徐冷型エボルーションナリー・プログラミング による大域最適化とニューラルネットワーク によるシステム同定問題への応用*

島 孝 司**

Global Optimization by Annealing Type of Evolutionary Programming and Its Application to System Identification Problem Using Neural Networks*

Takashi SHIMA**

The global optimization procedures which simulate the natural evolution are known as Evolutionary Programming or Simulated Evolution. We utilize several concept from natural evolution and construct the global optimization procedure. The features of the proposed method are summarized as follows :

- 1) Introduction of the concept of temperature to the reproduction process
- 2) Adjustment of the temperature and metric ; in early stages the generated points can be moved drastically in order to escape from the local minimum and the convergence to the global solution in the final stage should be satisfied.
- 3) The reproduction and selection processes are simple.

The numerical examples including Shekel function type problem are solved and the proposed method hardly misses the global solutions. The identification problem of a nonlinear system by neural networks is also solved successfully. The proposed method is very promising and can be applied to many types of real problems.

1. はじめに

本論では、進化論からいくつかの概念を借りて構成した、連続関数の大域最適化手法を提案する。自然界の進化を模倣した手法は、エボルーションナリー・プログラミング (EP) あるいはシミュレーテッド・エボルーション (SE)^{3,8)} として知られている。現在、研究の盛んな遺伝的アルゴリズム (GA) も、それらの手法の一つとして派生したものである⁹⁾。本論での目的は、自然界の進化を出来るだけ忠実に模倣しようとした手法の提案ではない。つまり、個々には進化論から借りた概念を使用しているが、全体として進化論的な意味づけを意識した手法の提案ではない。冒頭に述べたように、自然界の進化からいくつかの有用な概念を借りて構成した、大域最

適化手法の提案である。本手法の特徴、これまでの EP との差異は以下のように要約される。

- (1) 子の再生に温度の概念を導入したこと。
 - (2) 初期の世代では、活発に動き回ることにより局所最適解を脱出し、世代が進むに従って大域最適解そのものへ収束するよう、温度や距離を調整すること。
 - (3) 再生、淘汰の過程が非常に単純で簡素なこと。
- 等である。

温度と距離という簡単な概念の導入で、数度の試行に対してもほぼ逃すことなく、大域最適解そのものを発見する、非常に単純で簡素化された有効な手法が構成できる。これを示すことが本論の目的である。

以下では、まず、アルゴリズムの詳細を紹介する。つぎに、Shekel 関数などのテスト関数を用いた数値例を解く。また、類似手法との比較も行。最後に、ニューラルネットワークを用いた非線形システムの同定問題へ応用する。本論で提案する手法は、数値例に見られるように、ほぼ逃す事なく大域最適解を見つけ出しており、

* 原稿受付 1993 年 9 月 30 日

** 金沢女子短期大学 情報処理科 Kanazawa Women's College ; 10 Suemachi, Kanazawa city, Ishikawa 920-13, JAPAN

Key Words : global optimization, evolutionary programming, annealing, system identification, neural networks.

同定問題への応用に見られるように、いろいろなタイプの実システム問題への適用についても非常に期待できる手法である。

2. 徐冷型のエボリューションナリー・プログラミング (EP) による大域最適化について

本論では、以下のような大域最適化問題を考える。

$$\min f(\mathbf{x}) \text{ subject to } \mathbf{x} \in X \quad (1)$$

ここで、 $f(\mathbf{x})$ は多峰性の連続関数、 $\mathbf{x}$ は n 次元ベクトル、 X は n 次元空間 R^n の超立方体部分空間である。

前述したように、本手法は全体としては進化論的な意味づけを意識した手法ではない。ただ、進化論からいくつかの概念を借りて構成した手法であるため、本論では、これを徐冷型のエボリューションナリー・プログラミングと呼ぶことにする。

本手法では N 個の個体を考える。これら N 個の個体は複製子 (親) として働き、それぞれ、ただ一個の個体 (子) を再生するものと仮定する。

まず、一様分布に従い、最初の親 $\mathbf{x}_{p,i} (i=1, \dots, N)$ をランダムに X 内に発生させる。つぎに、各世代において、親は子を以下のように作成する。

$$\mathbf{x}_{c,i} = \mathbf{x}_{p,i} + \delta \mathbf{x}_i \quad (2)$$

$\mathbf{x}_{p,i}$ は i 番目の親で n 次元ベクトルである。 $\mathbf{x}_{c,i}$ は $\mathbf{x}_{p,i}$ により作られた子、 $\delta \mathbf{x}_i$ はある確率分布に従う n 次元の確率変数からなるベクトルである。 $\delta \mathbf{x}_i$ はどのような確率分布に従って発生されてもよいが、本論では、ロングジャンプが可能な、局所解を脱出するのに非常に都合の良い性質を持つ Cauchy 分布に従って、次のように発生されるものとする⁹⁾。

$$\mathbf{x}_{c,i,j} = \mathbf{x}_{p,i,j} + T_k \tan P_{i,j} \quad (3)$$

ここで、 $j (=1, 2, \dots, n)$ は要素を示す。 T_k は係数で温度を表わす。 k は世代を示す番号、 $P_{i,j}$ は $(-\pi/2, \pi/2)$ の間の値をとる一様乱数である。温度は世代に依存し、初期の世代では激しく変化するように高温に、世代を経るに従って大域最適解に収束するように減温される。このような徐冷型の EP は、著者の知る限り、全く新しい試みである。

上記の再生により全部で $2N$ 個の個体が存在することになるが、適応度により N 個の個体が淘汰され、残った N 個が次の複製子 (親) $\mathbf{x}_{p,i} (i=1, \dots, N)$ として働くことになる。よって適応度の高いものは、何世代も生き残ることになる。また、各世代で常に N 個の個体が存在することになる。一つの個体が一つの n 次元ベ

クトル $\mathbf{x}_{*,i}$ (添字 $*$ は p または c を示す、 $i=1, \dots, N$) に対応する。適応度としては目的関数値を用い、目的関数の値が良ければ適応度も高いことになる。この再生、淘汰の過程が非常に単純であることが本手法の特徴でもある。

さて、適応度の高いものが何世代も生き残ることになるが、親の適応度が高ければ、子の適応度も高くなる確率が大い。これが何世代も続くと同じような個体ばかりになってしまい、行き詰まる。つまり、局所最適解に陥り易くなる。これを防ぐために距離の概念を導入する。子を再生するときに、それぞれの子がある程度の距離だけ離れて作られるようにする。これは、最初の親をランダムに決定するときも同様で、個体どうしがある程度の距離だけ離れて作られる。ただ、親と子の間では、距離の概念は適用されないと仮定する。本論では、以下のような距離を採用する。

$$L_{h,i} = \|\mathbf{x}_{*,h} - \mathbf{x}_{*,i}\| = \sqrt{\sum_{j=1}^n (\mathbf{x}_{*,h,j} - \mathbf{x}_{*,i,j})^2} \quad (4)$$

ここで、 $*$ は p または c 、 $L_{h,i}$ は h 番目と i 番目の個体の間の距離を示す。

また、適応度の高いものは何世代も生き残ることから、同じような個体ばかりになり、局所最適解に陥ってしまうのを防ぐために、突然変異の概念も利用する。突然変異率を r とする。 $N_r (= N \times r)$ 個の個体をランダムに発生させる。ただし、 N_r は整数とする。各個体に $N - N_r + 1, N - N_r + 2, \dots, N$ と番号を付け、各世代で生き残った N 個の個体のうち適応度の良い順に並べたものの同じ番号を持つもの、つまり、 $N - N_r + 1, N - N_r + 2, \dots, N$ 番目のものと同じ番号の個体 (適応度の悪いもの N_r 個) の適応度を比べる。もし、突然変異の個体の適応度が良ければ置き換える。適応度が良い場合のみ置き換えを行うので、純粋な突然変異とはやや違っている。

上記説明に添ってアルゴリズムをまとめると、以下のよう表わされる。

- step 1 世代番号 $k=0$ とする。
step 2 N 個の n 次元ベクトル $\mathbf{x}_{p,i}$ (初期の親; $i=1, \dots, N$) を X 内に

$$L_{h,i} > L_k \quad (5)$$

$$h=2, 3, \dots, N, i=h-1, h-2, \dots, 1$$

を満たすよう、一様分布に従い、ランダムに発生させる。世代番号 $k=1$ とする。

- step 3 子を X 内に (3) 式に従って、距離 (5) 式を満たすよう発生する。

step 4 N 個の親と N 個の子の全部で $2N$ 個の個体が存在するが、これらを適応度（目的関数値）の良い順に並べ替え、良いものから順に N 個の個体を残し、次世代の親 ($\mathbf{x}_{p,i}, i=1, \dots, N$) とし、悪いもの N 個を淘汰する。

step 5 適応度の一番良いものとその適応度（目的関数値）($\mathbf{x}_1[k], f_1[k]$ と置く)を記憶する。また、 N 個の適応度の平均値 ($avf[k]$)も記憶する。

step 6 突然変異として N_r 個の n 次元ベクトル $\mathbf{x}_{m,N-N_r+i} (i=1, 2, \dots, N_r)$ を X 内にランダムに発生し、以下のように置き換える。

$$\begin{aligned} \mathbf{x}_{p,N-N_r+i} &= \mathbf{x}_{m,N-N_r+i} \\ \text{if } f(\mathbf{x}_{p,N-N_r+i}) &> f(\mathbf{x}_{m,N-N_r+i}) \\ \text{otherwise } \mathbf{x}_{p,N-N_r+i} &= \mathbf{x}_{p,N-N_r+i} \quad (6) \end{aligned}$$

ここで、 $i=1, 2, \dots, N_r$ である。

step 7 $k=k+1$ とする。 $k \leq k_f$ ならば step 3 へ戻る。 $k > k_f$ ならば終了。適応度の一番良いもの ($\mathbf{x}_1[k_f]$) を大域最適解とし、その適応度（目的関数値： $f_1[k_f]$) を大域最適値とする。

ここで、 L_k は k 番目の世代が、お互いにとるべき距離である。

(5) 式を満たす個体の発生法として、 h 番目の個体をランダムにまず発生し、先に発生された $h-1, h-2, \dots, 1$ 番目の個体とそれぞれ距離が L_k 分離れていれば、これを採用し、どれか一つとでも、距離が L_k 分離れていなければ、これをリジェクトし、再度、 h 番目の個体をランダムに発生するという方法を使用する。ただ、この方法ではリジェクトの作業をする分、計算時間がかかることになる。

突然変異においても、距離の概念を取り入れる事が出来る。また、突然変異が終了後、再度並べ替えを行った方がより厳密ではあるが、たくさんの世代を経た後では、あまり違いは見られない。本手法では、突然変異を採用しているが、EP では突然変異を省くことも可能である。その場合、多様性が少し失われるため、局所解に収束する危険性がやや高くなる。

(3) 式のなかの温度係数は、世代を経るに従って徐冷される。また、距離も世代を経るに従って小さくされる。よってアルゴリズム終了時には、 N 個の個体は大域最適解の周辺に集まる。もし距離を導入しなければ、アルゴリズムの終了時ではなく中間時に、局所解の周辺に集まる危険性が高くなる。

本手法では、アルゴリズムの終了時に最も適応度（目的関数値）の良いものを、大域最適解とみなしている。しかし、本手法では、上に述べたように、終了時に大域

最適解の周辺に個体が集まる。ある程度以上の個体がある点に集まれば、それを大域最適点とみなす方法も可能かもしれない。現時点では安全性の面から適応度の最も良いものを大域最適解とすることにし、ある程度以上の個体がある点に集まれば、それを大域最適点とみなす方法は今後の課題としたい。

本手法では、個体数、世代数、温度変化、距離の変化、突然変異率などは、最初は適当に定め、良い結果が得られるまで、試行錯誤的に変更する。個体数は $N=20 \sim 50$ 程度、世代数は $k_f=50 \sim 500$ 程度、温度は世代全般に渡って 0.25 程度から 0.001 に変化するように、距離は世代全般に渡って 5 程度から 0 に変化するように、突然変異率は $r=0.2$ 程度に選ぶと、多くの場合良い結果を得る。しかし、これらのパラメータの決定や徐冷スケジュール、距離のスケジュールの決定が、本アルゴリズムの非常にクリティカルな部分となっており、今後の課題でもある。とくに、温度が小さいのに距離が大きいと、子を発生するのに非常に時間がかかり、最悪の場合、発生しえない可能性もある。

3. 数値例と手法の比較

以下のテスト関数の大域最適化問題を、提案したアルゴリズムで解くことを考える。

3.1 数値例

【例 1】Shekel 関数

$$\max f(\mathbf{x}) = - \sum_{i=1}^m \frac{1}{\{(\mathbf{x}-\mathbf{a}_i)^T(\mathbf{x}-\mathbf{a}_i)+b_i\}} \quad (7)$$

subject to $\mathbf{x} \in X$

$$X = \{\mathbf{x} | 0 \leq x_i \leq 10 (i=1, \dots, n)\} \quad (8)$$

ここで、

$$\mathbf{x} = (x_1, \dots, x_n)^T$$

$$\mathbf{a}_i = (a_{i1}, \dots, a_{in})^T$$

$$b_i > 0, i=1, \dots, m, n=8, m=5$$

である。

(7) 式は $n=8$ より、8 次元の Shekel 関数と呼ばれる。各係数は Table 1^D のように与えられる。大域最適

Table 1 Coefficients in example 1

	$a_{i,j}$								b_i
i	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$	$j=8$	
1	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	0.1
2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.2
3	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	0.2
4	6.0	6.0	6.0	6.0	6.0	6.0	6.0	6.0	0.4
5	3.0	7.0	3.0	7.0	3.0	7.0	3.0	7.0	0.4

Table 2 Temperature cooling schedule in example 1 (k : generation)

T_k	k
0.25	1 ~ 20
0.2	21 ~ 40
0.167	41 ~ 60
0.125	61 ~ 80
0.1	81 ~ 100
0.09	101 ~ 120
0.08	121 ~ 140
0.07	141 ~ 160
0.05	161 ~ 180
0.025	181 ~ 200
0.01	200 ~ 220
0.008	221 ~ 240
0.005	241 ~ 260
0.0025	261 ~ 280
0.001	281 ~ 300

解は $\mathbf{x} = (4.000, 4.000, 4.000, \dots, 4.000)^T$ であり, 大域最適値は $f^{opt} = -10.077$ である. この問題を提案したアルゴリズムを用いて 5 回異なる乱数の種を与え, 解く (5 回マルチスタートさせる).

この例題では, 個体数 $N = 50$ とする. 全部で $k_f = 300$ 世代を考える. 温度変化は Table 2 に示すように与える. 増分発生式である (3) 式には, 小さな係数が乗算されていないため, 何千度という温度を使用する必要はない. また, 距離は以下のようにとる.

$$L_k = 5 \text{ for } k = 0, 1 \sim 20$$

$$L_k = 4 \text{ for } k = 21 \sim 100$$

$$L_k = 2 \text{ for } k = 101 \sim 180$$

$$L_k = 1 \text{ for } k = 181 \sim 260$$

$$L_k = 0 \text{ for } k = 261 \sim 300$$

ここで, 突然変異率 r を $r = 0.2$ とする.

5 回分の解を得たが, 大域最適解を逃すことは全くな

Fig. 1 $\bar{f}_1[k]$ and $\overline{avf}[k]$ in example 1

かった. $f_1[k]$ と $avf[k]$ の 5 回分の平均値を $\bar{f}_1[k]$ と $\overline{avf}[k]$ とし, その収束状況を Fig. 1 に示す.

【例 2】 Six-hump camelback 関数

$$\min f(x_1, x_2) = (4 - 2.1x_1^2 + x_1^4/3)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2 \quad (9)$$

$$\text{subject to } -3 \leq x_1 \leq 3, -2 \leq x_2 \leq 2 \quad (10)$$

この問題の局所解は 6 点存在し, そのうちの 2 点 $(-0.0898, 0.7126)^T, (0.0898, -0.7126)^T$ が大域最適解であり, 大域最適値は $f^{opt} = -1.0316$ である. 先の例題と同様に, この問題を提案したアルゴリズムを用いて, 異なる乱数の種を与え, 5 回マルチスタートさせて解く. 2 次元の簡単な例題なので, $N = 20$ とし, 全部で 50 世代を考える. 温度変化は以下に示すように与える.

$$T_k = 0.25 \text{ for } k = 1 \sim 10$$

$$T_k = 0.1 \text{ for } k = 11 \sim 20$$

$$T_k = 0.05 \text{ for } k = 21 \sim 30$$

$$T_k = 0.01 \text{ for } k = 31 \sim 40$$

$$T_k = 0.001 \text{ for } k = 41 \sim 50$$

また, 距離は以下のようにとる.

$$L_k = 0.2 \text{ for } k = 0, 1 \sim 20$$

$$L_k = 0.1 \text{ for } k = 21 \sim 40$$

$$L_k = 0.01 \text{ for } k = 41 \sim 50$$

突然変異率 r を $r = 0.2$ とする. 5 回の試行において, 大域最適解を逃すことは全くなかった.

例 1 と同じく, $f_1[k]$ と $avf[k]$ の 5 回分の平均値を $\bar{f}_1[k]$ と $\overline{avf}[k]$ とし, その収束状況を Fig. 2 に示す. また, 2 次元の単純な問題であり, $N = 20$ と個体数も少ないので, 4 回目の試行の結果として全個体の持つ値を適応度 (目的関数値) の良い順に Table 3 に示す. (ただし, 小数点以下 4 ケタまで表示する.) この試行では二つの大域最適解の周辺にそれぞれ 1/2 の確率で個体が集まっている. もちろん乱数の種が違えば 1/2 とは異なる確率で棲み分ける結果が得られる.

Fig. 1 や Fig. 2 からおよその収束状況が分かるが, 例題において, 乱数の種にもよるが, 大域最適解を逃す

Fig. 2 $\bar{f}_1[k]$ and $\overline{avf}[k]$ in example 2

Table 3 Individuals in example 2

Individual No.	x_1	x_2	f_{opt}
1	0.0898	-0.7124	-1.0316
2	0.0902	-0.7127	-1.0316
3	-0.0896	0.7124	-1.0316
4	0.0892	-0.7124	-1.0316
5	0.0898	-0.7121	-1.0316
6	-0.0901	0.7120	-1.0316
7	-0.0902	0.7117	-1.0316
8	0.0913	-0.7139	-1.0316
9	0.0921	-0.7138	-1.0316
10	0.0919	-0.7149	-1.0316
11	0.0931	-0.7144	-1.0316
12	-0.0896	0.7099	-1.0316
13	0.0922	-0.7151	-1.0316
14	-0.0939	0.7116	-1.0316
15	0.0931	-0.7151	-1.0315
16	-0.0939	0.7114	-1.0315
17	-0.0944	0.7120	-1.0315
18	-0.0937	0.7099	-1.0315
19	-0.0940	0.7092	-1.0315
20	-0.0936	0.7085	-1.0314

ことはなく、提案しているアルゴリズムが非常に有効に働いているのが分かる。

3.2 類似手法との比較

本手法と同様に進化の概念を適用した大域最適化手法に GA や、従来からの EP がある。本節では、これらの手法との比較を行ってみたい。

3.2.1 定性的な比較

EP の側からみた GA との比較が、参考文献 3) に詳細に述べられている。とくに、交叉が比較の対象となっているが、それらに関しては参考文献 3) を参考にしていただくことにして、ここでは、本手法との比較のみに限定し、簡単に以下の点にのみ触れておきたい。

- (1) GA では、連続関数の大域最適化問題に対して、染色体を構成する遺伝子数を非常に多くするか、または、何らかの局所的手法を併用しなければ、一般的には、大域最適解そのものが得られることはほとんどなく、近似解が得られるため、メタ最適化手法として使用される¹⁾。これに比べ、本手法では、大域最適解そのものが、ほぼ逃す事なく見つかる。

- (2) 大域最適解を見つけるという点に限れば、本手法のような単純な手法で十分であり、GA のようにコード化や交叉などの複雑な操作は全く必要ない。

EP は再生、淘汰、突然変異から構成されることが多く^{3), 4)}、また、突然変異を省略することも可能である。アルゴリズムは通常シンプルなものとなり、当然コンピュータ・コードも簡単なものとなる。GA と比較して、簡素なことが EP の特徴と言えよう。

さて、従来からの EP との比較に移る。本手法の特徴や従来からの EP との差異は、1. にすでに述べられている。これを要約すると、本手法の特徴は、温度と距離の概念を導入し、初期の世代では高温で活発に動き回り局所最適解を脱出し、世代が進むに従って温度と距離を小さく調整し、大域最適解そのものへ収束する事が出来ることであろう。温度と距離の導入により、従来の方法よりも、局所解に陥り易いという危険性を低くすることができ、数値例に見られるように、ほぼ大域解を逃すことのない方法が構成される。

以上、定性的な比較を行った。さて、次に一つの数値例を取り上げ定量的な比較を行う。

3.2.2 数値例による比較

【例 3】 以下のような非線形連立方程式を満たす実根を求める問題を考える。

$$g_1(x_1, x_2) = x_1^6 + x_1^4 x_2^2 + x_1^2 x_2^4 + x_2^6 - 100 = 0$$

$$g_2(x_1, x_2) = x_1^5 x_2^5 + x_1^3 x_2^3 + x_1 x_2 + 3 = 0$$

ただし、 $-5 \leq x_1 \leq 5$ 、 $-5 \leq x_2 \leq 5$ とする。

$f(x_1, x_2) = (g_1(x_1, x_2))^2 + (g_2(x_1, x_2))^2 = (x_1^6 + x_1^4 x_2^2 + x_1^2 x_2^4 + x_2^6 - 100)^2 + (x_1^5 x_2^5 + x_1^3 x_2^3 + x_1 x_2 + 3)^2$ と置き、 $f(x_1, x_2)$ の大域最適解を求めることにより、連立方程式の解を求める。目的関数 $f(x_1, x_2)$ はかなり非線形性の強い関数になっている。

上記問題を (1) 本研究で提案している手法と (2) シンプル GA にエリート保存戦略を付加した手法で解き、その結果を比較、検討する。まず、本研究の手法を適用する。10 回マルチスタートさせた結果、各試行の最も良い目的関数値を持つ個体として、Table 4 に示す解が

Table 4 Solutions by the proposed algorithm

Start No.	x_1	x_2	$f(x_1, x_2)$
1	2.13686	-4.67977×10^{-1}	2.69351×10^{-10}
2	2.13686	-4.67977×10^{-1}	1.78745×10^{-10}
3	-4.67977×10^{-1}	2.13686	2.13349×10^{-10}
4	4.67977×10^{-1}	-2.13686	2.69351×10^{-10}
5	-4.67977×10^{-1}	2.13686	1.89388×10^{-10}
6	-2.13686	4.67977×10^{-1}	1.95895×10^{-10}
7	4.67977×10^{-1}	-2.13686	1.79049×10^{-10}
8	2.13686	-4.67977×10^{-1}	1.84361×10^{-10}
9	2.13686	-4.67977×10^{-1}	1.78157×10^{-10}
10	-2.13686	4.67977×10^{-1}	1.90602×10^{-10}

得られる。ここでは、小数点以下5桁まで表示し、6桁目を四捨五入している。

各パラメータは以下のように設定している。個体数は $N = 10$ とし、全体で $k_f = 300$ 世代を考える。温度と距離の変化は次のように与えている。

$$\begin{aligned} T_k &= 0.075, L_k = 1.0 \text{ for } k = 1 \sim 50, \\ T_k &= 0.05, L_k = 0.5 \text{ for } k = 51 \sim 100, \\ T_k &= 0.01, L_k = 0.3 \text{ for } k = 101 \sim 151, \\ T_k &= 10^{-3}, L_k = 0 \text{ for } k = 151 \sim 200, \\ T_k &= 10^{-4}, L_k = 0 \text{ for } k = 201 \sim 250, \\ T_k &= 10^{-6}, L_k = 0 \text{ for } k = 251 \sim 300 \end{aligned}$$

ただし、 $L_0 = 2.0$ 、突然変異率を $r = 0.1$ とする。

Table 4 から、10 回の試行のすべてに 10^{-10} 程度の目的関数値が得られており、上の連立方程式の実根は 4 個存在することが分かる。乱数の発生しだいでは局所解に捕まり、目的関数値が悪くなることもあるかもしれないが、その確率は非常に低いと思われる。

つぎに、全く同じ問題をシンプル GA にエリート保存戦略を付加した方法で解く。染色体を構成する遺伝子の数は 10 とし、上と同じように 10 個の個体を考える。交叉率 $= 0.6$ 、突然変異率 $= 0.05$ 、各個体の適応度 s は目的関数値 f の線形スケールリングで与えることにし、 $s = 10 - 0.0009f$ (for $0 \leq f \leq 10000$)、 $s = 1$ (for $f > 10000$) とする。染色体を 2 進数とすると 10 ビットなので、10 進数に変換すると 0 から 1023 までの整数が扱える。 x_1, x_2 に対応する整数を a_1, a_2 とする。 x_i, x_2 は $x_i = (a_i - 511.5) / 102.3$ ($i = 1, 2$) により得られる。

このアルゴリズムを適用し、10 回マルチスタートさせて、300 世代目で得られた最も良い個体の値を Table 5 に示す。この問題では、最も良い目的関数値は (a_1, a_2) の値が (730, 463), (293, 560), (560, 293), (463, 730) のときに得られ、目的関数値は $f = 3.27470 \times 10^{-2}$ である。

$f = 3.27470 \times 10^{-2}$ という値は、本研究で提案している手法で得られる 10^{-10} 程度の値と比べると、かなり悪い値であり、連立方程式の解が得られているのかどうか

判断しかねる値である。これは、GA ではビット数を非常に多くするか、あるいは、なんらかの局所探索手法を併用しなければならないことを示している。これが、GA がメタ最適化に使用される理由であろう。

Table 5 を見ると、目的関数値 $f = 9.13986$ を持つ局所解に捕まっていると判断される場合が 4 回ある。10 回の試行で 4 回捕まるというのは、かなり高い確率である。全く同じ条件で 1000 世代目まで計算を続けると、以下のような結果が得られる。ただし、ここでは目的関数値のみ表示する。

$$\begin{aligned} [1.27852 \times 10^{-1}, 3.27470 \times 10^{-2}, 9.13986, 9.13986, \\ 3.27470 \times 10^{-2}, 3.27470 \times 10^{-2}, 3.27470 \times 10^{-2}, \\ 3.27470 \times 10^{-2}, 9.13986, 9.13986] \end{aligned}$$

上に得られた結果より、局所解からは脱出していないことが分かる。局所解に捕まるのは、エリート保存戦略が一因でもあろう。エリート保存戦略を採用したのは、個体数が少ない場合、シンプル GA だけではあまり良いパフォーマンスが期待できないためである。

さて、上と全く同じ条件で個体数を 20 個と 40 個と 60 個に増やして、それぞれ 10 回試行してみる。その結果、300 世代では、目的関数値として以下の結果が得られる。

個体数 20 個の場合：

$$\begin{aligned} [9.13986, 3.27470 \times 10^{-2}, 9.13986, 3.27470 \times 10^{-2}, \\ 3.27470 \times 10^{-2}, 3.27470 \times 10^{-2}, 3.27470 \times 10^{-2}, \\ 9.13986, 9.13986] \end{aligned}$$

個体数 40 個の場合：

$$\begin{aligned} [3.27470 \times 10^{-2}, 3.27470 \times 10^{-2}, 3.27470 \times 10^{-2}, \\ 3.27470 \times 10^{-2}, 9.13986, 3.27470 \times 10^{-2}, \\ 3.27470 \times 10^{-2}, 3.27470 \times 10^{-2}, 3.27470 \times 10^{-2}, \\ 3.27470 \times 10^{-2}, 6.82351] \end{aligned}$$

個体数 60 個の場合：

$$\begin{aligned} [9.13986, 3.27470 \times 10^{-2}, 3.27470 \times 10^{-2}, \\ 3.27470 \times 10^{-2}, 3.27470 \times 10^{-2}, 3.27470 \times 10^{-2}, \\ 3.27470 \times 10^{-2}, 3.27470 \times 10^{-2}, 3.27470 \times 10^{-2}, \\ 3.27470 \times 10^{-2}] \end{aligned}$$

上の結果から、個体数が 20 個程度では、局所解に捕まるのが改善されるとは言えないが、60 個では非常に良くなっているのが分かる。(60 個の場合、最初の試行で局所解に捕まるが、700 世代まで計算を続けると局所解を脱出し、目的関数値 3.27470×10^{-2} に収束する)。乱数の種が違えば、異なる結果が得られるが、上の結果から一般的に、各試行ではほぼ逃すことなく大域最適解を得るためには、個体数や世代数をあ

Table 5 Solutions by GA

Start No.	a_1	a_2	x_1	x_2	$f(x_1, x_2)$
1	730	464	2.13588	-4.64321×10^{-1}	1.27852×10^{-1}
2	464	730	-4.64521×10^{-1}	2.13586	1.27852×10^{-1}
3	291	511	-2.15543	-4.88759×10^{-3}	9.13986
4	291	511	-2.15543	-4.88759×10^{-3}	9.13986
5	463	730	-4.74096×10^{-1}	2.13589	3.27470×10^{-2}
6	730	464	2.13588	-4.64321×10^{-1}	1.27852×10^{-1}
7	463	730	-4.74096×10^{-1}	2.13588	3.27470×10^{-2}
8	293	460	-2.13588	4.74096×10^{-1}	3.27470×10^{-2}
9	291	511	-2.15543	-4.88759×10^{-3}	9.13986
10	291	511	-2.15543	-4.88759×10^{-3}	9.13986

る程度多くすることが必要であると言える。また、適応度を与える関数やパラメータの改善も必要かもしれない。

以上から、GA ではより良いパフォーマンスを得るために (1) 個体数や世代数をある程度大きくしなければならない、(2) ビット数をかなり多くするか、局所探索手法を併用しなければならない、などが必要である。しかし、本研究で提案している手法では、10 個という少ない個体数で、しかも 300 世代程度で 10^{-10} 程度の連立方程式の解と判断するに足る値がほぼ逃す事なく得られている。

計算時間については、本手法の上に述べた場合の一試行 (個体数 10, 300 世代) を基準とすると、卓上機で FORTRAN を使用した場合、GA の個体数 10, 300 世代の場合では、約 0.64 倍、個体数 20, 300 世代の場合では約 1.7 倍、個体数 40, 300 世代の場合では約 5.3 倍、個体数 60, 300 世代の場合では約 10.7 倍の時間がかかっている。パラメータの値やプログラムの作り方等により計算時間は異なってくる (特に本手法の場合、温度に対して距離を不適切に大きくとり過ぎると時間がかかる) が、これらの結果から、およそその計算速度の比較は可能であろう。GA の個体数 10, 300 世代の場合のみ、本手法よりも速い。しかし、その場合、Table 5 から分かるように、ほぼ 1/2 は大域解が得られていない。大域解の得られる率と計算時間を考慮すると、この問題に対しては、本手法が非常に有効に働いているのが分かったと思われる。

以上、かなり非線形性の強い問題に対して、本研究の手法とシンプル GA にエリート保存戦略を付加した手法とを比較した。本来、手法の比較は、いろいろな観点から、いろいろな問題に対して厳密に行うべきものである。ここでは、本手法の方が優れているとみられる一数列例を上げ、本手法が非常に有効であることを示した。

4. ニューラルネットワークによる非線形システムの同定問題への応用

階層型のニューラルネットワークによる学習問題においては、ニューロンユニットの持つ非線形性のため、2 乗誤差関数は多くの局所解を持ち、大域最適化問題として定義される。バックプロパゲーション (BP)¹⁰⁾ は勾配法に基づく手法であり、初期値がよほど適切に与えられない限り、近似解を見つけることは出来ても大域最適解を見つけることの出来ない手法である。これまでいろいろな大域最適化手法の適用が考えられているが、参考文献 4) では EP のニューラルネットワーク問題への適用が考えられている。また EP の一般的な同定問題への応用が

参考文献 3) に考察されている。本節では、提案するアルゴリズムをニューラルネットワークによる非線形システムの同定問題に適用し、テスト問題だけでなく応用問題に対しても有効であることを確認する。

【例 4】ここでは参考文献 5) を参考にして、プラントの方程式が以下のように離散的システムとして表わされると仮定する。

$$y_p(k+1) = f_p(y_p(k), u(k)) \quad (11)$$

ここで、

$$f_p(y_p(k), u(k)) = \frac{y_p(k) + u(k)}{1 + y_p(k)^2}, \quad (12)$$

$$-1 \leq u(k) \leq 1$$

とする。ただし、 $y_p(0) = 0$ かつ $u(0) = 0$ である。

Fig. 3 Neural network in identification problem

$f_p(y_p(k), u(k))$ を未知と仮定して、Fig. 3 に示す入力層、中間層、出力層からなる簡単な階層型ニューラルネットワーク $NN_f(y_p(k), u(k))$ で近似する。スレッシュホールドを外部に +1 のユニットとして表わしているため、各ユニットの出力は以下のように表わされる。

$$O_j^r = \alpha \left(\sum_{i=0}^{n(r-1)} w_{ij}^{r-1,r} O_i^{r-1} \right), \quad (13)$$

$$O_j^0 = I_j^0$$

ここで、 $r = 1, 2$, $\alpha(t) = 1 / (1 + \exp(-t))$ である。O と I の右上添字 $r = 0$ は入力層の、 $r = 1$ は中間層の、 $r = 2$ は出力層の入出力を表わしている。また、 $n(r-1)$ は $r-1$ 層のニューロンユニット数を表わす。 O_j^r は r 層の j 番目のユニットの出力、特に O_j^0 は入力層の j 番目のユニットの出力で、入力層は信号をそのまま通すだけなので、これはニューラルネットの入力 I_j^0 に等しい。 $w_{ij}^{r-1,r}$ は $r-1$ 層の i 番目のユニットから r 層の j 番目のユニットへの荷重結合係数を示す。 $O_0^0 = I_0^0$ は中間層の、 O_0^2 は出力層のスレッシュホールドのための +1 ユニットの出力である。出力層は 1 ユニットなので、この

ユニットの出力を $z_f(k+1) = O_1^2 = NN_f(y_p(k), u(k))$ とする。入力層は二つの入力ユニットと +1 の値を持つ スレッシュホールドのためのユニットからなるので、 $I_1^0 = O_1^0 = y_p(k)$, $I_2^0 = O_2^0 = u(k)$, $I_3^0 = O_3^0 = +1$ である。

ニューラルネットの出力 z_f は (0, 1) の間の値に正規化されているので, $[-1, 1]$ の間の値をとる一様乱数を入力し, (11) 式の出力の最大値, 最小値を目安に以下のように $z_p(k)$ をおく。

$$z_p(k) = 3.0(z_f(k) - 0.5) \quad (14)$$

時間単位 0 から 50 までの範囲で考える事にし, 最小化すべき誤差関数を以下のようにおく。

$$f = \text{err} = \sum_{k=1}^{50} (y_p(k) - z_p(k))^2 \quad (15)$$

入力 $u(k)$ に $[-1, 1]$ の間の一様乱数を与え, 誤差関数 err の最小値を提案するアルゴリズムで求める。誤差の評価はシミュレーションにより行う。各重みを $[-10, 10]$ の変数とし, アルゴリズムで重みを発生させ, シミュレーションにより誤差の値を得る。また, システムが最初からあまり不安定な挙動を示さないように, 重みの初期値だけは $[-2, 2]$ に発生する。ここでは, $N=20$ の個体を考える。また, $k_f=375$ 世代考える事にする。温度変化は Table 6 に示すように与える。さらに, 距離は以下のように与える。

$L_k = 0.5$ for $k=0, 1 \sim 75$; $L_k = 0.3$ for $k=76 \sim 150$

$L_k = 0.1$ for $k=151 \sim 250$; $L_k = 0$ for $k=251 \sim 375$

Table 6 Temperature cooling schedule in identification problem

T_k	k
0.1	1 ~ 25
0.075	26 ~ 50
0.05	51 ~ 75
0.033	76 ~ 100
0.025	101 ~ 125
0.0125	126 ~ 150
0.01	151 ~ 175
0.009	176 ~ 200
0.008	201 ~ 225
0.007	226 ~ 250
0.006	251 ~ 275
0.005	276 ~ 300
0.0033	301 ~ 325
0.0025	326 ~ 350
0.001	351 ~ 375

また, 突然変異率 r を $r=0.2$ としている。

このサンプル例では, 誤差の最後の世代における最も良い値は $f_1[k_f] = 5.873 \times 10^{-2}$ である。 $N=20$ 個の平均値の最後の世代における値は $avf[k_f] = 5.968 \times 10^{-2}$ である。最小値に対応する各重みが求められるが, この重みの値を使い入力 $u(k)$ を

$$u(k) = \sin \frac{2\pi k}{25} \quad (k=0, \dots, 50) \quad (16)$$

とした時の $y_p(k)$ と $z_p(k)$ のグラフを Fig. 4 に示す。 z_p が y_p をよく追従しているのが分かる。

Fig. 4 y_p and z_p in identification problem

5. おわりに

本論では, 進化論からいくつかの概念を借りて構成した大域最適化手法を提案した。本手法の特徴, これまでの EP との差異は, 本文中にもあるように, 子の再生に温度の概念を導入したことである。これにより, 初期では激しく動き回り局所解を脱出し, 世代が進むに従って大域最適解そのものへ収束することが出来る手法が構成される。また, 距離の概念の導入による助力もあり, GA のように近似解が得られ, 大域解そのものが得られる確率が低いという弱点を克服することが出来る。また, GA のようなコード化や交叉などの複雑な操作も必要ではない。

さて, 本手法は開発されたばかりであり, 本文中でも述べたように, 多くの課題が残っている。収束性に関して, 数学的に厳密な解析を行うことも重要な課題の一つである。また, 参考文献 2), 7), 9) にあるように, 求められた解が大域最適解である信頼度を与えることも非常に重要な事である。これらは, 本手法だけでなく, GA や EP など, 進化の概念を取り入れた方法全体の課題とも言える重要なものであるが, 本研究でも今後の重要な課題として取り組みたい。

数値例のところでも述べたように, 本手法を適用することにより, 大域最適解をほぼ逃す事なく見つけることが出来る。また, ニューラルネットワークによる同定問題に適用した結果, モデル出力がプラントの出力を追従

する良い結果が得られている。このことから、今後、課題の解決と共に、本手法のいろいろな実システム問題への応用が期待される。

参 考 文 献

- 1) 相吉，三室：大域的最適化のためのメタ最適化問題と遺伝的アルゴリズムによる解法；計測自動制御学会論文集，Vol. 28, No. 8, pp. 999~1006 (1992)
- 2) F. Archetti and F. Schoen：A Survey on the Global Optimization Problem：General Theory and Computational Approaches；Annals of Operations Research, Vol. 1, pp. 87~110 (1984)
- 3) D. B. Fogel：System Identification Through Simulated Evolution：A Machine Learning Approach to Modeling, Ginn Press (1991)
- 4) D. B. Fogel, L. J. Fogel and V. W. Porto：Evolving Neural Networks；Biological Cybernetics, Vol. 63, pp. 487~493 (1990)
- 5) K. S. Narendra and K. Parthasarathy：Identification and Control of Dynamical Systems Using Neural Networks；IEEE Trans. on Neural Networks, Vol. 1, No. 1, pp. 4~27 (1990)
- 6) 島：徐冷型ランダムトンネリングアルゴリズムによる大域最適化；計測自動制御学会論文集，Vol. 29, No. 11, pp. 1342~1351 (1993)
- 7) J. A. Snyman and L. P. Fatti：A Multi-Start Global Minimization Algorithm with Dynamic Search Trajectories；J. of Optimization Theory and Applications, Vol. 54, No. 1, pp. 121~141 (1987)
- 8) 須貝，平田：組合せ最適化アルゴリズムとその応用；計測と制御，Vol. 29, No. 12, pp. 1084~1091 (1990)
- 9) A. H. G. Rinnooy Kan and G. T. Timmer：Stochastic Global Optimization Methods；Mathematical Programming, Vol. 39, pp. 27~56 (1987)
- 10) P. D. Wasserman：Neural Computing, Theory and Practice, Van Nostrand Reinhold (1989)