- pp. 25-32.
- 7) 松尾日出男, 牧野修一, 宮田政信: 周辺噴流型 ACV の上下揺れについて, 日本航空宇宙学会誌, 第24巻, 第268号 (1976), pp. 246-251.
 - 8) MATSUO, H. and MATSUO, K.: Effects of Fan, Ducting and Powerplant Characteristics on the Cushion Stability of Air Cushion Vehicles, AIAA Journal of Aircraft, Vol. 18, No. 5 (May 1981), pp. 372-376.
 - 9) RIBICH, W. A. and RICHARDSON, H. H.: Dynamic Analysis of Heave Motion for a Transport Vehicle Fluid Suspension. Massachusetts Institute of Technology, Engineering Projects Laboratory Report 76110-3 (Jan. 1967).
 - 10) GENIN, J., GINSBERG, J. H. and TING, E. C.: Fluid Suspension Model for Air Cushioned Vehicles, Journal of Sound and Vibration, Vol. 25, No. 1 (1972), pp. 83-93.
 - 11) GOLDSCHMIED, F. R. and WORMLEY, D. N.: Frequency Response of Blower-Duct-Plenum Fluid Systems, Journal of Hydronautics, Vol. 11, No. 1 (1977), pp. 18-27.
 - 12) SWEET, L. M., RICHARDSON, H. H. and WORMLEY, D. N.: Plenum Air Cushion/Compressor-Duct Dynamic Interactions, Journal of Dynamic Systems, Measurement and Control, Vol. 97, No. 3 (1975), pp. 283-292.
 - 13) HINCHEY, M. J. and SULLIVAN, P. A.: Duct Effects on the Heave Stability of Plenum Air Cushions, Journal of Sound and Vibration, Vol. 60, No. 1 (1978), pp. 87-99.
 - 14) HINCHEY, M. J. and SULLIVAN, P. A.: Duct Effects on the Dynamic Fan Characteristics of Air Cushion Systems, Journal of Hydronautics, Vol. 13, No. 1 (1979), pp. 28-29.
 - 15) LAMB, H.: Hydrodynamics, Cambridge University Press, New York (1932), pp. 98-99.
 - 16) 矢野 鶴, 長田文一: ディーゼル機関掃気系におけるサージングの研究, 日本機械学会論文集, 第36巻, 第288号 (1970), pp. 1374-1384.

一 論 文 一

人工知能の探索手法を用いた航法について* On the Navigation Algorithm for Air-to-Air Combat Using a Game Tree Search Technique in Artificial Intelligence

坂 本 保**

Tamotsu SAKAMOTO

Key words: Aircraft; guidance and control; simulation; navigation; missiles; artificial intelligence; tree search

Abstract

A novel navigation algorithm using a tree search procedure developed in Artificial Intelligence has been proposed for both the air-to-air combat between fighter planes and the approach of missiles to air-targets. This algorithm is much easier with respect to the mathematical treatment of the nonlinearity of system equations and the boundary conditions than the differential game method. And it takes much more account of the reciprocal action between a pursuer and a evader than the usual navigation algorithm such as proportional navigation, pure pursuit and so on, by means of AND/OR game tree search which is much more efficient than the pure minimax procedure. This paper presents a comparison between the navigation algorithm using the game tree and the pure pursuit which

is often adopted especially in air-to-air combat simulation models, and shows a difference between the shallow tree search navigation algorithm and the deep one, by use of a very simple air-to-air combat model.

1. ま え が き

近来, デジタル計算機の発達は著しい。それに伴って, デジタル計算機の特徴を生かしたシミュレーションが可能かつ必要となってきた。

従来, 航空機同士の空中戦シミュレーションに用いられる航法は, ミサイルなどで用いられる航法を彼我の状態に応じて適用したり, 相手の動きを適当な方法で補間して, 自機の行動を決定するといった多分に経験的な手法を採用しているものが多いようである^{1)~4)}。しかし, このようなやり方では, 自機の行動は相手機の行動に盲目的に依存するおそれがあり, ま

* 昭和58年1月17日原稿受理

** 防衛庁技術研究本部第3研究所

第2図 アルファ・カット

第3図 ベータ・カット

A. I. では、この探索能率を向上させることを目的として種々の研究がなされてきているが^{11)~13)}、対象とするゲームに関する知識で探索効率を上げる情報 (heuristic information) の利用が、現実的にゲームを行わせるに際して不可欠であることが種々報告されている¹⁰⁾。

以下では、敵対性というゲームにおける最も基本的な性質を利用するアルファ・カット (alpha cut) とベータ・カット (beta cut) について簡単に説明する。

第2図にアルファ・カットの一例を示す。節点A (アンド節点) の取りうる得点を V_a とする。同様に、節点Aから生成を繰り返してできた子孫節点である節点B (オア節点) のとりうる値を V_b とする。ここですでに節点Aの状態から選ぶ“手”の一つから3点が節点Aに繰り上げられているものとする。節点Aから次々と子孫節点達が生成されて、いま、節点Bが作られたとして、節点Bから選ぶ“手”の一つから2点が得られたとする。すると、節点Bから生成される他の節点群は無視してかまわない。なぜなら、 $V_b \leq 2$ であり、 $V_a \geq 3$ であるからである。言い換えれば、節点Aから見た場合、すでに3点が得られる“手”が知られているのであるから、節点Bから下へ生成されるアンド節点の取りうる値がいかにであろうとも $V_b \leq 2$ であることを考えれば、対戦者相互の敵対性か

第4図 アルファ・ベータ手続きを用いた木探索

ら節点B以下は実現されえないので、もはやそれ以上探索を続行する必要はない。

ベータ・カットは、アルファ・カットと裏返しの関係にある。第3図はベータ・カットがなされた場合である。節点C (オア節点) ですでに4点以下に押さえられる“手”を発見してあるので、節点D (アンド節点) の得点 V_d が7点以上であることを考えると、節点Cから見た場合、節点D以下に生成される節点群は点数を下げる効果がない。ゆえに、節点Dから先の探索は不必要である。

探索効率の評価法はいろいろあるが²⁸⁾、末端節点 (terminal node) の数で効率を評価すると、純粋 Mini Max 法では、探索 D 、一つの節点から生成される子節点の数 (branching factor) を N とすると、末端節点の総数は、 N^D 個となる。しかし、一方、上記のアルファ・カットとベータ・カットを併用した場合は、最良の場合

$$N^{\lceil \frac{D}{2} \rceil} + N^{\lfloor \frac{D}{2} \rfloor} - 1$$

個となる ($\lceil x \rceil$ は x の切り上げ、 $\lfloor x \rfloor$ は x の切り下げ)。たとえば、 $D=6$ 、 $N=10$ とすると、純粋 Mini Max 法では 10^6 個、それに対して、アルファ・カットとベータ・カットを併用すると約 10^8 個のオーダーですむ。数百万個の節点を探索することは困難でも、数千個の節点の探索は、かなり容易でありうる。

この点は、実際に計算機に計算をさせる場合、非常に重要である。

第4図は、アルファ・カットとベータ・カットがなされた一様木 (uniform tree) の探索の例である。

点線の部分はカットを受けて生成されず探索されなかった個所である。ここで注意すべき点は、頂点節点の得点 11 は、探索の順序に依存しないが、途中の中間にある節点の点数とカットされる部分は探索の順序によって変化するという点である。しかし、頂点節点の得点は純粋 Mini Max 法を用いた得点と完全に一致することが数学的に証明されている^{18), 19)}。

5. アルゴリズム

あとに示される三次元空中戦シミュレーションで用いたアルゴリズムについて第5図、および第6図に概略を説明する²⁰⁾。

第5図の関数 F は、状態 P を与えると、可能な方策のなかから P の子孫以下に対して彼我が合理的に対応したと仮定した場合に最大の評価点を得ることのできる方策を決定し、それによって得られる評価値を F として与える関数である。

13行目と21行目の手続き `generate_child` は、節点 P から次々と新たに子孫状態節点 P_child を生成する作用素であり、これによって飛翔体の運動が規定される。

8行目の関数 `evaluate` は、状態 P に対して静的な評価値を与えるもので、一方の競技者から見た状態 P の有利不利に関する評価値を与えるものである。戦術はこの評価関数によって表現される。

まず8行目で節点 P が勝利または敗北状態 (評価値が $\pm\infty$) であるかを調べる。もしそうであれば、勝負は終了しているわけであるから、その下の子節点は生成する必要がなくその値が関数 F の値となる。また、最大打ち切り深さに達した場合も関数 F に静的評価値を与える。もし状態 P が上の条件を満たさない場合は、節点 P の評価値はただちには決まらず10行目から23行目までの手続きを行う。

11行目から13行目で節点 P から初めの子節点 P_child

`child` を生成する。14行目から22行目の間を、子節点 p_child が存在し ($p_child \neq \text{null}$)、節点 P を相手が生ずる可能性があり ($\alpha_child < \beta$)、かつ勝ち節が見つからない ($F < +\infty$) 場合に限り繰り返す。その他の場合は探索を打ち切り、枝刈りが行われる。

16行目と18行目の関数 G は、子節点 P_child の評価得点を与える関数である (第6図参照)。

15行目から20行目を繰り返すことで、 F 値は小さいほうから純粹 Mini Max 法による評価値に、必ずしも一致するとは限らないが、 α_child 値とともに近づいていく。しかし、親節点の存在しない根節点 (root node) に対しては一致する。

第6図の関数 G は、関数 F と双対関係にある。すなわち、関数 G は、状態 P を与えると、 P の子孫節点群に対して彼我が合理的な方策をとったとき、最小の評価値を与える関数である。関数 G 中の手続きは関数 F のそれと対称的な構造となっている (第5図参照)。

初期状態を根節点 P_0 として、関数 F によって評価すれば、最大打ち切り深さ内での評価値 ($F = \text{evaluate}(g_n g_{n-1} \dots g_2 g_1 P_0)$) と、それを実現させる一連の作用素列 ($g_1, g_2, \dots, g_{n-1}, g_n$) が得られる。ここで、 g_1 を P_0 に対する最適方策とする。

読みの深さは有限であり、かつ両者の相対的位置関係は刻々と変化していくので、状態の変化に伴って勝負のつくまで幾度も上記の操作は繰り返される。

手続き `generate_child` と評価関数 `evaluate` は、

```

1      Function F( var P : node ;
2                  var Alpha , Beta : value ;
3                  var Max_Depth : integer
4                  ) : value ;
5      var P_Child : node ;
6      Alpha_Child : value ;
7      begin
8          if ( |Evaluate( P )| = +∞ ) or ( P.Depth = Max_Depth ) then F := Evaluate( P )
9              else
10         begin
11             Alpha_Child := Alpha ;
12             F := -∞ ;
13             Generate_Child( P , P_Child ) ;
14             while ( P_Child ≠ null ) and ( Alpha_Child < Beta ) and ( F < +∞ ) do
15                 begin
16                     if G( P_Child , Alpha_Child , Beta , Max_Depth ) > F then
17                         begin
18                             F := G( P_Child , Alpha_Child , Beta , Max_Depth ) ;
19                             if F > Alpha_Child then Alpha_Child := F
20                         end ;
21                     Generate_Child( P , P_Child )
22                 end
23             end
24         end ;

```

第5図 得点を最大化しようとする立場の表現

```

1  Function G( var P : node ;
2      var Alpha , Beta : value ;
3      var Max_Depth : integer
4      ) : value ;
5  var P_Child : node ;
6      Beta_Child : value ;
7  begin
8      if (|Evaluate( P )| = +∞ ) or ( P.Depth = Max_Depth ) then G := Evaluate( P )
9      else
10         begin
11             Beta_Child := Beta ;
12             G := +∞ ;
13             Generate_Child( P , P_Child ) ;
14             while ( P_Child ≠ null ) and ( Beta_Child > Alpha ) and ( G > -∞ ) do
15                 begin
16                     if F( P_Child , Alpha , Beta_Child , Max_Depth ) < G then
17                         begin
18                             G := F( P_Child , Alpha , Beta_Child , Max_Depth ) ;
19                             if G < Beta_Child then Beta_Child := G
20                         end ;
21                     Generate_Child( P , P_Child )
22                 end
23             end
24         end ;

```

第6図 得点を最小化しようとする立役の表現

第7図 節点と生成作用素

微分ゲームでのシステム方程式と評価指数に当たるが、微分や積分などの解析的操作を基本的に行わないので、線型性や微分可能性はもちろん、連続性すら必要としない。また、制限条件が厳しくなれば、それだけ生成作用素に制限が加わり、枝刈り効果によって探索効率が向上する。

6. 三次元飛翔体モデル

ここでの三次元飛翔体モデルは非常に単純なものではあるが、前述のアルゴリズムとは分離して考えることができるため、より精密なモデルを構成するのは可

能である。本モデルでは、飛翔体は空間上の点で表現され、状態量は位置と速度である。

手続き generate_child としては、速度ベクトルの各成分（極座標）に正負の微小変化を加えるものと、何も加えないものの合わせて7種を考え、それによって新しい位置と速度を決めるものとした（第8図参照）。飛翔経路に関する制限は手続き generate_child の中で表現される。第9図で示されるように、速度 v は、 z 成分を v_z として、 $v_{\min} \leq |v| \leq v_{\max}$ かつ $v_{z\min} \leq v_z \leq v_{z\max}$ を満足する領域にあるものだけが実現されるとして、子節点の生成に制限を加えている。 $v_{\max}$, $v_{\min}$ は固定されているが、 $v_{z\min}$ は位置と速度により変化し、 $v_z < v_{z\min}$ では、上げ舵を取り続けていても地上に激突 ($z < 0$) する。

次に、評価関数について説明する。 θ_1 を評価得点を

第8図 速度空間の中の速度と速度変位

第9図 速度に対する制限

第10図 見越角と見越視線

最大化しようとする側から相手機をみたときの見越角 (look angle) とし, θ_2 を評価得点を最小化しようとする側から見た見越角とする. さらに, R, ε, C, n を, それぞれ相対距離, 微小正数, 正定数, および正の奇数とし, 評価値を E として,

$$E = C \cdot \frac{(\theta_2 - \theta_1)^n}{(R + \varepsilon)}$$

のように評価値を決めている. ただし, 評価得点を最大化する立場から見て, 相手機の背後についたとされる場合 ($\theta_1 \leq \pi/12(\text{rad})$, $\theta_2 \geq 5\pi/6(\text{rad})$, かつ $R \leq 3,000$ (m)) は, $E = +\infty$ とし, 逆に背後につかれたとされる場合 ($\theta_1 \geq 5\pi/6(\text{rad})$, $\theta_2 \leq \pi/12(\text{rad})$, かつ $R \leq 3,000$ (m)) は, $E = -\infty$ として評価関数を決めた.

両者は交互に方策を選択していくため, 一方が微小時間に一つの方策を選択する間, 他方は等速度運動すると見なし, 次の微小時間では立場を逆転させることで両者はおのおのの飛翔経路を変更するとして探索は行われる. すなわち, 時間に関する差分が両者で一致せず, 半周期ずつずれた形で時間差分が行われると考えてもよい.

実際には, 互いに相手の行動を読んで自分の行動を決定するにせよ, 読みの深さや制限条件などの違いに

より, 通常, 相手は予測されたとおりに行動するとは限らないし, 異種アルゴリズム (たとえば pure pursuit や比例航法など) を相手が採用している場合も同様である.

そこで, 両者の状態が時間とともに刻々と変化していくのに伴って, 次々と探索木を展開し直し, 探索を繰り返していく.

以下で用いられるパラメータは,

$$\Delta t = 0.5 \text{ (s)} \text{ (時間刻み)}$$

$$\Delta v_r^+ = 15 \text{ (m/s)}$$

$$\Delta v_r^- = -22 \text{ (m/s)}$$

$$\Delta v_\phi^+ = -\Delta v_\phi^- = \pi/15 \text{ (rad)}$$

$$\Delta v_\theta^+ = \pi/18 \text{ (rad)}$$

$$\Delta v_\theta^- = -2\pi/45 \text{ (rad)}$$

$$v_{\max} = 300 \text{ (m/s)}$$

$$v_{\min} = 100 \text{ (m/s)}$$

$$v_{z\max} = 150 \text{ (m/s)}$$

$$c = 1$$

$$n = 3$$

である.

第11図 Pure pursuit を用いた追跡機と直線飛行する目標機 (追跡機の初速度: 280 m/s, 目標機の初速度: 220 m/s)

7. 従来のアルゴリズムとの比較

従来の追跡アルゴリズムの代表として、追跡者が、pure pursuit を用いる場合を考える。pure pursuit とは、追跡者の速度ベクトルが見越視線(line of sight)と一致するように見越角の大きさに比例した縦、横加速度を加える誘導方法である。

pure pursuit は従来、とくに航空機同士の空中戦シミュレーションに、その変形をも含めてよく採られる航法である^{3),4)}。

第11図に pure pursuit を用いた追跡者が、等速直線運動する目標の背後に回り込んでいく様子が示されている。この場合、追跡者は追跡途中でオーバー・シュート(over shoot)を起こしている。これは、6章で述べられた速度変位に対応する限界加加速度が設けられているためである。しかし、あらかじめ早期に方向転換を大きく行っていれば、オーバー・シュートは避けられる。このことは次の第12図を見るとわかる。なお印は一定時間間隔ごとに付けたもので、追跡者と目標の同時刻における位置関係を示す。

Z 軸は高度方向であり、上方の図は Y-Z 平面、

第12図 木探索を用いた追跡機と直線飛行する目標機 (追跡機の初速度: 280 m/s, 目標機の初速度: 220 m/s)

すなわち立面図、下方の図は X-Y 平面、すなわち平面図である。両者は S 点からスタートし、E 点で終わる。

第12図は、前の第11図と同じ初期条件と制限条件の下で、等速直線運動する目標に対して、5章で示された木探索(tree search)を用いた追跡者が目標の背後に回り込んでいく様子が示されている。この場合は、pure pursuit を用いた場合と異なり、追跡中にオーバー・シュートを起こしていない。すなわち、その時々状態のみに依存する pure pursuit に比べて、現在の状態から将来の諸状態を生成して、そのなかから妥当な方策を決定する木探索では、将来起こりかねないオーバー・シュートを見越して、大きく進路変更をすることが可能である。ただし、追跡者に対しては、目標が等速直線運動するということは情報として与えていない。相手も自分の背後に回り込もうとしているとの仮定のもとに追跡者は自己の行動を決定してい

第13図 Pure pursuit を用いた追跡機と直線飛行する目標機の見越角と相対距離の時間変化

第14図 木探索を用いた追跡機と直線飛行する目標機の見越角と相対距離の時間変化

る。

第13図に第11図の、また第14図に第12図の、追跡者と目標の相対距離と見越角の時間変化を示す。横軸は時間、左縦軸は0から π (rad) までの見越角、右縦軸は相対距離である。見越角が0 (rad) で相手機は自機の真正面、 π (rad) で相手機は自機の真後ろということになる。相手機の背後に回り込もうとするということは、自機の見越角をできるだけ小さくし、相手機の見越角をできるだけ大きくすることを意味する。

第13図を見ると、pure pursuit を用いた追跡者は目標の背後に回り込む (追跡者の見越角 $\leq \pi/12$ (rad), 目標の見越角 $\geq 5\pi/6$ (rad), かつ相対距離 $\leq 3,000$ (m)) までの時間に56秒かかっている。一方、第14図を見ると、木探索を用いた追跡者は、目標の背後に30秒で回り込んでいる。また、追跡者の見越角の時間変化も木探索 (第14図参照) のほうが pure pursuit (第13図参照) のそれよりも急激であり、pure pursuit のほうでは大きく波打っているのに対して、木探索のほうは8秒以降は $\pi/16$ (rad) 以下である。すなわち、

第15図 Pure pursuit を用いた飛行機と木探索を用いた飛行機による空中戦 (両機の初速度: 250 m/s)

第16図 Pure pursuit を用いた飛行機と木探索を用いた飛行機による空中戦における両機の見越角と相対距離の時間変化

ほぼ真正面に目標を捕え続けている。

第15図は、pure pursuit を用いた飛行機と木探索を用いた飛行機が、相互に相手機の背後に回り込もうとする空中戦を想定したものである。

両者の運動に関する制限条件は同一であり、互いに相手機を正面に捕えている状態 (head on) からスタートしている。両機はシザーズ (scissors) を繰り返しながら下降していき、木探索を用いた飛行機が、pure pursuit を用いた飛行機の背後に回り込んでいく様子がわかる。

第16図は、第15図での両者の見越角と相対距離の時間変化である。第16図のグラフが波打っているのは、両者がシザーズを繰り返しているからである。時間の経過に伴って、木探索を用いたほうの見越角はしだいに小さくなり、pure pursuit を用いたほうのそれはしだいに大きくなり、かつ両者の相対距離は小さくなっている。これは、木探索を用いた飛行機が、pure pursuit を用いた飛行機の背後に回り込んでいく様子を定量的に表わしている。

8. 読みの深さの効果

第17図、および第18図に読みの深さの違いによる行動の違いを見る。

第17図は、上述の運動に関する制限条件の下に、探索を深く (deep: 読みの深さ7) 行って行動する飛行機と、探索を浅く (shallow: 読みの深さ1) 行って行動する飛行機が、相互に相手機を正面に捕えた状態からスタートして、互いに相手機の背後に回り込もうとしている様子を模擬している。

7章の場合と同様に、両者はシザーズを繰り返しながら下降していく。

第17図 深い木探索を用いた飛行機と浅い木探索を用いた飛行機による空中戦
(両機の初速度: 250 m/s)

第18図の両者の見越角と相対距離の時間変化から、探索の深いほうが探索の浅いほうよりも後方のよい位置を占めていることがわかる。

9. む す び

本研究では、おもに空中戦等を行う航空機、飛翔体に対して有効と思われる人工知能の分野の木探索手法を用いた航法アルゴリズムを提案し、それを用いて、幾つかのシミュレーションを行った。具体的には、木探索を用いた航法アルゴリズムと従来のそれ (pure pursuit) との比較を行うとともに、木探索に関して読みの深さの異なる場合の比較もあわせて行った。

今後の課題としては、木探索の性質を利用した多数機対多数機の場合への拡張や、計算時間が従来の実時間処理用航法アルゴリズム (pure pursuit や比例航法等) よりも必要とされると思われるため (たとえば、第11図は6秒、第12図 (読みの深さ6) は67秒計算に要した。) 計算速度向上に関する研究 (並列処理²¹⁾ や高速探索アルゴリズム^{22), 25)} 等) が考えられる。

なお、計算には、第一研究所計算機室の Hitac M-

第18図 深い木探索を用いた飛行機と浅い木探索を用いた飛行機による空中戦における両機の見越角と相対距離の時間変化

200H を使用した。

また、本研究を進めるにあたり、貴重な助言をいただきました関係諸氏に対しまして深く感謝いたします。

参 考 文 献

- 1) BURGIN, G. H., FORGEL, L. J. and PHELPS, J. P.: An Adaptive Maneuvering Logic Computer Program for the Simulation of One-on-One Air-to-Air Combat, Vol. I, NASA Contractor Report, NASA CR-2582 (1975).
- 2) BURGIN, G. H. and OWENS, A. J.: An Adaptive Maneuvering Logic Computer Program for the Simulation of One-on-One Air-to-Air Combat, Vol. II, NASA CR-2583 (1975).
- 3) HANKINS, W. W., III: Computer-Automated Opponent for Manned Air-to-Air Combat Simulations, NASA Contractor Technical Paper 1518, N79-33855 (Sept. 1979).
- 4) HUTCHESON, J. H. and SEGERBLOM, R. L.: TACTICS: A Three-Body, Three-Dimensional Intercept Simulation Program, The Rand Corporation Memorandum RM-5759-PR (Oct. 1969).
- 5) 加藤寛一郎: 比例航法に関する二、三の考察, 三研講演会パンフレット (1980).
- 6) 坂口 実: ゲームの理論, 森北出版 (1969).
- 7) 古田勝久: 線型システム制御理論, 昭晃堂 (1973).
- 8) 近藤次郎: 数学モデル入門, 日科技連 (1974).
- 9) MERZ, A. W.: The Homicidal Chauffeur, AIAA Journal, Vol. 12, No. 3 (Mar 1974), pp. 259-260.
- 10) 実近憲昭: ゲームと探索, 電子通信学会誌, 第65巻, 第4号 (1982), pp. 405-412.
- 11) WINSTON, P. H.: Artificial Intelligence, Addison-Wesley Publishing Company, Inc., New York (1977).
- 12) SLAGLE, J. R.: Artificial Intelligence, McGraw-Hill, Inc., New York (1971).
- 13) NILSSON, N. J.: Problem-Solving Methods in Artificial Intelligence, McGraw-Hill, Inc., New York (1971).

- 14) 田中幸吉: 人工知能から知識工学への展開, 数理科学, 第214号 (1981), pp. 7-10.
- 15) FEIGENBAUM, E. A.: Knowledge Engineering: The Applied Side of Artificial Intelligence, Stanford Heuristic Programming Project Memo/HPP-80-21 (1980).
- 16) 小川 均: 知識工学の最近の動きと文献, 数理科学, 第214号 (1981), pp. 49-54.
- 17) 後藤滋樹: 推論と問題解決, 電子通信学会誌, 第65巻, 第4号 (1982), pp. 379-385.
- 18) KNUTH, D. E. and MOORE, R. W.: An Analysis of Alpha-Beta Pruning, Artificial Intelligence, Vol. 6, No. 4 (1975), pp. 293-326.
- 19) BAUDET, G. M.: On the Branching Factor of the Alpha-Beta Algorithm, Artificial Intelligence, Vol. 10 (1978), pp. 173-199.
- 20) 房岡 璋, 金森 直: PASCAL—新しいプログラミング言語の紹介, システムと制御, 第23巻, 第4号 (1979), pp. 211-220.
- 21) AKL, S. G., BARNARD, D. T. and DORAN, R. J.: Design, Analysis and Implementation of a Parallel Tree Search Algorithm, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-4, No. 2 (Mar. 1982), pp. 192-203.
- 22) STOCKMAN, G. C.: A Minimax Algorithm Better than Alpha-Beta?, Artificial Intelligence, Vol. 12 (1979), pp. 179-196.
- 23) NEWBORN, M. M.: The Efficiency of Alpha-Beta Search on Trees with Branch-Dependent Terminal Node Scores, Artificial Intelligence, Vol. 8 (1977), pp. 137-153.
- 24) LEVI, G. and SIROVICH, F.: Generalized AND/OR Graphs, Artificial Intelligence, Vol. 7 (1976), pp. 243-259.
- 25) 茂木俊秀: 数理計画法の展開をさぐる(Ⅶ) 組合せ最適化とラグランジュ緩和, システムと制御, 第25巻, 第4号 (1981), pp. 222-230.
- 26) 上原祥雄: 最近の戦闘機の運動性における最適制御理論の応用, 計測と制御, Vol. 20, No. 10 (1981), pp. 930-936.