遺伝的アルゴリズムによる最適制御*

安　田　俊　一

1　はじめに

Holland (1975) によって提唱された「遺伝的アルゴリズム (Genetic Algorithms, 以下 GA)」は最適問題を解くためのアルゴリズムとして様々な問題へ応用され，その有効性は確立されている。しかしながら経済学などの社会科学の分野ではまだ応用例は多くない。工学と違ってそのような分野では最適問題を具体的な数値を用いて解くこと自体があまり意味があると考えられないことが原因の一つである。

しかし企業や家計などの経済主体を特定の環境の中で生存する生物とみなしてその行動を考えていく際には遺伝的アルゴリズムの考え方は非常に示唆に富むものである。後述のように遺伝的アルゴリズムは最適解の探索を解析的な条件から導くわけではないので，経済環境の中でゲームに参加する各主体の行動やそのルールなどを現実に近いような形でモデリングすることが可能である。このことは，たとえば多数の主体の相互関連とそこから生成されるマクロ的な構造が歴史的な時間の中で進化していく過程を考察するというような，従来の手法ではあまりに複雑で取り扱えなかったような問題に対する分析手法を提供するものと考えられる。

マクロ経済の構造をミクロ的な合理性から構築するに当たっては，各主体が動学的最適化問題を解くことによりその行動を決定する必要があるが，多少複

*この研究は 1995 年度松山大学研究助成金による成果の一部である。

雑なルールや条件を付け加えた動学的最適問題を解析的に解くことは非常に煩雑であり，多くの場合は解くことができない。そのため経済問題を定式化する際に定性的な解を得るためには変数や条件に様々な制約・仮定を加えることがふつうである。そうして得られた解は確かに問題の一側面を理解するために重要であり，それなりに意味を持つものであるが，特に上述のように多数の主体が参加する経済を考慮する場合には，解析的な解を得ること自体が非常に困難である。多数の主体が存在するような問題ではなくとも，通常の最大化原理などを用いた解析的な手法はコントロール変数や状態変数が複数になった時点で解くことがひどく困難になる。近年盛んに行われているゲーム理論を応用した分析では2人ゲームを仮定することが多いが，そのような理由から，解析的に解を決定できるモデルは限定されている。

したがって複雑な動学的最適化問題を解く場合には数値計算によって解の挙動を観察することが一つの方法である。当然，問題を解くアルゴリズムとしてGAは一つの候補として考えられるのであるが，現在のところ動学的最適化問題の数値計算にGAを応用した例はほとんどない。[1] 本稿の目的は動学的最適化問題を解くためのアルゴリズムとしてGAの有効性を確認することである。そのために通常1変数，あるいは1時点での最適化問題として定式化されるGAを多変数・多期間を取り扱えるような形に拡張したプログラムを作成し，それを用いて遺伝的アルゴリズムに関する実験を行う。

まず最初に遺伝的アルゴリズムを概観した後，標準的な問題に対してプログラムが有効であることを確認する。次に解析的に解くことが可能な動学的最適化問題に対して同じプログラムを用いて解を求め，その解が解析的に求められた解を非常にうまくシミュレートしていることを示す。最後に多期間・多変数の複雑な問題に対してプログラムを用いて解を求め，その性質を検討する。

---

1) 経済問題へのGAの応用としては各種資産への投資タイミングを検討したBauer (1994)，2期間のゲームを取り扱ったÖzyildirim (1997)，および競争市場をシミュレートしたDawid (1996) がある。

2 GA の 概 要

ここで遺伝的アルゴリズムの動作原理を簡単にまとめておこう。遺伝的アルゴリズムは生物に対する自然選択がより適合的な形質を発展させる性質を模したものである。生物集団における個体はそれぞれに異なる遺伝子をもっており，その遺伝子の発現によって自然環境に対する適合の程度がそれぞれ異なる。より自然に適合する個体は子孫を多く残すことができるので，遺伝を通じてそれらの性質は子孫に受け継がれる。その結果，よりよく自然に適合する個体が生物集団の中で増加する。

この過程を模するために，自然に対する適合の程度を測る指標を作成し，それが大きい個体に繁殖のチャンスをより多く与えてやる。そのことにより次世代により自然に適合した個体を増やしていくことができる。ただし，単にそれだけならば初期の集団においてたまたま高い適合を示した個体の性質のみが増え続けてそれよりも高い適合を持つ個体が出現しない。そこで遺伝子が子孫にコピーされる際に他の個体との遺伝子交叉や突然変異が起きるようにしておく。このような過程を「GA 過程」と呼ぶ。GA 過程が続くと，集団の中にはより適合的な遺伝子をもった個体の割合が多くなっていく。こうして自然にもっとも適合的な個体を作成していくのである。

ここでいう自然を「目的関数」，適合の程度を「関数値」に読み替えると，この過程が関数の最大化の手続きと同じであることは明白である。これが遺伝的アルゴリズムを最大化（最小化）問題の解法として利用する理由になっている。

以下では遺伝的アルゴリズムの基本的な手続きを簡単に述べる。これらについては基本的なテキスト[2]で多く解説されており，ここではごく簡単に触れるにとどめよう。

---

2) Goldberg (1989)，安居院・長尾 (1993)，伊庭 (1994)，Mitchell (1996)など。

2.1 遺伝子型と表現型

進化の過程で個体が自然淘汰に直面する場合，自然が選択するのは生物の身体上の特色や環境適応能力である。たとえば，「髪の色」「手の長さ」「移動速度」などは個体を構成する細胞が持っている「表に現れる」特徴であり，これを「表現型」と呼ぶ。

関数の最大化にそくしていえば，関数の値を決めるのはその関数に与えられる変数の内容である。この値に応じて関数値は変化する。また，ゲーム論では戦略が変われば利得は変化する。それゆえ，GA では関数における変数の値や戦略を表現型とする。たとえば，ここで取り扱う最適制御の問題においてはコントロール変数 u_t の実数列が表現型となる。

生物における表現型は親から子へと世代を通して遺伝していく。このシステムを担っているのが「遺伝子型」である。同一の遺伝子型は同一の表現型を生み出す。表現型は遺伝子型によって決定されるのである。そこで表現型を実数値や戦略とする場合にはそれを決定している遺伝子型を定義し，遺伝子型と表現型の間の変換ルールを定義しなければならない。

一般的に GA においては遺伝子型はビット列で表現される。これは染色体のモデル化であり，後述するように自然界の生殖をモデル化する際の基本となる。遺伝子型と表現型は一対一で対応しており，同一の遺伝子型が異なる表現型を実現することはない。それゆえ遺伝子型の多様さが可能な表現型の多様さを規定する。

たとえば表現型が整数で表される場合を考えよう。この表現型を担う遺伝子型を長さ 8 のビット列で表すと，たとえば次のように表現される。

10110011

そして遺伝子型から表現型への変換のルールがビット列を 2 進数と見なすことであれば，この遺伝子型が発現する表現型は 179 である。この場合には $2^8 = 256$ 通りの整数を表すことができる。

したがって最適化問題を解く場合にはビット列の長さが解の探索空間の広さ

を制限する。本稿で取り扱うような単純なケースではビット列を2進数とみなして遺伝子型から表現型（実数値）への変換を行うので，ビット列の長さを L とし，1ビットの増加による実数の増加分を d，探索範囲の下限を m とすると，探索範囲の上限は $d(2^L-1)+m$ となる[3]。

2.2 適 合 度

進化の基本的な原動力は自然選択によって「優位」な個体の遺伝子が残されていくことである。この「優位さ」は表現型によって規定される。たとえば，枯れた木の枝に擬態する虫を考えよう。個体の表面の色や模様（表現型）が木の枝にうまく似ていればその個体が外敵に襲われる確率は低くなり，子孫を残す可能性は高くなる。その結果そういった表現型を実現している遺伝子は子孫の中に生き残っていき，木の枝にうまく似せることができない遺伝子は減少していくだろう。木の枝によく似ていれば似ているほど，その遺伝子が生き残るチャンスは増加する。この「うまく似ている」程度が「適合度 (fitness)」である。

適合度が高い個体は**必ず**自分の遺伝子を次の世代に残す[4]というわけではなく，逆に適合度が低いからといってその個体の遺伝子が次の世代に**まったく**伝わらない，というわけではないことに注意しよう。適合度の高さはその個体が次の世代に遺伝子を伝える「確率」に影響するのである。

最適化問題の場合には適合度は目的関数の値そのものである。各個体はそれぞれ遺伝子を持っており，その型に応じた表現型（変数の値：実数値や戦略）が決まる。その表現型が自然（目的関数や利得関数）に評価を受け，適合度が決まる。そうして適合度が高ければ高いほどその個体の遺伝子が次の世代に伝えられる確率が大きくなる。

---

3) したがって少ない長さでより広範な探索範囲をカバーするためには遺伝子型を2進数（ビット列）ではなく，8進数や16進数，あるいは文字列を使うことが考えられるが，通常のGAではビット列を用いる。

4) 後述するようにGAではもっとも高い適合度を持つ個体を「必ず」次の世代に生き残らせる手法を用いることがある。

2.3 選　　択

遺伝子が次の世代に伝えられていくシステムがここから以下で述べる「選択」「突然変異」「交叉」である。遺伝的アルゴリズムとは「遺伝子型」「表現型」を持った個体と「適合度」を定義した上で,「選択・突然変異・交叉」を世代毎に繰り返して,より適合度が高い個体を生み出していく手続きにほかならない。その意味でこの節と以下の2節はGAのもっとも根本をなす部分である。

「集団」は様々に異なる適合度を持つ個体の集合である。この集団が自然淘汰のフィルターにかけられる場合，上で述べたように適合度が高い個体ほど遺伝子を残すチャンスは大きくなる。このことは適合度が高い個体ほど,「親」となる可能性が大きいことを意味する。集団の中から適合度に応じて「親」となる個体を選び出す手続きが「選択」である。これは自然界での「適者生存」に応ずる過程である。

n 個の個体からなる集団を考えよう。各個体が持つ適合度を $f_i(i = 1, 2, \ldots, n)$ とし,「親」として選択され，次の世代に子孫を残す確率を p_i とすると，この確率は適合度の関数として

$$p_i = F(f_i),\ 0 \leq p_i \leq 1,\ F' > 0$$

として表される。

GAでは選択をプログラムに実装する際の方法として「ルーレット方式」を用いることが多い。具体的には円周が集団全体の適合度の合計である円盤を各個体の適合度に比例した弧をもつ領域で区切ったルーレットを回し，玉が入った領域の個体を「親」として選び出す方法である。これは個体の適合度に応じてあたりが作ってある「くじ」を引くことと同じである。したがって，適合度が高い個体ほど「あたり」を引く確率は多いが，決してその確率は1ではなく，逆に適合度が低い個体も「あたり」を引く確率は小さいが，その確率は決して0ではない。「あたり」を引く確率は集団の中での相対的な適合度の大きさに比例する。

この方法で i 番目の個体が親として選択される確率は

$$p_i = \frac{f_i}{\Sigma f_i}$$

である。集団が全体で n 個の親から同数の子を発生させるとすると，個体 i が産む子供の期待値は

$$np_i = \frac{nf_i}{\Sigma f_i} = \frac{f_i}{\bar{f}}$$

ここで $\bar{f}$ は集団の平均的な適合度である。ここから，集団中の平均的な個体は平均1個の子を産むことがわかる。ルーレット方式では平均よりも適合度が高い個体は平均より多くの子孫を残し，平均よりも適合度が低い個体は残す子孫の数が平均よりも低くなる。この過程を通じて集団中に高い適合度を実現する遺伝子型を持った個体が増加するのである。

この他に集団の中で各個体に適合度による順位付けを行い，順位と選択確率を関連づける「ランク方式」，一定の数の個体をランダムに選び出して部分集合を作り，その中で最大の適合度を持つ個体を選ぶ操作を必要な数の親が得られるまで繰り返す「トーナメント方式」などがある。また，順位付けの指標として適合度をそのまま使うのではなく適切な変換を行うことがふつうである。[5)]

2.4 突然変異

世代間で遺伝子がコピーされる場合，自然界ではコピーのエラーが起こることがある。これが「突然変異」である。突然変異は集団の遺伝子がある状態に「固まる」ことを阻止する。進化の歴史では突然変異を繰り返すことによって，より複雑で環境に適した遺伝子を生み出してきたのである。

選択によって選ばれた親の遺伝子型が単に子孫へコピーされるだけでは親よりも適合度が高い個体は発生しない。集団の平均的な適合度を高めていくためには全世代の平均的な親たちが持つ遺伝子型が，よりよい遺伝子型へとなんら

5) 本稿で用いるプログラムはごく単純な変換（σ スケーリングと線形スケーリング）のみを用いている。

かの変化を引き起こすシステムが必要である。その意味からは突然変異がGAにおける最適値探索において本質的であるともいえる。

GAでは，遺伝子型が親から子へとコピーされる際に特定のビットが反転することによって突然変化が発生する。たとえば8ビットの長さの遺伝子型がコピーされる際に

11011100 → 10011100

のように，コピーに際して2ビット目が反転するようなものである。

遺伝子型が突然変異を起こすとそれが表す表現型が変化する。そうして表現型は自然淘汰のフィルターにかけられるので，より適合度が高くなる方向での突然変異は生き残り，逆の突然変異は淘汰されることになる。この過程によって集団中に適合度が高い遺伝子型をもつ個体が発生し，増加していくのである。

2.5 交　　叉

「交叉」は突然変異と並んでGAにおいてより適合度が高い遺伝子を作り出すもう一つのシステムである。これはより多様な遺伝子型を持つ子孫を作るメカニズムであり，[6] 2親の遺伝子を混合した子孫を作ることになる。

2親がそれぞれ1つずつの染色体を持っているとすると，それらの染色体の一部分ずつを混合することによってあたらしい染色体を作り出すことが交叉である。

いま，10011010という遺伝子型を持つ個体と11001101という遺伝子型を持つ親から子が作られるとしよう。このとき遺伝子座（染色体の中での遺伝子が存在する位置）4で交叉が起きると，「10001101」と「11011010」という遺伝子を持つ2つの子が作られる。こうすることによって子は両親から遺伝子を一部分ずつ受け取ることになる。

上の例は遺伝子座の1点で交叉が起こる例であった。これを「一点交叉」と

---

6) 人の性細胞の減数分裂ではそれに先立って染色体が一部遺伝子を交換する場合がある。Robert (1985) 第5章参照。

呼ぶ。これに対して2点で交叉が起こる「2点交叉」，任意の各点で交叉が起こる「一様交叉」などの方式がある。

GAのプログラム中では，交叉が起こる確率を「交叉確率」でコントロールしている。

2.6 スキーマ定理

以上のGA過程を経ることによって集団はどのように変化していくのであろうか。この過程を分析するために遺伝子型が持つ構造を考える。

遺伝子型がバイナリコーディング（遺伝子型を2進数列でプログラムする）される場合には各遺伝子座には「0」か「1」が入る。そのように表現された遺伝子型にはある「パターン」が存在する。たとえば「11100101」という遺伝子型と「11001011」という遺伝子型はともに「1*****1」という構造を持っている。あるいは「11100111」という遺伝子型と「01001110」という遺伝子型は「*1*0****」という構造を持っている点では共通している。ここで示した「1*****1」や「*1*0****」の中に現れている「*」は「1，0のどれでもかまわない」ということを意味し「ワイルドカード」と呼ばれる。また，このパターンを「テンプレート」と呼ぶ。「1」「0」「*」およびテンプレートによって構成される文字列の集合を「スキーマ」と呼ぶ。たとえば，スキーマ H_1 と H_2 を

$$H_1 = 111*1*0*,\ H_2 = *1*0****$$

と定義すると，

$$11111100, 11101101, 11101000, \ldots \in H_1$$

$$01001110, 11101011, 11000111, \ldots \in H_2$$

のように具体的な遺伝子型はあるスキーマの要素となっている。上述のように遺伝子型が表現型を担い，個体の適合度を決定しているのであるから，適合度による個体の選択，すなわち生存に有利な表現型の遺伝は，つまるところ，「有利なスキーマの遺伝」ということに他ならない。そこで，どのようなスキーマが遺伝されやすいかを考えてみよう。

上で示した2つのスキーマ H_1 と H_2 はGA過程で発生する交叉においてどちらが破壊されやすいだろうか。明らかに H_1 のスキーマに属する文字列(これをスキーマのインスタンスと呼ぶ) は H_2 のインスタンスに比較して交叉によって破壊されやすい。交叉点がテンプレートの内部に入り込む確率が大きいからである。たとえば交叉点が4ビット目と5ビット目の間に入り込んだ場合，スキーマ H_1 が次世代に伝えられるためには交叉の対となる染色体が「$****1*0*$」というスキーマのインスタンスでなければならない。これに対してスキーマ H_2 はどんなスキーマのインスタンスと交叉をおこなっても次世代に伝えられる。

ここから，GA過程全体を通して H_1 のスキーマよりも H_2 のスキーマの方がGA過程を通じて生き残りやすいのではないかという直感が得られる。以下ではそのことを確かめよう。

スキーマのワイルドカード以外の左端文字位置と右端文字位置の間の距離をスキーマの「定義長 $\delta(H)$」とよぶ。たとえば

$$\delta(H_1) = 7-1 = 6,\ \delta(H_2) = 4-2 = 2$$

また，スキーマ中のワイルドカード以外の文字の数をスキーマの「オーダー $o(H)$」と呼ぶ。たとえば

$$o(H_1) = 5,\ o(H_2) = 2$$

すると，個体の遺伝子型が l ビットで表される場合，一点交叉が確率 p_c で起こるとすると，交叉点が定義長の内部に入り込む確率は

$$p_c\frac{d(H)}{l-1}$$

であるから,[7] 1点交叉によってスキーマ H が次世代に生き残る確率 p_s^H は

$$p_s^H \geq 1-p_c\frac{d(H)}{l-1}$$

---

7) この確率は下限である。というのは本文中の例で示したようにペアになる染色体のスキーマがあるパターンを持っていると交叉後にもスキーマを維持する場合があるからである。

さらに，突然変異が発生する確率を p_m とすると，オーダーのそれぞれのビットに関して突然変異が**起きない**確率は $(1-p_m)$ なので，スキーマ H が突然変異によって**破壊されない**確率は $(1-p_m)^{o(H)}$ となる。

さて，時点 t において，集団中に含まれるあるスキーマ H のインスタンスの数（スキーマ H を持つ個体の数）を $m(H,\ t)$，集団内の H に含まれる個体の適合度の平均を $\mu(t,\ H)=\sum_{x\in H}f(x)/m(H,\ t)$ とする。GA 過程での選択が上述のルーレット方式によって行われたとすると，上述のようにスキーマ H を持つ個体が生むと期待される子孫の数は $f(H)/\bar{f}$ であるから，時点 $t+1$ における H のインスタンスの期待値は，交叉と突然変異を無視すれば

$$E(m(H,\ t+1))=\sum_{x\in H}\frac{f(x)}{\bar{f}(x)}=\frac{\mu(H,\ t)}{\bar{f}(t)}m(H,\ t)$$

である。ここで $f(x)$ は文字列 x（遺伝子型）を持つ個体の適合度，$\bar{f}(t)$ は時点 t での集団の平均適合度である。

これに交叉と突然変異を考慮すると，次世代のスキーマ数の期待値は

$$E(m(H,\ t+1))\geq\frac{\mu(H,\ t)}{\bar{f}(t)}m(H,\ t)(1-p_c\frac{d(H)}{l-1})(1-p_m)^{o(H)}$$

この式は，(1)破壊されにくい低次オーダー・短い定義長で，(2)平均的より適合度が高いスキーマは時間の経過につれて指数的に増加することを意味している。GA 過程では短くしかも適合度が高いスキーマが積み重なることによってよりよい個体を生み出していくと考えられており，これを「スキーマ定理」と呼ぶ (Goldberg(1989))。

GA 過程で遺伝子型が進化していくということは，生存に有利なスキーマが集団中に増加していくことに他ならない。これは関数の最適化の問題に即して考えれば，最適解候補のうちからよりよい適合度（関数値）をもたらした個体の染色体に含まれるよいスキーマが積み重なって，最終的に最適解を見いだすということである。

3　GAの実行例

一般的に使用されるGAでは遺伝子型を各個体について1つ配列としてコーディングする。[8] これは静的な最大化問題については問題ない。多変数関数については変数をまとめて一つの配列で遺伝子型を表し，表現型への変換に際して遺伝子型を分割すればよいからである。[9] しかしそのようなコーディングでは動学問題を扱うことが困難である。動学問題では各期間のコントロール変数の値が表現型となるので，問題の期間が長くなればなるほど遺伝子型を一つの文字配列で表現することが困難になるからである。[10] 本稿で使用するプログラムでは多期間・多変数を扱えるように遺伝子型を多次元配列として表現できるように階層的なクラスを用いて作成してある（付録A参照）。

このプログラムは1期間・1変数のGAを含むのでいくつかの例題について実行した例を以下に示す。

3.1　多変数関数の最大化

ここで取り上げる最大化問題は伊庭が紹介したテスト関数である。これは

$$f_1 = \sum_{i=1}^{3}(-x_i^2),\ -5.11 \leq x_i \leq 5.12$$

を最大化する問題である。

変数の定義域は $-5.11 \leq x_i \leq 5.12$ であるから1ビットが示す増加分を0.01とすると，各変数につき10ビットの配列で遺伝子型を用意すれば定義域に対応できる。[11] 遺伝子型から表現型への変換は，この場合，各変数について

---

8) たとえば安居院・長尾（1993）のC言語ライブラリ

9) 伊庭(1994)で紹介されている基本関数によるテストではすべてこの方法が用いられている。

10) たとえば3変数20期間の問題の場合，各変数の遺伝子型を10ビットで表現すると $3\times10\times20 = 600$ ビットの長い列になる。本文中に述べたようにこのような長い遺伝子型では有利なスキーマが現れても破壊されやすい。

11) だいたい正負が対称であると考えてよいので，$\frac{5.12\times2}{0.01} = 1{,}024 = 2^{10}$

表現型（実数値）＝－5.11＋10 進数に変換された遺伝子型×0.01

で実現する。

これらの遺伝子型・表現型をもった個体を作成し，その個体を集めて集団をつくる[12]（初期集団）。個体の表現型は目的関数によって評価され，適合度へ変換される。この場合の適合度は関数値そのものである。適合度に応じて集団中で次の世代に遺伝子を残すかどうかが判定され[13]（選択），親として選択された個体はペアを組み染色体をそれぞれ交換して（交叉），新しい世代の個体をつくる。

GA 過程が進行していくと，集団内には非常に「似た」遺伝子型を持つ個体の数が増加していく。突然変異のメカニズムによって，集団内のすべての個体が完全に同じ遺伝子型を持つ可能性はほとんどないが，「似た」個体が集団中に増えると集団中の平均適合度はなかなか増加しない。そこで，「似た」遺伝子型が集団中に一定割合以上に増えた場合を「集団が収束した」と判定して，そこで GA を終了させる。個体間の遺伝子型がどの程度「似て」いるかを計る尺度としては，２つの遺伝子型の異なるビットの数（ハミング距離）を用いる。ここで用いている収束条件は集団中のハミング距離が全体のビット数の２％以下になることである。

さて，この問題の解は，$x_i = 0$ であることが直感的には明らかである。GA はこれを見いだすことができるであろうか。

このGA 過程の実行に使用したパラメータは以下の通りである。

集団の個体数	50
交叉の方式	一点交叉
交叉確率	0.6
突然変異確率	0.005

12）初期集団の遺伝子型は一様乱数によってランダムに決定している。

13）集団と全く同じサイズの親を選択した場合には高い適合度を持つ個体の遺伝子型も交叉によって破壊されてしまうため，もっとも高い適合度を持つ個体はそのまま次世代に残す手続きを行うことがある。これを「エリート戦略」と呼ぶ。本稿では集団内でもっとも高い適合度を持つ２個体はそのまま次世代へ残すようにしてある。

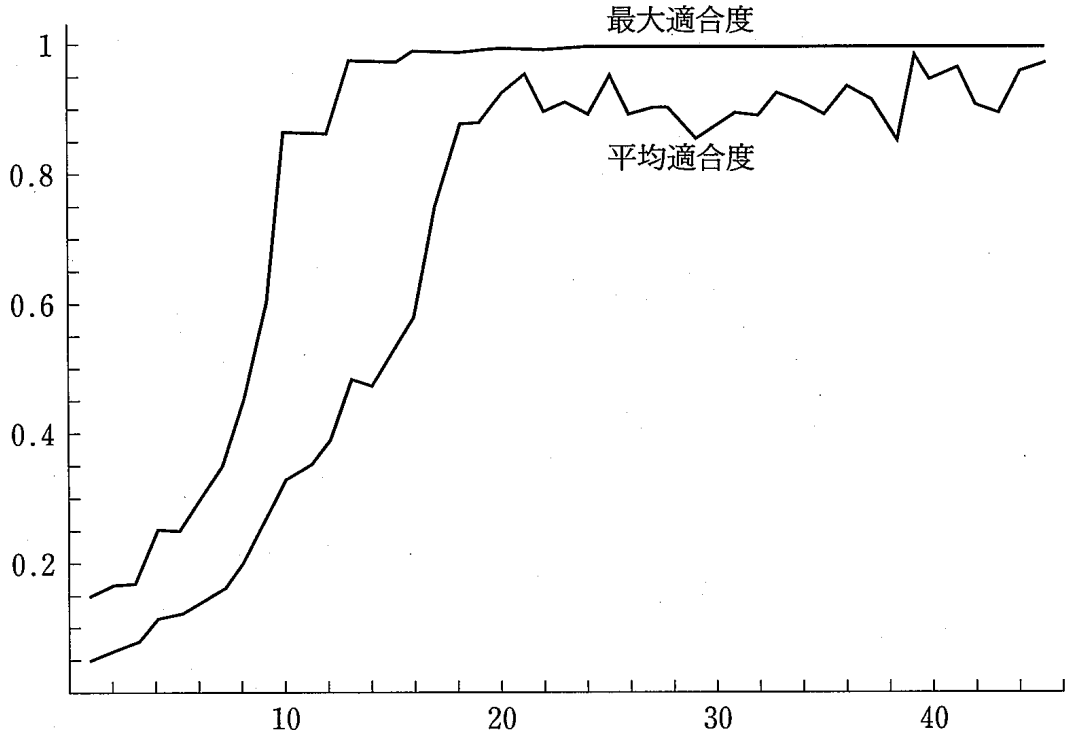

Fig 1：最大適合度と平均適合度

このパラメータの元でGAを実行した結果をFig 1に示す。グラフにはもっとも適合度が高い個体の適合度と集団の平均適合度が描かれている。ただし，適合度は$\frac{1}{1-f_1}$へ変換してあるのでグラフ中の最大適合度は1である。これは$f_1 = 0$に対応している。

このグラフはGA過程の典型的な様子を描いている。45世代で集団が収束し，もっとも適合度が高い個体は目的関数の最大値0を達成している。むろんその個体の表現型は$x_i = 0,\ i = 1, 2, 3$である。グラフからは判別しにくいが，集団の中での最適な個体は28世代目に出現している。GAは最適解を見いだすことができた。

一方，集団の平均利得は，最適個体が現れた以降微少な増減を繰り返しながら最適へ向かって収束している。ここでの収束判断はハミング距離を用いたものであるため，最後まで利得の低い個体も生き残っていることに注意しよう。

3.2 最適制御問題(1)：離散型

続いて1変数最適制御問題として，「ケーキの食べ問題」を取り上げる。これは「T 日以内に食べなければくさってしまうケーキをどのように食べればよいか」という問題で，離散型の多期間最適化問題として西村（1990）で例題に取り上げられている。

いま，0日にケーキを受け取り，それが腐って食べられなくなる期日が T 日後（$T-1$ 日までは食べられる），第 t 日に食べる量を u_t，第 t 日のはじめに残っている量を K_t とする。効用関数を $\sqrt{u_t}$ とすると，目的は $\sum_{t=0}^{T-1}\sqrt{u_t}$ を，制約 $k_{t+1}=k_t-u_t$ のもとで最大化することになる。ここでいくつかの思考をへて，問題は最終的に

$$\max_{k_t}\sum_{t=0}^{T-2}(\sqrt{u_t}+\sqrt{k_{T-1}})$$

$$s.t.\ k_{t+1}-k_t=-u_t,\ u_t\geq 0,\ k_t\geq 0$$

となる。[14)]

この問題の解は最適条件から導かれる階差方程式を解くことにより，

$$u^*(t)=\frac{k_0}{T},\ t=0,\ \ldots,\ T-2$$

で与えられる。

$T=10$ とした場合，サンメーションの中では $t=0,\ \ldots,\ 8$ が合計されることになるので，コントロール変数の期間としては9期間を仮定し，これを1変数多期間の最適化問題としてGAで解を求めた。GAの各パラメータは交叉確率を0.8，突然変異率を0.01としている以外は前節のパラメータとおなじである。なお，状態変数の初期値 $k_0=50$ としてあるので，理論上からの最適値は $u_t=5.0,\ (t=0,\ \ldots,\ 8)$ である。

実験は最大世代を4,000世代までとするGA過程の200回繰り返しを1

14) 西村（1990）の記述から定数項を省略してある。

セットとして，それを10セット行ったものである。最高値を出した実験は1，340世代で集団が収束し，もっとも適合度が高い世代の表現型，すなわち最適戦略は

$u[0]=5,\ u[1]=5,\ u[2]=5,\ u[3]=5.1,\ u[4]=5,$

$u[5]=4.9,\ u[6]=5.1,\ u[7]=5,\ u[8]=4.9$

であった。

これをみてわかるように，最適化の条件（この場合は階差方程式）を解くなどの方法をとることなく，遺伝子型の選択・交叉・突然変異という単純な繰り返しのみでほとんど理論上の最適値を得ている。

このような単純な例では理論的な最適値の計算の方が早くて正確であるので，GAによるアドバンテージは少ないが，以下の例のように複雑な問題ではそれは大きなものとなる。

3.3 最適制御問題(2)：連続型

上の例は離散型の問題であった。GAはコンピュータによる数値計算であるので離散型の問題については非常に適している。では連続型の問題ではどうであろうか。ここでは解析的にも解ける問題を用いて連続型の最適制御問題をGAがどの程度解決することができるのかをみる。

問題として取り上げるのは典型的な1変数連続型最適制御の問題である。[15)]

$$\max_{u_t}\int_1^5 (u_t x_t - u_t^2 - x_t^2)\,dt$$

$$s.t.\quad \dot{x} = x_t + u_t,\quad x_1 = 2.0$$

この問題の最適解は

$$u_t = \frac{-2e^{-\sqrt{3}(t+1)}\{e^{10\sqrt{3}}(3-\sqrt{3}) - e^{2\sqrt{3}t}(3+\sqrt{3})\}}{(3+2\sqrt{3}) - e^{8\sqrt{3}}(3-2\sqrt{3})}$$

15) Kemien and Schwartz (1982) p 120 Exercises 3 の問題。

となる。

この問題において上に示した解析的な解を求めて，それがどのような形になるのかを直感的に理解することは難しい。ここでのコントロール変数がなんらかの経済変数であるとして，必要なことはおおまかな解の形状であるような場合にはGAによるシミュレーション解で十分である。

GAでこのような連続型の問題を解く場合には，目的関数の積分として制約条件の微分方程式の評価が問題になる。ここでは積分の評価として単純な台形公式，微分方程式を解くアルゴリズムとしてRunge-Kutta法を用いている。[16)]

このケースでは積分区間が1.0から5.0であり，この区間を20分割してコントロール変数を求めている。GAの各パラメータは前節と同じで，やはり最大世代を4,000世代までとするGA過程の200回繰り返しを1セットとして，それを50セット，合計で10,000回の実験を行った。

Fig 2にもっともよい結果と理論から要請されるコントロール変数の形を示

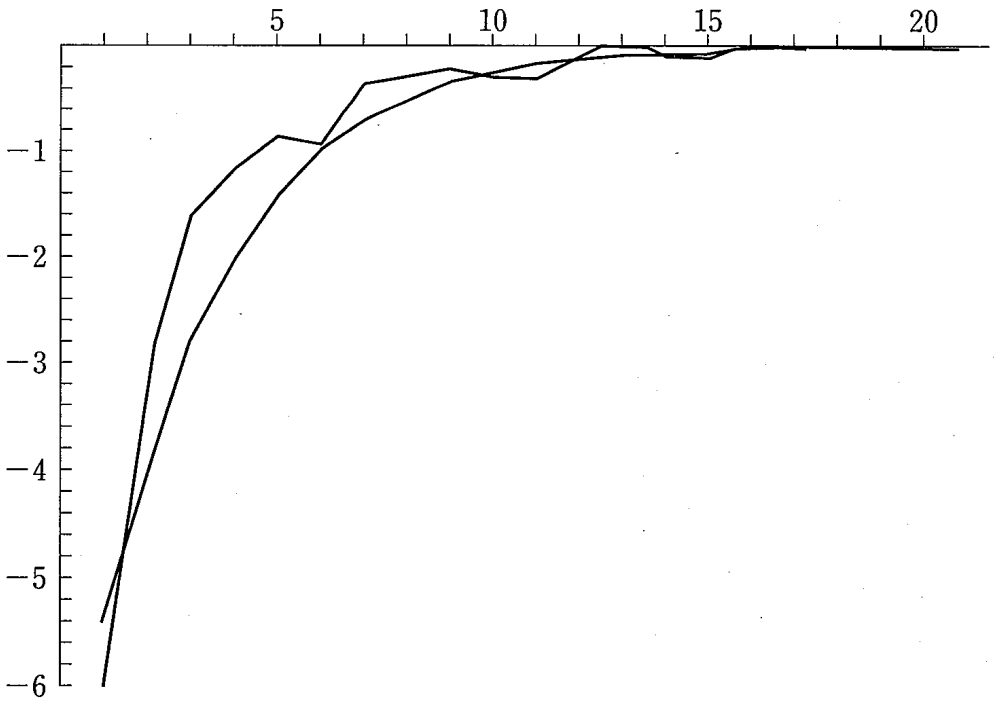

Fig 2：解析的な解曲線とGAによる解曲線

16) Plybon (1992) で紹介されているアルゴリズムを利用した。

す。なめらかな曲線が解析的な解曲線，そうでないものがGAの結果である。

GAによるシミュレーション解の形状をみると，部分的な区間で解析解から離れている部分はあるものの，おおまかな全体の形状は把握できる。この問題を解く場合に，通常は1階条件を導いて微分方程式を解くのであるが，GAではそのようなことは行っていない。単純に関数値を計算しているだけである。適合度の大きさによる選択と交配による単純な機構のみで上式のような関数がある程度の正確さで生み出されてくるのである。この点で，GAは最適解の探索に関して他の方法にくらべて大きなアドバンテージを持つものであるといえよう。

4 複雑な問題

前2節で見た問題はいずれも1変数の単純な問題であった。ここでは多変数で解析的な解が複雑な問題にGAを適用する。ここで取り上げる問題は筆者が分析したセキュリティ比率を求める問題である（安田（1996））。まず問題を簡単に述べておこう。

情報化の進展に伴い，新技術の開発に当たってはその漏洩を防ぐためのセキュリティ技術開発も必要となる。したがって，技術情報の外部へのもれを防ぐためには，限られた資源を技術そのものの開発投資とセキュリティへの投資へ分割しなければならない。

いま，t期において新技術の開発が成功するかどうかは開発しようとしている技術に関する知識ストック$Z(t)$の水準に依存しており，その技術をハッキングなどの外部からの攻撃をプロテクトできるかどうかはセキュリティに関する知識ストック$S(t)$に依存しているものとする。

$F(Z(t))$を知識ストック$Z(t)$以下で新技術の開発に成功する確率とすると，開発の“瞬間成功確率”は$\dot{F} = F'\dot{Z}$である。[17] 同様に$S(t)$以下ではセキュ

17) $\dot{x}$は$x(t)$の時間微分$\frac{dx}{dt}$をあらわす。

リティに失敗してしまう確率を $\varphi(S(t))$ とすると，“瞬間失敗確率”は $\varphi'\dot{S}$ である。[18)]

t 期に投入できる技術資源を $u(t)$ とし，そのうち，セキュリティ技術の開発に振り分ける比率を $\delta(t)$ とすると，$\dot{Z}$, $\dot{S}$ はそれぞれ

$$\dot{Z} = (1-\delta(t))u(t),\ \dot{S} = \delta(t)u(t)$$

ただし，$u(t)$ には上限があって $0 \leq u(t) \leq B$ であるとしておく。

また，開発に成功した場合の利得を Q，セキュリティに失敗した場合の損害を C，投入する技術資源の費用を $f(u)$ としておくと，この技術開発に関する t-期の期待利得は次のようになる。

- 開発成功の期待利得‥$QF'(1-\varphi)(1-\delta(t))u(t)$：開発に成功し，かつセキュリティも完全
- セキュリティ失敗時の期待損失‥$C\varphi'\delta(t)u(t)$：開発の成否に関わらずセキュリティに失敗したときには損害を受ける
- 開発失敗時の期待費用‥$(1-F)f(u)$

するとこの最適なセキュリティ比率を求める問題は以下のように定式化される。

$$\max_{u(t),\delta(t)} \int_0^T QF'(1-\varphi)(1-\delta(t))u(t) - C\varphi'\delta(t)u(t) - (1-F)f(u)dt$$

$$s.t.\quad \dot{Z} = (1-\delta(t))u(t),\ \dot{S} = \delta(t)u(t)$$

この解は，$\delta(t)$ に関して Bang-Bang 解になっており，$\delta(t)$ は 0 と 1 を交互にスイッチし，$\delta(t)=0$ の期間は $u(t)$ は上昇し，$\delta(t)=1$ の期間は $u(t)$ は一定であることが解析的な検討からわかっている。2 つのコントロール変数の対応は

$$\delta(t)=0 \text{ のとき } \dot{u} = \frac{1}{2}\theta u(t)$$

$$\delta(t)=1 \text{ のとき } \dot{u} = 0$$

18) $\lim_{z\to\infty} F(Z)=1$, $F(0)=0$, $F'>0$; $\lim_{S\to\infty}\varphi(S)=0$, $S(0)=1$, $S'<0$ と仮定する。

となっている。また，$\delta(t)$ のスイッチング時刻は特定化することができない(安田（同上))。

このように，多変数でしかもコントロール変数の1つがBang-Bang解であるような複雑な問題に対してGAはどの程度この特徴を表現することができるであろうか。これがこの節での課題である。

GAで取り扱いができるように関数 F, S を以下のように特定化しよう。

$$F(Z(t)) = 1-e^{-\theta Z(t)}$$

$$\varphi(S(t)) = e^{-\phi S(t)}$$

$$f(u(t)) = \frac{1}{2}u^2(t)$$

さらに，問題を離散型に変更し，$T = 40$ とすると，

$$\max_{u_t,\delta_t} \sum_{t=0}^{40} Q\theta e^{-\theta Zt}(1-e^{-\phi St})(1-\delta_t)u_t + C\phi e^{-\phi St}\delta_t u_t - \frac{1}{2}e^{-\theta Zt}u_t^2$$

$$s.t.\quad Z_t - Z_{t-1} = (1-\delta_t)u_t,\ S_t - S_{t-1} = \delta_t u_t$$

GAのために $Q = 400$, $C = 200$, $\theta = 0.05$, $\phi = 0.05$ という数値をあたえ，コントロール変数 u_t を表す染色体の長さ（ビット列の長さ）を9とし，1ビットの表現を0.01とした。これは $2^9 = 512$ なので，u_t の上限 B を5.11としたことになる。また，$u_0 = 1.0$, $\delta_0 = 1.0$ として初期値を与えてある。もう一方のコントロール変数 δ_t の表現には1ビットをあたえた。これは δ_t がBang-Bang解であることに対応している。

このGA過程の実行に使用したパラメータは以下の通りである。

Fig. 3(1)はシミュレーションによってえられたコントロール変数 u_t である。理論上からは $\delta_t = 0$ に対応するそれは増加を続けねばならないが，そうなってはいない。また $\delta_t = 1$ に対応する u_t の解は，パラメータがどのような値であれ，一定値をとるはずであるが，δ_t が，1から0へ切り替わる前に増加を始めている。いったん増加が始まると（16期め），u_t は急速に大きくなり，ほとんど上限の近辺で推移している。

集団の個体数	50
交叉の方式	一点交叉
交叉確率	0.8
突然変異確率	0.001

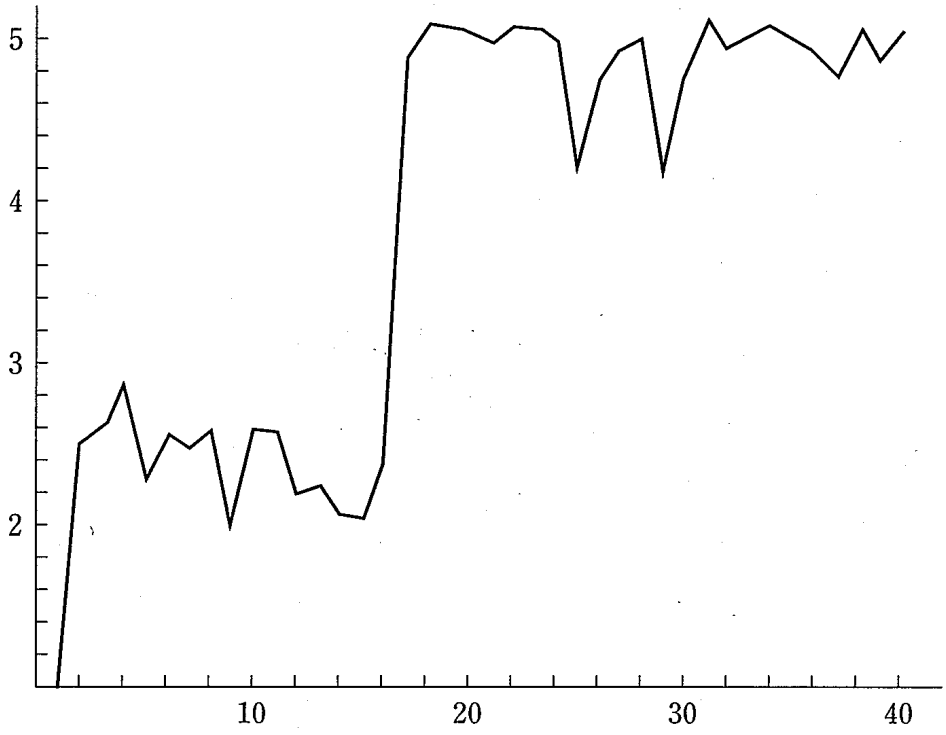

Fig 3(1) : GA が見いだした u_t

ここでのパラメータを用いると，$\delta_t = 0$ に対応する u_t の最適解は $\dot{u} = \frac{1}{20}u_t^2$ であるから，u_{t-1} の値が，ある程度大きければ (たとえば $u_{t-1} = 3.0$) であれば，u_t はすぐに上限に近づく ($u_t = 3.45$, $u_{t+1} = 4.40$, $u_{t+2} = 5.1$)。これは u_t の上限 B を大きくしても同じことでやはり同じように，u_t の増加が始まると急速に上限値に張り付く。その意味からは $\delta_t = 0$ となるより先に増加がはじまるものの，$\delta_t = 0$ の期間に u_t が上限近辺に張り付くことは理論的な考察からもいえることである。

一方，Fig 3(2)はシミュレーションによってえられた δ_t のグラフである。注目すべき箇所は，δ_t の切り替えが 24 期めにおこっており，スイッチングの特徴が現れていることである。スイッチング後 0 が続く。つまり形状としては解析的

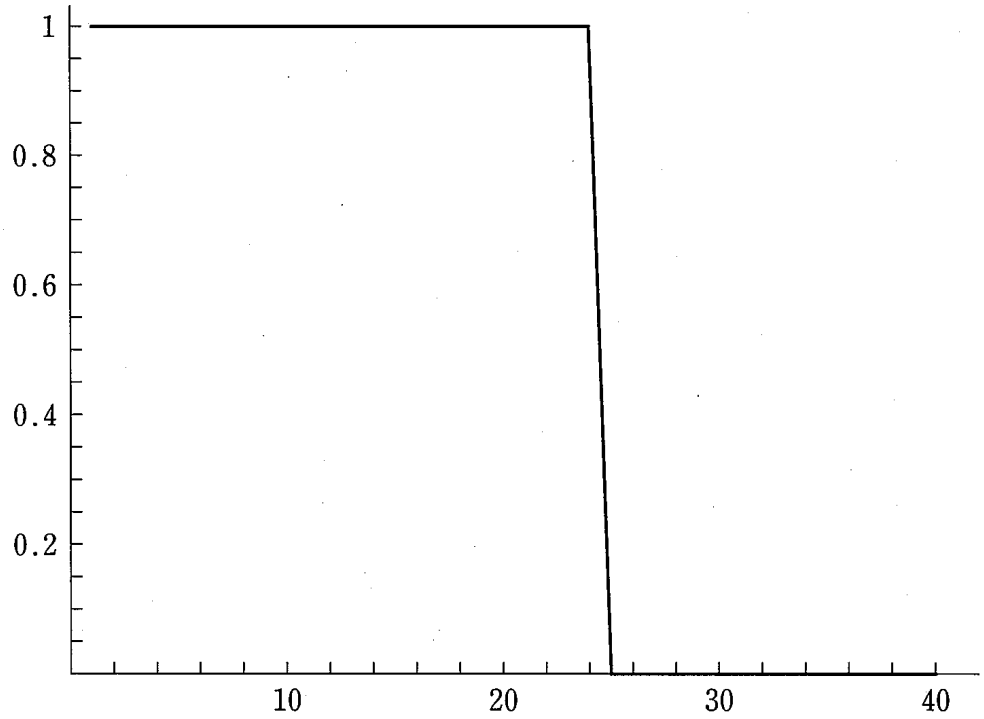

Fig 3(2) : GA が見いだした δ_t

な検討結果に合致する。また，このグラフは δ_t 初期値として1を与えたものであるが，この初期値を与えなくとも GA は24期まで $\delta_t = 1$ を見いだしている。

以上をまとめると，

- GA はこの問題の解の特徴をうまくとらえた（$\delta(t)$ のスイッチ）。
- それに伴う $u(t)$ の切り替え（一定値から上昇へ）もとらえている。
- しかし，スイッチ時刻が $u(t)$ と $\delta(t)$ で一致していない。

ということになる。特に $u(t)$ の形状に関しては解析的な検討と一致しない部分が多いが，それでもこのような複雑な問題に関して，理論から要請される特徴がある程度再現できていることは注目すべきことであろう。

解析的な検討から要求される基準を GA が満足に十分には満たしていない理由の一つとして，モデルの形があげられる。このモデルの理論的な検討では連続型の問題を使用して解を導いており，シミュレーションでは離散的なモデルとなっている。このこともコントロール変数 u_t の形状を再現できない理由のひとつとなっているかもしれない。

5 今後の課題

以上でみたように，Bang-Bangを含む多変数での問題に関してはGAはうまくは働かなかったが，それ以外の問題に対してはGAは非常にうまく働き，解を見いだしてくる。最後の節で取り扱った問題に対しても解の特徴の一部(δ_tのスイッチが起こること，それに対応するu_tの増加がみられること)を再現しており，最適制御の数値シミュレーションの方法としてGAはかなり有効であると考えることができる。

本論文で取り上げた例はいずれも1つの集団を用いたが，これを複数集団として相互に関連させることも可能であり，ゲーム理論への適用，特に進化ゲームへの適用が可能であろう。その方面へのシミュレーションの発展を今後の課題としたい。

Referenaces

Holland (1975)

John H. Holland

Adaptation in natural and artificial systems ; University of Michigan Press 1975

Goldberg (1989)

David E. Goldberg

Genetic Algorithms in Search, Optimization & Machine Learning ; Addison Wesley 1989

Kemien and Schwartz (1982)

Morton I. Kamien and Nancy L. Schwartz

Dynamic Optimization ; North Holland 1982

Bauer (1994)

Richard J. Bauer, Jr Genetic Algorithms and Investment Strategy ; Wiley 1994

Dawid (1996)

Herbert Dawid

Adaptive Learning by Genetic Algorithms

Lecture Notes in Economics and Mathematical Systems 441 ; Springer 1996

Plybon (1992)

Benjamin F. Plybon
An Introduction to Applied Numerical Analysis ; PWS-KENT 1992
Özyildirim (1997)
Computing open-loop noncooperative solution in discrete dynamic games
Journal of Evolutionary Economics vol 7 pp 23-40
Mitchell (1996)
Melanie Mitchell
An Introduction to Genetic Algorithms
(邦訳：「遺伝的アルゴリズムの方法」伊庭斉志監訳 東京電気大学出版局 1997年)
Robert (1985)
Robert Trivers
Social Evolution
(邦訳：「生物の社会進化」中嶋・福井・原田訳 産業図書 1991年)
伊庭 (1994)
伊庭斉志
「遺伝的アルゴリズムの基礎—GAの謎を解く—」オーム社 1994年
西村 (1991)
西村清彦
「経済学のための最適化理論入門」東京大学出版 1991年
安居院・長尾 (1993)
安居院猛・長尾智晴
「ジェネティックアルゴリズム」昭晃堂 1993年

A GAプログラムで使用したクラス

一般的なGAは一元配列を遺伝子型とする場合が多いが，本文中に述べたように本稿での目的はより一般的に多変数・多期間を取り扱うことであるため構造が複雑になる。そこでプログラムの構成自体を生物モデルのように階層化することにより多変数・多期間という重層的な構造を表現しようと試みた。

そのようなコーディングにはC++のクラス概念を利用するのがもっともよい。基本的なクラスは以下の通りである。

CGenUnitクラス このクラスは遺伝単位を表現する。遺伝子としてバイナリ文字列の配列を持ち，その表現型である実数値を保持する。この部分で多変数を維持できるように，各変数のバイナリ表現（遺伝子型）をまとめた上で1つの遺伝子型とし，表現型は変数の数に応じた長さの実数配列として保持するようになっている。

たとえば変数が3つある場合，それぞれの遺伝子型を10ビット表現で定義するなら

ば，30 ビットの長さのバイナリ配列と，長さ 3 の実数配列を持ち，実数配列に表現型が格納される。いわば 30 の長さのバイナリ配列が染色体を意味することになる。

CIndividual クラス このクラスは個体を表現する。CGenUnit クラスを期間分の配列としてもち，多期間を表現できるようにしてある。たとえば 40 期間の問題を取り扱う場合には長さ 40 の CGenUnit 配列を持つ。また，このクラスは自己の適合度の値，集団中での地位（適合度による集団内での順位），地位に基づく親となる確率を保持する。これを使って選択・交配がこのクラスを対象に行われる。

CPopulation クラス このクラスは集団を表現する。集団の現世代のメンバとして CIndividual クラスの配列を持ち，個体の評価関数をもつ。このクラスをいったん生成すると，初期集団を作成した後，評価関数に基づいた個体の順位付け・選択を行い次世代の「親」を決定する。その「親」は CIndividual クラスの配列として保持される。また，「子」として CIndividual クラスの配列をもっており，次世代にその「子」が集団のメンバに置き換わり，再び親が選択されるような関数（NextGen）を持っている。

これらのクラスを使うと，GA 過程は CPopulation クラスで決定された親の染色体を対象に行えばよい。アルゴリズムの流れは以下のようになる。

Step 1 CPopulation を生成。

Step 2 CPopulation の「親」を取り出し，ランダムにペアを組む。

Step 3 ペアリングされた親の各期間に対応する染色体に対して交叉・突然変異を発生させる。

Step 4 Step 3 で変化した染色体を集団の「子」へコピーする。

Step 5 「子」によって集団中のメンバを置き換え，再び親を選択。

Step 6 集団のハミング距離を測定。収束していなければ **Step 2** へ戻る。

本稿でもちいたプログラムは筆者のホームページ経由で利用可能である。[19]

19) Macintosh 用として Code Warrior™ のプロジェクトファイルとソース，GNU GCC 用として makefile とソースを準備している。