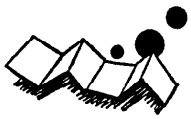
解　説



ゲームプレイングプログラムの近年の成果†

実　近　憲　昭††

1. まえがき

近年人工知能の研究は，問題解決，認知，言語理解等と急速にその分野を拡大しつつあるが，その原点においてゲーム（チェス）をする機械（計算機）に関する考察[3]が行われたことは決して偶然ではない．チェスのように古くから人知の限りを尽して戦い抜かれてきたゲームにおいては，ある種の思考法が文化遺産として伝えられ集積されてきており，これに対抗出来る機械こそ人工知能の名にふさわしいと考えられたからである．

ゲームはパズル等を含めると実に多種多様であり，その中から問題解決，学習，表現といった人工知能特有の研究テーマに最も適した，材料をその特別仕様に合せて，自由に選び出すことも可能である．また，ゲームのプログラムを作成することにより人間の心理的，あるいは論理的な思考のメカニズムの解明に役立てようとすることも行われている．

このように，ゲームの効用のスペクトルバンドは幅広いが，この解説で述べるゲームとしては，特に相手のあるものに絞ることにした．その理由は，パズル，トランプの Solitaire，ライフゲーム等々の一人ゲームは，その表現，構造共に全く千差万別であるため，ある分類に従って個別に採り上げて掘下げるか，あるいはゲームの次元よりもむしろ一般的な問題解決の枠組の中で論じた方がふさわしいと考えたからである．そして，個別に取扱うことは紙数の関係で不可能であるし，また，問題解決，あるいは探索の一般論の解説[1]は既に本誌の人工知能特集号でもとり上げられているので割愛させて載くことにした．

相手のあるゲームの利点の一つは，諸々の理論，もっともらしい仮説に基づいて書かれたプログラムも，人間との対決に於いて直ちにその真価が問われ，その結果は全くの専門外の人にさえ明白となることである．このフィードバックの早さは絶えず研究を刺激する．逆にゲームをする計算機の実力を A.I. 研究の進展具合を測定するベンチマークにしてもよいくらいである．しかしこのように相手のあるゲームに限定しても，その種類は相当なものであるが，研究目的でプログラムされたゲームとなるとかなり絞られてくる．

主なものを挙げると，完全情報ゲームではチェス，チェッカ，カラー，碁，将棋，連珠（五目並べ，三目並べ，立体四目並べ），ヘックス，Shannon のスイッチングゲーム，ニム（各種の変形）等々があり，不完全情報ゲームではポーカ，ブリッジ，ドミノ，バックギャモン，ブラックジャック等々がある．

これらのゲームは，研究目的に沿って慎重に選ばれ，その個有の役割を果している．例えばチェスはゲーム探索理論の開発，チェッカは学習機能の実現，碁はパタン認識に基づく推論，ポーカは駆引のメカニズムの解明，ヘックス，ニム等は必勝アルゴリズムの理論的解析の為の数学的モデルの提供[48]等々である．

2. 次の一手の決定

ゲームに限らず人工知能の問題一般についていえることであるが，ここでも知識と探索の二つの要素が基本的な役割を果す．

ゲームに関する知識としては，まずゲームの規則に関するものが含まれることはいうまでもないが，それ以外にも種々のレベルでの知識が存在する．例えば定跡に関する知識，短期的，局所的な戦術に関する知識，長期的，大局的な戦略に関する知識，形勢判断の材料およびその方法に関する知識，終盤における‘詰め’の知識等々枚挙にいとまがない．

人間はこれらの知識を個人的な経験，あるいは教科書を通じて，獲得してゆく．これらを知ることがゲームに勝つ為に明らかに有効であることは，経験を積んだプレイヤが等しく認めるところである．その様な知識の極限は，必勝法に関するものであろうが，これを

† Recent Development in Game Playing Programs by Noriaki SANECHIKA (Logical Systems Section Automatic Control Division, Electrotechnical Laboratory).

†† 電子技術総合研究所　制御部論理システム研究室

除けば，先に列挙した知識の例はすべて断片的であり，ある付帯条件のもとでのみ真となる性質のものがほとんどである．従ってこれらの条件付命題ともいうべき知識を十分活用するためには，ヨミ（探索）で補う必要が生じてくる．探索とは，適用しようとしている知識が真であるための条件が，その局面で成立しているか否かを確認する手続ともいえよう．

ところで知識が最低限に近ければ，完全探索を余儀なくされるし，知識が（質的に）増加してくれば，探索の部分はそれだけ減少する．必勝法の知識があれば最早探索は不要である．しかし現実のゲームにおいて必勝（最善）アルゴリズムを求めることは，一般に極度に難解な数学的問題であり，それどころか無意味でない（即ち完全探索と等価でない）必勝法は存在しないであろうことがほぼ断定出来るゲームのクラスさえある[2]．

現在，必勝法の知られているゲームのクラスは全体から見ればごく僅かな例しかない．いずれにせよ解かれたゲームは A.I. 研究の楽園から追放されたも同然である．

一般に探索に要する時間はヨミの深さの冪乗に比例して爆発的に増加するし，断片的な質の悪い知識を集めて数でこなそうとしても，たちまち記憶容量を越えてしまうだけでなく，そのデータベースの管理，情報検索に伴うオーバヘッドの増加も無視出来なくなってくる．要するに，時間と記憶空間の制約を考慮したシステムとしての最適設計の問題が即ゲームプレイングプログラムの問題でもある．この意味で計算機プログラムが次の一手にかける時間，あるいは一ゲーム当りの持時間に制限を設けることは本質的な条件のようである．

以上見てきたように知識（評価法，戦略とプラン）と探索は不即不離の関係にあるが便宜上，この項目ごとに分けて述べることにする．

2.1 ゲーム探索理論

以下，特に断らないかぎり2人零和完全情報ゲーム（例，チェス）を想定して話をする．説明の為に用語を整理しておく．

探索のモデルは通常図に示すような探索木で表わされる．各節点は一つの局面に相当し，それから出る枝は，その局面における合法手に相当する．ゲームは MAX と MIN の2人のプレイヤが参加する．頂上の根節点は探索開始局面であり，これを0レベルとする．偶レベルと奇レベルで2人のプレイヤの手番が入れ替る．

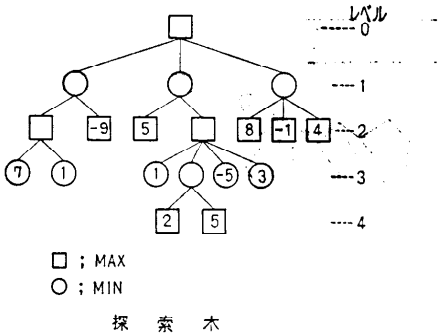


図 探索木

終節点はゲームの終局，あるいはそれ以下の探索を打切った局面に相当する．終節点 p に対してはある関数（静的局面評価関数）$S(\cdot)$ が定義されており，$S(p)$ は局面 p のある価値を表す．$S(p)$ の値の正負に応じてそれぞれ MAX, MIN 側に有利とする．

探索は根節点から始めて，一定の順序で各節点を展開して行き，終節点に至ってその値を比較しながら，自己側に有利な値を繰上げて行き根節点の値を決定した時点で一段落する．節点展開順序に従って，縦型 (depth first) と横型 (bredth first) と呼ばれる2つの探索型がある．前者は直系子孫節点優先であり，後者は兄弟節点優先である．

ゲームについてはほとんどの場合以下に述べる縦型のミニマクス法が採用されている．ミニマクス法とは，繰上げの過程で，MAX の手番ならば子節点の内最大の値をもつものを繰上げ，MIN の手番なら最小の値を繰上げるというものである．

筆者はこのミニマクス法の説明をすることにやや失意を感じざるを得ない．なぜなら Shannon はあの発端となったチェスをする機械[3] (1950) の中でこの原理について既に論じているのである．その後 α-β 法という改良が加えられはしたが，30年を経ようとしている現在，未だに釈迦ならぬミニマクス法の掌から脱出し得ないでいるからである．α-β* 法（より正確には α-β 手続きに基づくミニマクス法）は最初McCarthy によって考案されたらしい．（α-β 法の命名者でもある）Newell, Shaw 及び Simon 等[4]はゲームの木の刈込みについて発表したが，いわゆる branch-and-bound 形式のものであり，本来の α-β 法の説明は Hart 及び Edwards[5] によって成された．

このように比較的早くから知られていたにも拘ら

* α-β 史の小史が文献 6) §2 に記載されている．

ず，積極的にプログラムにも活用され，また理論的な解析が行われるようになったのは 60 年代の後半からであり，更に 70 年代に持越されている．α-β 法はミニマクス法と等価である．

等価の意味は，両者が同じ評価関数を用い，又同じ順序で後続節点が展開されるなら根節点への逆算値が同じであり，又繰上げ経験も一致するという意味である．異なる点は，ミニマクス法がすべての節点を調べているのに対し，α-β 法は探索の途中で得た情報をもとに，不必要な枝を刈込んでより小さな探索木で済ましていることである．不必要な枝の見分け方は次の通り．

今ある節点Pが MAX (MIN) の手番とし，その点への逆算値がある値 $\alpha(\beta)$ より小さく（大きく）ならないことが分っているものとする．一方節点Pの子孫節点で手番が MAX (MIN) である，ある節点Qの評価値xについて $x \leq \alpha\ (x \geq \beta)$ が確かめられた時点でQの兄弟節点は最早調べる必要なし（$\alpha(\beta)$ カット）とする．

α-β 法は各節点の展開順序によって，カットの生じ方に大きな差が出てくる．理想的な展開順序とは，各節点への逆節値が，幸運にも，常にその長男節点から繰上げられる場合である．均質木（最終レベルD以外のすべての節点から出る枝の数（分岐幅）が一定値Bをとる）の終節点の個数は B^D であるが，α-β 法が最大限に効果を発揮した場合約 $2B^{D/2}$ 個*の終節点を評価するだけでよいことが示されている[7]．

これはミニマクス法と比べて，同じ仕事をするのに約半分の深さのヨミを行うだけでよいことを意味する．

α-β 法はある意味では最良のアルゴリズムであることが示されている[6]．ここである意味とは，終節点の値を根節点に繰上げる様などんなアルゴリズムよりも，α-β 法はより少ない数の終節点を評価するだけでよい（但し，理想的な展開順序のもとで）という意味である．

一方理想的な場合だけでなく，平均的な α-β 法の刈込効果についても，限定された状況設定のもとではあるが，理論的解析が行われている[6),8)~10)]．均質木を扱い，その終節点の値を，ある確率分布に従ってランダムに与えたものについて α-β 法を適用して評価すべき終節点の個数の期待値を求めるものである．

Knuth 及び Moore[6] は深さ2における刈込（浅い刈込）のみ生ずるものとして，等価的な分岐幅のオーダーが $\Theta(B/\log B)$ であることを示し，深い刈込は2次的な効果しかないことを予測したが，これは，Baudet[10] によって確かめられている．

2.2 評価法

α-β 法の枠組の中で，次の一手決定のために必要となる評価すべき要素を列挙すると，

i) 探索打切条件の評価
ii) 終節点の評価（静的評価関数）
iii) ある節点の子節点の展開順序決定の為の評価
iv) ある節点の子節点の内将来性のないものを切落す為の評価（前向き枝刈**）

i) 打切条件：探索打切はゲームの終了時，あるいは予め定めた深さに達した時点で打切るのが基本である．（後述する前向き枝刈も探索打切の一つである．）以上の様な形式的なものと異なる条件として，打切時点で局面が'静か'であることが必要である．

終節点は静的評価関数で評価されるが，局面が静かでないと正しい評価が期待出来ない．例えば，チェスでは駒当りやチェックの存在する局面は不安定とみなされ，予定した深さに達しても特別に探索を延長して静かになるまで続けられるのが普通である．

しかし，局面の真の静けさ（さし迫った危険，急激な変化のないこと）を判定することはそれほど容易ではなく，むしろ最も扱い難い問題の一つに入っている．静的評価関数と打切条件の相互干渉で水平線効果[11] (horizontal effect) と呼ばれる不測の評価誤差が生じることが指摘されている．

一つは定められた深さの探索範囲内でのある不利な評価を避けて一時逃れの手を打ち，そのためかえって後で，より大きな不利を受けるものであり，もう一つは，同じく探索範囲内での小利に満足して折角の好チャンスを逃してしまう現象である．静けさ判定にあまりこるとその為に時間を費やすし（このレベルでのわずかな時間増は大きく積算される），又あまりに敏感な検出は，なかなか探索を打切らせない．別に目標指向に基づく静けさ判定法もある．これは根節点で存在したある目標が，終節点でも存続していれば，打切らずにその線に沿って追求する考えである．

ii) 静的局面評価関数：探索打切条件の満された時点でその局面（終節点）の評価を，それ以上の探索をしないで行う．その為に局面の優劣に強い相関があると思われるいくつかの評価因子を設定する．

---

* 正確には $2B^{D/2}-1$ (D: 偶数)，$B^{(D+1)/2}+B^{(D-1)/2}-1$ (D: 奇数)

** α-β 法による枝刈を後向き枝刈と呼ぶ場合がある．

評価因子に対する条件は，容易に検出可能で，その査定に手間取らないことも又重要となる．それらの因子はそれぞれ一つの数値で表現され，因子相互間の相対的価値によって係数を乗じ加算した線形多項式が一般に用いられる．ゲームの種類によっては，その進向段階に応じて，様相が大きく変動する場合があるが，この時その様相に応じて評価関数を替える工夫もなされている．線形多項式でない例としては，後述する(§4) Samuel[22] のチェッカプログラムで用いられた特徴表 (signature table) 法がある．これは一種の（多段）多値関数である．

ⅲ）探索順序決定：これは，α-β 法による枝刈が最も多く生じる，理想的探索順序になるべく近づける目的で行われる．固定順序付と動的順序付の二通りに分類される．前者は，一度探索順序を決めたら，後でこれを変更しない方式であり，後者は，一旦決めた後でも，その後の探索の途中で得られた情報に基づいて，順序変更もあり得るという方式である．固定順序付は，浅い探索によって行われる．例えば，展開しようとしている節点の子節点を（終節点で用いる）静的局面関数で評価して，有利な順に配列する．

あるいは，静的局面評価関数を流用する代りに，別途用意した手の評価関数を用いる場合もある．固定順序付は，浅い探索で決定した着手の価値と，終節点から繰上げられた逆算値の関には正の相関があるという仮定に基づいている．しかし静的局面評価関数，あるいは手の評価があまり正確でなければ相関度も弱まり，十分な枝刈効果が期待出来ない．この欠点を是正する為に，動的順序付が考えられた．2つ例を挙げる．

一つは Slagle 及び Dixon[7] によって考察された A-B 型 α-β 法である*．Slagle 等はカラー・プログラムを用いて，ミニマクス法，固定順序 α-β 法，A-B 型 α-β 法の比較を行っているが，固定順序型はミニマクスに比してかなり改善されたが，A-B 型は固定順序型をやや上回るにとどまる．これは α-β 法の理論的限界に近づいた為としている．

もう一つの動的順序付の例は反復型 α-β 法 (iterative or staged α-β pruning) と呼ばれるものである．この手法は Slate, Atkin[14]; Gillogly[15]; KAISSA（チェスプログラム名）を開発したソビエトチームの三者の間で独立に発見された．これは深さ D の探索を一度に実行しないで，i レベルまでの α-β 探索を行い，そこで得られた情報に基づいて新たな順序づけを行い，1レベル進めた $i+1$ までの探索を行うと言う手順を D 回反復する．

この手法の優秀性は，現在世界最強と思われるチェスプログラムCHESS 4.7 (North-Western 大学) あるいはKAISSA等に採入れられて確かめられている．この方法のもう一つの利点は，計算時間を制御しやすいことである．即ち制限時間がきた時点で，最後に実行したレベルでの結果を出力すればよいからである．

ⅳ）前向き枝刈：探索の過程で，局面の手の内，見込みのなさそうな枝を切捨ててしまう方式である．これは人間の探索の仕方と一致しており，常識的と思われる．しかし見込のないことを判定することは ⅰ）で述べた静けさ判定と同じく，あるいはそれ以上にゲームに関する深い知識を必要とする．いい加減な適用は，いわゆる勝手ヨミに繋がる．先細り n 次前向き枝刈，楽観悲観型前向き枝刈**等の種々の工夫がなされたが本質的改良ではない．前向き枝刈については別な問題点がある．それは唯一度の検査でその枝の運命を完全に決めてしまう点である．この点を改良し，探索途中で仕入れた知識を援用して，柔軟な枝刈を目指す試みが行われている．

その一つは，Harris[16]による Band width seach である．これは，アルゴリズム A*[13),17)]の一変種で，発見的評価関数が誤りを犯すものとして，その変動幅の上限，下限を与え，ミニマクス法に適用して，その収束性と共に，収束値の誤差範囲が，上の変動幅で押えられることを示している．

この方法は，普通のゲームプログラムと異なって，自己の手番のときだけ探索をするのではなく，常に探索を続行しており，自分の手番が来たときに一時探索を中止して，その時点での最善手を出力する．この方法とやや類似した方法，アルゴリズムB* が Berliner[18] により考えられた．

これは各節点が生成されるとき，その点の上限値，下限値を推定して割当てる．又節点 P（上限値 u，下限値 l）を展開して得た子節点 P_i，$1\leq i\leq d$ の上，下限値が u_i, l_i であったとすると，節点 P の値は次の修正を受ける．P の手番が MAX, MIN に応じてそれぞれ $u\leftarrow\max[u, \max_i u_i]$，$l\leftarrow\max[l, \max_i l_i]$；$u\leftarrow\min[u, \min_i u_i]$，$l\leftarrow\min[l, \min_i l_i]$．また P の先祖節点に対しても上と同様にして繰上げて行く．次に展開すべき節点は，根節点から始めて，双方のプレイヤが自己に有利な値を持つ経路を辿ればよい．この方

* 文献 12) 訳本 p. 20
** 文献 12) 訳本 p. 21 参照

法の収束性と共に，アルゴリズム A* よりは悪くはならないことが示されている．

以上の2方法は有望であるが，やや理論的な定式化にとどまり，誤差の定め方，上，下限値の定め方などに問題が残りそうである．

2.3 戦略とプラン

ミニマクス型の探索法の最大の欠点は，長期的な構想に基づく戦略，目標の設定，目標に沿った探索等がうまく出来ないことである．これは定められた深さでいきなり静的評価関数によって，局勢を数値に変換してしまう形式をとるかぎり本質的に免がれないことかも知れない．

相手（人間）が計算機と同じ評価関数を用いているとはとても考えられないし，ミニマクス的思考そのものにも問題がある（勝負手，駆け引き等の欠除）．最も致命的なのは計算機は自分の手番の度ごとに，前回より2手ばかり進行した局面について，毎回独立に探索木を作っているという事実である．前回のヨミ筋が継承されることにかなり偶然性を伴ってくる．この引継の悪さをいくらかでも改善しようとするのが静けさの判定（§2.2 i）の精密化の試みであった．

1970年以降におけるチェスプログラムの目覚ましい活躍ぶり（§4. 参照）は実は，超高性能大型計算機の能力と結びついた全幅探索型の α-β 法のおかげである（現在最強のものはエキスパートクラスに達している）．しかし最高レベルであるマスタ級になるには今一つ壁を破る必要がある．その為には本節のテーマである戦略とプランに基づいた戦斗法を実現することだといわれている[11]．（§2.2 iv）で述べた Harris の bandbreath 法及び Berliner のアルゴリズム B* は，無理な打切りは行わず，連続的にヨミ筋を追うという点で，同じテーマを扱ったものと考えてよい．まともに取組んだ例としては Pitrat[19] のチェスの手発見プログラムがある．これは完全なプレイをするのではなく，与えられた局面から有効手を見い出す為のものである．

これは初期局面から，いくつかのプランを作成し，その実行を試みる．プランが失敗したときのみ再度解析を行い，その失敗原因をつきとめ，それを排除する為のプランを作成するという手順を繰返す．プラン作成の動機となる目標が，駒取り，両当り，王の攻撃，ポーンの昇進のわずか4つから出発している．従ってヨミ筋はかなり限定されるか少なくとも一貫性は保証される．

実際のチェスプレイを考慮した，より実際的な方法が Frey, Atkin[20] によって提案されている．これは条件付評価関数法と呼ばれるもので，全幅探索（即ち前向き枝刈を行わない）α-β 法の利点を生かしながら，目標指向の考えをとり入れる．即ち，いくつかの（長期的，短期的）目標の特徴をよく摑まえた基本的なパタンを階層的に用意しておき，階層的なパタン整合を行い，その結果得られた目標指向成分を通常の静的局面評価関数に重ね合せて用いる．

このシステムの成否はデータベースとして用いる鍵パタンの整備次第にかかっているが，開発途中で，不完全であっても，静的評価関数で最低線は保証されているので実際的といえる．

3. 学　習

今までにゲームプログラムに於て扱われた学習は，いわゆる暗記学習，条件反射的学習の様なものが多くみられる．真の意味での学習は，認識，抽象，汎化というプロセスに深く関係しているので最も困難な，しかし挑戦に値するテーマの一つでもある．

学習をテーマとした報告をいくつか拾ってみると，Samuel[21] のチェッカ[22]，Pitrat[23] のチェス，Findler[24] のポーカ，Smith[25] のドミノ，Elcock, Murray[26] の五目並べ，Remus[27] の碁，等がある．これらは完全情報ゲームと不完全情報ゲームに二分される．その探索木の形状が，大雑把にいって，前者が細く深いのに比して，後者は広く浅い．従って学習形態も，前者が探索過程を反映して行われるのに比して，後者はある条件が満されればある決定を下すという一過性の決定構造の学習形態となりやすい．又学習するプログラムのチェックポイントとしては次の様なものが考えられる．（a）何を学ぶか，（b）何から学ぶか（教師，教材，経験），（c）学習の仕方，（d）学習結果の記憶形態，（e）学習効果等である．

先に挙げた学習プログラムについて以上の観点から簡単に眺めてみよう．

• Samuel のチェッカ：初期のもの[21]とその後の改良版[22]の2通りがあるが学習形態としては同じである．（a）静的局面評価関数（初版は線形多項式，改良版は特徴表（signature table）を学習対象とした．（b）教材はマスタプレイヤ同志の実戦譜（25 万局面）．（c）教材局面の後続局面をすべて求め，その内実際に打たれた手に相当する後続局面を模範とする．

初版の場合：線形多項式を構成する各評価因子の重

み係数を$C=(L-H)/(L+H)$で計算する．但し$H(L)$は学習開始後現時点に至るまでに，その評価因子の価が，模範手よりも高い（低い）値を示した手の数を累積したものである．

改良版の場合：この場合の評価関数は多段結合された多値関数と考えられる．下式は2段結合の例である．

$f_0(f_1(p_1, p_2), f_2(p_2, p_3), f_3(p_1, p_3))$．ここで変数$p_1$等は各評価因子に相当する．但しその値は正規化してある．各f_i等は特徴表と呼ばれる表で与えられる．特徴表の値$f(p_1, \cdots, p_n)$は$C=(A-D)/(A+D)$で与える．ここでA, Dは，表の各欄ごとに保存されており，$(p_1, p_2, \cdots, p_n)$の値が模範手のそれと一致するときは$D \leftarrow D$，$A \leftarrow A+$(模範手以外の手の数)とし，一致しないときは$D \leftarrow D+1$，$A \leftarrow A$とする．

（d） 初版：線型多項式の重み係数 改良版：特徴表の各欄の値の形で記憶する．

（e） Griffith[28] は特徴法の最初の考案者でもあるがこれとよく似た，しかし，より単純な構造を有する学習法，手相法 (Move-phase table) を提案し，同じくチェッカプログラムを用いて，線形多項式，特徴表法，手相表の三者について比較し，前者に対する後二者の優秀性を確認している．

・Smith[25] のドミノ

（a） 戦略を学習する．

（b） 自分自身との実戦譜を教材とする．

（c） 局面の状態を表す7個のパラメタを導入し，それに対して与えられた4つの純粋戦略に対する混合比を特徴表法を用いて（即ち，7個のパラメタを評価因子と見る．特徴表は1レベルのみ）学習する．

（d） 特徴表の値として記憶する．

（e） チャンピオン級の技量を獲得．

・Pitrat のチェス

（a） ‘ある特徴が盤上に存在すれば，それに付随した手（の集合）を考えよ’と言う知識片を学習する．この知識片はプランを立てるのに用いられる．

（b） 実戦譜を教材とする．

（c） 実戦譜の中から，ある得（このプログラムでは駒得のみ考慮）をもたらす手の組合せを切出し，これが成功した理由を理解する．ここで手の組合せの最初の手を根節点とする探索木を作りミニマクス手続きで根節点の値が正となった場合理解したと考える．成功した場合，この探索木を一定のルールに従って簡略化し一般化する．

（d） （c）で得られた抽象化された木を一つの知識とみなし，これを集積する．

・Remus[27] の碁

（a） 良い手（を含む局面パタン）を記憶学習する．

（b） 自戦譜を教材とする．

（c） 自戦譜の各手を，定められた規則に基づいて，良，悪，普通に分類し，これに応じて辞書 (lexicon) 中の対応する手の成功度を修正する．もし良と判定された手が含まれていなければ新規登録する．

（d） 辞書 (lexicon) 中に良い手を含むパタンを記録する．成功度が基準以下に下れば抹消する．

・Elcock, Murray[26] の五目並べ

（a） 手の評価法を学習する．即ち，各手は勝敗に関係した，n個の評価因子で記述されており，各因子がとる値により定まるn組数（記述子）の優先順位を学習する．

（b） 自戦譜を教材とする．

（c） 学習プログラムはサブゴール（必勝手と思われる手の記述子）のリストを有している．一方実戦譜を逆に辿り，敗着をつきとめ，その直後の必勝手の記述子をサブゴールリストに強さの順序を考慮して加える．

（d） サブゴールリストの形で記憶される．実際のプレイはこのリストを用いて行う．

・Findler[24] のポーカ：単に純粋なポーカゲームの状況だけでなく賭の履歴，相手の性格等まで含めた環境を考慮して種々な角度から学習するプログラムを作り実験している．その内比較的中心にあると思われる適切に相手を評価する学習プログラム (AEO) について説明する．

（a） 敵プレイヤの性格を学習する．

（b） ゲームの履歴，即ち，ドロー以降の賭の仕方と最後の対決時の手札のさらしに至るまでのすべての情報．

（c） 各敵プレイヤの性格に関する統計資料のデータベースを次の様に改修する．すべての可能な手札をその生起確率に基づいていくつかの(20)クラスに分割する．今手札クラスiから集められた．プレイヤjのある評価因子を$S(i, j)$と

し，その平均及び標準偏差をそれぞれ $m(i,j)$, $s(i,j)$ とする．ここで評価因子としては（実際の賭高）／（公正賭高），ポットの中の賭高，最後の賭高等々が考えられている．$\hat{S}$ を (b) の情報によって得た量とし，

$$|\hat{S}-m(i,j)| \leq w(i)\cdot s(j)$$

を満すか否かを調べる．ここで $w(i)$ は重みとし学習の始め一定値 (0.1) にセットする．条件が成立した手札 i に対しては $\hat{S}$ は正しい推定と見なす．手札をさらした後この推定が当った i に対しては $w(i)$ を 10% 増し，はずれなら 10% 減ずる．各人に対する評価因子の適切な選択も又別な手順（ベイジアン戦略）によって行われる．

(d) 各人に対する統計量 $m(i,j)$, $s(i,j)$, $w(i)$ の中に記憶される．

(e) 相手に対する推定した手札の強さと現実との差は平均して 2 (20 個の手札クラスの順位に於て）であり，良い戦略を持っているプレイヤに強く，ランダムなプレイヤには弱い傾向を持つと報告している．

4. 個別ゲームの成果

まえがきで述べたように個々のゲームはそれぞれ固有のセールスポイントを持っている．しかし，その中で特にチェスは際立った役割を果しており，話題も豊富なので便宜上チェスとそれ以外のゲームの二つに分けて足取を辿ることにする．

4.1 チェス

Shannon[3] (1950) は，チェスプログラムについて論じ，三つのタイプ A, B, C に分けた．

タイプAは，静的局面評価関数を用い，ミニマクス法により定められた深さまで完全探索を行ういわばミニマクス法の原型である．タイプBは，前向き枝刈を行うものであり，タイプCは戦略，プランを用いるものであった．そして，タイプAは計算能力の点で到底無理とし，タイプCは将来の夢であり，結局タイプBを最も有望とした．Kister 等[29]はタイプAのチェスプログラム (6×6 盤)，Bernstein 等[30]はタイプB，Newell, Simon 及び Shaw[4] はタイプCのチェスプログラムを作った．

以上の三つは最初の試みとして評価されるが，その技量は全く低いものであった．Kotok が McCarthy の指導下で作ったプログラムは普通のレベルでプレイした．後にこのプログラムは，Adelson-Velskii 等のモスクワチームのチームと初のコンピュータ同志の対戦（郵便通信）を行ったが負けている．1967 年 Greenblatt 等[31]の開発した B タイプのプログラム (Mac Hack 6) は人間の参加するアマチュアチェストーナメント戦で D 級*優勝した（総合実力は C 級にランクされている．）

このプログラムはタイプBの一つの頂点に達したと考えられている．それはその後この方向（タイプB）に沿った懸命な努力にも拘らず，大した進歩が見られなかった為である．これは前向枝刈を正確に行う為には意外なほど多量の '知識' を必要としたからであり，同時に生じる良い手の切落しは C 級以上の向上を阻んだものと思われる．Gilloglyl[15] のチェスプログラム TECH (technology) は実にAタイプに属し，固定長探索（中盤で約 5 手，但し駒当り，チェックの時のみ延長），評価関数は駒の損得を主体とした最も簡単なものである．一見時代に逆行する様であるが，訳の分らない前向枝刈に時間を費やすより，いっそのこと全幅探索とし，その代り，Shannon の時代には未だ発見されていなかった α-β 法の力を十分に活用しようとするものである．TECH は又相手の考慮時間中にも探索をすることを試みた最初の例である．1970 年 10 月に着手された TECH は 1971 年 3 月には 1968 年版の Greenblatt のプログラムと対戦し白番勝，黒番引分の成績を収め，8 月には ACM 後援第 2 回コンピュタチェスチャンピオン戦で 2 位（1 位は CHESS 3.5) となった．しかし，TECH の目的は必ずしも強くなることではなくて，単純な構造のもとで規格化されたチェスプログラムの作成であり，その為探索木の深さと展開節点数，各構成部の所要時間の統計をとるなどの実験を行っている．

さて，1970 年以降のコンピュータチェスの活動は目覚ましい．その舞台を提供したのは，1970 年を第 1 回とし，以降毎年行われている ACM 主催の米国コンピュータチェス選手権大会であり，もう一つは，1974 年に始まった世界コンピュータチェス選手権大会（3 年ごと開催，IFIP 主催）である．CHESS X. X は Slate 及び Atkin 等 (Northwestern 大学) によって書かれたチェスプログラムで，最初 1968 年に始まり，その後絶えず改訂を重ねて，最新版は，CHESS 4.7 に至っている．このプログラムは 1974 年を除いて，米国

* チェスの技量によって 8 段階に等級づけられる．グランドマスター，マスター，エキスパート，A, B, C, D, E

大会ですべて優勝している*. 又第1回世界大会(1974, ストックホルム) のチャンピオンの座はKAISSA (モスクワチーム) が獲得し, CHESS 4.0 は2位であった. しかし同大会で両者は顔を合せることがなく, 後で特別に行われた試合では引分けに終っている. 第2回世界大会 (1977, トロント) ではCHESS 4.6は見事優勝し, KAISSA は DUCHESS (Truscott (Duke 大), Wright (Durhan NC)) と共に2位の座を分けた**. CHESS X.X シリーズの成功の秘密は TECH と同じく全幅探索方式を採用したことであり, α-β 法の効果を最大限に引出す為に種々の工夫がなされた. その中には先に述べた (§2.2ⅲ)) 反復型 α-β 法も含まれる. もう一つのポイントは CHESS 4.4 (1975) 以来超高速大型計算機 (CDC CYBER 170 シリーズ) に乗換えたことである. KAISSA も又 A タイプに属し, CHESS と同じ思想に基づいており, 類似法*** (Methode of Analogy) その他の思想に基づき[32], 探索量の削減をはかっている. 人間との対戦に於ても進境は著しい. CHESS 4.5 は人間の参加するオープンチェス選手権大会 (1976, ミネソタ) に於て数人のエキスパートも交る A クラスで優勝(5勝1負)した. CHESS 4.6 (全米チェスプレイヤの 99.5% に勝つといわれる) は第2回コンピュータチェス世界選手権大会で, David Levey(グランドマスタ)と早指形式**** ではあるが対戦して勝っている. Leveyに関してはエピソードがある[33]. 第4回 Machine Intelligence Workshop (1968) に出席した Levey (当時エキスパート, スコットランドチャンピオン) は, その時 '10年以内にはチェスチャンピオンはコンピュータがとるであろう' という J. McCarthy, D. Michie 等の予想に同意せず, 彼自身が防衛することに1,250ポンド賭けた. 1977年の終りと 1978年の始めに Levey は CHESS 4.6, DUCHESS, KAISSA, Greenblattのプログラム***** と対戦してすべて破っている. そして愈愈, 賭の期限も尽きようとする1978年8月26日(土)に CHESS 4.7 (CDC CYBER 176/ランク2,030点(早指, 2,450点)) との間に最後の決戦が行われた. 結果は3勝1敗1引分でLeveyの勝に終った. Leveyは5年以内に彼を負すプログラムに5,000ドル提供することを約して, 再度挑戦に出ている.

第2回世界大会で始めてマイクロコンピュータ (INTEL 8080) によるチェスプログラム (A. Epsteinの作品) が参加し, フロックながら一勝した. 猛威を振うスーパーコンピュータ****** の中では今の所分が悪い. しかし, 昨年の1978年の12月に行われた第9回ACM主催のチェス大会[51]では, 2つのマイコンプログラム, SARGON II (Spacklen夫妻, 米), MIKE (M. Johnson, 英) が参加し, それぞれ3位, 8位にくい込んでおり, 今後とも急速なマイクロコンピュータチェスの進出が予想される.

4.2 チェス以外のゲーム

碁: チェスに於て一定の成果を上げた方法, 全幅探索方式は, いかにその陰で数多くの巧妙な工夫がなされているとはいえ, 最早碁には通用しないであろう. ゲームの複雑度を仮に可能なプレイの生じ方の回数で表すと, チェスと碁ではそれぞれ控えめにみて 5^{50}, 10^{100} 通りはあり, 又仮に計算機の速度が毎年2倍ずつ向上したとしても, 碁が現在のチェスの水準に到達するには200年を越える歳月を要することになる. すなわち, 碁に関する知識を最大限に活用して探索木の展開は極力避け, 戦略とプランの考えを導入することが本質的に必要とされる. その意味でチェス以後の研究課題にとりあげるゲームとして, 碁は有望株のトップにあげることが出来よう.

Remus[27] の学習する碁 (§3 参照) は 11×11 盤について行われたものであるが, 辞書の容量 (1,000手(1万5千局面)), 辞書の '改修基準' にあまり碁の知識が注入されていない点等を考慮すると, やや原理的な構想であって, 現実の碁には程遠い. Thorp 及び Walden[34] は, ややあいまいな碁のルールの一部を修正して, 一般の $N \times M$ 盤について理論的な解析を試み, 特に $1 \times N$ 盤の勝敗の結果について結論を得ている. 最初に完全なプレイをするプログラムは, Zobrist[35] によって書かれた. 盤上の局面を認識する手段として, ポテンシャル場の考えを導入している. これは石が周囲に及ぼす影響はその石に近い程大きいという一般的傾向の良い近似となっている. 又頻度の高い, 重要と思われる局所的な石の形 85 種をテンプ

---

* 1974年の1位は RIBBET (Waterboo 大). 1977年の第8回米国大会では DUCHESS と一位を分けた.

** KAISSA に虫がいたと云われる.

*** 探索の過程で以前に検出した類似の局面が再現し, その結果が不利であることがはっきりしている場合は省略する手法.

**** 1手5秒. CHESSは, 特に早指に強い. 又Leveyが同時に対戦した他のプログラムとの戦績は10勝, 1引分(CHAOS), 1敗 (CHESS 4.6).

***** Greenblattはその後チェスマシンを開発し, 高速化を行っている.

****** CDC Cyber 176 (CHESS. 4.6), IBM 370/168 (KAISSA, MASTER), IBM 370/165 (DUCHESS), AMDAHL 470 V/6 CHAOS, WITA), UNIVAC 1110 (BLACK KNIGHT), CDC 6600 (DARK HORSE) 等々.

レイトで用意し，その内 65 種は手の評価の為の重みづけとして用い，残り 20 種は，探索場所（この場合は石の捕獲の可能性がある場所）の検出に用いられる（高々2手までの浅い探索）．実際のプレイは，各テンプレイトマッチングで得た評価点を重ね合せて，その最高点の点に着手する．技量は人間の全くの初心者の域を出ていないが，ほとんどヨミを用いないで，平面的な視覚情報に主として依存している点は一つの進むべき方向と思われる．

Ryder[37] のプログラムは三つの機能に分けられる．一つは局面の認識要素の組織化で，string（タテ，あるいはヨコに隣接した石のつながり），army（模様），wall（壁），group（連結した string）等の諸概念をポテンシャルの考えを用いて定義している．但し平面的視覚パタンの形ではなく，再帰関数で与え，後での色々な処理の扱いを容易にしている．

第二にすべての string の捕獲の可能性のテストであり，これには本質的にある程度のヨミを必要とするが，比較的局所的に（主として標的 string のダメ）限定される為，探索木を小さく保てる．第三はより広い視野から着手を限定し，決定することである．

ここでは主として group の安全性と重要性がテストされ，15手程度に候補手を絞り深さ2までの探索を行う．Knuth（初心者）との対戦譜があり，プログラムが負けているが，全くの初心者よりは強いものと思われる．Reitman 及び Wilcox[38] は碁をパタン認識及びパタンに基づいた推論活動とみてプログラムを作った．基本的な機能は適切な多層構造パタン（string, group 等）を組織化するとともに各パタンのモニターを続け，攻撃と防御に関するより高次のプランに関与する状況の変化を抽出し，プランの為に有用なパタン情報を記述的に表現する．このプログラム（Interim）はその技量を 27 級に評価している（1969 年版の Zobrist のそれを 38 級としている）．

連珠，五目並べ：五目並べは tic-tac-tou（三目並べ）の延長でもあり（SCORE FOUR（立体四目並べ）は別の延長），ルールは簡明であるが変化は多く，チェッカとほぼ同規模と言われている．しかしチェッカと異なって，このゲームでは最初から終盤の様相を呈している程忙がしいのが特徴である．その為，五目並べは先手の優位がはっきりしており，これを修正して，黒（先手）のみに禁手のルールを設けたものが連珠である．従って連珠においては黒と白の作戦の立て方が対称ではなく，それが五目並べよりかなり複雑なものにしている．プログラムの観点から見ると，最終目標（五連達成）を直接目指した手，あるいは局面評価が（チェッカ等に比して）作り易く，それの精度を上げることが，直接プログラムの実力に反映され，それだけ長期的戦略，プラン作成の必要は少ないと言える．Elcock 及び Murray[26] は勝に至るいくつかの段階における中間目標の順序付けを学習する五目並べのプログラムを書いた（§3 参照）．又彼等は前と同じく敗着解析の手法で，中間目標（の記述）そのものを自動的に作り出すことも行っている[39]．真野[40]は Elcock, Murrey 等の盤面記述方法を，連珠の規則にあう様に修正してプログラムを作成している．小川[41]，駒木[42]等も連珠プログラムを書いているが，特に真野のプログラム（JIRO）と小川のプログラム（JARI）は試合（手紙通信）を行っており，前者が3勝1敗の成績を残している．

バックギャモン，ドミノ，ブリッジ：この三つのゲームはチェスと共に欧米の4大ゲームといわれる程ポピュラである．又すべて不完全情報ゲームである．バックギャモン（日本でも双六（絵双六ではない）の名で古代から存在した）は，サイコロの結果によって生ずる可能手が平均800通り以上もあり，通常の意味での探索は不可能である．従って，手の評価に対して最大限の知識が必要となる．又近代バックギャモンではダブルの規則（賭金を倍にする挑戦）の採用の為，局面の優劣，及びその差の測定を常に行う必要があり，ゲームを一層複雑なものにしている．最適なダブルの仕方に関していくつかの報告がある[43),44)]．Berliver[45),46)]はダブルのルールを含めた完全なゲームについてかなり上手なプレーをするプログラムを書いた．駒木[47]はダブルを含まないプログラムを作成している．ドミノとブリッジは4人（正式）で行うこと，及びベストプレイを要求する他に，相手（パートナ及び敵）の手の内を推定するプロセスが加わる点でやや似ている．Reisman[49] は初心者の認知のメカニズムを調べる目的で2人プレイヤのドミノプログラムを書いた．Pervin[50] のプログラム（4人）は，任意に4つの基本戦略（自己の為のプレイ，左右の敵への防害，パートナへの協力）を設定し，確率計算に基づいて混合戦略を用いている．

Smith[25] のドミノプログラム（§3 参照）はこの4つの基本戦略を基礎にして学習機能を持たせたものである．ドミノではプレイの進行につれてのみ相手の手の内を推論する他はないが，ブリッジでは，これ以外

に前半のビッドの過程で情報交換が行われ，正確な手の推論に基づく，適正なコントラクトを行うことが後半のプレイ以上に重要である．Steiner[53] はビッドをするプログラムを作成した．これは ACOL システム(英でよく使用されるビッドシステム)をもとに，ブリッジの知識を考慮して，直接推論を組立てるものである．確率計算に基づいた彼の初期のもの[52]に比較すればはるかにうまいビッドをする．

以上の他にポーカ，将棋，ヘックス，オセロ等の興味あるゲームが残ってしまったが，残念ながら紙数の関係上割愛せざるを得ない．それにしても東洋のドミノともいうべき麻雀が全く手つかずで残されているのは不思議である．

5. あ と が き

ゲームプログラムの歴史を振返ってみると，正にチェスに始まりチェスに終る観がある．

特に 1970 年以降のコンピュータ同志の花々しい合戦は印象的である．しかし，その成功の秘密が超高速大型計算機の威を借りた全幅探索という．A.I 研究の立場から見ると顔を覆いたくなる程の武骨な手段にあると聞けば驚かざるを得ない．あまりに探索型の思考法に安住しすぎているように思える．本来ゲームには種々の思考の型が存在したはずであり，それが生かせるゲームについてのコンピュータ合戦も必要である．日本にも GPCC (Game & Pyzzle Competition on Computer：計算機によるゲームとパズル競技会) が 1974 年に発足しており，各種のパズル，ヘックス等のゲームをとりあげているがやや宣伝不足である．私見になるが，対象とするゲームはあまり人工的でない，むしろ最もポピュラーなものの方が良い．その理由はそれらが人間によって厳しく鍛えられており，研究に価する思考形態が含まれているはずだからである．

最近の傾向として，マイコンのゲームの世界への進出は目覚しい．チェスのような複雑なゲームでさえ例外ではない．より柔軟な思考を可能にするマルチプロセッサシステムの開発が進めば将来，ゲームは，マイコンの独壇場になる可能性がある．

参 考 文 献

1) 志村正道：発見的探索理論，情報処理，Vol. 19, No. 1, pp. 929-935 (1978).
2) Schaefer, T. J.: On the complexity of some two-person perfect-information games, Journal of computer and system sciences 16, pp. 185-225 (1978).
3) Schannon, C.: Programming a digital computer for playing chess, Philosophy Magazine, Vol. 41, pp. 356-375 (1950).
4) Newell, A., Show, J. and Simon, H.: Chess playing programs and the problem of complexity, IBM J. Res. and Develop., Vol. 2, pp. 320-335 (1958).
5) Edward, D. and Hart, T.: The α-β heuristic, MLT Artificial Intelligence Memo, No. 30, (Oct. 1963).
6) Knuth, D. E. and Moore, R. W.: An Analysis of alpha-beta pruning, Artificial Intelligence Vol. 4, pp. 293-326 (1975).
7) Slagle, J. R. and Dixion, J. K.: Experiments with some programs that search game tree, JACM, Vol. 16, No. 2, pp. 189-207 (1969).
8) Fuller, S. H., Gaschnig, J. G. and Gillogly, J. J.: Analysis of the alpha-beta pruning algorithm, Carnegie-Mellon Univ., Computer Science Dept. Report (June 1973).
9) Newborn, M. M.: The efficiency of the alpha-beta search on trees with banch-dependent terminal node scores, Artificial Intelligence Vol. 8, No. 2, pp. 137-153 (1977).
10) Baudet, G. M.: On the branching factor of the alpha-beta pruning algorithm, Artificial Intelligence Vol. 10, No. 2, pp. 173-199 (1978).
11) Berliner, H. J.: Some necessary condition for a master chess program, Proc. 3rd IJCAI, pp. 77-85 (1973).
12) Slagle, J. R.: Artificial Intelligence; The Heuristic Programming Approach, McGraw-Hill (1971) (訳本：人工知能—発見的プログラミング—，産業図書).
13) Nilsson, N. J.: Problem Solving Methods in Artificial Intelligence, McGraw-Hill (1971) (訳本：人工知能，コロナ社).
14) Slate, D. J. and Atkin, L. R.: CHESS 4.5—The Northwestern University chess program, Chess Skill in Man and Machine, P. W. Frey (ed.), Splinger-Velag (1977).
15) Gillogly, J. J.: The technology chess program, Artificial Intelligence Vol. 3, No. 2, pp. 145-163 (1972).
16) Harris, L. R.: The heuristic search under conditions of error, Artificial Intelligence Vol. 5, No. 3, pp. 217-234 (1974).
17) Hart, P. and Raphael, B.: A formal bases for the heuristic determination of minimum cost paths, IEEE Trans. System Sci. Cybernetics, Vol. SSC-4, No. 2, pp. 100-107 (1968).
18) Berliner, H. J.: The B* tree search procedure;

Best-first search combined with branch and bound, Computer Science Dept., Carnegie-Mellon University, (July 1977).
19) Pitrat, J.: A chess combination Program which uses plans, Artificial Intelligence Vol. 8, No. 3, pp. 275-321 (1977).
20) Frey, P. W. and Atkin, L. R.: Creating chess player (part 4), BYTE pp. 126-144 (1979).
21) Samvel, A.: Some studies in Machine learning using the game of checkers, IBM J. Res. Develop., Vol. 3, pp. 211-229 (1959).
22) Samuel, A.: Some studies in machine learning using the game of checkers. II-Recent Progress, IBM J. Res. Develop., Vol. 11, pp. 601-616 (Nov. 1967).
23) Pitrat, J.: A program for learning to play chess, Pattern Recognition and Artificial Intelligence, C. H. Chen (ed.), Academic Press, pp. 399-419 (1976).
24) Findler, N. V.: Studies in machine cognition using the game of paker, CACM, Vol. 20, No. 4, pp. 230-245 (1977).
25) Smith, M. H.: A learning program which plays partnership dominoes, CACM, Vol. 16, No. 8, pp. 462-467 (1973).
26) Elcock, E. W. and Murrey, A. M.: Experiments with a learning component in a go-moku playing program, Machine Intelligence 1, pp. 87-103 (1967).
27) Remus, H.: Simulation of a learning machine for playing Go, Proc. IFIP pp. 192-194 (1962).
28) Griffith, A. K.: A comparison and evaluation of three machine learning procedures as applied to the game of checkers, Artificial Intelligence Vol. 5, No. 2, pp. 137-148 (1974).
29) Kister, J. P., Stein, S. and Ulam, W.: Experiment in Chess, JACM, Vol. 4, No. 2, pp. 174-177 (1957).
30) Bernstein, A. et al.: A chess-playing program for the IBM 704 computer, Proc. West. Joint Computer Conf., pp. 157-159 (1958).
31) Greenblatt, R. D. et al.: The Greenblott chess program, Proc. AFIPS, FJCC, pp. 801～810 (1967).
32) Adelson-Velskiy, G. M., Arlazarov, V. L. and Donskoy, M. V.: Some method of controlling the tree search in chess programs, Artificial Intelligence Vol. 6, No. 4, pp. 361-371 (1975).
33) Douglas, J. R.: Chess 4. 7 versus David Levy, BYTE, pp. 84-90 (Dec. 1978).
34) Thorp, E. O. and Walden, W. E.: A partial analysis of Go, The computer Journal, Vol. 7, No. 3, pp. 203-207 (1964).
35) Zobrist, A. L.: A model of visual organization for the game of Go, SJCC, pp. 103-112 (1969).
36) Zobrist, A. L.: Feature extraction and representation for pattern recognition and the game of Go, Thesis, Computer Science Dept., University of Wisconsin (1970).
37) Ryder, J. L.: Heuristic analysis of large trees as generated in the game of Go, Rep. No. CS-245, Computer Science Dept., Stanford University, (1971).
38) Reitman, W. and Wilcox, B.: Pattern recognition and pattern-directed inference in a program for playing Go, D. A. Waterman and F. Heys-Roth (ed.), Pattern-Directed Inference Systems, Academic Press, pp. 503-523 (1978).
39) Murrey, A. M. and Elcock, E. W.: Automatic description and recognition of board patterns in Go-Moku, Machine Intelligence 2, pp. 75-88 (1968).
40) 真野芳久：連珠をするプログラム，信学会全国大会，1322 (1973).
41) 小川 均：連珠と多相問題，bit, Vol. 5, No. 13, pp. 47-53 (1973).
42) 駒木悠二：連珠，bit, Vol. 4, No. 12, pp. 73-79 (1972).
43) Keeler, E. and Spencer, J.: Optimal doubling in backgammon, Operations Research, Vol. 23, No. 6, pp. 1063-1071 (1975).
44) Zadeh, N. and Kabliska, G.: On optimal doubling in backgammon, Management Science, Vol. 23, No. 8, pp. 853-858 (1977).
45) Berliner, H. J.: BGK—A program that plays backgammon, Computer Science Dept., Carnegie-Mellon Univ., (July 1977).
46) Berliner, H. J.: Experience in evaluation with BGK—a program that plays backgammon, 5th IJCAI, pp. 428-433 (1977).
47) 駒木悠二：バックギャモン，bit, Vol. 7, No. 4, pp. 60-63.
48) 一松 信：石とりゲームの数理，森北出版，(1968).
49) Reisman, S.: Dominoes; A computer simulation of cognitive processes, Simulation and Games, Vol. 3, No. 2, pp. 155-163 (1972).
50) Pervin, Y. A.: Algorithmization and programming of the game of dominoes, Problems of Cybernetics III, Pergamon Press, pp. 957-972 (1962).
51) Shershow, H.: The 9th annual ACM chess tournament in Washinton, D. C., Personal Computing, Vol. 3, No. 3, pp. 58-65 (1979).
52) Stainer, A. M.: BRIPP—a bridge playing program, Essex Univ., (1974).
53) Stainer, A. M.: BRIBIP; a bridge bidding program, 4th IJCAI, pp. 374-378 (1975).

(昭和54年4月18日受付)