ゲーム情報学研究の事例

将棋

なぜ将棋？

2002 年の秋に中東のバーレーンで行われたチェスの対局で、最強のチェスプレーヤーの一人であるクラムニクがコンピュータと引き分けた。使用されたコンピュータは、Pentium III 900MHz を 8 台搭載した汎用サーバである。当時チェス世界ランキング 1 位のカスパロフが IBM のディープブルーに敗れたのは 1997 年であるが、今回はディープブルーとは違って個人が使う PC とさほどかわらない性能のコンピュータである。チェスに関しては、コンピュータが人間のチャンピオンに追いついたといってよい。

現在のコンピュータ将棋の実力は、持ち時間の短い勝負であれば、だいたいアマチュア五段といったところである。アマチュアのトップまでもう少しではあるが、プロのトップまではまだ遠い。図 1 は今までのコンピュータ将棋の実力の伸びを大雑把に示したものである。今のペースのままだとプロのトップに並ぶのはまだ 10 年ほど先ということになる。

将棋は取った駒を再利用できるというルールがあるため、チェスよりも分岐数が多く難しいとはよくいわれることである。しかしよく考えてみると、分岐数が多いということは、そのゲームは人間にとっても難しくなっているのだから、相対的にコンピュータだけが弱くなるというのも変な話である。実のところ、将棋や囲碁が人間に追いついていないのは、分岐数が多いからというよりは、しらみつぶし型探索の効率化という点に研究の力点がおかれてきたからかもしれない。

人間が実際に直面するさまざまな知的作業は、探索問題として考えた場合、将棋や囲碁よりもはるかに難しい。いいかえれば、分岐数のはるかに大きな問題である。そのような問題に対しては、かつてチェスで成功したような、すべての可能な手を探索する手法（全幅探索）はほとんど無力であり、いかにして探索の範囲を限定するかということが非常に重要なテーマとなる。

その点将棋というゲームは、チェスよりも分岐数が多いため、全幅探索では強いプログラムを作ることは難しい。また同時に、囲碁のように、単純に探索の問題に帰着するのが困難なほど分岐数が多いというわけでもない。そのような点で、コンピュータ将棋は探索アルゴリズムを研究する上で適度な難しさの研究対象といえるだろう。

本稿では、コンピュータ将棋選手権で上位で活躍しているプログラムや、商品化されているようなプログラムがどのような工夫をしているのか、また、克服すべき課題は何なのかについて、実用的な側面と研究的な側面の両方から簡単に紹介する。

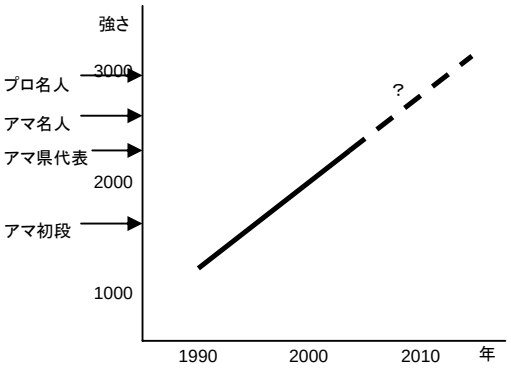

図 1　コンピュータ将棋の棋力[1]

探索

コンピュータ将棋で一般的に用いられている探索手法は、ミニマックス法とαβ枝刈りを利用したアルゴリズムである。図 2 のその例を示す。図中のノードは局面、エッジは指し手に対応する。一番上のノード（ルートノード）が探索を開始する局面である。探索を行う場合、普通は最初に何手先まで読むかを決めておき、ルートノードからその手数まで進んだノード（葉ノード）において評価関数によってその局面の優劣を数値化する。ミニマックス法とは、自分の手番では最も自分に有利な手を選び、相手の手番では自分にとって最も不利な手を選択することを前提として最善手を選択する手法である。αβ枝刈りとは、そのような選択をする場合に、不必要なノード展開を防ぐための方法である。実際には、さらに反復深化法といって、先読みの深さを徐々に深くしていくという方法がとられる。

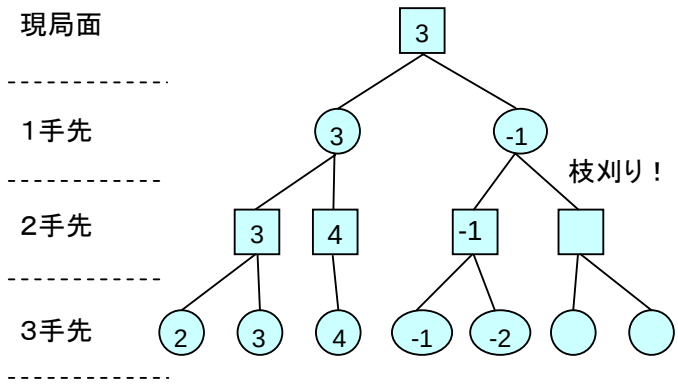

図 2　ミニマックス法とαβ枝刈り

先にも述べたように将棋は分岐因子が大きいため、全幅探索では強いプログラムを作ることは難しい。特に終盤になると持ち駒が増えてくるために、可能な指し手の数が爆発的に増えてしまうからである。人間の場合、初段程度の棋力しかなくても、終盤で 10 手先まで読むことは珍しくないが、全幅探索で終盤に 10 手先まで読むのは、高速なコンピュータを利用しても実用的な時間で探索を終えるのはほとんど不可能である。

そこで重要になるのが、読む必要のない展開を省略し、また読むべきところは通常よりも深く読ませるといった、探索範囲の制御方法である。これまで、探索範囲の制御方法としては主に、固定深さを基本として、それにさまざまなヒューリスティクスによる部分的な探索延長や前向き枝刈りを組み合わせたものが多かった。たとえば、王手がかかっている局面では探索範囲を一手延長するとか、あるいは逆に、残り探索深さが 5 手以内のときにはただで飛車を捨てる手は読まない、という具合である。

しかし最近このような、深さ打ち切りにヒューリスティクスを組み合わせる手法とは異なるアプローチを採用したプログラムもいくつか登場している。そこで、ここではそのような変り種の探索手法をいくつか紹介する。

局面の実現確率を利用した探索[2]

将棋を指す人で、先読みの範囲を手数で決める人はいないだろう。人間は、実際に起こりそうな展開であれば深く読むし、そうでない展開についてはほとんど読まない。そのような思考法をコンピュータで実現するために、探索範囲の決定に局面の実現確率を用いるアルゴリズムがある。

そのアルゴリズムでは、ルート局面（探索を開始する局面）の実現確率を１とし、指し手ごとにそれに対応する遷移確率を掛けていき、実現確率の値が閾値を下回った時点で探索打ち切りとする。

指し手の遷移確率はプロの実戦譜から推定されている。確率の高いカテゴリとしては、「駒得をしながら王手をかける手」や「駒得をしながら直前に動いた駒を取る手」などがある。したがって、このような手順を含む展開は深くまで読まれる。逆に、「駒をただで取られる手」などは確率が低いために、このような手が含まれる展開は浅いところで探索が打ち切られることになる。

２００２年に行われた世界コンピュータ将棋選手権では、この探索手法を利用したプログラムが優勝している。また、この手法を利用したことで、プログラムの強さが級位から２段程度まで上昇したという報告もある。

Alpha-beta-conspiracy search（ABC 探索）[3]

詰め将棋の世界では、証明数を利用した手法が大きな成功を収めた。それらのアルゴリズムのもとになった考え方が、McAllester による共謀数という考え方である。共謀数とは、ルート局面の評価値の安定性を、その値が変わるのに必要なノードの数で評価する。つまり、ルート局面がある一定値以上変化するために、より多くのノードの評価値が変わる必要がある場合には、その評価はより信頼できるというわけである。

共謀数の考え方をベースにした共謀深さという値を探索範囲の制御に利用するのが、Alpha-beta-conspiracy search (ABC search)と呼ばれる手法である。このアルゴリズムを利用すると、強制手（それ以外の手を指すと極端に不利になってしまう手）を含む手順が自然に深く読まれるようになる。

２００３年に行われた世界コンピュータ将棋選手権では、、このアルゴリズムを利用したプログラムが６位に入っている。

ProbCut [4]

ある局面の評価を考えたとき、浅い深さで探索した場合と深く探索した場合の評価結果に強い相関があることが知られている。このことを利用すると、実際に深い探索を行わなくても、浅い探索の結果からある程度どういう値になるかを予測することができる。

α β法を利用する場合、探索ノード中の局面ごとにウィンドウと呼ばれる評価値の範囲が設定される。これは、探索結果がこのウィンドウの範囲外になった場合、探索結果がルートの評価値に影響しないことを意味している。このことと、浅い探索の結果を利用して枝刈りを行うのが ProbCut と呼ばれる手法である。たとえば、現在探索中のノードのウィンドウが[100,200]であるとする。浅い探索の結果、この局面の評価値が 300 となった場合、深い探索を行なわずに上限の 200 という値を返したとしても、ほとんどの場合探索結果は変わらないということになる。もちろん、浅い探索を行うためのコストは余計にかかるため、その分の無駄は生じるが、それでも深い探索を省略できる効率化は大きい。

ProbCut は最強のオセロプログラムの１つであるロジステロで使われた手法である。この手法を将棋に適用した研究報告がいくつかあり、全幅探索ではその効果がある程度確かめられている。しかし、様々な探索延長や前向き枝刈りを行う実際の将棋プログラムと組み合わせたときの効果はまだ明らかではない。

評価関数

将棋の場合、ゲーム木の末端（どちらかの玉が詰んでいる状態）まで探索できることはかなり少ない。したがって、探索木の葉ノードでは、その局面がどちらがどれくらい優勢であるのかを評価関数によって数値で表現することになる。

評価関数は、将棋プログラムの性能に非常に大きく影響を与える部分であるが、体系的な設計方法はまだ確立されていない。そのため、実際の将棋プログラムの評価関数の設計は、

個々のプログラマの将棋の知識を手作業によって評価関数に変換しているというのが現状である。

	YSS7.0	激指	TD 法[5]
飛	1040	950	973
角	890	800	714
金	690	600	602
銀	640	600	499
桂	450	400	260
香	430	400	217
歩	100	100	100
竜	1300	1300	1568
馬	1150	1150	1304
成銀	670	600	187
成桂	640	600	387
成香	630	600	248
歩	420	600	549

表１　駒の価値

評価関数の主な要素は以下の３つであるといわれている。

（1）駒の損得

それぞれの駒に点数を割り当てて、自分側の駒と相手側の駒の総得点の差を評価する。表１に駒の価値の設定の例を示す。もちろんプログラムによってその値は微妙に異なっているが、駒の価値がおおむね駒のききの数[1]に比例しているところが面白い。

（2）駒の働き

駒の働きを評価する目的は遊び駒をなくすことである。働いている駒を正確に定義することは難しいが、自分の玉を守ることに役立っているか、相手の玉を攻めることに役立っていることを働いていると考えて、自玉および相玉から離れるほど点数が下がるようにする手法などが用いられる。

（3）玉の危険度

玉の危険度とは、自玉がどれだけ詰まされたり、必至をかけられたりする可能性があるかを示す指標で、自玉近傍の相手のききの数などを基準に評価することが多い。

昔の将棋プログラムでは、コンピュータが駒得だけを目指して遊び駒をつくって必敗形になる、ということがよくあったが、最近のプログラムでは、駒の働きや自玉の危険性もだいぶ正確に評価できるようになっている。

評価関数を設計する上で難しいのは、単に上記の３要素を合計すればよいというのではなく、序盤・中盤・終盤といった局面の進行度に応じて、それぞれの重要性が変わってくるところにある。

[1] 駒が動ける升目の数。例えば、歩は１、桂馬は２、銀は５。

評価関数の自動チューニング

普通、評価関数の中の様々なパラメータはプログラマが手作業で調整しているが、それらのパラメータを自動的に学習しようとする試みもある。TD 法と呼ばれる方法で、基本的なアイデアは、もし評価関数の性質がよければ、ある局面で探索した結果の評価値と、一手進めた局面で探索した結果の評価値は大きく変わることはないだろうという仮定である。極端な場合を考えてみよう。たとえばまったくでたらめな評価関数を用いたとすると、ある局面での評価と一手進めた局面での評価は多くの場合非常に異なった値になる。ところが、実際の将棋では、不利な局面から一手で有利な局面になったり、その逆ということはまれである。このことを利用すると、ある局面での評価結果と次の局面での評価結果のずれが少なくなるようにパラメータを調整すればよいということになる。

この手法を将棋の駒の価値の学習に利用した結果が、表１における一番右側の列である。その後、駒の価値だけでなく玉の危険度などのパラメータを学習した結果も報告されている。しかし、実際の将棋プログラムではパラメータの数が何百とあるため、すべてのパラメータを完全に自動学習するのは難しい。

終盤で必要な処理

詰みチェック

詰みの有無は勝敗に直結するため、終盤になると詰みのチェックすることが非常に重要になってくる。そのため多くのプログラムでは、ある局面で詰みがあるかどうかをチェックするアルゴリズムを指し将棋のアルゴリズムとは別に用意している。詰め将棋のためのアルゴリズムとしては、証明数、反証明数を利用した手法が、大きな成功を収めており、多くの将棋プログラムがこれらのアルゴリズムを採用している。そのため実戦で 30 手を超える詰み手順が出現することもしばしばである。

詰みチェックのルーチンは探索中に何度も呼ばれるため、詰みをみつけることももちろん重要であるが、詰みが存在しない局面において、詰まないということを少ない探索量で判定できるということも重要である。反証数を利用した方法でも、不詰めの判定にはそれなりに時間がかかるため、まだ改良の余地がありそうである。

必至

詰みの有無を見つけることに関してはコンピュータはすでに人間を超えている。しかし、実際の終盤では詰まして勝ちということも少なくないが、必至をかけて勝つ[2]ということも多い。

必至探索に関してはいくつかの手法が提案されているが、

[2] 必至をかけられた側は、その局面で相手玉を詰ますことができない限り負けになるため。

実際の将棋プログラムに組み込むためにはそのオーバーヘッドや探索量が重要である。いくら必至を見つけることができても、その処理に時間がかかってしまっては、トータルの強さの向上には結びつかない。そのため、完全な必至探索を行うアルゴリズムが実装されている将棋プログラムは非常に少ない。

必至の完全な探索ではないが、オーバーヘッドの非常に少ない手法として類似ハッシュを利用した方法がある[6]。実戦で必至を掛け損なう原因の多くは、詰めろ[3]をかけても相手からの連続王手によって、一見詰めろがはずれたように見えてしまうことにある。類似ハッシュを利用すると、似たような局面での詰みのチェックが高速に行えるため、連続王手後の局面で高速な詰みチェックを行うことで、この問題をかなり軽減することができる。

定跡データベース

序盤に関しては探索をしないで定跡に頼るという方法もある。プロの将棋の場合、最初の手は角道を開けるか飛車先を突く手かのほとんどどちらかであるが、これらの手が最善であることを探索によって決めさせるのは難しい。そこで、このような序盤に関しては、探索をせずに定跡データベースを利用して指し手を決定する方法がよく利用されている。

定跡データベースには、局面とその局面における指し手がハッシュテーブルなどの形式で保存されており、もし現在の局面がデータベース中の局面と一致していればその手を指すようにする。

定跡データベースの作成に関しては、定跡書などから手作業で打ち込んだり、プロの実戦譜から自動的に抽出などの方法がとられている。ただ、定跡書などで互角だといわれている局面が、コンピュータにとってみるとかなりどちらかに形勢が大きく傾いている場合も多く、その利用には注意が必要である。

水平線効果対策

10 年ぐらい前の将棋ソフトで遊んだことのある人ならおなじみかもしれない。水平線効果の典型的なパターンは、コンピュータが不利になると、突然無駄に駒を捨てだすという現象である。これは、駒を捨てることによって、不利な局面を探索範囲の外に押しやってしまうことが原因である。

例えば探索範囲が 2 手であるとしよう。いま自分の角がとられそうになっているとき、相手の飛車の頭に歩を打つとする、そうすると、「歩を打つ」「とり返す」で 2 手消費されるため、自分の角がとられる状況が探索範囲の外にでてしまうのである。

水平線効果の問題は、数年前までは非常に大きな問題とされていて、それぞれのソフトがその対策に工夫をこらしていた。ひとつの対策としては、水平線効果が疑われる手を指したときは、その手の探索を 2 手延長するというものである。つまり、駒を捨てても結局は損をするというところまで探索させるという方法である。ただ、この探索延長を行うと、非常に探索量が増えてしまうという問題がある。他の対策として類似ハッシュを利用する方法もある。

水平線効果はトータルでの探索が深くなると自然に発生の頻度が少なくなっていく。そのため、最近ではコンピュータの性能向上によって探索可能な量が増えてきたことで、個別的な水平線効果対策の重要性は昔に比べて低くなりつつある。

探索の高速化

コンピュータは一秒間に何局面ぐらい読んでいるのだろうか？　もちろんプログラムによって異なるが、現在最新のマシン（Pentium IV 3.0GHz や Athlon 3000 XP+）などで、数十万局面／秒というのが現状である。もちろん、探索が全幅探索であったり、評価関数を非常にシンプルにした場合は、より速くすることも可能である（プログラムによっては秒間 1 千万局面（！）近いものもある）。しかし現実的に強いプログラムを作ろうとすると、局面ごとにかなり複雑な処理を行わなくてはならないため、結果的にこれぐらいの速度になってしまう。

探索速度を向上させることは、ストレートに強さの向上に結びつくために、強いプログラムを作成する上で非常に重要である。コンピュータ将棋の棋力の向上は、ハードウェアの進歩によるコンピュータの計算速度向上によるところも大きい。

現状のコンピュータを利用してさらに速度を向上させる手法としては、複数プロセッサを利用した探索の並列化が考えられる。しかし、アルファベータ枝刈りを基本とした探索アルゴリズムは、逐次的に処理する場合に最も効率がよくなることから、効率的に並列化を行うことは簡単なことではない。

近年では、選手権に参加するプログラムにも、探索を並列化したプログラムがいくつか登場するようになっている。多くの場合、デュアルプロセッサを利用した並列化であるが、その場合で、約 1.5 倍程度の速度向上のようである。

探索量と強さの関係

探索量と強さの定量的な関係を明らかにすることは重要である。探索量を増やせば強くなることは経験的に知られているが、たとえば探索量を 2 倍にしたときにどれだけ強くなるかということの具体的な数値はまだよくわかっていない。

この関係をもっとも簡単に調べる方法は、自己対戦による方法である。つまり、同じプログラム同士を思考時間を変えて対戦させて勝率を調べればよい。この方法による勝率の変化に関してはいくつか報告があるが、おおむね、思考時間を 3 倍にすることで、1 級から 1 段程度棋力が向上するというこ

[3] 相手が防ぐ手を指さなければ詰ますことができる状態

とのようである。

ただし自己対戦による手法では、探索量が大きい側が探索量の小さい側の探索内容を完全に包含してしまうため、必要以上に勝率に差がついてしまう。そのため、自己対戦による勝率の上昇というのは、本来の強さよりもかなり過大に評価されている可能性が高い。

それに加えて、コンピュータチェスの世界では、探索量を増やしていくと、だんだんとその効果が少なくなっていく Diminishing return という現象が報告されているため、将棋でも同様なことが起きる可能性がある。

探索量と強さの関係は、今後の将棋プログラムの強さの変化を予測する上でも、また将棋ハードウェアなどを開発する上でも非常に重要な情報であるため、詳細な研究が期待される。

展望

コンピュータ将棋の実力は、ようやくプロのレベルまであと少しという段階に来た。今までの進歩のペースだと、プロのトップまでにはまだ 10 年近くかかることになるが、探索アルゴリズムなどの進歩によってはそれを大幅に短縮することも可能である。もし読者の中にいいアイデアを持っている人がいたら、ぜひコンピュータ将棋の世界に挑戦して欲しい。

参考文献

[1] Hiroyuki Iida, Makoto Sakuta, Jeff Rollason, Computer Shogi, Artificial Intelligence 134 (1-2), pp. 121-144, 2002

[2] Yoshimasa Tsuruoka, Daisaku Yokoyama and Takashi Chikayama, Game-tree Search Algorithm based on Realization Probability, ICGA Journal, Vol. 25, No. 3, pp. 145-152, 2002

[3] D. McAllester and D. Yuret, Alpha-Beta-Conspiracy Search, ICGA Journal, Vol. 25, No. 1, pp. 16-35, 2002

[4] Michael Buro, ProbCut: An Effective Selective Extension of the Alpha-Beta Algorithm, ICGA Journal, Vol. 18, No. 2, pp. 71-76, 1995

[5] Donald F. Beal and Martin C. Smith, First Results from Using Temporal Difference Learning in Shogi, In the Proceedings of Computers and Games (CG) 1998, pp 113-125, 1998

[6] 松原仁編著, コンピュータ将棋の進歩３, 共立出版, 2000

著者紹介

鶴岡慶雅（つるおか　よしまさ）

科学技術振興事業団 戦略的基礎研究推進事業（CREST）研究員

1997 年東京大学工学部電気工学科卒業。2002 年同大学院博士課程終了。工学博士。

自然言語処理に関する研究に従事。

tsuruoka@is.s.u-tokyo.ac.jp

http://www-tsujii.is.s.u-tokyo.ac.jp/~tsuruoka/atlab.html