人　工　知　能　106－14
（１９９６．１０．３１）

大きな突然変異率と集団エリート選択を用いた遺伝的アルゴリズム（GALME）の提案

下平　丞作士

文教大学情報学部情報システム学科
〒253　神奈川県茅ヶ崎市行谷 1100
Tel: 0467-53-2111
E-mail: f2610390@ca.aif.or.jp

概要　遺伝的アルゴリズム（GA）は、関数の最適化のための有力な手段の一つであるが、伝統的 GA は局所的探索を行なうのには適していないといわれており、GA の性能を向上させるための様々な研究が行なわれている。筆者は、最適化の手段としての GA の性能を向上させるための実験的検討を行なった結果、大きな突然変異率（ある場合には，世代についての減少関数として制御する）と集団エリート選択を用いた GA（GALME)が、局所的最適値を避けて真の最適値を得るのに極めて有効であることが分かった。本論文では、伝統的 GA と対比して GALME とその理論的根拠について述べ、多峰性関数と騙し関数を用いた計算機実験により、性能を検証した結果を示す。GALME のアルゴリズムは、伝統的 GA と同程度に簡単であり、並列処理にも適している。ここで行なった実験の範囲では、その性能は伝統的 GA よりも著しく良いことが分かった。

キーワード:　遺伝的アルゴリズム、突然変異率、集団エリート選択、最適化

Proposal of a Genetic Algorithm Using Large Mutation Rates and Population-Elitist Selection (GALME)

Hisashi Shimodaira

Department of Information and Communication, Bunkyo University
1100 Namegaya, Chigasaki-City, Kanagawa 253 Japan
Tel: +81-467-53-2111
E-mail: f2610390@ca.aif.or.jp

Abstract　Although the genetic algorithm (GA) is one of prospective means for function optimization, it is recognized that the traditional GA can not often obtain the global optimum. In this paper, I propose a GA using large mutation rates (in some cases, it is controlled as a function of generation) and population-elitist selection (GALME) to obtain the global optimum effectively and presents the results of computational experiments, compared to the traditional GA. Within the range of the experiments, it is turned out that the performance of GALME is remarkably superior to that of the traditional GA .

Keywords:　Genetic algorithm, Mutation rates, Population-elitist selection, Optimization

1. はじめに

遺伝的アルゴリズム（GA）は確率的な多点探索法の一種であり、関数の最適化のための有力な手段の一つである。関数の最適化においては、局所的な最適値に落ち込まないで、真の最適値にできるだけ速く到達することが必要である。そのためには、探索空間をくまなく大域的に探索した後、それまでに遺伝子型に蓄えられた探索空間についての情報を用いて所的探索を行なうことが望ましい。しかしながら、伝統的 GA は細かく調節して局所的探索を行なうのには適していないことが広く認められている [2]。

最適化の手段としての GA の性能を向上させるための研究主なものとしては、次のようなものがある。伝統的 GA とは異なった GA の変種が、Eshelman [3]、Syswerda [4]、Whitley ら[5]、Mahfoud ら[6]によって提案されている。また、伝統的 GA の枠組みの中で、突然変異率や交叉率を制御することにより、性能を向上させる試みが、Fogarty[7]、Bäck[8]等によって行なわれている。一方、新しい枠組みの GA についての研究が、Goldberg ら[9]によって行なわれている。

筆者は、最適化の手段としての GA の性能を向上させるための実験的な検討を行なった結果、大きな突然変異率（ある場合には、世代についての減少関数として制御する）と集団エリート選択を用いた GA (GALME: Genetic Algorithm using Large Mutation Rates and Population-Elitist Selection) が、局所的な最適値を避けて真の最適値を得るのに極めて有効であることが分かった。以下、伝統的 GA と対比して GALME とその理論的な根拠について述べ、計算機実験の結果を示す。

2. 伝統的 GA と擬似焼きなまし法

ここでは、後の説明に必要な伝統的 GA と擬似焼きなまし法の概要について述べる。

伝統的 GA[3]、[10]の枠組みを，図１に示す．ここで、tは世代、structure は個体の遺伝子型、P(t)、C(t)は、世代 t の個体の集団を表す。伝統的 GA では，次のような処理が行なわれる。(1) 集団の個体数 Nは一定であり、その初期化は乱数を用いて行なわれる。(2) 親となる個体の選択(select_r)は、適応度の高いものほど有利なように行なわれる。(3) 確率に基づいた交叉と突然変異により、遺伝子型の組換えが行なわれる。組換えオペレータとして、１点交叉または２点交叉が用いられる。突然変異は集団の多様性を維持するため

```
begin
  t=0;
  initialize P(t);
  evaluate structures in P(t);
  while (termination condition not satisfied) do
  begin
    t=t+1;
    select_r P'(t-1) from P(t-1);
    recombine structures in P'(t-1)
        forming C(t);
    evaluate structures in C(t);
    selects P(t) from C(t) and P(t-1);
  end;
end;
```

Fig.1 Skeleton of the traditional GA

に用いられ、その確率は小さな値がとられる。(4) 生き残る個体の選択(select_s)では、通常、全ての C(t)を P(t)とする。ただし、世代間ギャップ（G) [10]を考慮するときは、P(t-1)の中の N(1-G)個の個体が（ランダムに）選ばれ、そのまま P(t)で生き残る。エリート戦略では、P(t-1)の中で最も適応度の高い個体が選ばれ、そのまま P(t)で生き残る。

擬似焼きなまし法（SA）[11]は、物理的な焼きなまし過程に類似したシミュレーションを行なう最適化手法である。GA とは異なり、単一の解を扱う。SA においては、解はそのコストに基づいて評価される。関数の最小値を求める場合、良くない解は、良い解よりもコストが高い。コスト f(i)の現在の解 i に対して、近傍探索オペレータを施して、コスト f(j)を持つ解 j を生成したとする。解 j が次の時点の解として受け入れられる確率 p は、次のようにその時点の温度 T とコスト差に依存する。

$$\text{if } (f(j) \leqq f(i))\ p=1$$
$$\text{if } (f(j) > f(i))\ p=\exp[(f(i)-f(j))/T] \qquad (1)$$

このようにして次々に解を求めながら、収束条件を満足するまで繰り返し計算を行なう。温度は最初は高く、次第に低減される。温度が高いとコストの高い解でも受け入れられるが、温度が低くなるとそのようなことはほとんどなくなり、解が収束する。SA の特徴は、解が局所的な最適値に落ち込んだ場合にこれから脱出できるように、現在の解よりも良くない解を確率的に次の解として採用することにある。SA は、ある条件の下で、真の最適値への漸近的な収束が証明されている[11]。

3. 既往の研究

本研究と関連のある既往の研究の概要について説明する。主として、伝統的GAとの差異について述べる。

Eshelman[3]の提案したCHCでは、$select_r$においてP(t-1)からランダムに2個の個体を非復元抽出してペアを作り、P'(t-1)としている。$select_s$ではC(t)とP(t-1)を競争させ、それらの中から適応度の高い順にN個の個体を選んでP(t)としている。Eshelmanはこれを集団エリート選択と呼んでいる。ペアを作る際に2個の個体のハミング距離を計算し、それがしきい値よりも小さい場合には、ペアとしない（近親相姦の排除）。しきい値は、集団の収束状況に応じて適応的に減少させる。組換えオペレータとして、非常に破壊的作用の大きな一様交叉の変種（HUX：値が異なる遺伝子座のみについて、その個数の半分をランダムに入れ替える）を用いている。組換えの過程では突然変異を用いないで、その代わりに集団が真の最適値に到達しないで収束したとき、最も適応度の高い個体を用いて、高い比率で突然変異を施して個体を生成した後、再スタートする。CHCの特徴は、集団エリート選択と破壊的作用の大きな一様交差を組み合わせて用いている点にある。伝統的GAよりも高い性能を達成しているが、そのアルゴリズムはかなり複雑であり、ハミング距離の計算等にもコストがかかる。

Syswerda[4]の定常状態GAでは、$select_r$において、適応度に応じて2個の個体を選択してペアを作り、交叉と突然変異を施して2個の子供を生成する。集団の2個の個体を削除し、生成した子供を挿入する。削除する個体は、適応度の最も低い個体から始めて、逆ランク法によって選択する。Whitleyら[5]のGENITORアルゴリズムでは、1個の子供を生成し、集団に加える。これらのアルゴリズムの特徴は、世代という概念は用いず、1サイクルに1または2個の子供を生成し、集団に加える点にある。しかし、削除される個体と加えられる個体の適応度の大小関係は考慮されておらず、論理的な明確さを欠いている。

Mahfoudら[6]の並列擬似焼きなまし法では、ランダムに2個の個体を非復元抽出して、集団の個体全てをペアにする。各ペアに交叉と突然変異を施して、2個の個体を生成する。各ペアについて生成した子供とその親の間でトーナメントを行い、式(1)のような擬似焼きなまし法の受け入れ確率を用いて、次世代の個体を選択する。突然変異率は世代の関数とし、最初は大きくとり、次第に減少させている。このアルゴリズムの特徴は、次世代の個体の選択に擬似焼きなまし法の手法を用いていること、および突然変異率の制御を行なっている点にあるが、集団全体における各個体の適応度の大小関係は考慮されていない点でGAの枠組みの中にあるとは言い難い。

伝統的GAの枠組みの中で、その性能の改善を試みた研究の概要は，次のものがある．Fogarty[7]は、特定の応用例において、ある初期状態から出発した場合に、突然変異率を変化させると、性能が向上したと述べている．Bäck[8]は、各個体の遺伝子型に突然変異率を表す部分を設け、突然変異率を適応的に変化させる方法を提案している。

4. GALME

GALMEの骨組みを図2に示す。集団の個体数はN個である。集団の初期化は、一様乱数を用いて行なう。突然変異率 p_m は一定とするか、または世代 t の関数として外部から制御する。親の選択（$select_r$）では、P(t-1)からランダムに2個づつ個体を非復元抽出し、集団の個体全てをペアにしてP'(t-1)とする。各ペアの個体について確率 p_m で突然変異を施し、交叉させ、C(t)を生成する。生き残る個体の選択（$select_s$）では、C(t)とP(t-1)を合わせた集団の中から、適応度の大きい順にN個の個体を選んで、P(t)とする。図2の繰り返し計算の回数は、規定の世代数（*GMAX*）を上限とする。*GMAX*以内で真の最適値に到達しない場合にはループを終了し、突然変異率を初期値に設定し、すべての個体を一様乱数によって生成し、集団を再初期化して再スタートする。

```
begin
  t=0;
  initialize P(t);
  evaluate structures in P(t);
  while (termination condition not satisfied) do
  begin
    t=t+1;
    calculate pm for t;
    selectr P'(t−1) from P(t−1) by randomly
    pairing all individuals;
    apply mutation with pm and crossover to
      each pair in P'(t−1) forming C(t);
    evaluate structures in C(t);
    selects P(t) from C(t) and P(t−1) according
      to their fitness values;
  end;
end;
```

Fig.2 Skeleton of GALME

伝統的GAでは、突然変異は解が局所的な最適値に落ち込むことを避けるために、交叉オペレータが全ての遺伝子を持つように保証するバックグランドオペレータであるとされており[12]、一般に小さな値が用いられている[10]。これに対し、GALMEでは、P'(t-1) のペアに対してかなり大きな確率で突然変異を施し、必ず交叉させてC(t)を生成する。これは、できるだけ親と異なった子供を生成し、それまでに探索していない領域を探索するためである。また、突然変異率を世代の関数として、初期の段階で大きな値をとることは、解空間をもれなく探索するとともに、局所的な最適値に落ち込みそうになったときには、これから脱出することを可能にする。つまり、SAと同様に、近傍探索の範囲を大きくとり、集団エリート選択により、その時点の最良の解よりも劣る解を集団の中に残すことにより、局所的な最適値から脱出できるようにするわけである。しだいに突然変異率を小さくすることは、それまでに得られた解を用いて局所的探索を行なって、真の最適値を発見することを可能にする。

Bäck[8]の自己適応型の突然変異率を用いた計算結果によると，最適の突然変異率はかなり大きな値であり、世代についての減少関数になっている。また、最適の突然変異率についてのHesserらの理論的研究[13]によれば、それは時間についての負の指数関数になっている。これらの結果は、GALMEで採用している突然変異率のとり方についての、経験的・理論的な裏付けとなる。

5章の計算機実験では、次のような突然変異率の変化のタイプを用いた。

$$
\begin{aligned}
&\text{Type1} \quad p_m = \text{constant}. \\
&\text{Type2} \quad p_{m,t+1} = \beta\, p_{m,t}, \quad \text{if } p_0 \geqq p_{m,t} > p_{min}. \\
&\text{Type3} \quad p_{m,t+1} = \beta_1 p_{m,t}, \quad \text{if } p_0 \geqq p_{m,t} \geqq p_b. \\
&\qquad\quad p_{m,t+1} = \beta_2 p_{m,t}, \quad \text{if } p_b > p_{m,t} > p_{min}. \qquad (2)
\end{aligned}
$$

タイプ2は突然変異率を1段階で漸減させる方法であり、タイプ3は突然変異率を2段階で漸減させる方法である。ここに、p_0、p_b、p_{min}はそれぞれ突然変異率の初期値、境界値、下限値であり、β、β_1、β_2はそれぞれその低減率である。$p_{m,t+1}$がp_{min}より小さくなる場合には、それ以後p_{min}とした。

伝統的GAでは、適応度の高い個体は次世代に複数個の子孫を残せるのに対し、GALMEでは、適応度の大小にかかわらず親となるチャンスは1回であり、生成された子供と親は、適応度が上位からN番目以内である場合に生き残る。したがって、伝統的GAでは、性能の良いスキーマを含む個体は、性能の悪いスキーマを含む個体に比べて、急速にその数が増加するのに対して、GALMEでは、その増加のスピードは緩やかではあるが、着実に増加する。これは、解が局所的な最適値に落ち込むのを避けるために、有利に作用する。というのは、伝統的GAでは、その局所的な最適値に関するスキーマを含む個体が急速に増加し、集団を支配してしまうのに対して、GALMEでは、そのスピードが緩やかであり、その間に大きな突然変異率によって個体を変化させることにより、これから脱出できる可能性があるからである。一方、性能の良い個体が増加するスピードが緩やかであることは、真の最適値に到達するスピードをも低める作用もあるが、局所的な最適値を回避することによるスピードの向上がこれを補償し、全体として性能の向上につながるものと考えられる。

交叉や突然変異は，親と異なった子供を生成することにより、それまでに探索していない空間の領域を探索するという効果をもつが、一方、既に得られた性能の良いスキーマを破壊するという副作用がある。この副作用の影響は、伝統的GAとGALMEでは決定的に異なる。伝統的GAでは、交叉や突然変異によって親よりも適応度の低い子供が生成された場合にも、原則として親はその子供によって置き換えられる。エリート戦略をとる場合には、ごく少数のエリートのみが次世代で生き残る。これに対し、GALMEでは、生成された子供が親よりも適応度が低ければ、決して置き換えられない。上記の理由で、伝統的GAでは、突然変異率を大きくとればランダムサーチに近くなり、かえって性能が悪くなる。GALMEでは、集団エリート選択を用いているため、上記の理由で大きな突然変異率をとることができ、性能向上の効果をもたらす。突然変異率を世代の関数として初期の段階で大きくとることは、探索空間をくまなく大域的に探索して、探索空間についての情報を各個体の中に蓄積する効果がある。

GALMEでは，生き残った個体は、必ず親となることができる。したがって、性能のよいスキーマを含んだ個体は生き残って、次々とその子孫を作ることができる。

一方、GALMEでは、適応度が低い個体でも上位N番目以内であれば保存され、必ず親となることができる。これは、ある時点では真の最適値に関するスキーマを含む個体の適応度が低い場合にも、その個体の子孫が将来真の最適値に近い適応度の個体を生み出すチャンスを与える。これは、SAにおいて、ある確率

で現在の解よりも性能の悪い解を受け入れて、局所的な最適値から脱出することと類似している。突然変異率を世代についての減少関数とし、初期の段階で大きな値をとる場合には、集団エリート選択のこのような効果と相まって、局所的な最適値からの脱出がより促進されるであろうことが、SA における温度に基づいた解の受け入れ確率の制御方法からの類推により、推測される。

GALME で用いている手法が上記のような性能の向上効果をもたらすということは仮説であり、理論的、実験的な検証が必要である.

伝統的 GA と GALME の計算量を比較すると、次の通りである。関数の評価回数は、ともに1世代について N回である。親となる個体の選択は、伝統的 GA でどのような方式を用いるかにもよるが、ほぼ同じであると考えてよい。生き残る個体の選択では、GALME では、$2N$個の個体から適応度の大きい順に N個を選ぶ分だけ多い。GALME のアルゴリズムは伝統的 GA と同程度に簡単であり、並列処理にも適している。

GALME は基本的には、Eshelman[3]の集団エリート選択と、Mahfoud ら[6]、Hesser ら[13]の突然変異率を世代についての減少関数とするアイデアでから成っている。しかし、これによって、大きな突然変異率を用いることができ、次章に示すように、これによって性能が著しく向上することを発見したことに新規性がある。

5．計算機実験

計算機実験により、伝統的 GA と GALME の性能の比較を行った。これは、新しい GA の性能の評価のための原器として、伝統的 GA が用いられているためである。簡単な関数では、伝統的 GA と GALME の性能の差違はほとんどない。ここでは、GA で解くことが難しく、しばしば性能の評価に用いられている多峰性関数[3]、[14]と騙し関数[3]、[9]についての結果を示す。それぞれの関数は、唯一の真の最大値を持っており、これを求めた。計算は、各計算条件について集団初期化等に用いる一様乱数のシーズを変えて 20 回行なった。繰り返し計算回数の上限値(GALME で再スタートを用いない場合と伝統的 GA については、$GMAX$=10000、GALME で再スタートを用いる場合にはトータルで 10000）以内に，真の最大値に到達した個体が初めて現われたときに収束したものとした。性能の評価は、収束した回数と収束するまでの関数の評価回数の平均値によって行なった。関数の評価回数を尺度とするのは、実用上、GA 自体よりも関数の評価の方が計算コストがかなり大きいためである。

交叉の方法として、GALME と伝統的 GA の両者とも、2点交叉を用いた。GALME については、2つの関数を用いて、2点交叉と HUX[3]の性能の比較するための予備的実験を行なった。どの場合にも、2点交叉の方が HUX よりも性能が良かったため、前者を用いることとした。

5.1 多峰性関数

用いた関数[14]は、次式の通りである。

$$f6 = 0.5 + \frac{0.5 - \sin^2\sqrt{x^2 + y^2}}{[1 + 0.001(x^2 + y^2)]^2} \qquad (3)$$

この関数は、z軸について軸対称であり、原点で最大値 1.0 を持つ。図3に x軸と z軸を含む断面を示す。GA では、xと yについて-100 から 100 までの範囲を、22 ビットの2進数でコード化した。離散化された探索点が原点に一致しないため、この場合の真の最大値は、少数点以下に9が8個連なる数である。

GALME では、突然変異率の制御方法の3タイプについて、突然変異率とその低減係数等のパラメターを少しずつ変えてランし、最も性能が良い（収束回数が最も多く、かつ関数の評価回数が最も少ない）場合を探した。伝統的 GA では、De Jong の標準的なパラメータ値[10]と Grefenstette のオフライン性能を重視したパラメータ値[10]を参考にして,パラメータの値を少しずつ変えてランし、最も性能が良い場合を探した。5.2 の騙し関数の場合も同様である。ちなみに、前者は、N=50、交叉率 p_c=0.6、突然変異率 p_m=0.001、世代間ギャップ G=1.0、スケーリングはしない、エリート戦略を使用、である。後者は、N=80、p_c=0.45、p_m=0.01、G=0.9、1世代前の適応度の値でスケーリングをする、適応度に比例して親となる個体を選択、

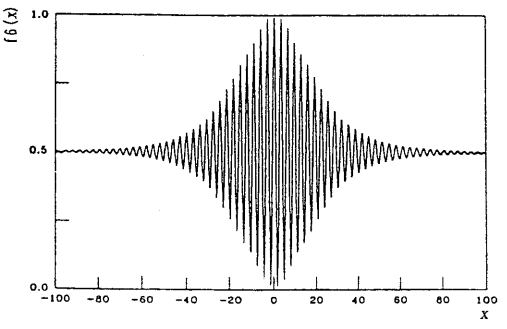

Fig.3 Section of f6 function

Table 1 Performance for function f6 (Number of convergence to global optimum and average of function evaluation times)

Algorithm	Computation condition	Num.of conv.	Aver.of func. eval. times
GALME	N=100, Mutation=Type1 (p_0=0.08), $GMAX$=10000, Restart=No	18	67466
	N=100, Mutation=Type2 (p_0=0.3, p_{min}=0.01, β=0.9995), $GMAX$=10000, Restart=No	20 (all)	200180
	N=100, Mutation=Type3 (p_{0a}=0.15, p_{0b}=0.06, p_{min}=0.01, β_1=0.99088, β_2=0.99985), $GMAX$=300, Restart=Yes	20 (all)	22315
Traditional GA	N=50, p_c=0.6, p_m=0.001, G=1.0, Scaling=No, Selection strategy=Elitist selection, $GMAX$=10000	9	26066

である。表1に、それぞれの場合について最も性能が良かったときの計算条件と計算結果を示す。

GALME では，突然変異率の制御方法タイプ1の場合、p_0の値がこれより大きくてもこれより小さくても性能が悪くなった。タイプ2の場合、p_0とβがこれより若干大きい値でも全て真の最大値が得られたが、関数の評価回数が多くなった。p_0またはβをこれより小さい値にすると、真の最大値に収束する回数が減少した。タイプ3の場合、p_{0a}、p_{0b}、β_1、β_2の値がこれより若干大きくてもこれより若干小さくても、全て真の最大値に収束したが、関数の評価回数が多くなった。

表1の伝統的 GA における計算条件は、De Jong のパラメータ値である。選択の戦略については、親となる個体は適応度を用いたルーレット方式で選択し、次世代の集団は最も適応度の大きな個体2個をそのまま生き残らせ、他は生成した子供で構成した。交叉率または突然変異率の値をこれより大きくまたはこれより小さくすると、真の最大値に収束する回数がこれより少なくなった。なお、Grefenstette のパラメータ値では、全く真の最大値が得られなかった。

5.2 騙し関数

Goldberg[9]のオーダ3の騙し関数を用いた。この関数では、30 ビットの0と1からなるストリングについて、10 個のサブ関数が表2に示す値を持つものとし、その合計を関数値とする。タイトな配置の問題では、ストリングを順番に3ビットずつ区切ってサブ関数とする。ルースな配置の問題では、最初のサブ関数はストリングの 1、11、21 番目のビットからなり、2番目のサブ関数は 2、12、22 番目のビットからなるものとする。真の最大値は 300 である。表3と表4に、最も性能が良かったときの計算条件と計算結果を示す。

Table 2 Goldberg's order-3 deceptive problem

f(000)=28	f(001)=26	f(010)=22	f(011)=0
f(100)=14	f(101)=0	f(110)=0	f(111)=30

GALME では、タイトおよびルースな配置の両者とも、突然変異率の制御方法タイプ1の場合の性能が最も良かった。この場合、突然変異率の値をこれより若干大きくまたはこれより若干小さくしても、全て真の最大値に収束したが、関数の評価回数がこれより多くなった。タイプ2、3の場合にも、突然変異率とその低減率を適切にとれば、全て真の最大値が得られたが、関数の評価回数がタイプ1の場合よりも多くなった。

伝統的 GA の計算条件は、De Jong のパラメータについて、突然変異率の値のみを大きくとったものである。タイトおよびルースな配置の両者とも、突然変異率をこれより小さい値にすると性能が悪くなった。なお、Grefenstette のパラメータ値を用いた場合、真の最大値が得られた回数は、タイトな配置については 13、ルースな配置については 0 であった。

5.3 考察

多峰性関数 f6 の計算結果から、次のことが分かる。GALME では、突然変異率は一定とするよりも、世代についての減少関数にする方が性能が良い。突然変異率の初期値をある程度大きくとって、ゆっくりと減少させれば、確実に真の最大値がえられるが、収束のスピードがおそい。突然変異率の低減率を第1段階で大きく、第2段階で小さくとり、再スタート機能を用いることにより、より確実により早く、真の最大値に到達できる。伝統的 GA では、真の最大値が得られた回数は、45%である。

騙し関数の計算結果から、次のことが分かる。GALME では、突然変異率は世代についての減少関数にするよりも、一定とする方が性能が良い。これは、騙し関数においては積み木仮説のような作用によって真の最大値が導かれないためであり、初期の段階で突然変異率を大きくとって、大域的探索を行ってもあまり役立たないからである。伝統的 GA と比べて、いずれの場合も、関数の評価回数は著しく少ない。伝統的 GA では、オーダ3の騙し関数を解くのは難しく、

Table 3 Performance for tightly ordered deceptive function (Number of convergence to global optimum and average of function evaluation times)

Algorithm	Computation condition	Num.of conv.	Aver.of func. eval. times
GALME	N=50, Mutation=Type1 (p_0=0.095), $GMAX$=10000, Restart=No	20(all)	20895
Traditional GA	N=50, p_c=0.6, p_m=0.003, G=1.0, Scaling=No, Selection strategy=Elitist selection, $GMAX$=10000	20(all)	256207

Table 4 Performance for loosely ordered deceptive function (Number of convergence to global optimum and average of function evaluation times)

Algorithm	Computation condition	Num.of conv.	Aver.of func. eval. times
GALME	N=50, Mutation=Type1 (p_0=0.085), $GMAX$=10000, Restart=No	20(all)	68642
Traditional GA	N=50, p_c=0.6, p_m=0.0034, G=1.0, Scaling=No, Selection strategy=Elitist selection, $GMAX$=10000	20(all)	168012

ことにルースな配置の場合を解くのは不可能であるとされていたが[9]、エリート戦略を用い、突然変異率をある程度大きくとることにより、確実に解けることが分かった。

GALME では、伝統的 GA で一般的に用いられている突然変異率よりもはるかに大きな値を用いたときに、良い性能が得られている。したがって、GALME では、突然変異は単なるバックグランドオペレータではなく、探索オペレータとしての役割を果たしていることが分かる。また、突然変異率を世代の関数とし、初期の段階で大きくとることは、騙し関数のような特殊な関数を除いて、性能向上に極めて有効である。しかし、突然変異率の適切な大きさや変化のさせ方は、問題によって異なるようであり、GA が一般に持っているパラメータのチューニングという問題点がある。

GALME と伝統的 GA の性能を比較すると、上記の2つの関数については、適切なパラメータ値を用いれば、GALME の方がより確実に著しく速く、真の最大値に収束することが分かった。

既往の研究で提案された有力な方法と比較すると、関数 f6 については、Syswerda[4]、Whitley[5]の手法による結果[15]では、評価回数 4000 回でほぼ頭打ちになっており、小数点以下の 9 の数は、3よりも小さいのに対して GALME では平均 22315 回で常に真の最大値が得られている。タイトな配置の騙し関数について、真の最大値に収束したときの関数の評価回数は、CHC[3]では平均 20960、並列疑似焼きなまし法[6]では 23000 回、メッシーGA[9]では 40600 であるのに対して、GALME では平均 20895 である。ルースな配置の騙し問題については、列疑似焼きなまし法[6]では 90000 回であるのに対して、GALME では平均 68642 回である。ただし、並列疑似焼きなまし法の問題は 24 ビットであり、ここで扱った問題より易しい問題である。これらの結果から、GALME がこれらの手法と同等かそれ以上の性能を有していることが推測さよう。

6. おわりに

提案した GALME は、伝統的 GA と比べて、計算量が若干多いが、そのアルゴリズムは伝統的 GA と同程度に簡単であり、並列処理にも適している。その性能は、ここで行なった計算機実験の範囲では、伝統的 GA よりも著しく良いことが分かった。また、既往の研究で提案された手法と比較しても、同等かそれ以上の性能を有しているものと推測される。

GALME は基本的には、Eshelman[3]の集団エリート選択と、Mahfoud ら[6]、Hesser ら[13]の突然変異率を世代についての減少関数とするアイデアでから成っている。しかし、これによって、大きな突然変異率を用いることができ、性能の向上に顕著な効果があることを発見したことに新規性があり、その有用性と相まって新しい手法を構成しているものと考えられる。実際、その点において高い評価を得ている[1]。ちなみに、公知の技術を組み合わせた発明について、その組み合わせが自明のものでなく、その組み合わせによって顕著な効果が得られる場合には、新規性があるとするのが通説である。

今後の研究課題は、次の通りである。更に複雑な多峰性関数や巡回セールスマン等の組合せ最適化問題に適用し、性能を検証する。パラメータのチューニング方法について検討する。CHC[3]や GENITOR アルゴリズム[5]と性能比較を行なう。良い性能が得られる

理由を、理論的に解明する。また、GALME は、疑似焼きなまし法と類似した点があるため、同様な考え方でその収束性を証明できるのではないかと考えている。

参考文献

[1] Shimodaira, H.: A New Genetic Algorithm Using Large Mutation Rates and Population-Elitist Selection, Proc. of the 8th IEEE International Conference on Tools with Artificial Intelligence, November 16-19, 1996, France, to appear.

[2] Grefenstette,J.J.: Incorporating Problem Specific Knowledge into Genetic Algorithms, Genetic Algorithms and Simulated Annealing, pp.42-60, Morgan Kaufmann (1987).

[3] Eshelman, L.J.: The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination, Foundation of Genetic Algorithms, pp.265-283, Morgan Kaufmann (1991).

[4] Syswerda, G.: Uniform Crossover in Genetic Algorithms, Proc. of the Third International Conference on Genetic Algorithms, pp.2-9, Morgan Kaufmann (1989).

[5] Whitley D.: The GENITOR Algorithms and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best, Proc. of the Third International Conference on Genetic Algorithms, pp.116-121, Morgan Kaufmann (1989).

[6] Mahfoud, S.W. et al.: A Genetic Algorithm for Parallel Simulated Annealing, Parallel Problem Solving from Nature, 2, pp.301-310, Elsevier Science (1992).

[7] Fogarty T.C.: Varying the Probability of Mutation in the Genetic Algorithm, Proc. of the Third International Conference on Genetic Algorithms, pp.104-109, Morgan Kaufmann (1989).

[8] Bäck, T.: Self-Adaptation in Genetic Algorithms, Proc. of the First European Conference on Artificial Life, pp.263-271, MIT Press (1992).

[9] Goldberg D.E. et al.: Messy Genetic Algorithms: Motivation, Analysis, and First Results, Complex Systems, Vol.3, pp.493-530 (1989).

[10] Grefenstette, J.J.: Optimization of Control Parameters for Genetic Algorithms, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-16, No.1, pp.122-128 (1986).

[11] Aarts E. et al.: Simulated Annealing and Boltzmann Machines, John Wiley & Sons (1989).

[12] Holland, J.H.: Adaptation in Natural and Artificial Systems, p111, MIT Press (1992).

[13] Hesser, J. et al.: Towards an Optimal Mutation Probability for Genetic Algorithms, Parallel Problem Solving from Nature, pp.23-32, Springer-Verlag (1990).

[14] Schaffer J.D. et al.: A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization, Proc. of the Third International Conference on Genetic Algorithms, pp.51-60, Morgan Kaufmann (1989).

[15] Lawrence, D.; Handbook of Genetic Algorithms (邦訳あり), p.27, Van Nostrand Reinhold (1991).