Vol.2016-GI-35 No.7
2016/3/8

Df-pn探索による6×6 TRAXの解析

桑原 和人[1,a)] 鎌田 雅和[1] 中島 麻衣[1] 西谷 旦[1] 菅原 真[1,b)] 保木 邦仁[1,c)]

概要：二人零和完全情報ゲームの TRAX を，df-pn 探索を用いて解析した．盤面の大きさを 6x6 に制限した場合に，初期局面の探索が終了することが確認されて，引き分けという出力が得られた．

Analysis of 6×6 TRAX by using df-pn search

Kuwabara Kazuto[1,a)] Kamada Masakazu[1] Nakajima Mai[1] Nishitani Tan[1] Sugawara Shin[1,b)]
Hoki Kunihito[1,c)]

Abstract: We analyzed two-player zero-sum perfect-information game TRAX by using df-pn search. We confirmed that the initial position had been searched with an output of draw when the board size is reduced to 6×6.

1. はじめに

TRAX は [*1]，1980 年にニュージーランド人の David Smith 氏が考案したターン制の二人零和確定完全情報ゲームである．他のボードゲームとは異なり，盤面の大きさには基本的に制限がないという特徴がある．

TRAX にはリミテッドトラックスと呼ばれる盤面の大きさを制限してプレイする遊び方が存在する．リミテッドトラックスでは盤面の縦幅，横幅が 8 より大きくなるような場所にはタイルを配置することが出来ない．盤面が全て埋まった時点で決着が着いていなかった場合は引き分けとなる．本論文ではこれを 8×8 TRAX と呼ぶ．

本研究ではこのような盤面の大きさを制限した TRAX を AND/OR 木を用いて解析する．ハッシュ表を使用したり，df-pn 探索を用いたりすることによって大幅に木探索が効率化されることを示す．また，盤面の大きさを 6×6 に制限した場合に，初期局面の探索が終了することが確認されて，引き分けという結果が得られたため，これを報告する．

[1] 電気通信大学
The University of Electro-Communications, Chofu, Tokyo 182–8585, Japan
[a)] kaz03301@gmail.com
[b)] s1211086@gmail.com
[c)] k.hoki@uec.ac.jp

[*1] Official TRAX Website: http://www.traxgame.com/ (last access, 2016)

図 1 Trax で使用するタイル

2. Trax のルール

TRAX のルールについて説明する [1].

2.1 ゲームの進行

TRAX は，タイルを 1 枚ずつ交互に並べていく 2 人用対戦ゲームである．並べるタイルは図 1 に示されるように 6 種類ある．先手が白，後手が赤のプレイヤである．タイルを配置する際，赤のラインは赤，白のラインは白につながるようにタイルを配置しなければならない．自分の色にも相手の色にもつなぐことができる．

例えば図 2 の左の盤面は，3 手までタイルを配置した状態であるが，次の 4 手目を配置する場合右下のように異なる色のラインがつながる手は配置することができない．

2.2 連鎖ルール

TRAX では，通常，先手・後手は交互に 1 枚ずつタイルを配置していく．しかし，連鎖ルールが適用される場合は，例外的に 1 ターンで 2 枚以上のタイルが配置される．連鎖の発生条件は，タイルを 1 枚置いた後，タイルがない

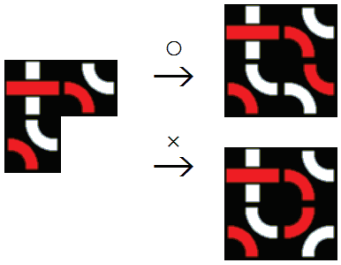

図 2　不正なタイルの配置

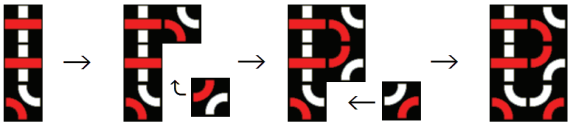

図 3　連鎖が 2 回発生し，計 3 枚のタイルが 1 ターンで配置された例

図 4　3 本の白色ラインが 2 度の連鎖発生後に 1 箇所に集まった例

図 5　赤のループと白のビクトリーラインの一例

場所に集まる同色ラインの数が 2 になることである．このような空きスペースは自動的に埋められる．連鎖ルールに従ってタイルを配置した後，さらに連鎖発生条件が満たされた場合にも同じように空きスペースが自動的に埋められる (図 3 参照)．

また，ある空きスペースに同色のラインが 3 本以上集まったときは，そのターンのプレイヤの手は無効となり，ターン最初の盤面に巻き戻される (図 4 参照)．

2.3　勝利条件

ループかビクトリーラインが形成されるとゲーム終了となる．ループとは，両端がつながったラインである (図 5 左参照)．ビクトリーラインとは，盤面の最上端と最下端を縦断，または最左端と最右端を横断する，長さ 8 列以上のラインである (図 5 右参照)．白のループまたはビクトリーラインが形成されると白の勝ち，赤のループまたはビクトリーラインが形成されると赤の勝ちとなる．手番プレイヤでない方のプレイヤの勝利条件が満たされる場合もある．また，白と赤両方が同時に勝利条件を満たしたときは手番プレイヤの勝ちとなる．

2.4　リミテッドトラックス

TRAX にはタイルの枚数に制限がない．一方，8×8 TRAX と呼ばれる変種ではタイルを配置できるスペースが制限される．ルールは通常の TRAX に準ずるが，場に配置できるタイルは縦横 8 列までとなる．64 枚すべてのタイルを配置しても勝負がつかなければ引き分けとなる．

本研究では，8×8 よりサイズの小さい TRAX も扱う．また，3 本以上の同色のラインが集まる空きスペースができるような手は非合法手として扱い，空きスペースが残っていて合法手がない場合には手番プレイヤの負けとする．

3.　探索プログラムの実装

AND/OR 木を探索してゲーム局面の勝敗を求める [2]．AND/OR 木は Min-Max 木の特殊形であり，各プレイヤの利得は 0 と 1 の 2 値で表される．

3.1　AND/OR 木

AND/OR 木は，AND 節点と OR 節点から構成され，根節点を現在の局面とする根つき木である．各節点は各ゲーム局面に対応し，節点間の枝はゲームの進行に対応する．

OR 節点では，利得を 1 にしたい方のプレイヤが手番を持つ．AND 節点では利得を 0 にしたい方のプレイヤが手番を持つ．各節点は 1，0，不明のいずれかの値を持つ．

子節点を持たない節点を先端節点と呼ぶ．ゲーム終了条件を満たした先端節点を終端節点と呼び，利得がそのまま終端節点の値となる．

子節点を持つ節点を内部節点と呼ぶ．OR 内部節点の値は，値が 1 の子節点が 1 つでも存在すれば 1，すべての子節点の値が 0 なら 0，それらでなければ不明となる．AND 内部節点の値は，値が 0 の子節点が 1 つでも存在すれば 0，すべての子節点の値が 1 なら 1，それらでなければ不明となる．

3.2　深さ優先探索の反復深化

AND/OR 木を幅優先探索すると，木のサイズが有限の場合，原理的には根の値を必ず見つけることが出来る．しかし，幅優先探索には訪問した全節点を保存するために大量のメモリを必要とするという実用上の欠点がある．一方，深さ優先探索は保存する節点は少ないという利点があるが，短手数でゲームが終了するような簡単な手順を見逃す可能性がある．深さ優先探索のようにメモリ消費を抑え，かつ幅優先探索のように最短手順を見つけることが可能な手法が反復深化である．

反復深化では深さ優先探索の深さを閾値として探索を終了させる．最初は深さの閾値を 1 として探索を行い，解が見つからなければ閾値を 2 にして探索を行う．このようにして閾値を 1 つずつ増やしながら探索を根節点に値がつくまで繰り返す．

3.3 トランスポジションテーブル

本研究では，任意の 2 局面に対して，タイルの配置と手番が同じであれば，その局面に到達する手順が異なっていたとしても同じ局面とみなす．そして，そのような複数の局面を 1 つの節点に対応させる．したがって，木ではなく非巡回グラフ (DAG) の探索を行うこととなる．なお，TRAX では 1 ターンあたり 1 枚以上のタイルが配置され，配置したタイルが消滅することもないため巡回は存在しない．

反復深化を行ったり，DAG を探索したりするため，同一節点を 2 回以上訪問するようなことが頻繁に起こる．そこで，探索の効率化をはかるため，トランスポジションテーブル (以下 TT) を利用する．TT とは探索した局面の結果を保存するハッシュ表である．ある局面を探索しようとした時，その局面の結果が TT に保存されていればこれを参照するだけで探索を終了出来るため，探索時間が短縮される．2 局面の比較やハッシュキーには，手の乱数値の排他的論理和をとった 64 ビットのゾブリストハッシュ法を使用する [3]．また，ハッシュ表はチェーンハッシュ法を用いて実装した．

3.4 深さ優先証明数探索 (df-pn 探索)

Df-pn 探索は Allis らの証明数探索を深さ優先で探索する方法である [4]．300 手詰め以上の詰将棋を全て解いたことで有用性が示された [5], [6]．証明数探索で次の最有力節点を選ぶときに前回と同じ経路をたどったとすると，根まで戻って潜るという操作は無駄である．Df-pn 探索では，最有力節点が探索中の節点の下にあるときは戻らずそのまま探索を行うことが出来る．

Df-pn 探索では，探索打ち切りの条件に用いられる閾値として深さ優先探索の探索深さは用いない．証明数と反証数を閾値に用いて打ち切り条件を設定し，各節点を訪問する順番を制御する．

3.5 対称性を考慮した同一局面の検出

探索の効率化のため，回転，反転，平行移動して等しくなる局面を同一局面とみなす．このために，ゾブリストハッシュキーを生成する際には，回転および反転された n×n TRAX の盤面を，n×n の平面に再配置する．この時に，全てのタイルは形を変えずに左上に寄せる．このようにして生成された 8 つの 64 ビットハッシュキーのうち，最も値が小さいものを，局面のキーとする．

3.6 子節点情報の配列への保存

Df-pn 探索では，訪問した節点の証明数，反証数などの情報を TT に保存する．本研究では，TT への保存に加えて探索経路上の各深さの節点に対する全子節点の情報 (ハッシュキー，証明数，反証数) を TT とは別の配列に保存する．探索中の節点の証明数と反証数を再計算するときにはこの配列を参照した．これにより，毎回子節点の局面を生成してゾブリストハッシュキーを求め，TT を参照する場合に比べて約 4 倍の高速化が達成された．

3.7 ガベージコレクション (GC)

TT のエントリが全て埋まってしまった時に GC を行い，空きエントリを確保する．ある局面を根とする部分木のサイズが大きいほど，その局面情報は価値が高いと考えられる．なぜならば，部分木のサイズが大きいほど再計算に時間がかかるからである．したがって，エントリを捨てる時は部分木のサイズが小さいようなエントリから捨てていくのが良い．

本研究の実装は，small Tree GC に基づく [7]．TT のエントリにある節点の情報を登録する際，この節点以下の探索に要した訪問節点数を保存する．TT のエントリが全て埋まったら，次の操作を行う．

(1) 閾値 θ を 1 に設定する
(2) 訪問節点数が θ より小さいエントリを全て空にする
(3) TT の空き容量が一定の割合 r を超えていたら GC を終了する
(4) そうでなければ θ を 1 増やし，手順 2 へ戻る

本研究では r は 0.3 に設定した．

3.8 深さを制限した浅い AND/OR 木探索の併用

1，2 手でゲームが終了するような局面に対しては，TT を用いたり証明数，反証数を用いたりすることによる探索の効率化はあまり望めない．そのため，df-pn 探索の内部節点で浅い深さ d の AND/OR 木探索を行う．

この内部節点 v を根とした浅い深さ d の AND/OR 木探索により，v の値が判明した場合，この結果を TT に保存する．ただし，GC で用いられる情報である部分木の訪問回数は 0 として保存する．ここで，この浅い探索で訪問した節点情報は TT に登録しない．浅い AND/OR 木探索が終了しても v の値が判明しない場合，浅い AND/OR 木探索によって値が判明しなかった子節点を用いて df-pn 探索を継続する．これにより，TT エントリへの登録回数を減らし，GC の発生回数が削減される．

このように，浅い AND/OR 木探索を df-pn 探索に併用する方法は先行研究にもみられる [8]．この先行研究では，詰将棋において df-pn 探索が効率化され，探索節点数および探索時間が削減されることを見出した．なお，この先行研究では新規節点のみに対して浅い AND/OR 木探索を行うことを目指したが，本研究ではこれを訪れた全ての df-pn 探索の内部節点に対し行った．

本研究では，d は $0, 1, 2$ のいずれかとしている．$d = 0$ の場合はこの探索を全く行わず，生成した子節点全てに対し df-pn 探索を行った．

図 6　6 時間以内に結果が得られなかった問題

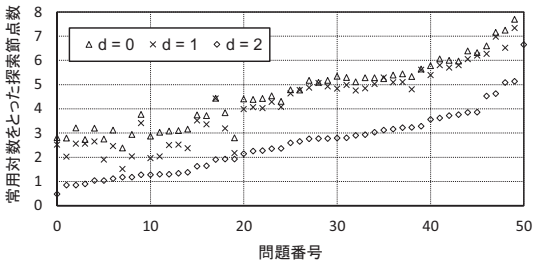

図 7　TRAX puzzles を解くのに要した探索節点数

4. 実験結果

TRAX に df-pn 探索を適用し，ゲームの結果を求め，性能の検証を行う．

実験に使用した環境は以下の通りである．

OS　Windows 8 Pro 64-bit

CPU　Intel(R) Xeon(R) CPU E3-1220 v3 @ 3.10GHz

TT には 120,000,000 エントリー (約 3.4GB) を確保した．

4.1 TRAX puzzles を用いた検証

TRAX オフィシャルサイトで公開されている，TRAX puzzles[*2] を df-pn 探索で解き，探索節点数を比較する．全部で 52 問の問題からなり，難易度 (easy，intermediate，hard) と次の手番 (白，赤) で分かれている．全て次の手番が勝つ局面である．

制限のない TRAX は実装出来ないため，ゲームを 255×255 TRAX とみなして探索し，縦横の列が 255 に達しないことを確認した．また，探索時間が 6 時間経過しても解を見つけることが出来なければ，探索を打ち切った．次の手番が勝った場合の利得は 1，負けた場合の利得は 0 とした

図 7 に結果を示す．勝敗結果を得ることが出来た場合には全て題意と同じ結果が得られた．図 6 の問題のみ時間内に解を見つけることが出来なかった．浅い AND/OR 木探索の深さ d が大きいほど df-pn 探索節点数が減少し，TT の使用エントリ数も減少することが分かった．

4.2 TT と df-pn 探索の性能検証

n×n TRAX では引き分けが存在するため，結果は勝ち，負け，引き分けの 3 通りになる．AND/OR 木探索は利得が 3 通りある場合を扱うことが出来ないので，先手勝ちもしくは引き分けの利得を 1，先手負けの利得を 0 とした探索と，先手勝ちの利得を 1，先手負けと引き分けの利得を 0 とした探索の両方を行う．

[*2] TRAX puzzles http://www.traxgame.com/games_puzzles.php (last access, 2016)

表 1　3×3 TRAX，先手勝ちと引き分けの利得を 1 とした場合．根節点の値は 1．

	探索時間 (s)	探索節点数	深さ
反復深化	2.2	2,899,038	8
反復深化 (TT 利用)	0.0	7,495	8
df-pn ($d = 0$)	0.0	1,063	-

表 2　3×3 TRAX，後手勝ちと引き分けの利得を 0 とした場合．根節点の値は 0．

	探索時間 (s)	探索節点数	深さ
反復深化	2.8	4,561,403	8
反復深化 (TT 利用)	0.0	8,765	8
df-pn ($d = 0$)	0.0	2,548	-

表 3　4×4 TRAX，先手勝ちと引き分けの利得を 1 とした場合．根節点の値は 1．

	探索時間 (s)	探索節点数	深さ
反復深化 (TT 利用)	35.3	2,775,271	11
df-pn ($d = 0$)	1.4	69,150	-

表 4　4×4 TRAX，後手勝ちと引き分けの利得を 0 とした場合．根節点の値は 0．

	探索時間 (s)	探索節点数	深さ
反復深化 (TT 利用)	36.5	2,817,170	11
df-pn ($d = 0$)	2.3	95,557	-

n が 3 と 4 の探索結果を表 1 から表 4 に示す．TT と df-pn 探索は TRAX の探索を大幅に効率化することが明らかとなった．3×3 TRAX と 4×4 TRAX は引き分けという結果が得られた．

4.3 浅い AND/OR 木探索を併用することによる df-pn 探索の効率化検証

n が 5 と 6 の探索結果を表 5 から表 8 に示す．浅い AND/OR 木探索の深さ d が大きいほど df-pn 探索節点数が減少し，d を 0 から 1 にした場合には探索時間も減少することが示された．しかし，d を 1 から 2 にした場合には探索時間が増加してしまうことも示された．5×5 TRAX と 6×6 TRAX も引き分けという結果が得られた．

5. おわりに

サイズを制限した TRAX の初期局面を df-pn 探索を用いて解析した． 盤面の大きさを 6×6 に制限した場合に，初期局面の探索が終了することが確認されて，引き分けという結果が得られた．深さを閾値とした AND/OR 木探索よりも，証明・反証数を閾値とした df-pn 探索の方が効率よく局面を探索することが示された．Df-pn 探索を行う場合，各探索節点で深さの浅い AND/OR 木探索を実行すると探索効率が向上することが確認された．

表 5 5×5 TRAX，先手勝ちと引き分けの利得を 1 とした場合．根節点の値は 1.

d	探索時間 (s)	探索節点数	GC 発生回数
0	90.6	2,400,757	0
1	41.6	1,258,075	0
2	84.5	385,322	0

表 6 5×5 TRAX，後手勝ちと引き分けの利得を 0 とした場合．根節点の値は 0.

d	探索時間 (s)	探索節点数	GC 発生回数
0	150.0	4,086,531	0
1	79.4	2,466,704	0
2	212.7	1,004,546	0

表 7 6×6 TRAX，先手勝ちと引き分けの利得を 1 とした場合．根節点の値は 1.

d	探索時間 (s)	探索節点数	GC 発生回数
0	7,249.0	97,333,448	0
1	3,563.6	66,475,706	0
2	8,780.8	26,060,356	0

表 8 6×6 TRAX，後手勝ちと引き分けの利得を 0 とした場合．根節点の値は 0.

d	探索時間 (s)	探索節点数	GC 発生回数
0	28,529.5	367,199,042	2
1	14,190.9	242,817,026	2
2	22,059.7	66,786,537	0

参考文献

[1] Bailey, D.: *Trax Strategy For Beginners*, D.G. Bailey, New Zealand, second edition (1997).
[2] 小谷善行，岸本章宏，柴原一友，　鈴木豪：ゲーム計算メカニズム: 将棋・囲碁・オセロ・チェスのプログラムはどう動く.
[3] Zobrist, A. L.: A new hashing method with application for game playing, *ICCA journal*, Vol. 13, No. 2, pp. 69–73 (1970).
[4] Allis, L. V., van der Meulen, M. and Van Den Herik, H. J.: Proof-number search, *Artificial Intelligence*, Vol. 66, No. 1, pp. 91–124 (1994).
[5] 長井歩，　今井浩：df-pn アルゴリズムの詰将棋を解くプログラムへの応用，情報処理学会論文誌，Vol. 43, No. 6, pp. 1769–1777 (2002).
[6] Kishimoto, A., Winands, M. H., Müller, M. and Saito, J.-T.: Game-tree search using proof numbers: The first twenty years, *ICGA Journal*, Vol. 35, No. 3, pp. 131–156 (2012).
[7] Nagai, A.: A New Depth-First-Search Algorithm for AND/OR Trees, Master's thesis, The University of Tokyo, Tokyo, Japan (1999).
[8] 金子知適，田中哲朗，山口和紀，　川合慧：新規節点で固定深さの探索を併用する df-pn アルゴリズム，第 10 回ゲーム・プログラミングワークショップ，pp. 1–8 (2005).