a root node in several times and is able to find a solution with less number of search node than an alpha-beta search. Previous researches reported that MTD-f, which uses an approximate minimax value at first for the NULL-window search, showed the best results. We show the nature of MTD-f for random-game trees in this paper. From this observation, we propose several methods for the improvement and analyze these performances. These method are concerned how to determine initial values of NULL-windows. As a consequence, MTD-f with MTD-step, called MTD-f-step, and MTD-f-alpha-beta can find solutions 3 percent faster than MTD-f in our experiments.

1. はじめに

ゲーム木探索には、αβ探索などの深さ優先探索手法の他に、αβ探索よりもさらに小さな探索木で探索が可能となる、SSS*などの最良優先探索がある。SSS*は探索が効率的に行えるが、記憶容量を多大に必要とすることや手の並び替えに時間を要するなどの理由で、実際に実行することは難しいとされていた。

その後、SSS*のような最良優先探索を深さ優先探索に置き換えて実行する MTD という手法が提案された[1]。この手法によって、最良優先探索における問題点であった記憶容量や手の並び替

えの問題を解消することが可能となった。[1]において、MTD はチェッカー、オセロ、チェスの三種類のゲームで適用され、提案された手法の一つ MTD-f が高い探索性能を有していることが示された。

本稿では、この MTD をランダム木と将棋において適用し、その探索ノード数や探索時間の違いをαβと比較することで、その性能を示す。また、近似値の設定次第で性能が悪化する MTD-f に対して、MTD-step を組み合わせた MTD-f-step と、ウィンドウ探索を組み合わせた MTD-f-αβについて提案し、その効果を検証した。

2. MTD 法における Null-Window の初期値設定

MTD は Pearl が論文中で示した Test を拡張したものである。Test とは、探索木の部分木におけるミニマックス値が、決められたしきい値よりも大きくなるか、そうではないかの二値を返すアルゴリズムであり、null-window 探索と同等のアルゴリズムである。ゲーム木探索で求めなければならないのはミニマックス値であり、二値の問題ではない。そこで、MTD はミニマックス値の上限と下限を再帰的に狭めながら求め続けることで、ミニマックス値に近づけていくという方法を用いている。その場合、同じノードを何度も探索することから、MTD はトランスポジションテーブルを使用して、無駄な再探索を実行することを防いでいる。MTD の中で再帰的に呼び出されるコード部分を MT と呼ぶ。

MTD には、二つの設定項目がある。一つ目は、探索を実行するときのしきい値の初期値であり、二つ目は、一回 MT の呼び出しを行った後で、次のしきい値を設定する方法である。この設定を変更することで、MTD はいくつかの種類を設定することができる。次に、[1]で示された種類を示す。

2.1. MTD+∞

一つ目の設定項目であるしきい値の初期値を＋∞に設定し、次のしきい値に前回の探索の結果得られた値を使用する手法を MTD+∞と呼ぶ。この探索手法は、SSS*と同様の探索手法にあたる。

2.2. MTD-∞

MTD+∞との相違点は、開始時のしきい値を-∞にすることである。一見すると MTD-∞は MTD+∞と差がないように思われる。しかし、実際にはある深さにおける探索量に変化が現れる。MTD+∞は上限の値を徐々にミニマックス値へと近づける手法である。その場合、Max ノードは全展開し、Min ノードは一つだけ展開するようになる。逆に、MTD-∞は下限をミニマックス値に近づける手法であるので、Max ノードは一つだけ展開し、Min ノードは全展開する。したがって、読む深さが奇数となる場合、MTD-∞は MTD+∞に比べて探索量が少なくなる。

2.3. MTD-f

MTD-f は、開始時のしきい値に無限大の値ではなく、ミニマックス値の近似値を使用する。ミニマックス値が近似値を上回ると判断されれば、しきい値は下限に変更されるので、MTD-∞と同じ探索を行い、下回ると判断された場合は、しきい値は上限に変更されるので、MTD+∞と同じ探索を行うようにする。

2.4. その他の MTD

[1]で示された MTD の種類には、他に MTD-bi, MTD-step, MTD-best がある。MTD-bi は上限と下限の値の中間値を次のしきい値に設定する方法である。また、MTD-step は開始時のしきい値を+∞とし、次のしきい値に関しては、得られた値よりもいくらかの幅(step)だけ、マイナス方向へ大きく飛ぶようにする。つまり、一回の MT 呼び出しごとに小さなジャンプを繰り返すのではなく、一回で大きなジャンプを行おうというものである。

上記二つの手法は、その性質上トランスポジションテーブルに、上限と下限の両方を保存することで最大の効果を発揮する手法である。片方だけ保存する方法でも、探索を実行することは可能であるが、その性能は落ちてしまう。これに対し、MTD-∞や MTD+∞、MTD-f は一つだけで十分である。[1]では、トランスポジションテーブルの情報を変更するのが難しいという理由から、MTD-step と MTD-bi に関しては、その効果の検証を行っていない。

今まで説明した MTD は、最善手のミニマックス値を求めることができるのに対し、MTD-best は最善手のミニマックス値を求めず、最善手がどの手であるかだけを調べる探索手法である。よって、さらに小さな探索木で探索が実行できる可能性がある。具体的には、最善手だと思われる手の下限値だけを調べ、その値がその他の兄弟ノードにおける上限値を上回る場合には、最善手だと睨んだ手が最善手であると証明されることになる。この方法では、最善手であると睨んだ手が最善手でない場合、無駄な探索が増加することになる。

3. MTD-f 法のランダムゲーム木における性質

MTD-f の改良を考える前に、まず MTD-f の性質について調査することにした。具体的には、ランダム木において MTD を適用し、その性能を調べる方法で行う。

3.1. ランダム木探索実験の方法

実験は、深さ固定のランダム木を生成し、同一の探索木を MTD-f と NegaScout を使用して探索し、その結果を比較する方法で行った。深さは 3 から 6 までとし、探索延長や反復深化は導入されていないものとする。また、ランダム木の分岐数は 50 である。

MTD-f では、近似値 f をあらかじめ設定する必要がある。今回の実験では、探索する深さ-2 で探索した結果を近似値として使用している。

500 個のランダム木に対して実験を行い、その性能を調べた。

3.2 ランダム木探索実験結果

500 個のランダム木に対して各深さで探索を実行した場合の、総探索ノード数との関係を示すグラフをそれぞれ、通常の場合を図 1 に示した。

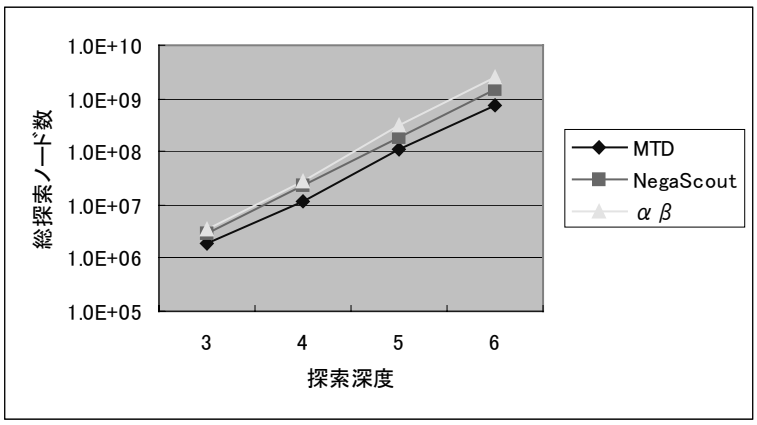

図 1 探索深度と総探索ノード数のグラフ

全体的にみて、MTD が NegaScout よりも高い性能を示していることがわかる。深さ 6 における総探索時間を比較した場合、MTD の方が約 1.8 倍速いという結果になった。浅い探索では差がほとんど認められないが、ランダム木では評価関数に要する時間が 0 に近いので、実際のゲーム木ではある程度の違いが現れる可能性もある。MTD はα β探索よりも小さい探索木を構築するた

め、評価関数の呼び出し回数が少ないからである。

MTD は呼び出し平均回数が少ないほど、速く探索することができる。今回の実験では、平均 4 回呼び出しを行っている。MTD は最低 2 回呼び出しを行う必要があることから、かなり少ない回数で探索を実行していることがわかる。

αβ探索は各ノードにおいて長男ノードの評価値が最も高い場合、探索量が最小となる。そのときの探索木を最小木と呼ぶ。では、MTD が最小木となるゲーム木を探索した場合はどうなるのだろうか。NegaScout と比較した場合の実験結果を表 1 に示した。

表 1 最小木を探索した場合の比較

深	MTD 探索量	NS 探索量	MTD 時間	NS 時間
5	135201	135196	0.11	0.093
6	385189	385182	0.313	0.281
7	6761030	6759788	2.532	2.343
8	19280065	19253820	12.875	12.641
9	338319068	337771768	123.813	117.922

この結果を見ると、NegaScout と MTD の探索量はほとんど変わらないことがわかる。これは、MTD が最小木となるゲーム木を探索する場合、最小木を構成するからである。ただし、2 回探索して最小木を構成することから、一部分のノードが重なっている。そのため、MTD の探索量は NegaScout よりも多くなる。しかし、呼出回数がほぼ 2 である場合、最小木となるような木であっても、MTD の探索量はそれほど大きくならないことがわかる。つまり、MTD の呼び出し回数が多くならない限り、MTD がαβの探索量よりも劣るような状況は少ないと考えられる。実際、予備実験として、ランダム木に降順の傾向を持たせた場合で探索を実行してみても、MTD はαβ探索よりも高い性能を示していた。よって、MTD を実行する上で重要な点は、MT の呼出回数をいかに減らすかという点であるといえる。

4. MTD-f の改良手法

実際のゲーム木探索、特に将棋などのプログラムでは、深さによって値の変更が激しいことが予想されるため、MT の呼び出し回数は必然的に多くなると考えられる。近似値の設定方法がミニマックス値に近い値を設定できるような精度の高い方法であれば、呼出回数の増加を防ぐ事はできる。実際、高性能の評価関数を使用した場合、MT の呼び出し回数は最小に近い回数で済む事が示されている[3]。しかし、すべてのゲーム木において、それが可能となるわけではない。実戦で使用する上では、優れた近似値を設定することが難しい場合も考えられる。

MTD-f は近似値がミニマックス値から離れた設定になるにつれて、呼び出し回数が増加し、αβ探索よりも性能が悪くなる[1]。それを助長する要素として考えられるのが、MTD の上限や下限における変更ステップの狭さである。MTD-f などでは、しきい値の変更量が小さいために、目的のミニマックス値への移動が小さく、結果的に MT の呼び出し回数の上昇を招いていると考えられる。

ミニマックス値へと近づく速度の遅さを解消することで、より速く近づけるようにする方法として[1]で提案されているのが MTD-step である。しかし、提案された手法は、しきい値の初期値が+∞に設定されており、MTD+∞に近い動作となることが予想される。そこで、MTD-f に近い形で実行できるようにして、さらに早くミニマックス値へと近づけるようにした改善案を次に示す。また、上限、下限を求め続ける MTD の性質を利用して、ウィンドウ探索と組み合わせることで、ミニマックス値を早く見つけ出す手法についても提案する。

4.1. MTD-f-step

MTD-step はその開始値を+∞に設定する手法であった。それに対し MTD-f-step は初期値に MTD-f で使用されている近似値を使用する。これによって、MTD-f に近い動作が実現される。また、[1]における MTD-step の説明では、ステップの大きさは徐々に小さくなるとの説明しかない。そこで、ステップサイズについて設定する必要がある。

後で示す実験では初期のステップサイズを 100 に設定した。そして、しきい値がミニマックス値を飛び越えた場合、ステップサイズを 50 に減らす。その次の MT 呼び出しの後で、しきい値が再びミニマックス値を飛び越えてしまったかに関わらず、ステップサイズを 25 に設定する。さらに次の MT 呼び出し後ではステップサイズを 12 に変更する。その次の呼び出し後ではステップサイズを 0 に設定し、MTD+∞(MTD-∞)と同様の実行を行ってミニマックス値に近づけるという方法を行っている。

この手法で行う場合、開始時のしきい値 f がミニマックス値に極めて近い値であったとしても、MTD-f よりも多く MT 呼び出しを行う可能性が高い。しかし、大きく値を外れた場合のリスクを軽減することができる。

4.2. MTD-f-α β

MTD-f-α β は MTD-f とウィンドウ探索を組み合わせた探索手法である。はじめは MTD-f-step と同様に、初期値を近似値 f に設定し、ステップサイズ 100 で MT を呼び出しつづける。そして、しきい値がミニマックス値を飛び越えた時に、MT の呼び出しを止めて、得られた上限、下限をもとにα β探索によるウィンドウ探索を行うものである。ウィンドウ探索とは、探索の結果得られる値の範囲をあらかじめ限定することで探索量を削減する探索手法である。通常はある程度の予測を立てて実行し、予測が外れた場合は再探索を実行する。しかし、MTD によって得られた上限と下限を使用する場合、再探索を行う必要はない。

4.3. 実行する上での問題点

この二つのアルゴリズムを実行する上で、より高い性能を出すためには、トランスポジションテーブルには上限と下限の二つの値を保持する必要がある。しかし、時間の関係上、その実装を行うことはできなかった。二つの値を保持しなくても、最善手を選び出すことは可能であるが、探索ノード数や探索時間が増加する可能性がある。二つのアルゴリズムの収束状況を図 2 に示す。

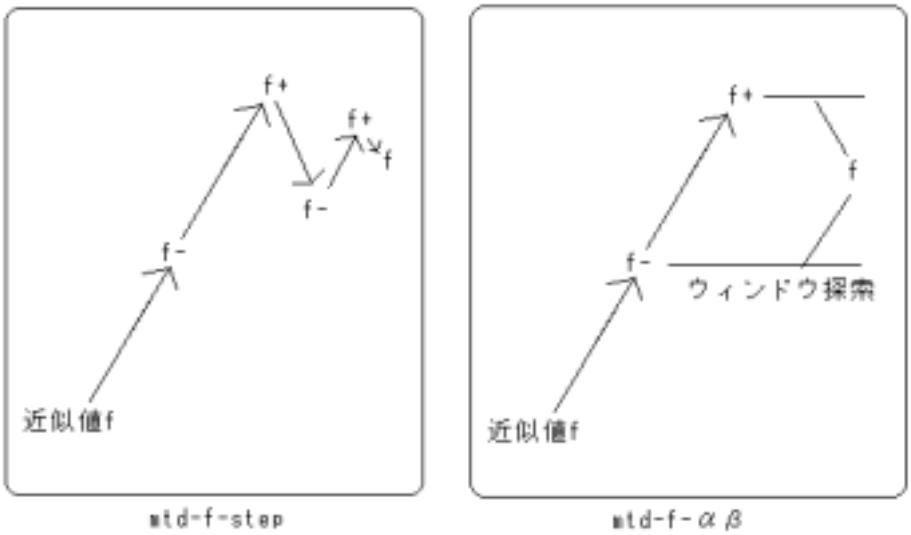

図 2　mtd-f-step, mtd-f-α β の収束

5. αβとの比較実験

5.1. 実験に使用したプログラム

MTD を将棋に適用し、その探索ノード数をαβ探索の場合と比較する実験を行った。その実験環境を次に示す。

実験には、あたらしく本研究室で開発された将棋システム「まったりゆうちゃん」を使用した。このシステムは、古いバージョンの将棋プログラムであり、コンピュータ将棋選手権で一次予選を突破したプログラムとは異なっている。よって、反復深化法や確率延長は導入されていないものである。プログラムには多様な探索延長が組み込まれているが、MTD とαβを比較する関係上、その一部を外して行った。具体的には、あるノードにおいて長男となる子ノードの探索を延長する事と、シンギュラー延長を外している。また、詰めルーチンも外している。しかし、基本的には元システムと同じアルゴリズムを使用するようにしている。

シンギュラー延長とは、あるノードにおける各子ノードの評価値を調べた際に、一つの子ノードの評価値が他の兄弟ノードの評価値より比較的離れている場合、このノードの探索結果を問題視し、さらに深い深さで再探索を行うものである。将棋においては、水平線効果などの関係上、不自然に値が高くなったノードを信用しない方が良い場合があるためである。しかし、MTD が得る評価値は上限、下限という暫定的なものであり、αβが行うシンギュラー延長とは異なる結果が得られることになる。よって、シンギュラー延長を外すこととした。

このようにして、αβ探索と異なる結果が得られる可能性を生み出す部分を外して実行を行ったが、実際には最善手が一致しないケースが現れた。原理的に言えば、MTD とαβは同じ最善手を選び出す。しかし、MTD は基本的にαβ探索よりも小さな探索木を生成する。したがって、αβ探索は MTD が取得しなかった情報をトランスポジションテーブルに取得している。ここで、探索延長が導入されている場合、より深い探索結果がトランスポジションテーブルに格納されている可能性がある。この情報が使用された場合、MTD とαβ探索の結果は一致しない可能性が高い。このことは、トランスポジションテーブルを用いたαβ探索が、MTD よりも大きな探索木を探索できる性質を持っていることを意味する。また、この逆の可能性、つまり MTD が、αβ探索では調べない情報を得る可能性については調べていない。

5.2. 実験方法

実験は、同じ探索木を NegaScout と MTD とで探索させて、その探索ノード数、探索時間、MT の呼び出し回数を比較することで性能を調べた。方法は、NegaScout 探索するシステムを後手として、NegaScout 探索と MTD を同じ探索木に対して実行するシステムとで対戦させ、二つのアルゴリズムで同じ木を探索しているプログラムからデータを得た。探索する木を同じものにするため、静的評価値に乱数を付加しないように設定している。なお、対戦相手となる NegaScout 探索だけのシステムは、静的評価値に乱数が付加されている。また、定跡や駒組み部分に関しては両者とも乱数が付加されている。対戦は時間制限で手数制限なしの 255 対局を行った。深さの設定は、通常の探索を 15、取り合い探索を 25 に設定して行った。探索は通常、一つ深くなるごとに 10 減らしていく。ただし、探索延長が行われた場合、10 以外の値で深さを減らしている。

実験に使用した MTD は、MTD+∞, MTD-∞, MTD-f、MTD-f-step、MTD-f-αβの五種類とした。MTD-f の近似値に関しては、二手前の自分の局面で得られた最善手の評価値を使用している。また、MTD-f-αβはウィンドウ探索のとき、NegaScout 探索を使用して実行している。評価値の誤差分布を図 3 に示す。ただし、この図は、無限大になった情報は除外したものである。この誤差分布の平均値は-3.5、標準偏差は 94.3 である。ここで、この将棋システムにおける歩一枚の価値は約 20 である。

MTD-f-step と MTD-f-αβに関しては、ステップサイズを変更した場合についても調査している。MTD-f-step では、ステップサイズの初期値を 100 ではなく 50 から始めた場合についても調査した。また、MTD-f-αβはステップサイズを 100, 50, 25 の三種類で実験している。また、MTD

は詰めが近づいた場合、評価値の変動が激しくなり、呼出回数が大幅に増加する。そこで、1000回呼び出しても解が得られない場合は、その探索を中断している。

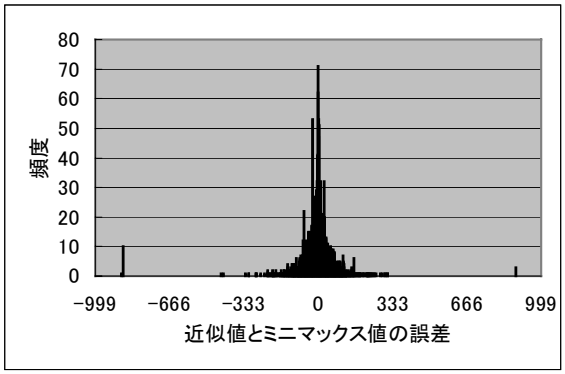

図 3　将棋プログラムにおける評価値の誤差分布

5.3. 対戦実験結果

対戦の結果得られた各種データを表 2 に示す。

表 2　MTD+∞,-∞,-f, mtd-f-step, mtd-f-αβの NegaScout との性能比較

手法	NS を 1 としたときの 探索量	探索時間	NS Nodes/s	MTD Nodes/s	最善手 不一致数	MT 平均 呼出回数
mtd+∞	1.482	0.849	84165	146990	40	43.32
mtd−∞	1.274	0.806	86487	136761	42	32.97
mtd_f	0.900	0.646	85161	118700	43	12.00
mtd_s_100	0.914	0.662	84720	117008	33	6.98
mtd_s_50	0.878	0.634	83069	114937	40	6.50
mtd_a_100	0.911	0.670	85753	116658	30	2.11
mtd_a_50	0.872	0.637	83441	114166	48	2.35
mtd_a_25	0.852	0.629	84135	113983	42	2.81

表において探索量とは、探索ノード数のことであり、最善手不一致数とは、αβ探索の結果とMTD の最善手が異なる場合の数である。mtd_s とは MTD-f-step のことであり、mtd_a とは MTD-αβを意味している。その右側に書かれた値が、ステップサイズである。

実験の結果、すべての手法において NegaScout 探索よりも高い性能を得ることができた。一秒あたりの探索ノード数は NegaScout に比べ、MTD の方が多い。これは、MTD は性質上、同じノードを複数回探索するので、トランスポジションテーブルからデータを高速に呼び出す回数が多いためである。

MTD-f-step や MTD-f-αβはステップサイズの値によって、探索の性能が異なっていることがわかる。今回の実験結果では、ステップサイズを小さめにとる方が性能が高くなっていることがわかる。また、全手法において、もっとも速い探索を実行できたのは、MTD-f-αβのステップサイズが 25 のときであった。また、MTD-f-step もステップサイズの初期値を 50 に設定した場合、MTD-f よりも速くなっている。

6. 考察

実験では、ステップサイズの設定次第では、MTD-f-step や MTD-f-αβが MTD-f よりも速く

なるという結果を得た。これは、MTD の呼出回数に関係があるものと思われる。MTD-f の MT 呼出回数が平均約 12 回であるのに対し、MTD-f-step は約 7 回程度と少ない。よって、MTD-f-step や MTD-f-αβは、近似値の設定を補助することができるといえる。

MTD-f-αβに関しては、呼び出し回数が大きくなるにつれて、よい結果になっている。呼び出し回数が 3 以上になった場合について検証してはいないが、回数が大きくなりすぎると悪化することを考えると、呼び出し回数は 3 回ぐらいが最善となる可能性が高い。

予備実験として、将棋の探索を深くして実験を行ったところ、MTD-f と MTD-f-αβの性能が NegaScout とほぼ変わらない程度になるという結果が出た。これは、MTD-f だけでなく、MTD-f-αβも同様の結果になっている。探索の速度が悪くなった原因として考えられるのは、深さが深くなることで探索量が増加し、MT の呼び出し回数増加によるオーバヘッドがシビアになったことが考えられる。しかし、MTD-f-αβも悪くなっていることが、これに矛盾する。

もうひとつの可能性として考えられるのが、前向き枝刈りによる MTD の効果減少である。MTD は後ろ向き枝刈り手法であり、前向き枝刈り手法が強く働いている場合、その効果が減少する可能性が高い。将棋プログラムで使用されていた前向き枝刈り設定を使用して、ランダム木において深さを 6、7 に設定して探索したときの総探索ノード数と総探索時間を表 3 に示す。深さ 6 において、MTD の探索速度は NegaScout の約 1.6 倍となっている。これに対し、前向き枝刈りを導入しなかった場合の探索速度は約 1.8 倍であった。よって、前向き枝刈りが MTD の性能を下げていることがわかる。これに加え、将棋プログラムでは探索延長が使用されているために、探索木の深い位置においては分岐数が少なくなっている。よって、MTD は強い前向き枝刈りと同時に使用しないほうが良いと言えるかもしれない。

表 3 ランダム木で前向き枝刈りを使用した場合の総探索ノード数と総探索時間

	総探索ノード数				総探索時間		
深さ	MTD	NegaScout	αβ	深さ	MTD	NegaScout	αβ
6	343739537	628103273	1069207656	6	325.60	528.57	829.17
7	1816564793	2806926235	5916587602	7	1489.67	2319.56	4371.32

7. おわりに

今回の実験では、αβ探索と性能を比較するために、いくつかの探索延長をはずして行っている。しかし、実戦で使用することを考えた場合、それらの導入方法についても検討する必要があると思われる。また、詰めルーチンとの競合も存在するものと思われるので、その問題解決も重要である。αβ探索において使用されている既存の手法が、MTD に対して単純に適用できるかどうかわからない。それらについて検証することも、実戦で使用する上で重要であるといえる。

また、MTD-f-step や MTD-f-αβは、トランスポジションテーブルに二つの値を入れることや、ステップの大きさを検討することで、さらに高速化が期待できる。

また、MTD-∞は、マイナス方向から近づくために、特定の深さでは MTD+∞よりも高い性能を示す。そこで、わざと近似値をいくつか減らして実行する方法も考えられる。

参考文献

[1] Aske Plaat, Jonathan Schaeffer, Wim Pijls, Arie de Bruin：A New Paradigm for Minimax Search, Technical Report TR-CS-94-18, 1994

[2] 松原　仁：コンピュータ将棋の進歩 2 , 共立出版株式会社, 1998

[3] 三輪　誠：ゲーム木探索の大規模並列化, 第 65 回全国大会, 2003

[4] Aske Plaat, Jonathan Schaeffer, Wim Pijls and Arie De Bruin：Best-First and Depth-First Minimax Search, Artificial Intelligence 87(1-2): 255-293 , 1996