将棋終盤で発展してきた探索手法のLines of Actionへの適用

作田 誠[†] 長嶋 淳[††]
橋本 剛[†††] 飯田弘之[††]

将棋終盤の探索で成功している λ^1-探索（詰み探索）とキラー木ヒューリスティクス（詰み部分木の応用）を，Lines of Action の終盤探索において，反復深化法および PDS に組み込む形でテストした．λ^1-探索はそのまま生成着手を限定した形では勝ち手順を見つける能力は低かった．また λ^1-着手を優先的に並べ替える手法では本質的な探索節点数は少し減少したが，実行効率が悪く解答能力は低下した．キラー木ヒューリスティクスについては，深さ優先探索に組み込むタイプのアルゴリズムを提案し実装・テストした．その結果反復深化法で 5%程度，PDS で 20%程度の探索節点数・解答時間の減少を達成でき，解答能力も高まった．

Application of the Methods Developed in the Endgame Search of Shogi to Lines of Action

Makoto Sakuta,[†] Jun Nagashima,[††] Tsuyoshi Hashimoto[†††] and Hiroyuki Iida[††]

This paper examines the application of the λ^1-search and the killer-tree heuristics to the endgame search of Lines of Action. Both methods are incorporated into two depth-first searches: iterative deepening and PDS. The λ^1-search, in which move generation is limited to λ^1-moves, has shown a poor ability to find a winning solution. With using the λ^1-sorting, in which λ^1-moves are sorted at the top, the number of searched nodes has been a little reduced, but the solving ability has declined because of the inefficiency of move generation. A new algorithm, in which the killer-tree heuristics is incorporated into a depth-first search, has been proposed, implemented and evaluated. Then, the number of searched nodes and the solving time has reduced by 5% in iterative deepening and by 20% in PDS. Consequently, the solving ability has been enhanced.

1. はじめに

ゲームやパズルを題材とした探索による問題解決は人工知能研究の重要分野である．とりわけ，詰将棋問題を題材とした AND/OR 木探索アルゴリズムの進歩は顕著である．代表的な例として，野下による深さ優先反復深化法のプログラム T2 によって短手数問題では人間エキスパートの能力を陵駕し，伊藤・河野・野下らの最良優先探索プログラムによって 100 手を超す長手数の問題も解けるようになった[4),7)]．最近では脊尾あるいは長井による局面表を活用した証明数探索の深さ優先版アルゴリズムによって，500 手以上の超長手数の難問題をも非常に少ない探索節点数と短時間で解けるようになった[12)]．

コンピュータ将棋の分野で発展してきた高性能な詰み探索とそれを元にした詰み部分木の応用という手法は他の問題領域ではほとんど研究されていない．今回我々はそれらの手法を Lines of Action というゲームの終盤探索に応用し評価した．これにより，手法の有効適用領域に関する知見，さらには，将棋や他ゲームの性質の違いに関する知見が得られると期待できる．

2. λ-探索とキラー木ヒューリスティクス

2.1 λ-探　索

チェスでは全幅探索を基本にしているものの，いろいろな探索延長を組み入れることで選択的探索を可能にしている．Deep Blue においては，評価値の特に高い着手，チェックを防ぐ着手，チェックメイト脅威を防ぐ着手などに対して相応な延長ポイントを与えることにより強制手順を中心とした探索延長が実現されて

† 静岡大学大学院理工学研究科
Graduate School of Science and Engineering, Shizuoka University
†† 静岡大学大学院情報学研究科
Graduate School of Information, Shizuoka University
††† 静岡大学工学部
Faculty of Engineering, Shizuoka University

表 1 λ-着手のレベル分類
Table 1 Classification of λ-moves.

	λ^0-着手	λ^1-着手	λ^2-着手	λ^n-着手
一般ゲーム	勝ちになる着手	次にほうっておくと勝ちになる脅威を与える着手とその防ぎ手	次にほうっておくと λ^1 着手の連続で勝ちになる脅威を与える着手とその防ぎ手（λ^1-着手も含む）	次にほうっておくと λ^{n-1} 以上の着手の連続で勝ちになる脅威を与える着手とその防ぎ手（λ^{n-1}-着手以下も含む）
将棋	詰みになる指し手	王手とその防ぎ手	詰めろとその防ぎ手（王手とその防ぎ手も含む）	$(n-1)$ 手すきとその防ぎ手（$(n-2)$ 手すき以下の手とその防ぎ手も含む）

いる[1]．

一方，将棋終盤では，詰み探索，すなわち，着手を王手とそれを防ぐ手に限定した探索が重要で，人間はもちろんのことコンピュータにおいても，勝つためのきわめて重要な探索となっている．詰み探索は実戦の終盤において不可欠なもので，最近では探索の根節点だけでなくある程度の深さまでの内部節点において詰み探索が実行されるようになっている．将棋分野では以前から終盤を玉に対する脅威度，攻め手・受け手の種類で分類・定式化する試みがあった[6]が，最近 λ-探索という，攻め方からあるゴールに対する脅威を与える着手（threat）とそれに対する受け方の防ぎ手（defense）に限定する探索が脅威の緊急度でレベル分けして定式化された[14]．**表 1** に一般のゲームと将棋における λ-着手の分類を示す．λ^1-探索に相当する詰み探索の分野では近年きわめて効率的で解答能力の高い探索法が開発されてきた．その中でも証明数探索を transposition table（以下，局面表と記す）を活用して深さ優先探索にした PDS[8),9)] あるいは df-pn[10),11)] の解答能力が優れている．

また，λ^2-探索に相当する必至探索は詰み探索の高速化，アルゴリズムの工夫などにより近年大きく高性能化された．λ^3 以上の探索も考えられ，将棋の高段者は λ^3 および λ^4 に相当する 2 手すき，3 手すきの寄せといった考え方をする．しかしそれらはコンピュータ将棋では着手生成のコストが大きくなりすぎるためいまだ実現されておらず，むしろ λ^3 以上の着手をレベル分けしない探索の方が優位でないかと思われる．なお最近，必至探索においてすら λ^1-探索と λ^2-探索を分離しない高効率な探索法が発表された[10]ことは注目すべきである．

2.2 キラー木ヒューリスティクス

将棋終盤では，詰めろの判定および詰めろをかけられたときに受け方の着手がそれを防いでいるかどうかの判定が重要である．これらの判定を効率的に行う手法が棚瀬により「詰み部分木の応用」として提案された[13]．それ以前，詰将棋において類似局面で詰み手順を再現する手法を Kawano[5]が提案し，Seo のプログラム[12]でも合駒や成れる駒の不成の探索で活用されていたが，上記手法はそれを実戦終盤で有効利用できる形に拡張したものといえる．

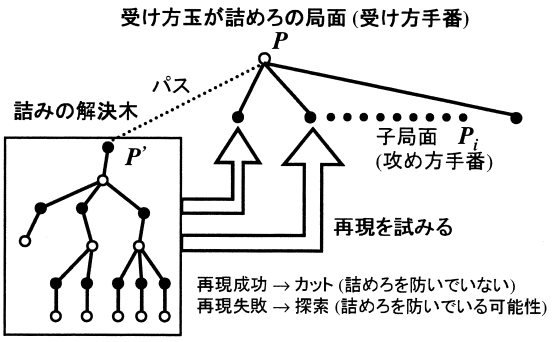

図 1 キラー木ヒューリスティクス（詰み部分木の応用）
Fig. 1 Killer-tree heuristics.

受け方の詰めろを防ぐ着手の判定で説明する（**図 1** 参照）．受け方手番のある局面 P で詰めろをかけられているかどうかは，その局面の手番を攻め方にした局面 P' で詰み探索を行うことによって判定できる．この探索において局面表を使用する．詰めろがかかっていたときは局面表中に詰ますための着手の木ができることになる．局面 P での受け方の各合法手 m_i において，それを指した局面 P_i から上記の着手の木が再現でき同じく詰みに至るかどうかを調べる．すべてが再現でき詰みに至れば，その着手 m_i は詰めろを防いでいない．着手木が再現できない，あるいは詰みに至らない場合は，その着手 m_i は詰めろを防いでいる可能性があるので，実際に詰み探索を行って防いでいるかどうかを判定する（実際には，直下の子局面だけでなく，相手からの王手と自分側の防ぎ手ペアが連続した後自玉が王手でなくなった局面での適用を提案している）．

この手法では，多くの詰み手順に影響を与えない着手を詰み探索をすることなく排除でき，探索コスト

```
int IDwithKTH( 問題局面 ) {   // 反復深化
    for( 深さ = 1; 深さ ≤ 最大深さ; 深さ += ステップ ) {
        r = semekata( 問題局面，深さ，無効局面 );
        r が「勝ち」なら r を返して戻る;
    }
    「未解」を返して戻る;
}

int semekata( position, depth, killer ) {   // 攻め方手番
    position が末端ならその値を登録し返して戻る;
    position を局面表から検索;
    解決済みならその値を返して戻る;
    depth が 0 なら「未解」を登録し返して戻る;
    killer が有効のとき
        killer を検索;
        局面表にある最善手 km を得る;
        km が position でも合法なら
            position で km 実行;
            killer で km 実行;
            r = ukekata( position, depth - 1, killer );
            position で km を戻す;
            killer で km を戻す;
            r が「勝ち」なら r を登録し返して戻る;
        killer を無効化;
    position の合法着手 m[0], ..., m[n_m − 1] を生成;
    各着手 m[i] について
        m[i] が km と等しければスキップ;
        position で m[i] 実行;
        r = ukekata( position, depth - 1, killer );
        position で m[i] を戻す;
        r が「勝ち」なら r を登録し返して戻る;
    「未解」を登録し返して戻る;
}

int ukekata( position, depth, killer ) {   // 受け方手番
    position が末端ならその値を登録し返して戻る;
    position を局面表から検索;
    解決済みならその値を返して戻る;
    depth が 0 なら「未解」を登録し返して戻る;
    position の合法着手 m[0], ..., m[n_m − 1] を生成;
    killerroot = 無効局面;
    各着手 m[i] について
        position で m[i] 実行;
        killer が有効なら
            m[i] が killer でも合法なら，killer で m[i] 実行;
            そうでなければ killer を無効化;
        killer が有効なら
            r2 = semekata( position, depth - 1, killer );
        そうでなければ
            r2 = semekata( position, depth - 1, killerroot );
        killer が無効 かつ killerroot が無効 かつ r2 が「勝ち」なら
            (ある一定の基準を満たせば)  killerroot = position;
        position で m[i] を戻す;
        killer が有効なら，killer で m[i] を戻す;
        r2 が「勝ち」でなければ,「未解」を登録し返して戻る;
    「勝ち」を登録し返して戻る;
}
```

図 2 キラー木ヒューリスティクスを組み込んだ反復深化法
Fig. 2 Pseudo-code of the iterative deepening with the killer-tree heuristics.

が大幅に縮小される．また，棚瀬は詰めろを防ぐ手のチェック手法しか示していないが，逆に，ある攻め方局面の詰み探索における不詰めに至る着手木を再現テストすることによって，明らかに詰めろになってない着手を排除できる．実際には，詰み探索で不詰みを求めるのには詰みよりもはるかに多くのコストがかかるので，不詰めの木を再現することは多くない．現実的な必至探索においてはある程度以上のコスト内で詰みが見つからなかったら不詰みと見なして再現する．これは安全確実ではないがほぼすべての問題で有効に使

える手法である．必至探索では，これらの手法を活用することにより人間エキスパートを凌ぐ速さで問題を解けるようになった[3]．なお，一般的なゲームで考えたとき「詰み部分木の応用」という用語はあまりふさわしくない．本稿では，今後 AND/OR 木への適用だけでなくミニマックス木への適用も視野に入れ，キラー着手[2] を木構造に拡張したものととらえ，キラー木ヒューリスティクス（killer-tree heuristics；KTH と略記）と名付ける．

本研究では，KTH として，将棋の詰めろ防ぎのときのように詰み探索と詰めろ防ぎの探索が分離している形でなく，基本となる深さ優先探索に組み込む形の実装を提案する．受け方手番の局面で，ある子局面 x が攻め方の勝ちになった場合，それ以降の x の兄弟局面の探索において子局面 x から求められる着手木をできる限り再現する．この手法を組み込んだ標準的深さ優先探索の探索深さによる反復深化版の擬似コードを**図 2** に示す．これは「勝ち」か「未解」を返す 2 値探索で，「負け」は調べていない．IDwithKTH() が本体で，攻め方手番および受け方手番のコード semekata() および ukekata() が相互に呼び出される．通常は killer は無効だが，KTH が働いているときは有効になる．killerroot は killer の根節点である．

KTH のアイデア自体はゲームに依存しないが，それをどういうときに使用するべきかはゲームの性質に依存する可能性がある．コードでは子局面が勝ちになったときに killerroot にその局面が設定されることで KTH が働くが，ゲームの性質により何らかの基準を設定できるときはその基準が満足されるときに限り KTH を利用するようにできる．ただし，本研究での実装では何も基準を設けず，解けた子局面があったら必ず KTH を起動する．

なお KTH については，将棋の詰めろ防ぎの探索と同じく，受け方手番の局面で，パスをして攻め方手番とした局面で攻め方の勝ちがあるかどうかを調べ，勝ちがあった場合，各子局面の探索においてパス局面から求められる着手木をできる限り再現する，という形の実装も考えられる．この実装は図 2 の KTH に比べて，パスから始めているので解手順が子局面に至る着手の影響を受けないという利点がある一方，本来の探索空間に存在しない局面を余分に探索するという欠点がある．本研究では局面表の登録数をできるだけ少なく抑えるため，パスを使わない実装とした．

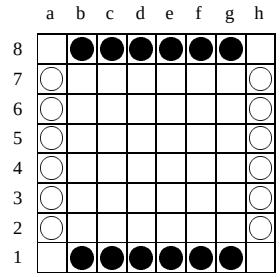

図 3 開始局面
Fig. 3 Starting position.

3. Lines of Action（LOA）

3.1 ル ー ル

Lines of Action（LOA）は 1960 年頃に考案された 8×8 の正方盤を使って黒側と白側とで対戦する 2 人零和完全情報ゲームである．LOA では，Mind Sports Olympiad で人間の世界選手権が行われたほか，Computer Olympiad でもコンピュータプログラムどうしでのゲームが競われている．なお，本稿では盤位置を一般的なチェス式で表す．以下にルールを示す．

(1) 開始局面では各側 12 個の石があり，黒石を行 1 と行 8 の列 b–g に置き，白石を列 a と列 h の行 2–7 に置く（**図 3** 参照）．
(2) 黒番から始めて交互に動かす．
(3) 手番のプレイヤは自分の石の 1 つを必ず動かさなくてはならない．移動は直線的に 8 方向可能だが，移動のマス数はその方向（逆側も含めて）にある双方の石数の合計と一致していなければならない．
(4) 自分側の石は飛び越すことができる．しかし自分側の石があるマスには移動できない．
(5) 相手側の石は飛び越すことができない．ただし，ちょうど相手側の石のマスに重ねることで相手石を取れる．
(6) 自分側のすべての石を連結することで勝ちとなる．連結は縦横でも斜めでもよい．石数が 1 個になったときも連結したと見なす．

また，以下はまだ議論の余地があるが比較的一般的なルールで，本研究の LOA ではこれらを採用している（Computer Olympiad のルールとは違っている）．

(7) 同時に黒側の石も白側の石も連結した場合，最後の着手をプレイした方のプレイヤの勝ちとなる．
(8) 手番側のプレイヤに合法着手がなければ，そのプレイヤの負けとなる．

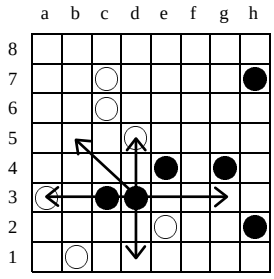

図 4 d3 の石の可能着手
Fig. 4 Possible moves of the disc d3.

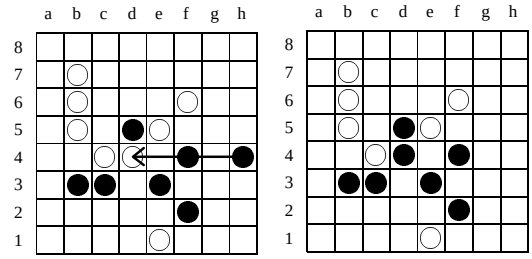

図 5 終局の一例 左 (a) 黒番局面 右 (b) 黒勝ち
Fig. 5 Example of game ending: left (a) A position (Black to move); right (b) Black won.

(9) 同一局面（手番も含めて）が生じたら引き分けとなる．

例を使って可能着手を説明する．**図 4** で黒番として d3 の黒石の移動可能なマスは，まず上下方向（列 d）には 2 個の石があるので 2 マス移動となり，d1 と d5 に移動できる．d5 のときは移動と同時に白石を取ることになる．次に左右方向（行 3）では 3 個の石があるので a3 と g3 に移動できる．a3 のときは黒石を飛び越して移動して白石を取ることになる．また右下/左上方向では 2 個の石があるので b5 に移動できる．相手の石は飛び越せないので f1 には移動できない．最後に右上/左下方向では 4 個の石があるが，左下方向は盤からはみ出すので駄目で，右上方向は着地地点 h7 にちょうど自分側の黒石があるので駄目である．結局，d3 の黒石について全部で 5 通りの合法着手がある．なお，全黒石で 27 通りの合法着手がある．

図 5 は終局の例で，(a) の黒番局面から h4:d4 と白石を取って (b) になると，黒石がすべて連結したので黒の勝ちとなる．

3.2 ターゲットとした理由

LOA は終盤でも合法着手数が比較的多く，平均合法着手数は将棋終盤のように 100 を超すことはないが 25 程度だと報告されている[15]．今回 LOA をターゲットとした理由は，終盤でも合法着手数が比較的多い点，終盤データベースが有効利用できない点，終盤で見つかる勝ち手順が比較的長い強制着手手順を持つ点などで LOA の終盤が将棋終盤と似ており，2 章で述べた将棋由来の手法が有効に応用できるのではないかと考えたからである．これらの手法は今のところ他のゲームでの応用がほとんど知られていないが，応用を試み評価することで，これら手法の有効適用領域に関する知見，さらには，将棋や他ゲームの性質の違いに関する知見が得られると期待できる．

また LOA は探索空間にサイクルを持つ点も将棋と同じで，サイクル問題があるときの木探索アルゴリズムの性能を評価する対象にもなるが，本稿ではこの点に関しては触れない．

一般に，勝ち手順が比較的長い強制着手手順を持つ問題では証明数探索が有効だと考えられる．実際，LOA でも（元の最良優先の）証明数探索（PN）と PN^2，PN*について終盤探索を調べ比較的良い結果が報告されている[16),17)]．

将棋では λ^1-探索（詰み探索）によって多くの勝ち手順が見つかるが，LOA でも λ^1-探索が有効である可能性がある．ただし，将棋では λ^1 の攻め方着手は王手，受け方着手は王手を防ぐ手としてはっきり識別でき，またそれらの着手を効率的に生成できる．しかし，LOA では λ^1 の攻め方着手（受け方パスなら次に勝ちになる着手）も受け方着手も効率的に生成するのは難しい．そこで着手生成のオーバヘッドと探索局面数の減少とのトレードオフがプラスに働くかどうかが問題となる．なお，λ^1-探索が有効でなければ，より高次の λ^2-探索などは有効になりえない．

また，LOA 終盤では合法着手数が比較的多く，その中には勝ち手順に影響を与えない着手も含まれている．着手がどれだけ局面全体に影響を与えるかに依存するが，KTH も有効である可能性がある．

4. 実　　験

テストセットとして 122 題の LOA 終盤局面の問題集（Winands 氏による）を使った．いずれの問題も手番側の勝ち手順があることになっている（解けなかった問題もあるので未確認．解けた問題はすべて手番側の勝ちとなった）．問題は 5 手から 9 手の勝ちがある比較的簡単な問題が 58 題，それ以上の難しい問題が 64 題ある（詳細は**表 9** 参照）．これらは詰将棋や必至の問題とは異なり着手生成に制限がない．多くの問題は LOA の実戦の終盤局面を取り出したものなので，本稿の結果は LOA 実戦終盤の探索についても通用すると思われる．

基本探索アルゴリズムとして，標準的深さ優先探索

表 2 反復深化法での KTH，λ^1 ソートの実験結果（時間制限 20 分）
Table 2 Summary of results obtained by the iterative-deepening method (within 20 min).

	# solved	for all problems			for only solved problems		
	out of 122	movecount	ttcount	time	movecount	ttcount	time
	84	1.54E+8	4101852	432.7	3.39E+7	859059	84.9
with KTH	87	1.47E+8	3869015	415.7	3.77E+7	979491	99.6
with λ^1-sorting							
	83	4.81E+8	1923420	450.6	1.09E+8	335361	97.8
with KTH	84	4.75E+8	1888059	445.0	1.14E+8	389752	103.3

movecount: count of making moves in the search, ttcount: number of registered nodes in the transposition table, time: solving time by seconds

表 3 反復深化法での KTH，λ^1 ソートの実験結果（ttlimit=300 $\times 10^4$）
Table 3 Summary of results obtained by the iterative-deepening method (ttlimit=300 $\times 10^4$).

	# solved	for all problems			for only solved problems		
	out of 122	movecount	ttcount	time	movecount	ttcount	time
	76	5.11E+7	1311436	140.8	1.05E+7	289408	27.4
with KTH	78	5.01E+7	1258502	138.8	1.00E+7	276117	26.1
with λ^1-sorting							
	80	3.84E+8	1167482	350.3	7.99E+7	205409	70.1
with KTH	81	3.82E+8	1164467	348.4	9.01E+7	235369	79.0

の探索深さによる反復深化版（以下，ID と略記）および証明数探索の深さ優先探索版である PDS を選び，KTH あるいは λ^1-探索の手法を組み込み，効果を調べた．λ^1-探索はまず PDS においてそのままの形で実装しテストした．また別に，ID と PDS において，着手のソーティングの際に各着手が λ^1-着手になっているかどうか調べ λ^1-着手であれば先頭の方に並べる形でテストした．これにより深さ優先探索では λ^1-探索と似た効果が得られる．以下では後者の手法を λ^1 ソートと呼ぶ．

実験は Athlon 1.2 GHz の Windows 2000 マシン（メモリ 512 MB）で行った．ID では 1,600 万エントリ，PDS では 800 万エントリの局面表を使用し，garbage collection は行わなかった．各問題について，時間制限を設けて時間内で解けないものを未解とした実験，および，局面表の登録数に限界値を設けてそれを超えたら未解とした実験を行った．ID および PDS とも，時間制限は 20 分とし，局面表登録数の限界値は 10 万，30 万，100 万，300 万のどれかとした．

5. 結果と考察

ID での時間制限および局面登録数制限（ttlimit と表す）300 万での実験結果のまとめをそれぞれ**表 2** および**表 3** に示す．データは全 122 題あるいは解けた問題の平均値で，時間データはすべて秒単位である．movecount は探索中の局面更新数で探索コストの目安となる．一方，ttcount は局面表に登録された節点数で探索に本質的に関わる節点数を表す．時間制限実験の結果を見ると，KTH により，解ける問題数が増加し，探索節点数・時間とも 5%程度の減少が見られた．また，λ^1 ソートをした場合，ttcount で表される本質的な探索節点数は 50%程度に減少した．しかし現在の λ^1-着手生成の実装では，すべての着手について 1 手先局面に更新し勝ちとなる着手があるかどうかを調べ局面を戻す操作を行っているため，効率が非常に悪く探索時間・解答能力では λ^1-着手を考慮しない場合に及ばなかった．一方，局面登録数制限の実験結果を見てみると，λ^1 ソートにより少し解答能力が増している．ただし，本質的な探索節点数の減少は実際にはわずかであることが分かる．時間制限データでの本質的な探索節点数の大きな減少は着手生成が遅いことによる見かけ上の寄与が大きい．局面登録数制限を他の値にしたときの解けた問題数を**表 4** に示したが，いずれも λ^1 ソートにより 3〜9 題多くの問題を解いている．このことは，将棋プログラムのように λ^1-着手の生成を効率化することで λ^1 ソートが有効になる可能性があることを示唆している．

表 4 ttlimit を変えた反復深化法（KTH，λ^1 ソート）での解けた問題数
Table 4 Number of solved problems by the iterative-deepening method with changing 'ttlimit'.

ttlimit $\times 10^4$	10	30	100	300
	44	58	68	76
with KTH	45	63	70	78
with λ^1-sorting				
	50	67	75	80
with KTH	49	67	75	81

表 5 PDS を使った λ^1-探索の実験結果
Table 5 Summary of results obtained by the λ^1-search using PDS.

	# proved	for all problems			for only proved problems		
		movecount	ttcount	time	movecount	ttcount	time
all 122 problems	14 (11%)	3.70E+5	268.7	0.26	2.19E+6	1589.2	1.43
58 problems with less than 10 steps	13 (22%)	7.20E+5	513.8	0.51	2.22E+6	1598.2	1.43

表 6 PDS での KTH, λ^1 ソートの実験結果（時間制限 20 分）
Table 6 Summary of results obtained by PDS using KTH and λ^1-sorting (within 20 min).

	# solved	for all problems			for only solved problems		
	out of 122	movecount	ttcount	time	movecount	ttcount	time
	113	3.37E+7	1274703	162.7	2.03E+7	753998	87.0
with KTH	116	2.71E+7	1041613	121.6	1.86E+7	705558	80.0
with λ^1-sorting							
	99	4.27E+8	679258	339.7	1.75E+8	297179	139.5
with KTH	104	3.72E+8	615895	298.1	1.74E+8	290245	141.8

表 7 PDS での KTH, λ^1 ソートの実験結果（ttlimit=300 $\times 10^4$）
Table 7 Summary of results obtained by PDS (ttlimit=300 $\times 10^4$).

	# solved	for all problems			for only solved problems		
	out of 122	movecount	ttcount	time	movecount	ttcount	time
	105	2.11E+7	786673	91.8	1.15E+7	428325	49.3
with KTH	106	1.86E+7	711211	80.9	9.26E+6	365734	40.6
with λ^1-sorting							
	104	5.37E+8	789270	426.7	2.73E+8	406644	217.1
with KTH	105	4.39E+8	679208	349.5	1.87E+8	303460	152.5

次に PDS での結果を示す．第 1 に，λ^1-探索をそのままの形で実装した場合を見てみる．PDS で攻め方の着手生成を λ^1-着手に限定して探索した結果のまとめを**表 5** に示す．すべての問題が解決された（prove あるいは disprove された）が，多くの問題（108 題）で解なしとなった（disprove された）．これは解手順すべてが λ^1-着手である問題の割合が将棋の場合ほど大きくないことを意味する．10 手未満の短手数の問題（58 題）に限っても解を得られたのは 13 題にすぎなかった．ただ問題解決時間が非常に短いので，LOA 実戦の終盤探索で最初に λ^1-探索を試みる価値はあるかもしれない．

第 2 に，時間制限および局面登録数制限（ttlimit）300 万での KTH と λ^1 ソートの効果を調べた結果をそれぞれ**表 6** および**表 7** に示す．時間制限実験の結果を見ると，KTH を使うことによって，解けない問題は 9 題から 6 題に減少し，探索節点数・時間とも 20%程度の減少が見られた．PDS では深さによる反復深化法と違って指向的な深い探索が行われるため，より効果が大きかったと思われる．また，λ^1 ソートをした場合，ttcount で表される本質的な探索節点数は減少したが，解答時間は倍近くかかるようになり，結果的に制限時間がある場合の解答能力は低くなった．一方，局面登録数制限の実験結果を見てみると，λ^1 ソートにより解答能力の改善は見られない．局面登録数制限を他の値にしたときの解けた問題数を**表 8** に示したが，10 万，30 万，100 万のときはいずれも λ^1 ソートにより 1〜6 題多くの問題を解いている．これらから，λ^1 ソートは全幅探索である ID に適用したときほどは有効でないものの，特に探索節点数が少ないときには効果があることが分かる．

表 8 ttlimit を変えた PDS（KTH, λ^1 ソート）での解けた問題数
Table 8 Number of solved problems by PDS with changing 'ttlimit'.

ttlimit $\times 10^4$	10	30	100	300
	47	73	87	105
with KTH	48	73	92	106
with λ^1-sorting				
	50	74	91	104
with KTH	54	77	93	105

次に，それぞれのアルゴリズムバリエーションで最も多くの問題を解いた KTH を入れた ID および KTH を入れた PDS（いずれも時間制限 20 分）のデータを基に，各問題の手数と探索時間の関係をまとめておく．**図 6** および**図 7** にそれぞれのプロットを示し，また，表 9 に各手数で解かれた問題数を示す．なお，ID で

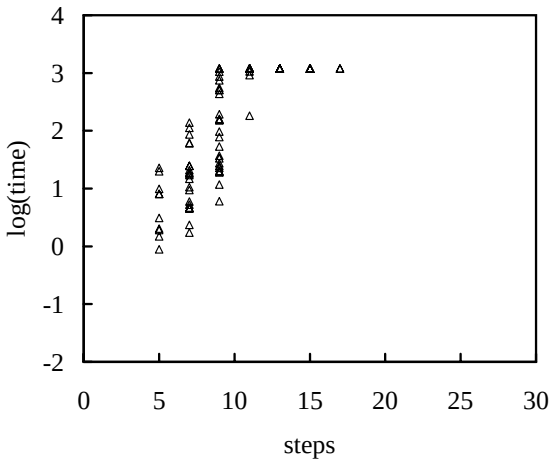

図 6 KTH を入れた反復深化法で解けた問題の手数とその解答時間
Fig. 6 Steps and solving time of solved problems by the iterative deepening with KTH.

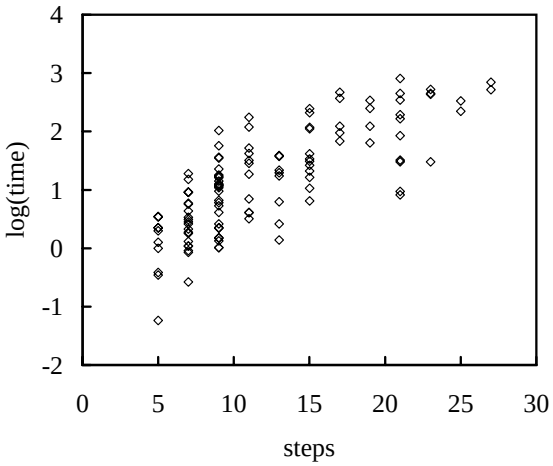

図 7 KTH を入れた PDS で解けた問題の手数とその解答時間
Fig. 7 Steps and solving time of solved problems by PDS with KTH.

表 9 KTH を入れた反復深化および PDS で解けた問題数の手数分布（時間制限 20 分）
Table 9 Steps of solved problems by the iterative deepening and PDS with KTH (within 20 min).

steps	5	7	9	11	13	15	17	19	21	23	25	27
ID	10	21	27	11	7	9	2	0	0	0	0	0
PDS	10	21	27	11	8	12	5	4	10	4	2	2

解けた問題についてはそれで得られた解決木の高さを問題の手数とした．PDS のみで解けた問題については，それで得られた解決木の高さを手数としたが最短手数の保証はない．大まかには，ID も PDS も右肩上がりの分布をしており，手数が長いほど解きにくくなる．ID は手数の増加に対して探索節点数の増加が著しいと考えられ，19 手以上の問題は解けていない．

6. ま と め

将棋終盤で成功した λ-探索と KTH を LOA の終盤探索でテストした．λ^1-探索については，勝ち手順中の λ^1-着手の割合が高くないことから，そのままでは勝ち手順を見つけるのには向かなかった．λ^1-着手を優先して探索する λ^1 ソートの手法では，本質的な探索節点数は少し減少した．現在は λ^1-着手の生成の効率が悪いため時間的には良い結果は出なかったが，効率を高めれば有望な手法となる可能性がある．

一方，KTH は反復深化法で 5%程度，PDS で 20%程度の探索節点数・時間の減少を達成でき，有効であった．今回 LOA の終盤探索における KTH の実装では，KTH を働かせる際に何ら基準を設けなかったが，問題領域に特化した何らかの基準を設けてそれが満足される場合に限り KTH を働かせることによりいっそう効率化できる可能性がある．LOA の場合考えられる基準は，得られた着手木の高さがある範囲かどうか，あるいは，その子局面に至る着手が相手石を取る手かどうか，などである．これらの効果を調べるためには，パラメータを細かく変動させた数多くの実験が必要であり，今後の課題といえる．

λ-探索や KTH が有効であるかはゲームの性質に大きく依存する．λ-探索は，ゴールに向う際のレベル分けがはっきりしており，かつ，λ-着手が効率的に生成できるゲームに適していると思われる．LOA の我々の実装ではレベル分けしない探索の方が優位であった．KTH は 1 つの着手が局面全体に与える影響が小さいものほど効果が大きいといえる．LOA では 1 着手の影響が小さくはないと考えられるが，合法着手数が多いため勝ち手順に影響を与えない着手が少なからずあったと思われ，KTH は有効であった．

謝辞 LOA 終盤の問題集を提供していただいた Winands 氏に感謝する．

参 考 文 献

1) Campbell, M., Hoane, Jr., A.J. and Hsu, F.-h.: Deep Blue, *Artificial Intelligence*, Vol.134, No.1-2, pp.57–83 (2002).
2) Gillogly, J.J.: The technology chess program, *Artificial Intelligence*, Vol.3, pp.145–163 (1972).
3) 橋本 剛，作田 誠，飯田弘之：必至問題を解くプログラムとその評価，人工知能学会論文誌，Vol.16, No.6, pp.539–547 (2001).
4) 伊藤琢巳，野下浩平：詰将棋を速く解く 2 つ

のプログラムとその評価，情報処理学会論文誌，Vol.35, No.8, pp.1531–1539 (1994).
5) Kawano, Y.: Using Similar Positions to Search Game Trees, *Games of No Chance*, Nowakowski, R.J. (Ed.), MSRI Publications, Vol.29, pp.193–202, Cambridge University Press (1996).
6) 香山健太郎：終盤の定式化についての一提案，コンピュータ将棋協会誌，Vol.10, pp.67–69 (1997).
7) 松原　仁（編）：コンピュータ将棋の進歩，共立出版 (1996).
8) Nagai, A.: A new AND/OR Tree Search Algorithm Using Proof Number and Disproof Number, *Proc. Complex Games Lab Workshop*, pp.40–45, ETL, Tsukuba (1998).
9) Nagai, A.: A New Depth-First-Search Algorithm for AND/OR Trees, M.Sc. Thesis, Department of Information Science, The University of Tokyo, Japan (1999).
10) Nagai, A.: Df-pn Algorithm for Searching AND/OR Trees and Its Applications, Ph.D. Thesis, Department of Information Science, The University of Tokyo, Japan (2002).
11) Nagai, A. and Imai, H.: Application of df-pn+ to Othello Endgames, *Game Programming Workshop in Japan '99*, Hakone, Japan, pp.16–23 (1999).
12) Seo, M., Iida, H. and Uiterwijk, J.W.H.M.: The PN*-search algorithm: Application to tsume-shogi, *Artificial Intelligence*, Vol.129, pp.253–277 (2001).
13) 棚瀬　寧：IS 将棋のアルゴリズム，コンピュータ将棋の進歩 3，松原　仁（編），1 章，pp.1–14，共立出版 (2000).
14) Thomsen, T.: Lambda-Search in Game Trees — with Application to Go, *ICGA Journal*, Vol.23, No.4, pp.203–217 (2000).
15) Winands, M.H.M.: Analysis and Implementation of Lines of Action, M.Sc. Thesis, Universiteit Maastricht, The Netherlands (2000).
16) Winands, M.H.M. and Uiterwijk, J.W.H.M.: PN, PN^2 and PN* in Lines of Action, *Proc. CMG 6th Computer Olympiad: Computer-Games Workshop, CS 01-04. IKAT*, Universiteit Maastricht, Maastricht, The Netherlands, Uiterwijk, J.W.H.M. (Ed.) (2001).
17) Winands, M.H.M., Uiterwijk, J.W.H.M. and van den Herik, H.J.: Combining Proof-Number Search with Alpha-Beta Search, *Proc. 13th Belgium-Netherlands Artificial Intelligence Conference* (*BNAIC 2001*), Krose, B., de Rijke, M., Schreiber, G. and van Someren, M. (Eds.), pp.299–306 (2001).

(平成 14 年 2 月 21 日受付)
(平成 14 年 9 月 5 日採録)

作田　誠（正会員）
2001 年静岡大学大学院理工学研究科博士後期課程修了．博士（工学）．現在，同大学院研究生．ゲーム・パズルを題材とした探索・推論・学習等に興味を持つ．

長嶋　淳
2002 年静岡大学情報学部卒業．現在，静岡大学大学院情報学研究科在学中．

橋本　剛
1994 年京都大学農学部卒業．1996 年東京大学大学院理学系研究科在学中に中国雲南民族学院へ留学．1997 年東京大学を中途退学し，台湾文化大学に留学．1998 年静岡大学大学院理工学研究科博士後期課程入学．2002 年同修了．博士（工学）．現在，学術振興会特別研究員として静岡大学工学部システム工学科に在籍．主にコンピュータ将棋等ゲームの研究，開発や生物進化のモデリングに取り組んでいる．

飯田　弘之（正会員）
1962 年生．将棋プロ棋士六段．1994 年東京農工大学大学院博士後期課程修了．博士（工学）．科学技術振興事業団博士研究員，オランダマーストリヒト大学客員教授等．現在，静岡大学情報学部助教授．ゲーム情報学，エンターテイメントコンピューティング等に興味を持つ．