連珠における追い詰め探索法の考察と試作

2000MT005 新井 勇植　2000MT007 浅野 弘揮　2000MT088 鈴木 徹
指導教員　真野 芳久

1　はじめに

今日、将棋プログラムが大きな進歩を遂げている。1990 年代に大きく日の目を浴び、詰将棋を解くプログラムを始めとする研究が一気に進んだ。今や将棋のプログラムは 4、5 段が集まるアマチュア将棋大会で優勝してしまう程にまで達した。

それに対して、連珠は他のゲームに比べるとコンピュータ分野での研究があまり進んでおらず、複雑さは他のゲームと同程度の割にはソフトの完成度も低いようである。

そこで我々は、コンピュータ連珠について研究を行うことにした。本研究では、発展する詰将棋に使われているテクニックから連珠に応用できるものがないか探し、連珠に適したシステムを作ることを目指す。そして対局の終盤における追い詰め、または詰連珠を解けるプログラムを試作する。

2　連珠について

2.1　対象とする問題

詰連珠の中で我々は四追い勝ち問題を取り上げる。攻めの手を四だけに限定した詰連珠で相手の防ぎ手が必ず 1 つに限定される。連珠終盤にはほとんどの場合に四追いを行う状況が出てくるので、四追い問題を解くことは他の問題を解くための基礎となる。

2.2　コンピュータ連珠の特徴

詰連珠をコンピュータで解かせる時の難しさの理由の一つとして候補手同士は影響しないことが多いということが挙げられる。影響しないということは、それらの手をどの順番で打っても結果は変わらないということであり、人にとっては単純そうでも計算量は爆発的に増える。

3　詰将棋における解決法調査

次に連珠終盤の追い詰めを解決するために、近い分野で使われている手法を調査し、連珠に応用することを考えた。詰将棋は詰連珠より一般に知られており、研究の成果が出て来ている。我々は特に詰将棋で使われている解決のためのいくつかの工夫が連珠終盤の追い詰め解決に適用可能であると考え、調査を行った。

3.1　詰将棋における問題解決

詰将棋と詰連珠は、駒と石の違いなどはもちろんだが、それぞれ固有の解決すべき問題を持っている。詰将棋では千日手の存在と無駄合いの問題がその代表的なものである。これらの問題は局面にハッシュ表を割り当て、そのハッシュ表の作成の仕方を工夫することで解決している。

表 1　詰将棋プログラムの成果の変遷

年	主な成果
1994 以前	13 手の問題を解ける程度
1994	T2 が 25 手前後のほとんどの問題を解く ITO が 661 手の問題を解く
1995	脊尾詰がこれまで解かれていなかった長手数問題を解く
1997	脊尾詰が発見されている最長手数の問題を解く
2001	長井のプログラムが発見されている 300 手以上の長手数問題を全て解く

3.2　詰将棋での成果

詰将棋プログラムの現状と成果を表 1 にまとめた。

注目すべきは 1997 年に脊尾詰が発見されている最長手数問題「ミクロコスモス」を解き、また最近の詰将棋プログラムの成果としては発見されている 300 手以上の全ての問題を解いているという点である [5]。

3.3　詰将棋での解決法

詰将棋プログラムの解答能力が向上していった理由に探索法の進歩が挙げられる。表 1 のプログラムについてその探索法を進歩の過程にそって述べる。深さ優先探索、最良優先探索の動きなどは 5 節で述べ、本節ではそれぞれのプログラムでの使われ方のみについて触れる。

■T2　T2 では反復深化を応用し、最大の深さを 2 ずつ増やし、詰みの有無を $\alpha\beta$ 法を用いて調べている。

T2 はこの探索法に加え、着手の順序付け、ハッシュの導入など、多くの工夫を行なっているが、深さ優先探索では手数が増えると探索木も爆発的に大きくなるので 27 手以上の問題はほとんど解けなかった。T2 は $\alpha\beta$ 法を使って詰将棋を解く限界であったようだ [4]。

■ITO　最良優先探索は深さ優先探索が正解の手数までの全ての攻め手、防ぎ手を調べる方法であることに対して、探索した全ての局面の中から最も「見込み」のありそうな局面から探索を行う方法である。この方法では全ての局面を「見込み」がありそうかといった観点から評価値を付ける必要がある。ITO の探索法では玉の自由度を評価値の基準としている。

■脊尾詰　脊尾詰では共謀数という概念を用いている。

共謀数とは AND/OR 木において、「ルート局面を解くために解かなければならない先端局面数」のことをいう [3]。脊尾詰は共謀数としてその局面を詰ますために詰まさなければならない先端局面数 (証明数) を使った。

脊尾詰はこの証明数と反復深化を組み合わせて使う。この方法は前述した反復深化のしきい値を証明数として

探索を行う方法である。このことによって探索の動きは深さ優先探索のように記憶領域の消費は少なくなり、最良優先探索にあった問題点を克服している。

■長井のプログラム　脊尾詰では証明数のみをしきい値としたが、長井のプログラムでは不詰を示すために示さなければならない先端局面数 (反証数) も用いた。反証数は詰将棋でいえば王手の掛け方の総数にあたる。長井のプログラムでは証明数と反証数を対等のしきい値として使うことで、脊尾詰よりさらによい結果が得られた。

3.4　連珠への応用の検討

我々は詰将棋プログラムの解決法から詰連珠についても同様の探索法を応用可能であろうと考え、この進歩にそって詰連珠プログラムを試作する。

4　連珠における代表的な問題と対策

見かけ上の分岐の回避

ゲーム木を探索する中で、ある局面から分岐した手順が何手か後で再び同一局面に合流することがある。これを見かけ上の分岐といい、どのゲームにも存在すると考えられる。連珠は、この見かけ上の分岐が起こることが多いと言われている。

見かけ上の分岐の回避を取り上げる理由は、計算量が大きく減るからである。異なる手順から同じ局面に合流したにもかかわらず別の局面と見なした場合、その先は全く同じ探索を何度も行うため、結果として計算量が無駄に増えてしまう。さらに分岐の中に分岐があると、組合せの効果でゲーム木は爆発的に広がる。この見かけ上の分岐を回避することによって計算量を減らし、メモリと時間を節約できると考えられる。

そこで見かけ上の分岐を回避する手法を 2 つ考えた。

局面記録法

第一は探索途中の全ての局面を記録しておき、再び同じ局面に遭遇したらそれ以上はすでに探索を終えたものとして、探索を打ち切る方法である。この方法は後向き枝刈りであり、計算量を減らしながらも $\alpha\beta$ 探索と同じ結果が期待できる。

局面記録法はメモリを多量に使用するためゲーム木が大きい問題に対しては適用できない可能性もある。これについては 5 節で考察する。

予測回避法

もう一方は、見かけ上の分岐が起こりそうな局面に遭遇したら予め計算して回避する方法である。

例を挙げると**図 1** の局面 S において追い手 A_1 と B_1 があり、仮に A_1 を打ち進める。この時、打ち進めた後の局面でも B_1 が着手可能であれば見かけ上の分岐が起こり得ると予測し、直前の着手によって新たに生まれた追い手 A_2 を打ち進める。これを繰り返し、直前の着手によって新たに追い手が生まれなくなるか、B_1 が打てなくなるまで打ち進める。同様に局面 S から B_1 も打ち進める。

この後は、見かけ上の分岐が起こっていることを確かめるために局面 S から追い手 A と B の組合せによって生まれる局面について対応する追い手が打てるかを調べ、**図 2** のゲーム木を作り上げる。

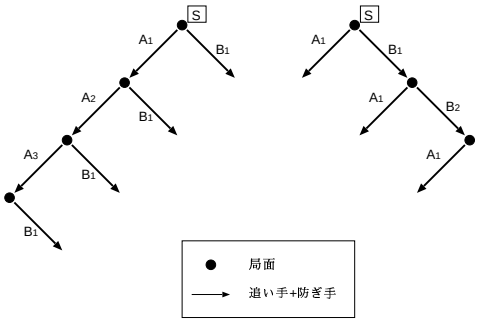

図 1　予測回避法ゲーム木モデル 1

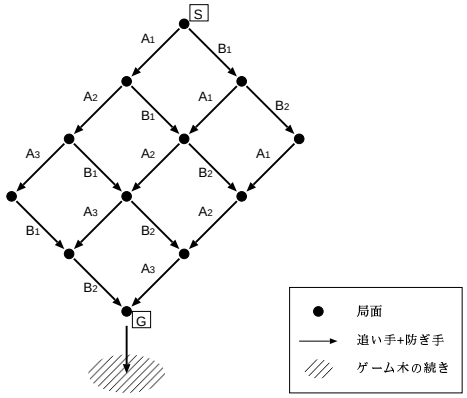

図 2　予測回避法ゲーム木モデル 2

この例では S から G までの間で見かけ上の分岐が起こっており、これを回避することで、回避しない時よりも局面 G 以降の計算量を「S から G までのゲーム木のたどり方の総数」分の 1 に減らすことができる。

予測回避法は見かけ上の分岐かどうかの判断を行うため、局面記録法に比べて時間がかかると思われる。

5　プログラム作成に向けての検討と全体設計

前節までで検討した内容を踏まえて、実現に必要な工夫や全体の設計について具体的に述べる。

5.1　試作プログラムに採用する方法の検討

5.1.1　局面表現と点価値表

局面は空白、黒、白の 3 値を 15×15 の配列に盤外を加えた、17×17 の配列に格納することで表現する。局面情報から各点が黒白それぞれの立場で、またそれぞれの方向 (縦、横、斜め 2 方向の 4 方向) において持つ価値を 18 種類の数値で表現し、$2 \times 15 \times 15 \times 4$ の配列に格納する。以下この表を点価値表と呼ぶ。点価値表の作成法については [1] を参照されたい。

点価値表は局面の変化に合わせて修正する。また、探索を行う際に候補手は点価値表から作成する。

5.1.2　禁手判断について

連珠における禁手判定は重要である。これは白番には相手に禁手を打たせるよう仕向けることで勝つという方法が戦術の一つになるためである。

禁手判断の解決は簡単な禁手判定については点価値表の作成する過程で行っている。しかし、三三や四四を判定することは難しい。これは三三の性質上、判断の構造が再帰的となるためである。我々のプログラムでは判断に負担がかかる点の判定は保留する方法を用いた。

5.2　探索法について

我々はゲーム木を探索する方法として深さ優先探索、最良優先探索を用いる事にした。

5.2.1　深さ優先探索

深さ優先探索は行きがけ順にゲーム木を走査して、勝ちとなる端末局面を探す。メモリをあまり使用しない方法ではある。しかし、$\alpha\beta$ 法を使って計算量を減らしているが最終的には探索時間が非常に長くなる。

5.2.2　最良優先探索

最良優先探索は探索中のノード群の中から静的評価で最も良いものを選んで探索する方法である。引き続く探索ノード間には何ら関係ないために各ノードは自局面の内容を持っている。

評価の高いノードから優先して評価しているので有利な場面を優先して探索することができる。結果的に解となりうる手順を見つけるための時間が少なくて済むが、全てのノードに対して局面を保存させるのでメモリ不足になることが多いという欠点を持つ。

この欠点を補うために、局面をハッシュテーブルに保存させ、各ノードはそのアドレスとそれに加える一手を情報として持たせることにした。リストからノードを取り出す場合に、アドレスにある局面とノードの持つ次の一手で評価する局面を作っている。これにより、保存する局面の数を減らし、メモリの消費を少なくしている。

5.3　ハッシュ導入について

前述したように見かけ上の分岐の回避における局面記録法や最良優先探索において、局面を記録する場合に問題となるのが大量のメモリを使うことである。

我々が作成する四追いプログラムを将来、追い詰めソフトの一つのモジュールとして使う時のことを考えても、できる限り使用メモリは少ない方が望ましい。そこでハッシュを用いて局面を bit 列で表現することにした。必要とするハッシュは以下の特徴を持っているべきである。

- 膨大な数になることが予想されるのでメモリ節約のため短い bit 列
- 異なる局面のハッシュ値はできる限り異なる値となる
- 処理の高速化のため短時間で計算できる

このような条件を踏まえて用いるハッシュ法は [2] を参考にしたものである。

その方法は全ての座標の黒石白石にあらかじめ 32bit の乱数を割り当てておき、石が置かれていれば該当する乱数を取り出し、全ての排他的論理和をとったものを局面のハッシュ値とするというものである。

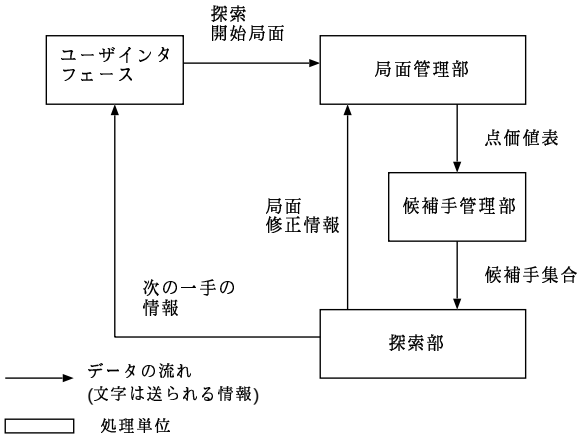

図 3　データの流れ

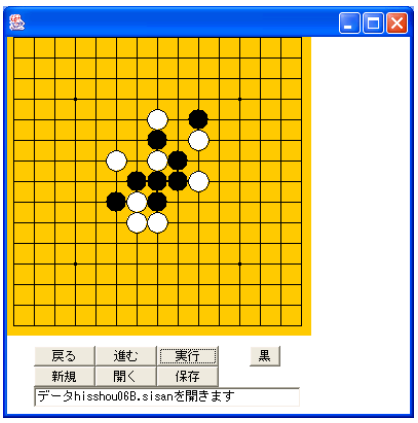

図 4　ユーザインタフェース実行画面

5.4　設計全体図

試作したプログラムの構造をデータの流れに注目して図 3 に示す。

ユーザインタフェース図 4 は問題局面の入力と探索結果の出力を行う。局面管理部では局面情報を保持するとともに局面情報から点価値表を生成する。候補手管理部では点価値表から候補手となる点の集合と、禁手となる点の集合を作成する。探索部では探索アルゴリズムにしたがって解を導き出す。

6　実験と考察

6.1　実験方法

詰問題の解は、最短手数で詰ませられることと、攻め手に対して相手がどんな受け方をしても詰ませられるということを示すことが暗黙の了解である。ただし我々が扱うのは四追い問題なので、相手の受け方は 1 通りしかない。そこで詰みまでが最も短い手筋を解とし、次の 1

表 2　見かけ上の分岐による組合せ爆発が顕著に起こっている例

詰問題タイトル	手数	見かけ上の分岐を回避しないときのノード数	見かけ上の分岐を回避したときのノード数
百選 52 問	125	2,453,207	1,669
苦楽部 18 問	67	784,073	1,044
苦楽部 4 問	49	1,240,081	2,264
苦楽部 6 問	27	872,998	1,857
百選 46 問	41	354,217	820

手を示す。よって 1 つの詰手筋を見つけても更に短い手筋がないかを調べることとする。

6.2　見かけ上の分岐の回避の手法について

見かけ上の分岐を回避する手法として局面記録法を用いて比較実験した。

表 2 は実際の詰問題を解く過程でノードとなる局面の生成数を表にしたものである。

この表では手順が異なる局面は別の局面として爆発的に増えた時の数と、見かけ上の分岐を回避し同一局面をまとめて 1 つとした時の数を比較している。特に差が激しい例だが、主に 40 手以上の問題について計算量が数百倍に増えていることがわかる。

以上の結果により、見かけ上の分岐を回避することは計算量に大きく効果があることがわかる。

6.3　局面記録法について

我々が用意した詰問題は、短手数のものから詰みと同時に盤面が石で全て埋まる完全問題と呼ばれるものや、組合せによって変化が多くなりそうであることが一目でわかるような問題など様々なものがある。

その中で、局面記録法で見かけ上の分岐を回避してなお、局面数の多かったものは最大約 45 万局面だった。

局面を記録する方法として用いたのは交点 1 つ (黒、白、空白) を 2bit で、1 列 (15 交点=30bit) を 4Byte で、1 局面 (15 列) を 60Byte で表現する方法である。この方法ならば 45 万局面で約 27MByte となり、メモリが足りなくなることはなかった。

ここでハッシュを使うと 1 局面あたり 32bit(4Byte) なので、45 万局面で約 1.8MByte となる。この程度ならばメモリが足りなくなることはなさそうである。

なお、違う局面でも同じハッシュ値となる「衝突」は理論上起こり得るが、用意した詰問題を解く過程では衝突は起こらなかった。理由としては四が打てる局面だけの探索なので衝突の条件を満たし難いためと推測できる。

局面記録法を用い見かけ上の分岐を回避すると、手数 x の詰問題について評価ノード数をおよそ $10^{-0.06x}$ 倍にできることがわかった。長手数になると見かけ上の分岐が頻繁に起こっているためと思われる。

6.4　証明数・反証数の導入について

ここでは、見かけ上の分岐は局面記録法で回避しつつ、深さ優先探索と証明数・反証数をしきい値にした最良優先探索について比較実験を行った。

その結果、評価ノード数を減らすことはできたが、時間については明確な効果は得られなかった。深さ優先探索に比べて 1 局面の点価値を調べる処理が多いため、と推測できる。

6.5　考察

我々の用意した問題全てにおいておよそ半分以下に減らすことができた。そして 40 手以上の問題に対しては 1% 以下に演算量を減らすことができた。

以上のように見かけ上の分岐についてはっきりとした効果が得られた。

7　まとめ

我々は連珠終盤の追い詰めや詰連珠、特に四追い問題について詰将棋からの応用を検討しつつ、プログラムを試作した。また、連珠の性質のひとつである見かけ上の分岐を回避することでプログラムの効率化を図り、結果として効率化に結びついたといえる。

今後の課題

我々が試作したプログラムに詰将棋プログラムから応用したものはごく一部であり、多くの応用範囲は残されている。また、それ以外にも実現すべき課題が残されている。その一部を以下に挙げる。

- 四だけでなく三、フクミ手、ミセ手を使う詰問題に対応できるよう拡張
- 点価値評価のさらなる高速化
- 変化別詰めを全て示せるような表示方法の工夫
- 連珠に適したハッシュ法の工夫
- 詰問題を自動作成するシステム

謝辞

本研究を進めるにあたり、粘り強く我々にご指導いただいた真野先生、また関係諸氏に深く感謝します。

参考文献

[1] 真野芳久：連珠プログラムと対抗戦, bit, Vol.7 No.4, pp.300-305 (1974.4).

[2] 野下浩平：詰将棋を解くプログラム T2, コンピュータ将棋の進歩, 共立出版 (1996).

[3] 脊尾昌宏:共謀数を用いた詰将棋の解法, コンピュータ将棋の進歩 2, 共立出版 (1998).

[4] 長井歩:詰将棋, 情報処理, Vol.44 No.9, pp.905-909 (2003.9).

[5] 長井歩：df-pn アルゴリズムと詰将棋を解くプログラムへの応用, アマ 4 段を超える - コンピュータ将棋の進歩 4 -, 共立出版 (2003).