革新的遺伝的アルゴリズムにおける確率共鳴信号の解析

コース　　基盤科学コース
学籍番号　　１１０４０２
氏名　　陳　金琳
指導教員　　ルジェロ・ミケレット

Recently, the Genetic Algorithms, a new learning paradigm that models a natural evolution mechanism, have received a great deal of attention regarding their potential as optimization techniques for solving combinatorial optimization problems. However, all previous studies are in ideal conditions, instead noise is present in actual life application of Genetic Algorithm. The effect of noise in these algorithms is not known. In this study, I developed an original Genetic Algorithms in python to study this type of linear equation:

$$ax + by + cz + dw = const.$$

Here, I incorporate noise into the program, to study quantitatively what is the effect of noise in the performance of Genetic Algorithms, also I discovered that the particular effect of "stochastic resonance" comes out.

１．研究背景と目的

人工知能に関しての様々な問題において、複雑で大きな探索空間の中で最適な解をすばやく見つけることが常に望まれている。しかし、問題の規模が大きくなると厳密解を求めるには、計算の量が増えることとなり、現実的ではなくなり、そこである程度満足できる解を探す近似解法が利用される。その際、探索空間についての知識が先見的に利用できる時は早く解を探索できる。しかし、そうでない時は動的にその探索空間についての知識を蓄積しながら探索手順を制御しなくてはならない。このような問題は適応型探索問題と言われ、一般にその取扱が難しいと言われている。遺伝的アルゴリズムはこのような問題において、特に有効な探索アルゴリズムである。

また、我々の周囲には自然系及び非自然系に関わらず、ノイズ（雑音）が満ち溢れている。従来では、このノイズはシステムに好ましくないため、除去してきた。しかし、近年ではノイズの存在が有益な効果を果たしていることが判明された。それは確率共鳴（Stochastic Resonance）と呼ばれる現象であり、弱い信号にある微弱なノイズを加えると、元の信号が強まるという。

以上より、本研究では従来では計算量が多く、最適な解を求めるに時間がかかるものであり、次のような線形方程式

$$ax + by + cz + dw = const.$$

私は遺伝的アルゴリズムのオリジナルアルゴリズムを設計し、この線形方程式の問題解決のやり方は深く解析した。特に世界中の今までの研究では遺伝的アルゴリズムの応用中、ノイズ関しての研究がなかった。そのため、私は遺伝的アルゴリズムにのノイズを入れて、ノイズの影響に関して定量的な解析を目的とする。さらに、ノイズに関わる確率共鳴現象（Stochastic Resonance）の研究を行った。

2．方法

本実験では、６つ元素を含む空リストを作り、この６つの元素はランダム的に生成し、この一つのリストは遺伝子をもつ個体と呼ぶ。それぞれの個体を評価するため、評価値をつけ、個体の評価値と呼ぶ。個体の評価値はリスト中の全元素の総和とする。それぞれの個体に対する目的関数 F の値を適応度と呼ぶ。初期集団の個体は適応度により、適応度が高い個体は親として選ばれ、ペア交叉を経って、次世代を創る。目的値は 371 とし（Target＝371)、リスト中の元素の和は 371 に近い個体（リスト)、つまり近似目的世代を作ってもらう。、自然進化メカニズムに真似した遺伝的アルゴリズムのオリジナル的アルゴリズムのプログラムを設計し、それぞれの目的に応じて、設定を変更しながら、プログラムを実行する。そして、得られたデータをグラフにし、結果を詳しく解析した。

適応度関数　　$F = \frac{\text{個体の評価値}}{\text{T}}$　　T＝371

3．結果

ノイズが環境の下で遺伝的アルゴリズムの働きは黒色のグラフのようにできている。ホワイトノイズが[-0.371, 371]の範囲内では、実行して反映されたグラフは赤色の方である。緑色のリンクのところを注目すれば、見た通りで赤色のグラフ即ちノイズがある時の方が先に０の値に近づいたことが分かる。つまり、本実験に使用された遺伝的アルゴリズムは、ノイズが[-0.371, 0.371]ある時の方が、ノイズがない時より、早く目的値に接近したことを示し、応答が良くなっていることが分かった。これは確率共鳴の現象が起こっていると考えられる。

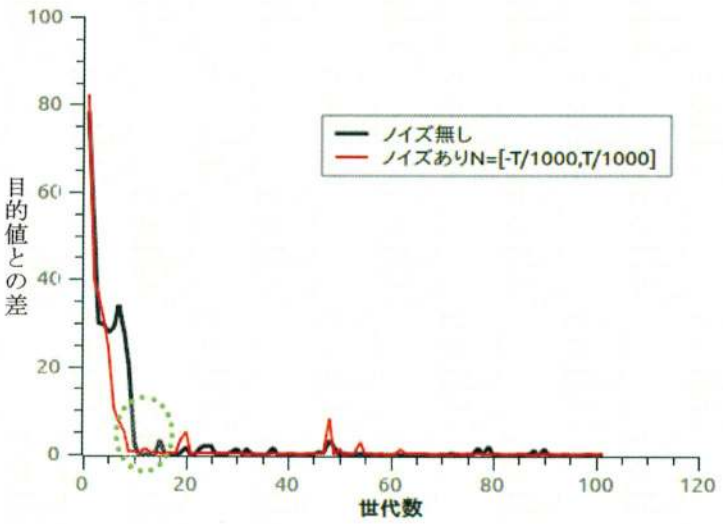

図　8　ノイズあると無しの時の遺伝的アルゴリズムの働き

4．結論

本研究は自然から学んだ知恵でできた遺伝的アルゴリズムは最適問題の解決手法として適応していることを実証した。そして、ノイズ導入実験により、ノイズがある方がない方より確実に早く目的値に接近したことが分かった。このことにより、確率共鳴は遺伝的アルゴリズムにも役割を果たしていることは、このオリジナル研究に成功した。また、ノイズの定量的分析を行い、ノイズ範囲を大きくすると、遺伝的アルゴリズムの働きにずっと悪い影響を与えることではなく、共鳴している可能性を示した。

＜卒業論文＞

革新的遺伝的アルゴリズムにおける確率共鳴信号の解析

横浜市立大学

国際総合科学部

国際総合科学科

基盤科学コース

学籍番号　１１０４０２

陳　金琳

指導教員

横浜市立大学　ルジェロ・ミケレット教授

目次

第1章　序論

1.1　研究背景

遺伝的アルゴリズム（Genetic Algorithms）は、ミシガン大学の John Holland により、1975 年に提案され、その後彼の弟子たちを中心に発展してきた。遺伝的アルゴリズムは生物進化のメカニズムを真似する人工モデルとして提唱されたものである。しかし、応用面での実用的な効果はあるものの、理論的な研究はそれほど進展してなかったためか、提唱後 20 年あまりも経過した最近になるまで、それほど注目されなかった。しかし、近年になって、遺伝的アルゴリズムは最適化問題の解法として注目され、応用されるようになってきている。

人工知能に関しての様々な問題において、複雑で大きな探索空間の中で最適な解をすばやく見つけることが常に望まれている。しかし、問題の規模が大きくなると厳密解を求めるには、計算の量が増えることとなり、現実的ではなくなり、そこである程度満足できる解を探す近似解法が利用される。その際、探索空間についての知識が先見的に利用できる時は早く解を探索できる。しかし、そうでない時は動的にその探索空間についての知識を蓄積しながら探索手順を制御しなくてはならない。このような問題は適応型探索問題と言われ、一般にその取扱が難しいと言われている。

遺伝的アルゴリズムはこのような問題において、特に有効な探索アルゴリズムである。つまり、問題に固有知識を利用できないためにうまく探索空間を狭められず、組み合わせ爆発を起こしてしまうような大きな探索空間を持つ問題に対しても適用することのできる汎用の探索法である。

また、我々の周囲には自然系及び非自然系に関わらず、ノイズ（雑音）が満ち溢れている。従来では、このノイズはシステムに好ましくないため、除去してきた。しかし、近年ではノイズの存在が有益な効果を果たしていることが判明された。それは確率共鳴（Stochastic Resonance）と呼ばれる現象であり、弱い信号にある微弱なノイズを加えると、元の信号が強まるという。

1.2　研究目的

本研究では、従来では計算量が多く、最適な解を求めるに時間がかかるものであり、次のような線形方程式

$$ax + by + cz + dw = const.$$

ｘ，ｙ，ｚとｙは変数で、a, b, c とｄは定数です。この計算問題は通常のやり方では求めることができないか計算量は重いこととなる。ここで、私は遺伝的アルゴリズムのオリジナルアルゴリズムを組み立て、この線形方程式の問題解決のやり方は深く解析した。特に世界中の今までの研究では遺伝的アルゴリズムの応用中、ノイズのことは無視されている。そこで、私は遺伝的アルゴリズムにのノイズを入れて、ノイズの影響に関して定量的な解析の目的とし、ノイズに関わる確率共鳴現象（Stochastic Resonance）の研究を行った。

第２章　　遺伝的アルゴリズムと確率共鳴の概念

2.1　遺伝的アルゴリズムのイメージ

生物が持つ形質には、生物個体の先天的持つ遺伝形質と、生物が生まれてから死に至るまでの間の、生存期間中に得る獲得形質の２つがある。その２つの形質ともに環境へ適合するために効果的に働いており、環境適合する形質を備えた生物個体が、その時点での大勢を占めることになる。そして、大勢を占めた生物個体は、繁殖によってその子孫たちを残し得るため、ますますその時点において、環境によく適応した生物群が増殖していくこととなる。その反面、適応ではない生物個体は、生存確率が低くなり、生存していても少数でしかなく、その繁殖による子供を残せる確率はますます低いものになってしまう。このような現象は、進化の流れの中では自然選択と呼ばれている。

進化的アルゴリズムと思われるように、この処理プロセスは進化する生物を創りだす。この生物は生存本能を持ち、生き残るために連続的にその目標を達成する必要がある。もし、目標を達成できなかった場合には、突然変異によって、親と少し異なる複数の子供が生み出される。この子供たちは、一般的には歴史的なデータ（生き残るために有利であった過去データ）を用いて比較され、どの子供がもっと目標を達成する可能性があるかが決定される。最良の生物（子供）が新たな親として育成され、残りの生物が淘汰される。新しい親は、引き続きその目標を達成できなくなるまで生き続ける。このようにして、進化過程をシミュレーションするのが、遺伝的アルゴリズムの基本である。

2.1.2　遺伝的アルゴリズムの処理プロセス

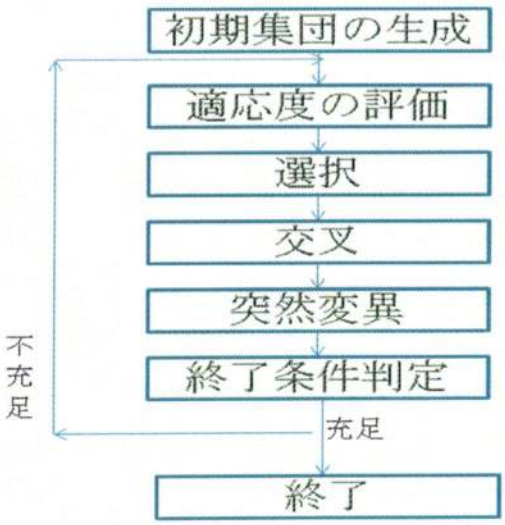

図１　遺伝的アルゴリズムの処理プロセス

遺伝的アルゴリズムの処理プロセスは図１に示したように、大き分けて次の６段階で実施される。

① 初期集団の生成：　ランダムに初期集団を生成する。

② 適応度の評価：　適応環境における各個体の適応度の算定を処理し、評価される。

③ 選択：　適応度の高い個体を選び、適応度の低い個体を削除処理する。

④ 交叉：　選ばれた個体ペアに対して交叉をさせ、新しい個体を創る。

⑤ 突然変異：　個体群の多様性を保つため、親と異なる遺伝子を創り、働きを調節する。

⑥ 終了条件判定：　終了条件が満たされば終了し、満たされなければ②へ戻る。

2.2　確率共鳴のメカニズム

確率共鳴とは、ノイズを加えることにより、システムの応答が入力信号に対して共鳴しやすくなる現象である。もともとは大氷河期の周期的到来を説明するに導入された概念であった。現在では、ある閾値をもつ興奮系図２のように、入力された信号は閾値を超えなければ、出力できない。しかし、あるノイズを加えることで、閾値を超えて出力される確率が高くなる現象を説明する。

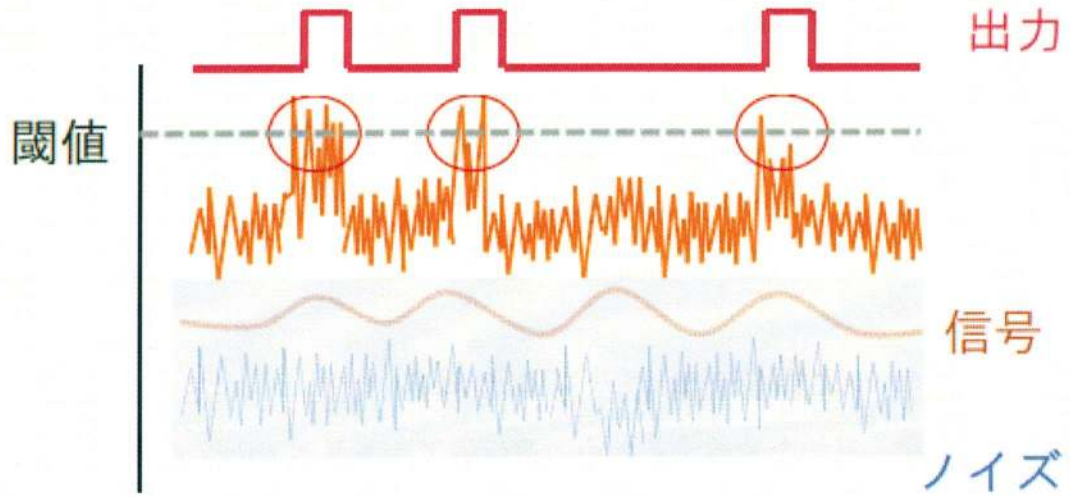

図２　確率共鳴のメカニズム

第３章　遺伝的アルゴリズムにおける確率共鳴の解析実験

3.1　実験準備

本実験では、６つ元素を含む空リストを作り、この６つの元素はランダム的に生成し、この一つのリストは遺伝子をもつ個体と呼ぶ。それぞれの個体を評価するため、評価値をつけ、個体の評価値と呼ぶ。個体の評価値はリスト中の全元素の総和とする。それぞれの個体に対する目的関数 F の値を適応度と呼ぶ。初期集団の個体は適応度により、適応度が高い個体は親として選ばれ、ペア交叉を経って、次世代を創る。目的値は 371 とし（Target＝371）、リスト中の元素の和は 371 に近い個体（リスト）、つまり近似目的世代を作ってもらう。以上本実験の要旨である。

適応度関数　　$$F = \frac{\text{個体の評価値}}{\text{T}}$$　　T＝371

3.2　実験手法

3.21　ノイズがない時に遺伝アルゴリズムの働き

1）実験始まる前に、目的を達成するための遺伝的アルゴリズムのプログラムを設計し、起動する。（本実験に使うプログラムは論文の最後に添付していている）

2）様々な設定を行う。まず、初期集団の個体数を 100 に設定し、リスト中の元素の値を 0～100 に設定し、選択率を 0.2 に設定し、ランダムで親を生成する率を 0.05 付加し、突然変異率は 0.01 に設定する。また、100 回繰り返してもらう。つまり、100 世代を作ってもらう。

3）プログラムを実行し、目的値にアプローチするプロセスをグラフに表す。

4）ここで、突然変異率の大きさは目的値にアプローチするスピードに関係している化どうか調べるため、突然変異率の大きさを 0.01、0.05、0.21、0.31、0.41、0.51 と 0.61 におき、プログラムを実行する。取り得たデータをグラフにする。

3.22　ノイズがある時の遺伝的アルゴリズムの働き

1）3.21 の 1）に組まれたプログラムにノイズ関数を組み込む。

ノイズ関数　$N = X * \frac{T}{1000}$　　T=371（Target）

2）ノイズの範囲を X の変化で加減し、プログラムを実行し最良のノイズ範囲を調べる。取り得たデータをグラフにする。

第４章　　実験結果と考察

4.1　実験結果

ノイズがない時、遺伝的アルゴリズムを使って、実際で目的値（Target）にアプローチするプロセスを以下の図３に示す。横軸は世代数を表し、縦軸は各世代において、最良個体の評価値と目的値との差を表す。分かりやすいため、近似線も引いた。

近似線の関数　　$f = A*\exp(-x/\tau)+y_0$

A=42.8728
t=4.5079
y_0=0.0982

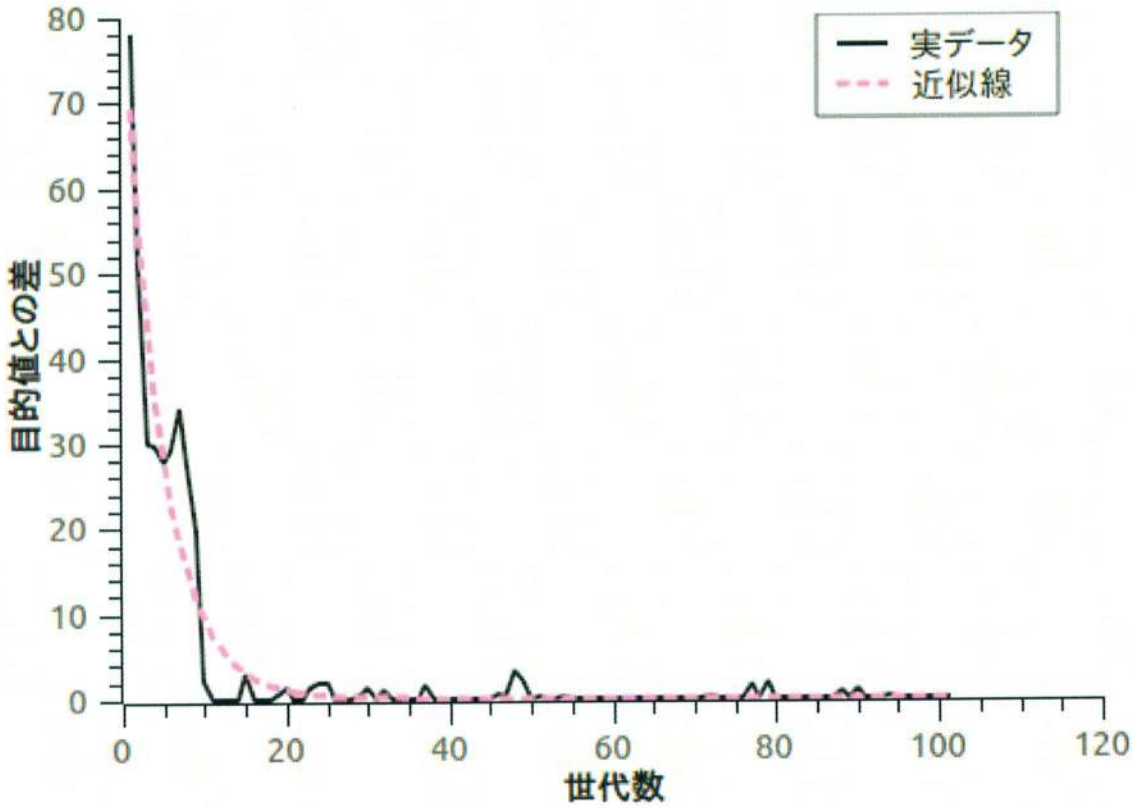

図　３　目的値にアプローチするプロセス

以上のグラフにより、世代数が増加するにつれ、最良個体の評価値と目的値の差が減少していることが分かった。まるで、世代交替が進んでいるうち、各世代の個体はより良い子供を創るため、各自の評価値を目的値に近づこうとしている様子に見える。以上により、遺伝的アルゴリズムの働きが自然進化メカニズムに似ていると言える。これは遺伝的アルゴリズムにある学習階段である。平坦になったグラフにまた突起しているところが見えるが、それは突然変異が起こり、高い適応度を持つ個体を破壊してしまたからである。

次に、突然変異率を設定した値に変化させながら、とれたデータをグラフにした。横軸は世代数で、縦軸は各世代において、最良個体の評価値と目的値との差を表す。見やすいため、30 世代までのグラフを拡大し、図４のように表示した。

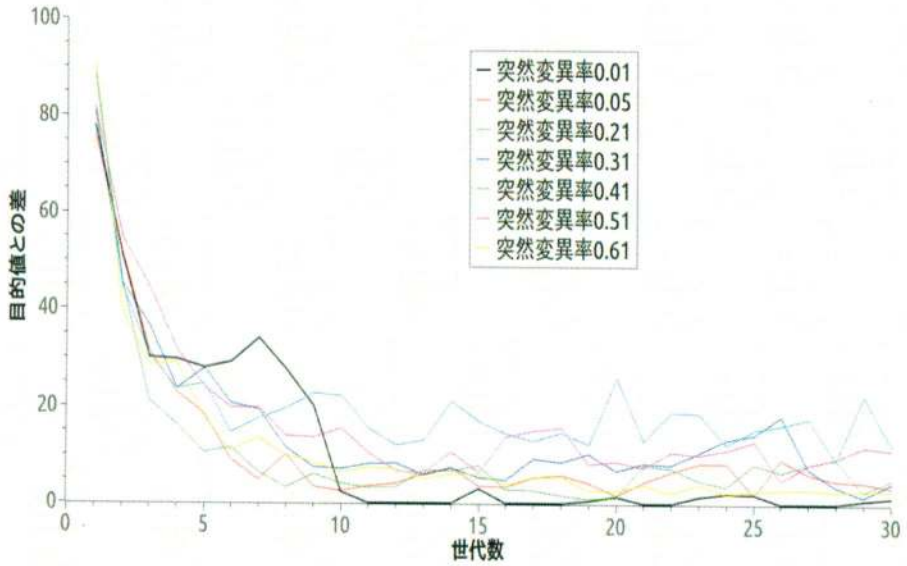

図　4　突然変異率変化により遺伝的アルゴリズムの働きⅠ

図４のグラフにより、一つ面白い現象が見られた。突然変異率は 0.01 から 0.41 までの間に、大きさを上げるにつれ、各世代の最良個体の評価値と目的値の差が早くお落ちるに対し、突然変異率の大きさをそれ以上上げると、各世代の最良個体の評価値と目的値の差が前より遅く落ちることを発見した。

図４の動きを分かりやすいため、それぞれの突然変異率の下で引いたグラフの近似線をとった。図４の近似線をとる時に使う関数は

$$f = A*\exp(-x/\tau)+y_0$$

それぞれの近似線をとった時の収束時間定数を元に図５のグラフ描いた。収束時間定数τは、目的値に接近する速度を示している。τが小さければ、速度が早いことを示している。

図５の近似線の関数

収束の時間定数　　$\tau = a_0+a_1 \cdot x+a_2 \cdot x\hat{}2$

$a_0 = 5.4898$

$a_1 = -1.7404$

$a_2 = 0.1870$

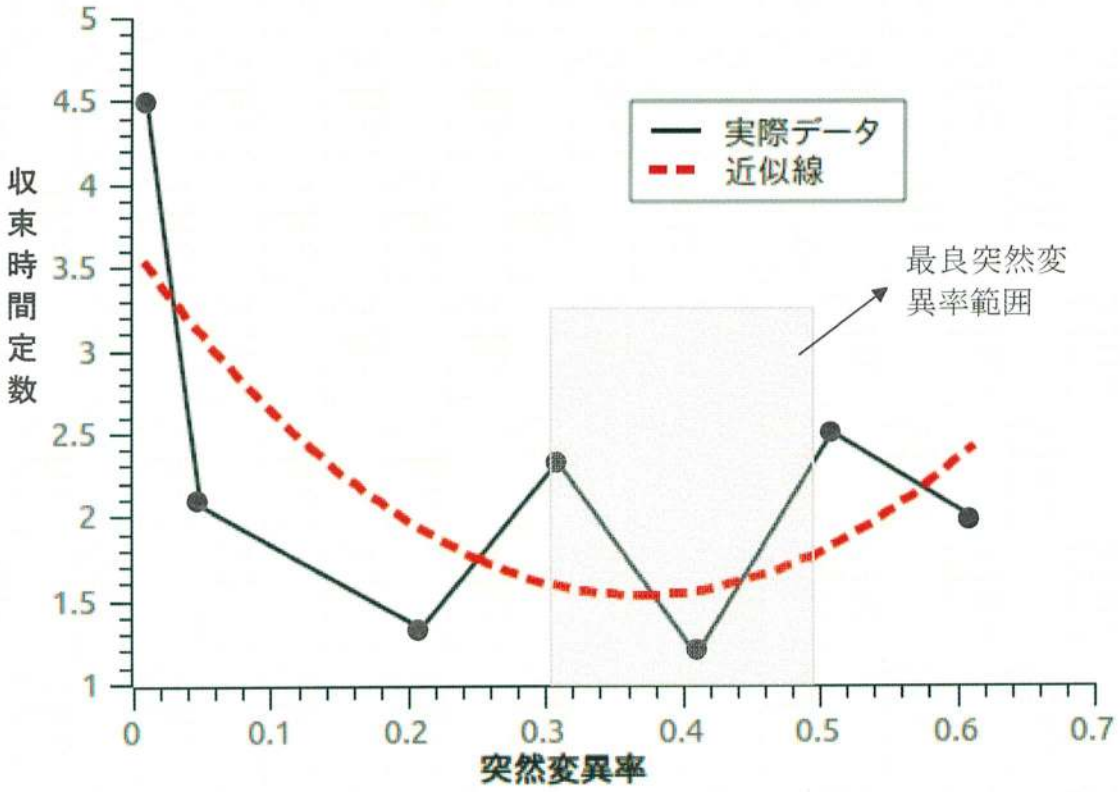

図　５　突然変異率の変化と遺伝アルゴリズムの働きⅡ

図５のグラフの近似線に示した通り、収束の時間定数が低ければ、目的値に接近する速度が速くなるとする。自然システムでは、突然変異率が小さいため、本実験に使

う遺伝的アルゴリズムのプログラムにも突然変異率の設定を小さくした。突然変異とは、親と異なる遺伝子を創るという。だから、親がいい遺伝子を受け継ぐことがなくても自分自身に変異ができるため、突然変異率が大きくなると、変異が起こりやすくなる。そのため、目的値に早く接近することが考えられる。しかし、過剰な突然変異は適応度の優秀な固体を破壊してしまうため、図５のように高い突然変異率の下では、目的値に接近する速度が落ちてしまった。

図５により、突然変異率は 0.3～0.5 の間に、収束の時間定数が低いところにある。つまり、この範囲内の突然変異率は本実験に使う遺伝的アルゴリズムの働きを促進する傾向があることが分かった。

次に、ノイズをプログラムに組み、ノイズの範囲を７つのポイントに分けた。それぞれは

ポイント１：　－T/1000≦N≦T/1000
ポイント２：　－1.5T/1000≦N≦1.5T/1000
ポイント３：　－2T/1000≦N≦2T/1000
ポイント４：　－2.5T/1000≦N≦2.5T/1000
ポイント５：　－3T/1000≦N≦3T/1000
ポイント６：　－3.5T/1000≦N≦3.5T/1000
ポイント７：　－4T/1000≦N≦4T/1000

以上の７つのノイズ範囲が存在する環境の下で、遺伝的アルゴリズムの働きが図６のようになった。横軸は世代数であり、縦軸は各世代において、最良個体の評価値と目的値との差を表す。

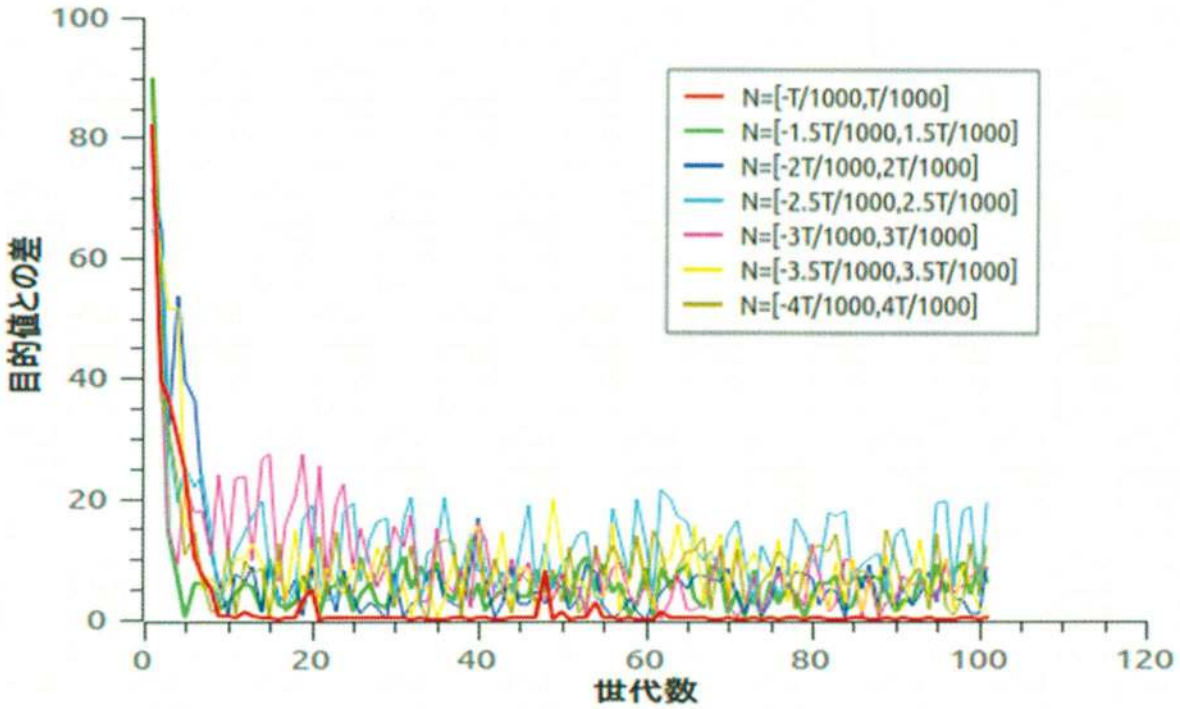

図　６　ノイズによる遺伝的アルゴリズムの働き

図６では、グラフが重なっていて見づらいため、２番目からのグラフのデータをすべてプラスαにして、離れるようにして、図７のグラフを描いた。

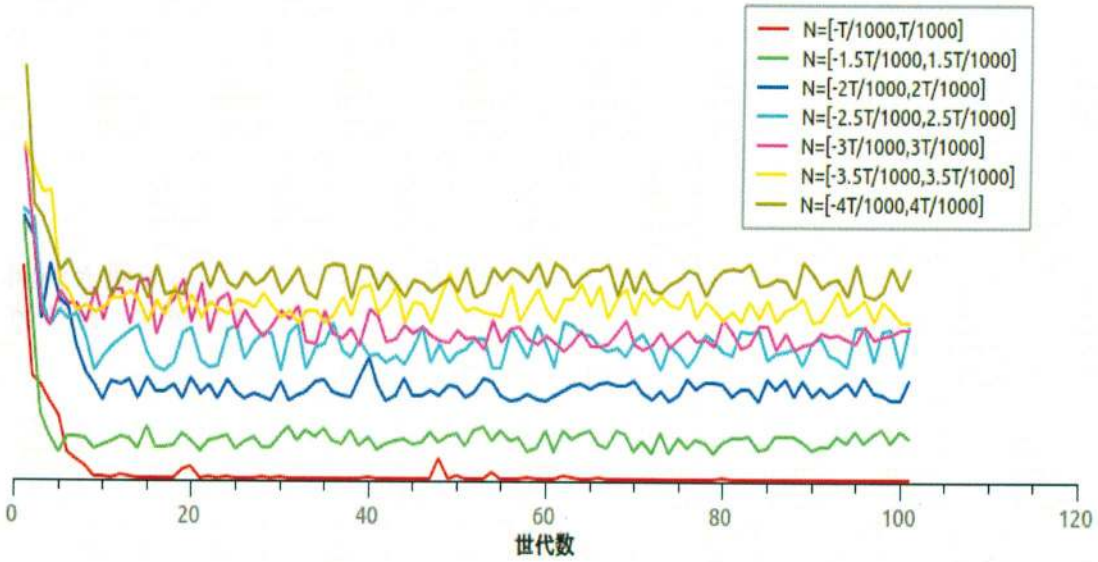

図　７　ノイズによる遺伝的アルゴリズムの働き

図７により、赤色のグラフが一番良い働きをしていることが分かった。ノイズがそれ以上大きくなると、常に目的値と近いところに保つことができなくて、赤色のグラフのように、動きが落ち着いていることなく、激しい変化していることが分かる。つまり、本実験において、ノイズのある環境の下で、遺伝的アルゴリズムはノイズの範囲が[-T/1000, T/1000]の時では、応答が一番いいことが分かった。

そして、図７にある赤色のグラフをと図３にある実際データで引いたグラフを比較するため、つまり遺伝的アルゴリズムの働きはノイズがある時と無いときの比較を行い、図８に示す。

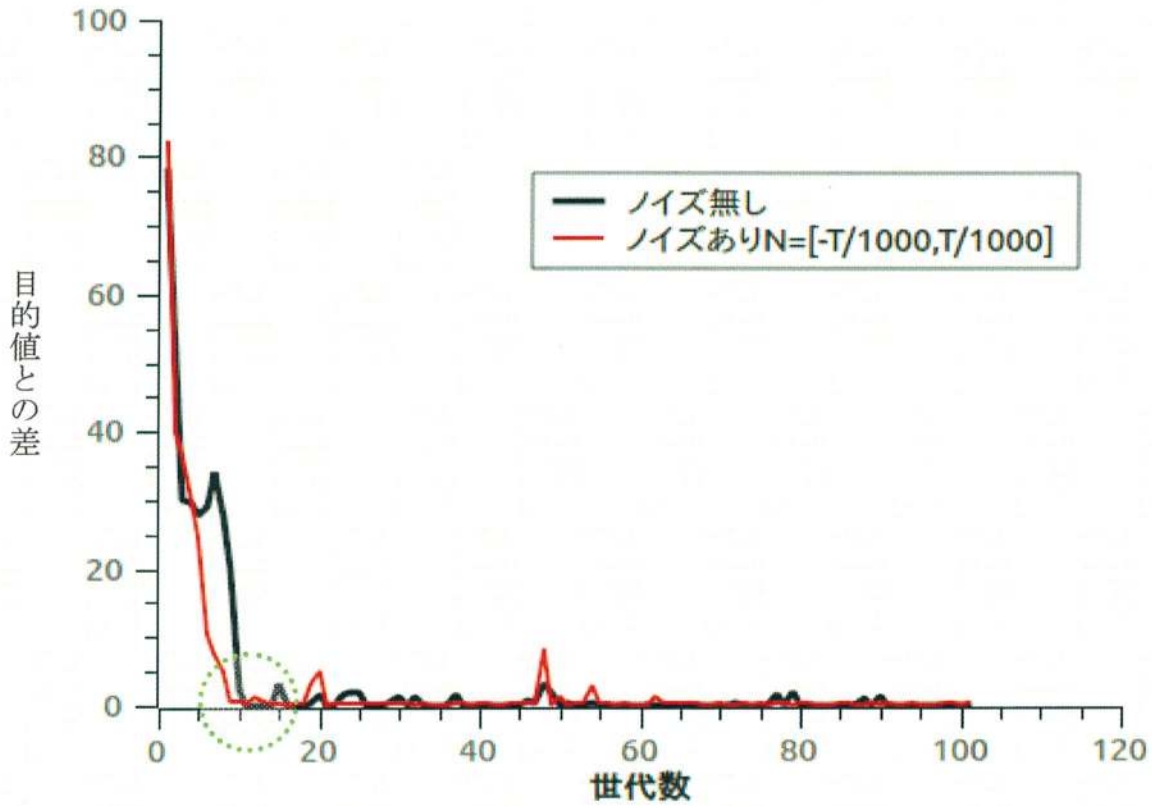

図　８　ノイズあると無しの時の遺伝的アルゴリズムの働き

図８が示した通り、ノイズが環境の下で遺伝的アルゴリズムの働きは黒色のグラフのようにできている。ノイズが[-T/1000, T/1000]の範囲内では、実行して反映されたグラフは赤色の方である。緑色のリンクのところを注目すれば、見た通りで赤色のグラフ即ちノイズがある時の方が先に０の値に近づいたことが分かる。つまり、本実験に使用された遺伝的アルゴリズムは、ノイズが[-T/1000, T/1000]ある時の方が、ノイズがない時より、応答が良くなっていることが分かった。これは確率共鳴の現象が起こっていると考えられる。

最後に、図６の各グラフの近似線をとり、その時の近似線の関数は

$$f = A*\exp(-x/\tau)+y_0$$

そして、それぞれのグラフを以上の関数で取った収束時間定数τを記録し、グラフにして分析した。図９に示す。

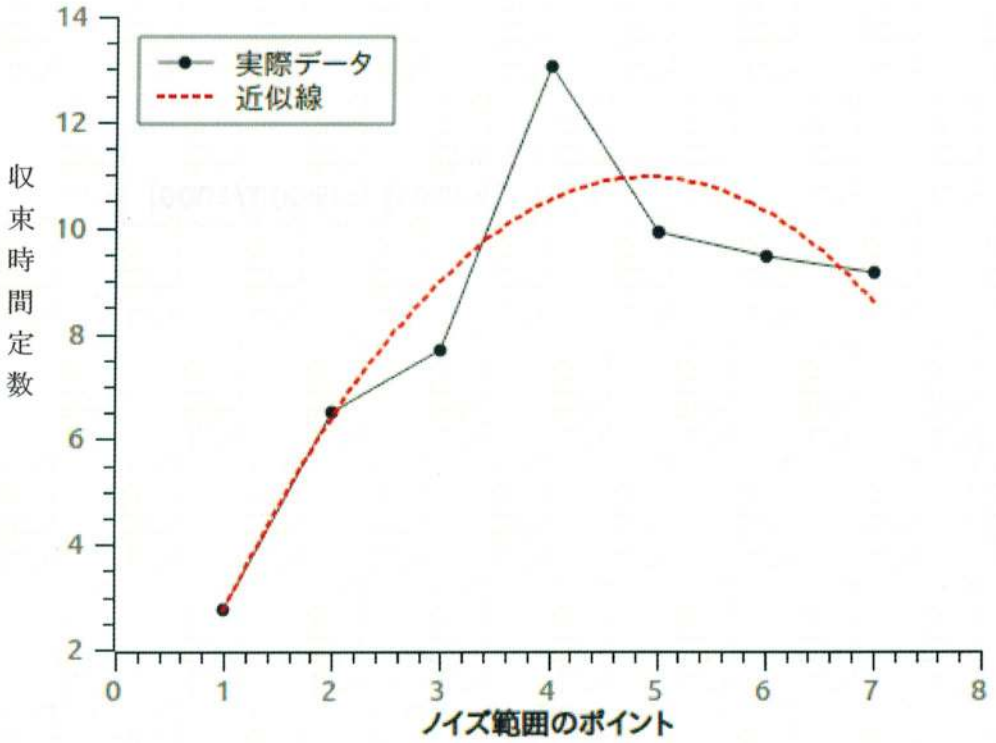

図　９　ノイズ影響の分析

図９より、ノイズ範囲を上げると、収束時間定数のτがずっと大きくなるに限らない。つまり、ノイズ範囲は大きくすると、遺伝的アルゴリズムの働きが悪化する一方ではないことが分かった。図９のグラフの近似線によれば、ノイズ範囲がポイント５のところにピークが現れ、その以降はまた落ちる方向に向かっていることが分かる。これは、遺伝的アルゴリズムの働きがノイズによる共鳴現象だと考えられる。

第５章　　結論

5.1　まとめ

本論文は、従来では計算量が多く、解を求めるのに時間がかかる多元線形方程式は遺伝的アルゴリズムの利用で、簡単で早く解決できることを検討した。そして、遺伝的アルゴリズムの働きには確率共鳴現象が起こるかどうか検討した。

初めに、自然進化メカニズムに真似した遺伝的アルゴリズムのプログラムを設計し、調べにより設定を行い、それぞれの目的をもって実行した。最初に、本実験に使われる遺伝的アルゴリズムのプロセスをグラフにし、得られたグラフは目標個体に接近しようとしている様子だった。このことにより、遺伝的アルゴリズムのプロセスは自然進化メカニズムに似ていることを分かった。

次に遺伝的アルゴリズムの働きに関係している可能性がある突然変異率の影響を調べた。結果として、突然変異率が 0.3～0.5 の範囲内では、遺伝的アルゴリズムの働きがいいことを示した。また、過剰な突然変異率は優秀な固体を破壊してしまうことが図５により分かった。

また、ノイズ関数を遺伝的アルゴリズムのプログラムに組み、７つポイントの範囲のノイズにおき、遺伝的アルゴリズムの働きを調べた。その結果として、ノイズが[-T/1000, T/1000]ある時の方は、ノイズがない時より、遺伝的アルゴリズムの応答が良くなっていることが分かった。これは確率共鳴により遺伝的アルゴリズムの働きを促進することを検証した。

以上より、本研究は自然から学んだ知恵でできた遺伝的アルゴリズムは最適問題の解決手法として適応していることを実証した。そして、確率共鳴は遺伝的アルゴリズムにも役割を果たしていることは、このオリジナル研究に成功した。また、ノイズの定量的分析をし、ノイズ範囲が大きくすると、遺伝的アルゴリズムの働きにずっと悪い影響を与えることではなく、共鳴している可能性を示した。

5.2　応用的意義

遺伝的アルゴリズム手法は素早く最適問題を解決してくれるのが特徴で、忙しい人のために、予定とかイベントを入力するだけで、自動的にスケジュールを作成してくれるアプリケーションにの応用が期待できる。また現在、プログラムによってピュータが様々なデータを処理することができているが、一般的には人間の手でより良い手順書が与えられる。ここで、遺伝的アルゴリズムの応用により、コンピュータも人間のように自ら学び、自ら問題を解決するようなプログラムが作られることが期待されている。

参考文献

[1] 棟朝雅晴　「遺伝的アルゴリズムーその理論と先端的手法」森北出版株式会社 2008

[2] 米澤保雄　「遺伝的アルゴリズムー進化理論の情報科学」森北出版株式会社 1993

[3] 伊庭斉志　「遺伝的アルゴリズムと進化のメカニズム」株式会社岩波書店 2002

[4] Goldberg, D.E.: Genetic Algorithms in search, Optimizations and Machine Learning, Addison Wesley, 1989

[5] Illinois Genetic Algorithm Laboratory
http://www-illigal.ge.uiuc.edu/ イリノイ大学GA研究室のサイト

謝辞

本研究を進めるにあたって、指導教員であるルジェロ・ミケレット教授は、科学的知識が薄い私に一から指導し、研究活動に終始世話して頂きました。また、研究活動において、困難に遇った時、ミケレット教授はいつも励ましてくださって、心から感謝致します。ありがとうございました。

また、外国人である私を暖かく手伝って下さった研究室の皆様にお礼を申し上げます。優しい研究室の皆様の御蔭で、本研究がスムーズに完成でき、本当にありがとうございました。

##本実験で使用するプログラム##

```
from random import randint, random
from operator import add
import matplotlib.pyplot as plt

def individual(length, min, max):
    'Create a member of the population.'
    return [ randint(min,max) for x in xrange(length) ]

def population(count, length, min, max):
    """
    Create a number of individuals (i.e. a population).

    count: the number of individuals in the population
    length: the number of values per individual
    min: the minimum possible value in an individual's list of values
    max: the maximum possible value in an individual's list of values

    """
    return [ individual(length, min, max) for x in xrange(count) ]

def fitness(individual, target):
    """
    Determine the fitness of an individual. Higher is better.

    individual: the individual to evaluate
    target: the target number individuals are aiming for
    """
    sum = reduce(add, individual, 0)
    return abs(target-sum)

def grade(pop, target):
    'Find average fitness for a population.'
    summed = reduce(add, (fitness(x, target) for x in pop))
    return summed / (len(pop) * 1.0)

def evolve(pop, target, retain=0.2, random_select=0.05, mutate=0.01):
    graded = [ (fitness(x, target), x) for x in pop]
    graded = [ x[1] for x in sorted(graded)]
    retain_length = int(len(graded)*retain)
    parents = graded[:retain_length]
    # randomly add other individuals to
    # promote genetic diversity
    for individual in graded[retain_length:]:
        if random_select > random():
            parents.append(individual)
    # mutate some individuals
    for individual in parents:
        if mutate > random():
            pos_to_mutate = randint(0, len(individual)-1)
            # this mutation is not ideal, because it
            # restricts the range of possible values,
            # but the function is unaware of the min/max
            # values used to create the individuals,
            individual[pos_to_mutate] = randint(
```

```
                min(individual), max(individual))
    # crossover parents to create children
    parents_length = len(parents)
    desired_length = len(pop) - parents_length
    children = []
    while len(children) < desired_length:
        male = randint(0, parents_length-1)
        female = randint(0, parents_length-1)
        if male != female:
            male = parents[male]
            female = parents[female]
            half = len(male) / 2
            child = male[:half] + female[half:]
            children.append(child)
    parents.extend(children)
    return parents


target = 371
tg1p=1*target/100

p_count = 100
i_length = 6
i_min = 0
i_max = 100
p = population(p_count, i_length, i_min, i_max)
fitness_history = [grade(p, target),]
for i in xrange(100):
    X=2*random()-1 # from -1 to 1
    rTarget=target+X*tg1p
    p = evolve(p, rTarget)
    fitness_history.append(grade(p, rTarget))

f = file('test11.txt', 'w') # open for 'w'riting
for datum in fitness_history:
   print datum
   f.write(str(datum))# write text to file
   f.write('¥n')
#plt.plot(fitness_history,".-",color='red')
#plt.show()


f.close()
```