ゲーム情報学 5－9
（2001．3．16）

反復深化探索に基く協力詰将棋の解法

星　由雄† 野下浩平† 柳井啓司†

協力詰将棋（協力詰）は詰将棋のルールを変形したパズルである．協力詰の問題はOR木の探索問題として定式化できる．本論文では探索問題として協力詰をとりあげ，これを解くための新しいアルゴリズムを提案する．探索法として，反復深化法と局面表（トランスポジション表）の技法を組み合わせた深さ優先探索を用いる．局面表に新しい技法を導入し，探索の高速化と記憶領域の節約を図る．本論文のアルゴリズムを用いて作成したプログラムは，協力詰の最長手数問題である49909手詰の「寿限無3」をはじめ，これまでコンピュータでは解けなかったいくつかの難しい問題を解くことができる．また実例によって，新しい技法は局面の数が多い問題で特に効果が大きいことを示す．

A New Algorithm for Solving the Cooperative Tsume-Shogi Based on Iterative-Deepening Search

Yoshio Hoshi,† Kohei Noshita† and Keiji Yanai†

The cooperative Tsume-shogi, or helpmate, is one of the most popular variants of Tsume-shogi. Solving a helpmate problem can be formulated as an OR-tree searching. We present a new algorithm for solving helpmate. The basic framework of our algorithm consists of depth-first searching, iterative-deepening and a transposition table. Several new techiniques for handling information in the transposition table are introduced, by which we achieve considerable memory-savings as well as speed-ups. Our program has solved the longest-step problem named Jugemu 3 with 49909 steps, as well as several hard problems, all of which have been intractable by other previous programs. By experiments we show that our techniques are effective particularly for hard problems which need a large transposition table.

1. はじめに

近年，多くの探索問題がコンピュータによって解かれている．しかし，探索問題はその深さに伴って局面の数が爆発的に増加する特徴があり，プロセッサの速度向上だけでは解けない問題がほとんどである．そこで探索問題を解くためには，アルゴリズムの改良が不可欠となっている．

探索アルゴリズムには数多くの研究がある[11]．その中で基本的なものとして，深さ優先探索 (depth-first search, DFS)，幅優先探索 (width-first search, WFS)，最良優先探索 (best-first search, BFS) がある．このうち WFS と BFS は，探索中の節点の集合を記憶するために，一般に大きい作業用記憶領域が必要である．一方 DFS は，探索の深さに比例する作業用記憶領域で済む線形記憶領域の探索法である．1985年に IDA*が発表された[6),14)]．これは反復深化法 (iterative-deepening) と局面表（トランスポジション表，transposition table）の技法を組み合わせた深さ優先探索で，BFS の代表的なものである A*と同等の探索を線形記憶領域で実現する方法である．チェスで用いられてきた反復深化法は探索の深さを順次増加するのに対して，IDA*の反復深化法は評価関数で閾値を定め，それを順次増加するものである．同様の性質を持つ再帰的最良優先探索 (recursive best-first search, RBFS)[7] も考案されている．なお WFS は，評価関数で定める閾値を探索の深さにとる IDA*の特殊なものとして実現できる．

詰将棋はよく知られているように玉を詰めるパズルである．与えられた問題図に対して，先手と後手は交互に指し，先手は王手の連続によって後手玉を最短手数で詰め，後手は詰手順が最長手数となるように王手を回避する．

協力詰将棋（協力詰）は，詰将棋のルールを変形したパズルである．与えられた問題図に対して先手と後手は協力して最短手数で後手玉を詰める．詰将棋と同様に数多くの問題が作られているが，協力詰には詰将

† 電気通信大学情報工学科
Department of Computer Science, The University of Electro-Communications

棋よりはるかに詰手数の長い問題がある．

詰将棋や協力詰は，局面を節点，候補手を枝とする木で表現できる．詰将棋の問題がAND-OR木の探索問題として定式化できるのに対し，協力詰の問題はOR木の探索問題として定式化できる．

詰将棋は日本人になじみが深く，また人工知能のよい例題と考えられ，これをコンピュータで解くための研究が盛んに行われてきた．1990年当時，詰将棋を解くプログラムはDFSを用いたものが主流であり，長手数問題はほとんど解くことができなかった[2),3)]．1994年に，伊藤，河野，野下はBFS[4)]を用いたプログラムによって，超長手数問題の「寿」(611手詰)を解いた．1997年に，脊尾は共謀数を用いた探索法[12)]によって，詰将棋の最長手数問題である「ミクロコスモス」(1525手詰)を解いた．最近では証明数と反証数を用いた探索法[9)]によって，ほぼ全ての詰将棋問題をコンピュータで解くことができるとの報告がある[10)]．

協力詰を解くプログラムは，石黒によるfm[5)]が知られている．また，野下によるT3[4)]がある．T3は，最良優先探索で詰将棋を解くプログラムであるが，これをOR木探索用に調整したものである．1994年に，fmとT3の性能を評価する実験が行われた．性能評価には，橋本によって選択された95題の協力詰の難しい問題が使用された．当時，fmとT3はそれぞれ85題を解いている．

本論文では，探索問題として協力詰をとりあげ，これを解く新しいアルゴリズムを提示する．反復深化法と局面表の技法を組み合わせた深さ優先探索を用いる．ここでの新しい技法は，主として局面表に関するものであり，より多くの局面を枝刈りすることで計算時間を短縮する技法と，局面情報を整理することで記憶領域を節約する技法とがある．

本論文のアルゴリズムを用いたプログラムは，協力詰の最長手数問題である「寿限無3」(49909手詰)[13)]を，コンピュータで初めて解くことに成功している．また，このほかにもいくつかの問題を新たに解いている．本アルゴリズムは，局面の数が多い問題で特に計算時間を短縮する効果が大きい．また，局面表に登録する局面情報を著しく減らすことができる．

本論文のアルゴリズムは，協力詰特有の知識を利用していないため，他の探索問題への応用も容易である．

2. 協力詰について

協力詰は，詰将棋のルールを変形したパズルである．協力詰では与えられる問題図に対して，先手と後手は協力して最短手数で後手玉を詰める．先手の着手は王手，後手の着手は王手を回避する手であることが義務付けられる．協力詰の問題図では，駒の初期配置と詰手数が与えられる．また，無駄合の概念は無い．無駄合とは，詰手順に影響しない後手の合のことである．以上の条件を満たす詰手順が一意的に定まり，駒余でない問題が完全作である．駒余は，後手玉を詰めた局面で先手に持駒が残るものである．協力詰の問題には，作成者の意図とは異なり完全作でないものがある．完全作でない問題には，不詰によるものと余詰によるものとがある．不詰は，問題図から詰局面が得られない問題である．余詰は，問題の作成者が意図とは異なる詰手順が存在する問題である．なお余詰には，それが軽微であるとして許されるものがある．

本論文のアルゴリズムは，問題図から出現しうる全ての局面を網羅的に探索し，主として詰手順の最短解を出力するものとして作られている．

3. 探索アルゴリズム

3.1 探索木

協力詰の問題は，局面を節点，候補手を枝とするOR木の探索問題として定式化できる．問題図で与えられる駒の初期配置が根である．また，候補手が0の局面は葉である．葉には，先手の候補手が0となる不詰局面と，後手の候補手が0となる詰局面とがある．探索の最大深さは，問題図で与えられる詰手数である．

本論文では，探索木の節点のうち，その性質によって次の4種類の節点を定義する．

- 打切り局面
- 既知局面
- 確定局面
- 保護局面および非保護局面

ここで打切り局面は，探索で指定される深さにある節点であるが，葉でない局面である．既知局面，確定局面，保護局面，非保護局面については以下の必要なところで適宜説明する．本論文では，誤解のおそれがない場合，節点とそれに付与された駒の配置を併せたものを，単に局面と呼ぶこととする．

3.2 局面表

局面表は，走査した局面の情報を登録して，繰り返し出現する同一局面の再計算を，局面表の参照によって省略するものである．このように，同一局面の探索を省略することを枝刈りという．局面表は，駒の配置をキーとするハッシュ法によって実現する．駒の配置は，確率アルゴリズム的手法[8)]によって圧縮したビット列で表現する．ビット列の長さは64ビット以上が望ましいとされている．局面表に登録する局面情報は，

駒の配置を表すビット列と，その局面の出現する深さである．本論文のアルゴリズムの性質から，同一局面同士はより浅い（根に近い）ものを優先する．

3.3 深さ優先反復深化

深さ優先探索 (DFS) は，根から葉に向かって一直線に走査し，葉に達すると一つ手前の節点に戻り，次の枝に対して再び一直線の走査を繰り返す．DFS は，詰手数に対して線形量の記憶領域で探索できる．反復深化法は，まず浅い木の探索を行い，段階的に木を深くしながら探索する局面を広げる．このとき新たに走査した局面は，局面表に登録される．DFS に反復深化法を組み合わせた深さ優先反復深化探索 (depth-first iterative-deepening, DFID) は，最初に探索する木の深さ $Depth_0$ と，探索する深さの増分 $Delta$ によって n 回目の反復深化での探索木の深さ $Depth_n$ を定義できる．一般に，DFID は DFS よりも枝刈りする局面が増加する．なぜなら，浅い木の探索で得た結果を深い木の探索で利用できるからである．

3.4 幅優先探索と同じ振舞をする探索法

幅優先探索 (WFS) は，浅い局面を優先的に探索する．WFS は探索途中での木の末端局面を全て記憶する必要があり，一般に探索が深くなるにつれて膨大な記憶領域が必要になる．

DFID で $Depth_0 = 1$, $Delta = 1$ とすると，未走査の局面を走査する順序が幅優先探索と同じになる．本論文では，この探索法を IDWFS と呼ぶ．IDWFS は，局面表を用いて WFS の特徴を利用した枝刈りを行うことができる．同一局面同士では，最初に登録した局面よりも短手数で詰めることはできない．なぜなら，WFS では同一局面同士では最も浅い局面がより早い段階で局面表に登録されるからである．したがって，同一局面は最初に登録した局面以外を枝刈りできる．ここで，この枝刈りされる局面を「既知局面」と定義する．

協力詰では，先手局面と後手局面との間で同一局面が生じることはない．そこで，先手局面（あるいは後手局面）のみに注目すると，$Delta = 2$ としても先手局面（あるいは後手局面）は未走査の局面を走査する順序が WFS と同じになる．よって，IDWFS は $Delta = 2$ としてよい．$Delta = 2$ の IDWFS は，反復深化の回数が $Delta = 1$ の場合の半分になる．

3.5 確定局面

探索結果として詰または不詰が判明している局面を確定局面と定義する．探索結果が判明している局面とは，その局面を根とする部分木の末端局面が全て葉であることをいう．よって，確定局面であることは，次のいずれかによって決定することができる．

- 葉
- 全ての子が葉，確定局面，既知局面の局面

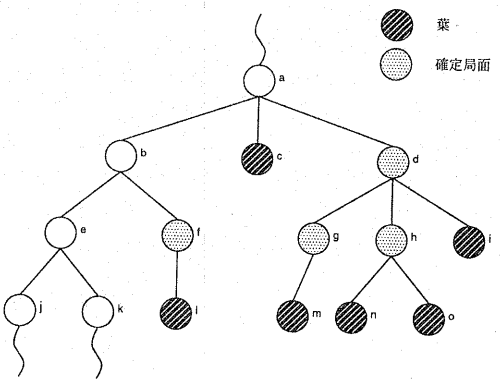

図 1　確定局面の例

葉は，詰局面か不詰局面であるので，葉は探索結果が判明している．葉は候補手が 0 であるため，末端局面である．これより，葉は確定局面である．

全ての子が葉，確定局面，既知局面の局面を考える．IDWFS において既知局面は無視できる（WFS も同様のことがいえる）．このとき，葉と確定局面は探索結果が判明している局面である．よって，この局面は子によって探索結果が与えられる．また，葉と確定局面はそれ自身を根とする部分木の末端局面が葉である．これより，全ての子が葉，確定局面，既知局面の局面は確定局面である．

探索によってある局面が確定局面であることが判明すると，この局面が確定局面であることを局面表に登録する．後の探索で確定局面の同一局面が出現した場合，この局面を枝刈りする．

本方法の確定局面については，IDWFS によって同一局面のうち最も浅い確定局面のみを局面表に登録するという点が本質的である．なお，この最も浅い確定局面の考え方によって，OR 木のループに関するいわゆる GHI 問題 (graph history interaction problem)[1] は生じない．本方法は，一般の IDA*には直接適用できないことに注意されたい．

3.6 保護局面と非保護局面

次の反復深化後の探索で必ず走査する局面を保護局面と定義する．また，保護局面でない局面を非保護局面と定義する．非保護局面とは，次の反復深化後の探索では確定局面による枝刈りで探索されなくなる局面である．

全ての局面は最初，保護局面とする．局面は，局面表に登録された時点では後の探索で重要な局面かを判定することはできないからである．

確定局面によって枝刈りされる局面は，それ以前の探索でその局面を根とする部分木が探索されている．よって，この部分木に含まれる局面は局面表に記憶されている．一方，確定局面の同一局面は枝刈りされるため，確定局面を根とする部分木は，その確定局面自身を除いて探索されない．この部分木に含まれる局面のうち，他の探索で参照される同一局面を除く全ての局面は，後の探索で同一局面が出現しない限り参照されない．これらの局面を非保護局面とする．

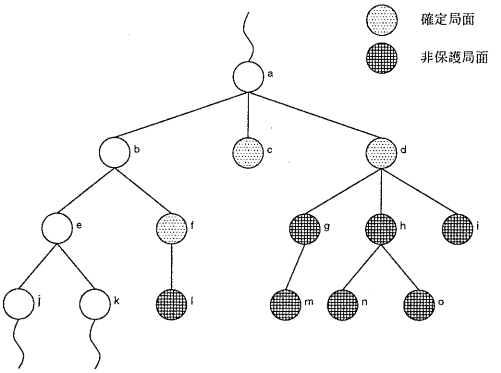

図 2　保護局面と非保護局面

非保護局面の登録されたアドレス（局面表の中の登録位置）へ新しい局面の登録要求があると，非保護局面の局面情報は削除される．また，非保護局面が後の探索で参照されると，非保護局面は保護局面に戻される．本論文ではこれらの操作を ProtectGC と呼ぶ．

局面表には，後の探索で重要な局面と，そうでない局面とが混在している．保護局面と非保護局面によって局面を差別化し，局面情報を整理する．参照される見込みの低い非保護局面を整理することで，限られた局面数しか登録できない局面表の記憶領域を有効に利用する．

ProtectGC は，ガーベジ・コレクション (garbage collection, GC) に似た振る舞いをする．一般に GC は，局面表が溢れた場合に局面情報を整理する．一方，ProtectGC は，反復深化の時点で局面情報を整理する．削除した非保護局面が後の探索で出現しても，再計算されるので一貫性が保証される．整理された局面情報はただちには削除せず，新しい局面によって上書きされるまで保持する．よって，局面情報が削除されるまではその非保護局面の局面情報を探索に利用できる．

3.7　候補手生成の軽量化

ある局面から次の局面へと走査する時，最も計算時間を必要とするのは候補手の生成である．そこで，局面で生成した候補手を局面表に登録することが考えられる．しかし，記憶領域の大きさは限られているため，全ての局面で全ての候補手を登録することは困難である．そこで，ここでは候補手が高々1 つである局面についてのみ候補手を登録する．候補手を生成する前に局面表を参照し，候補手が登録されていればこれを探索に使用する．この工夫は特に新しいものではないが，本論文の計算実験では高速化の効果が著しい．

4.　性能評価実験

協力詰を解くプログラムを作成し，性能評価実験を行う．実験は，Linux の動作する，Athlon 1.2GHz の CPU と，1.25GB のメモリを搭載した AT 互換機上で行う．

本実験ではアルゴリズムの異なるいくつかのプログラムを作成している．作成したプログラムを表 1 に示す．表中の○はプログラムで使用したアルゴリズムである．また局面表とは，局面表に登録可能な局面数の最大値である．

表 1　実験に使用したプログラム

	探索法	a	b	c	局面表
A	IDWFS, $Delta = 1$	○	○		16777216
B	IDWFS, $Delta = 1$	○			16777216
C	DFS	○			16777216
D	DFID, $Delta = 1$				16777216
E	DFID, $Delta = 1$	○			16777216
F	IDA*	○			16777216
G	IDWFS, $Delta = 1$	○	○		100663296
H	IDWFS, $Delta = 2$	○	○	○	4194304

a ：確定局面による枝刈り
b ：ProtectGC
c ：候補手の記憶による高速化

各プログラムについて簡単に説明する．A は，本論文で提案する探索法で基本となるものである．B は，A で ProtectGC を用いないものである．C，D，E，F は，従来の探索法によるプログラムである．これらは，本論文で提案した IDWFS との比較のために用いる．ただし，F は IDA*であるが，評価関数として局面までの探索に要したコストの総和と，局面から問題で指定された詰手数までに必要とする探索の深さの和を用いる．G は，局面表に登録できる局面情報の数を大きくとり，局面の数が多い問題を解くために使用する．H は，局面表に登録できる局面情報の数を少なく

とり，また，局面表に候補手を登録することで候補手生成を高速化したものである．

4.1 解答能力の評価

協力詰を解くプログラムの解答能力を評価する．

4.1.1 橋本孝治による 95 題

評価に用いる協力詰の問題は，橋本孝治によって選ばれた 95 題である．これは 1994 年に，石黒俊太郎による fm と，野下浩平による T3 の解答能力の評価に用いられている．当時，fm と T3 はそれぞれこのうちの 85 題を解いている．なお，fm は後にさらに 3 題を解いたとの報告がある．1994 年当時，解けなかったり，解の出力に長時間を要した 11 題についての結果を表 2 に示す．本実験で使用したプログラムは A である．表中，○は解けた問題，×は解けなかった問題である．

表 2　解答能力の評価

問題	手数	fm	T3	A	計算時間 (s)
30	67	×	×	○	426
31	739	×	×	○	351
67	357	×*	×	○	662
72	135	○	×	○	135
79	77	×	×	○	86
88	35	×	○	○	69
91	1325	×	×	×	-
92	2157	×*	×	○	1168
93	5349	×	×	×	-
94	19447	×*	×	○	10152
95	5119	×	×	×	-

星印の問題は 1994 年以降に解けたとの報告がある．

プログラム A は，95 題のうち 92 題を解いた．また，解けた問題のうち 4 題（問題 13，61，74，77）は木全体を探索できなかった．これらはいずれも多数の余詰を含む問題である．余詰は，詰局面での駒の配置が微妙に異なるものが多数生成されている．これらの余詰に至るまでの局面の数は膨大であるため，局面表の溢れが生じている．この 4 題以外にも多数の余詰を含む問題が存在するが，これらはいずれも局面の数が多い．解けない 3 題もまた，局面表の溢れによるものである．

4.1.2 長手数問題

協力詰の長手数問題に対する解答能力を評価した．協力詰の問題で 1000 手詰以上の完全作は 9 題発表されている．このほかに，完全性の検討されていない問題が 3 題ある．本実験では，これらを問題集として使用した．本実験で使用したプログラムは G である．実験結果を表 3 に示す．

本プログラムは，コンピュータとしては初めて協力詰の最長手数問題である「寿限無 3」（49909 手詰）を解くことに成功した．また，1000 手詰以上の問題のうち，「寿限無 3」を含む 5 題を解いた．

表 3　長手数問題

作者	作品名	手数	G
加藤徹	寿限無 3	49909	○
加藤徹	寿限無 2	38889	○
加藤徹	寿限無	19447	○
加藤徹		5349	×
西田尚史	ゴールドベルク	5119	×
西田尚史		3987	×
森茂		2157	○
鮎川哲郎		1325	×
加藤徹		7181	×
加藤徹		4197	×
加藤徹		4143	○

上の 8 題は完全作扱い．下の 3 題は完全性が未検討．また，鮎川哲郎 3451 手詰問題（完全作）は問題図不明．

次に，協力詰の問題としては特に手数の長い「寿限無」，「寿限無 2」，「寿限無 3」についての実験結果を示す．

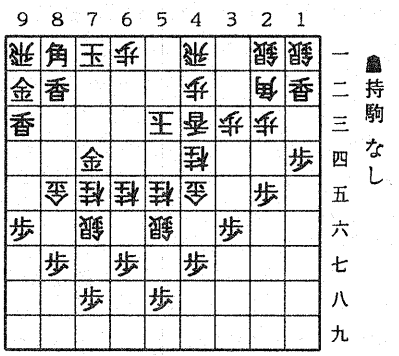

図 3　寿限無 3（49909 手詰）

本実験で使用したプログラムは，A，H，およびプログラム H で問題ごとに局面表のサイズを調整したものである．

表 4　「寿限無」問題の計算時間

	寿限無	寿限無 2	寿限無 3
詰手数	19447	38889	49909
探索局面数	3452219	32829309	13418156
A (s)	10512	34426	38682
H (s)	1508	5254	5455
調整版 (s)	480	5254	3047

調整版プログラムで局面表が登録できる局面数の最大値は以下の通り．
寿限無：262144，寿限無 2：4194314，寿限無 3：524288

IDWFSは探索時に非常に多くの局面を枝刈りする．よって，計算時間は詰手数と比べてさほど長時間にはなっていない．本論文の探索法は局面表の大きさによって計算時間が大きく変化する．つまり，それぞれの問題ごとに適した局面表の大きさがある．「寿限無」問題は，局面の数が比較的小さいため，局面表の大きさは比較的小さくできる．また，局面表に候補手を登録することで，さらに計算時間が短縮されている．これは，枝刈りによって局面の候補手が1つだけとなる局面が数多く存在することを示している．探索が深くなると，探索木は浅い局面で非常に細長い形状になる．よって，探索が深くなっても局面の数が爆発することなく探索できる．

4.2 アルゴリズムの比較

探索アルゴリズムの性能評価のため，既存の探索アルゴリズムを用いたプログラムによる比較実験を行った．実験に使用したプログラムはB，C，D，E，Fである．橋本による問題集より問題14，30を例に，実験結果を示す．問題14，30はいずれも67手詰と比較的短手数の問題であるが，問題14は詰手数に対して局面の数が少ない問題であるのに対し，問題30は詰手数に対して局面の数が非常に多い問題である．走査局面数とは，走査した局面の延べ数である．また，探索局面数とは，出現した駒の配置の総数である．

表5は，問題14におけるアルゴリズムの比較実験の結果である．問題14では，DFSが高速に解を出力している．これは，DFSでは反復深化を行わないためである．反復深化法を用いた探索では，反復深化の際に局面表を再設定する時間が必要である．DFS以外のアルゴリズムの計算時間は，その大部分が局面表を再設定する時間である．プログラムDとEは，確定局面を用いた枝刈りを行うかどうかの点で異なる．プログラムBとEは，既知局面を無視できるかどうかの点で異なる．これらより，IDWFSは枝刈りによって走査局面数を大幅に減少することがわかる．

表5　アルゴリズムの比較（問題14）

プログラム	走査局面数	計算時間 (s)
B	18508	26
C	7862	2
D	52704	26
E	31135	25
F	116724	34

詰手数：67，探索局面数：2862

表6は，問題30におけるアルゴリズムの比較実験の結果である．問題30では，IDWFSが高速に解を出力している．走査局面数の多い問題30では反復深化で必要な局面表の再設定時間は計算時間全体と比較して相対的に小さい．よって計算時間は走査局面数に大きく依存している．また，問題14の結果と比較すると問題30では既知局面を無視する効果が大きく，走査した局面の数は著しく減少している．

表6　アルゴリズムによる比較（問題30）

プログラム	走査局面数	計算時間 (s)
B	27146076	432
C	113330341	1762
D	204106708	3108
E	189986648	2966
F	955175792	14629

詰手数：67，探索局面数：7909636

4.3 局面表を整理する効果

ProtectGCによって局面表を整理する効果を検証した．実験に使用したプログラムはAとBである．橋本による問題集より，問題30，94を例に，実験結果を表7に示す．問題94は，長手数問題の「寿限無」（19447手詰）である．最大保護局面数とは，探索中に保護局面となる局面の数の最大値である．

表7　ProtectGCの効果

問題	ProtectGC	最大保護局面数	計算時間 (s)
30	使用 (A)	1818028	426
	不使用 (B)	7909636	432
94	使用 (A)	60992	10152
	不使用 (B)	3452219	9575

ProtectGCの使用による計算時間の変化はわずかである．問題30では77%，問題94では98%程度の局面が非保護局面として整理されている．整理される局面の数は問題によってばらつきがあるが，橋本による問題集では探索局面数全体の約75%である．これは，局面表に登録される局面情報を1/4程度に整理したことを示している．本プログラムで解けない問題は全て局面表の溢れによるものである．したがって，ProtectGCを用いて記憶領域を有効に利用することで，解けない問題を解きうる．

5. ま と め

本論文では，探索問題として協力詰を解くためのアルゴリズムを提案した．反復深化法と局面表の技法を組み合わせた深さ優先探索で，幅優先探索と同じ振る舞いをする探索法を実現した．また，局面表の新しい技法を提案した．これにより，探索で走査する局面を減少することで，計算時間を短縮した．また，局面情報の整理によって，局面表の記憶領域を有効利用でき

ることを示した．

本論文で提案したアルゴリズムを用いて作成したプログラムは，協力詰の最長手数問題である「寿限無3」をコンピュータとして初めて解いた．また，これまでコンピュータで解くことができなかった問題を数題解いた．なお，本アルゴリズムは協力詰に特化した知識を利用していないため，他の探索問題への応用も容易である．

参考文献

1) Breuker, D. M., Herik, H. J. van den, Uiterwijk, J. W. H. M. and Allis, L. V.: A Solution to the GHI Problem for Best-First Search, CG'98, LNCS 1558, 25–49 (1999).
2) 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, 情報処理学会誌, 35, 8, 1531–1539 (1994).
3) 伊藤琢巳, 河野泰人, 脊尾昌宏, 野下浩平: 詰将棋を解くプログラムの進歩, 人工知能学会誌, 10, 6, 853–859 (1995).
4) 伊藤琢巳, 河野泰人, 野下浩平: 非常に手数の長い詰将棋問題を解くアルゴリズムについて, 情報処理学会論文誌, 36, 12, 2793–2799 (1995).
5) 神無七郎: フェアリー詰将棋のホームページ, http://www.coms.ne.jp/~k_7ro/.
6) Korf, R. E.: Depth-First Iterative-Deepening: An Optimal Admissible Tree Search, *Artificial Intelligence*, 27, 1, 97–109 (1985).
7) Korf, R. E.: Linear-Space Best-First Search, *Artificial Intelligence*, 62, 1, 41–78 (1993).
8) Levy, D. N. L. and Newborn, M.: How Computers Play Chess, *W. H. Freeman and Conpany*, New York (1990).
9) Nagai, A.: A New AND/OR Tree Searching Algorithm Using Proof Number and Disproof Number, *Workshop of the First International Conference on Computers and Games (CG'98)*, 40–45 (1998).
10) 長井歩: コンピュータ詰将棋の最新成果: 超長手数問題への挑戦, コンピュータ将棋協会誌, 13, 34–42 (2000).
11) Pearl, J.: Heuristics, *Addison-Wesley*, Reading (1985).
12) 脊尾昌宏: 共謀数を応用した詰め将棋プログラムについて, ゲーム・プログラミングワークショップ'95論文集, 128–137 (1995).
13) TETSU（加藤徹）: おもちゃ箱, CD-ROM (1999).
14) Zhang, W. and Korf, R. E.: Performance of Linear-Space Search Algorithms, *Artificial Intelligence*, 79, 2, 241–292 (1995).