Emaergent Computation(EC)

指導院生：渡邉 真也　奥田 環
チーフ：青井 桂子　　サブチーフ：伏見 俊彦　小池 政輝

2001 年 4 月 20 日

1　GA の歴史

GA は，1960 年代の終わりから 1970 年代のはじめに，J.Holland と彼の同僚やミシガン大学の学生らによって，自然界のシステムの適応過程を説明し，生物進化のメカニズムを模擬する人工モデルとして提唱されてきた手法であると見なされている．その後，応用面での実用的な成果はあるものの，理論的な研究のは点がそれほど見られなかったため，比較的最近になるまで世間からはそれほど注目されなかった．しかし 1985 年に，カーネギー・メロン大学において，第 1 回の遺伝的アルゴリズムに関する国際会議が開催されたことが契機となり，最適化・適応・学習のための方法論として急速に注目されはじめ，多方面に応用されるようになってきている．

2　GA の概要とプロセス

2.1　GA の概要

遺伝的アルゴリズム (Genetic Algorithm: GA) は，適応範囲の非常に広い，生物の進化を工学的にモデル化し，また参考にした学習的アルゴリズムである．基本的な GA として，Goldberg によって提案された**単純 GA(simple genetic algorithm: SGA)** があり，そこから種々なバリエーションが提案されている[1]．

GA では自然界における生物の進化過程と同様に，ある**世代 (generation)** を形成しているいくつかの**個体 (individual)** の集合，すなわち**個体群 (population)** の中で，環境への**適合度 (fitness)** の高い個体が高い確率で**選択 (selection)** される．その個体に対して，遺伝子の**交叉 (crossover)** や**突然変異 (mutation)** が，ある確率で発生することによって次の世代の個体群が形成されてゆき，最後に得られた個体群の中で最も適合度の高い個体が最良システムとなる．

各個体は，それぞれ**染色体 (chromosome)** によって特徴付けられており，染色体は**遺伝子 (gene)** の集まりで構成されている．また，通常 GA では 1 染色体で 1 個体を表現 する．染色体上で各遺伝子のおかれている位置を**遺伝子座 (locus)** といい，**対立遺伝子 (allele)** としては，2 進数のビット {0,1} を用いることが多い．GA で扱う情報は**遺伝子型 (GTYPE)** と**表現型 (PTYPE)** の二層構造からなる．

- **GTYPE**…GA のオペレータの操作対象，染色体の構造にあたる (ex. 01110, 00110; 0,1 のビット列).
- **PTYPE**…環境に応じて PTYPE から適合度が決まる，環境内での構造を示す (ex. 9, 5; 具体的な値).

なお，PTYPE から GTYPE へ写像することを**コード 化**[2]**(coding)**，逆に GTYPE から PTYPE へ逆写像することを**デコード 化 (decoding)** という．PTYPE によって適合度が決まり，その適合度が大きいものほど子孫を作りやすく，小さいものほど死滅しやすい (淘汰) ようになっている．このことにより，次世代の各個体の適合度は前世代よりも良いことが期待される．

[1] 分散 GA(DGA), 環境分散 GA(DEGA) など
[2] 具体的には，binary-coding, Gray-coding などの手法がある．

LINGO

世　　代 ··· ＧＡにおける手順の繰り返しの１単位
個　　体 ··· 解に対応するもの
適 合 度 ··· 個体の良好さを示す値
染 色 体 ··· 個体を記号列や文字列で表したもの
遺 伝 子 ··· 染色体を構成する最小単位, GA の最小構成要素
遺伝子座 ··· 各遺伝子のおかれている場所をさす
対立遺伝子 ··· 各遺伝子がとることのできる遺伝子候補
遺伝子型 ··· オペレータの操作対象となる染色体構造
表 現 型 ··· 与えられた環境内での構造表現型によって適合度が決まる
コード化 ··· 遺伝子型から表現型への変換 Binary-coding, Gray-coding などがある
局 所 解 ··· 最適解ではない，極大 (極小) 値などの解
早熟収束 ··· 最適解ではなく，局所解を発見して探索を終了してしまった状態
選　　択 ··· 適合度の高い個体を次世代に残すためのオペレータ．ルーレット選択・ランク選択・トーナメント選択などがある
エリート ··· 母集団の中で最も適合度の高い個体エリート戦略では次世代にエリート個体が必ず保存される
交　　叉 ··· ２個体の染色体を組み替えるオペレータ一点交叉・多点交叉・一様交叉が代表的
交 叉 率 ··· 交叉を行う確率を定めたもの
突然変異 ··· ある遺伝子を他の対立遺伝子に置き換えるオペレータ
突然変異率 ··· 突然変異を行う確率 1/L(遺伝子長) が一般的
致死遺伝子 ··· 解の候補となり得ない遺伝子型の個体 GA における探索の妨げとなるため致死遺伝子の発生を防ぐ必要がある
多 様 性 ··· 母集団全体で各個体にどれだけ差があるかを示す

2.2 GA のプロセス

1. 問題のモデル化：対象問題から, 染色体の遺伝子配列・適合度関数・GA パラメータの数値を決定する．
2. 初期集団の生成：初期集団になる遺伝子型をランダムに生成する．
3. 染色体の評価：その環境における各個体ごとの適合度を算出する．
4. 適合度での選択：各個体の適合度に基いて次世代個体を決定する操作．
5. 交叉処理：２つの染色体を互いに入れ替え，新しい個体を生成する．
6. 突然変異操作：個体上のある遺伝子を一定の確率で変化させる．
7. 終了判定：終了条件を満たしていれば，母集団内の適合度が最も高い染色体を解とする．

このような GA が古典的な探索法と大きく異なる点は，Goldberg によれば，次の 4 つの特徴にある．

- パラメータをコーディングしたものを直接利用する．
- 一点探索ではなく，多点探索である．
- サンプリングによる探索で，ブラインドサーチである．
- 決定論的規則ではなく，確率的オペレータを用いる探索である．

2.3 GA の特徴

2.3.1 GA の長所

- 任意の解の良さ (適合度) を一意に決められる問題であれば適応できる (適応できる問題が広い)．
- 組合わせ爆発を起こすような広大な探索空間を持つ問題に対しても適用できる．
- 局所解をもつ問題 (騙し問題) に対しても有効である．

2.3.2　GA の短所

- 広域な探索は得意だが，局所探索は不得手である．
- より良い解を求めるには個体数を増やす必要があるが，個体数を増やすと計算に時間がかかる．
- 早熟収束によって，局所解へ収束してしまう．
- パラメータの設定が複雑である．

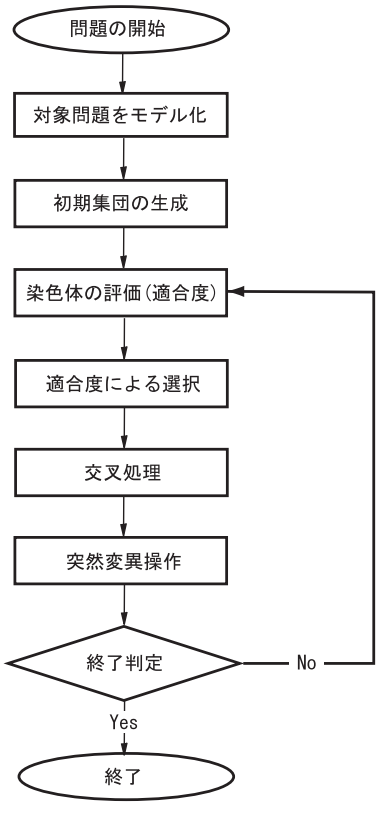


Fig. 1　GA の構成プロセス

3　GA における最適化探索

最適化探索においてもっとも重要となることは，

“ **最適値 (optimum value) を正しくしかも早く見つけられるか** ”

ということである．よって，これが満たされない場合には，探索は失敗に終わったことになる．失敗の主な原因は，

“ **最適値ではない解を見いだして終わる** ”

ということである．最適値ではない解 (不要な解) は**局所解 (local optimum)** と呼ばれる．局所解で探索が終了した場合には，最適解である最大 (最小) 値ではなく，極大 (極小) 値を見つけて探索が終わってしまったことになる．GA においては，このような現象を**早熟収束 (Premature Convergence)** と呼ぶ．この問題を解決するために，GA の基本技法として，次の技法がある．

- **スケーリング技法**
- **選択方式**
- **GA オペレータ**

3.1　スケーリング技法

適合度の評価が不適当であると，選択の際に不都合が生じることがある．その際に，選択率を適当なものに変換する手法が**スケーリング技法 (scaling)** である．

まず，不都合がどのように起こるかの例を示す．選択には様々な手法があるが，ここではルーレット方式の場合を例にとって説明する．

三つの個体の適合度が，次のようであったとする．

$f_1 : 10.0$
$f_2 : 5.0$
$f_3 : 20.0$

このとき，選択にかかる割合が適合度の値に比例すると仮定する．つまり各個体の適合度を f_i，選択される割合を $pselect_i$ とすると，

$$pselect_i = \frac{f_i}{\sum f_i} \tag{1}$$

である．これは，選択の項目で説明するルーレット選択の手法である．さて，上記の適合度に式 を用いると，採用される割合は，

f_1 28.6% $(10.0/(10.0 + 5.0 + 20.0))$
f_2 14.3% $(5.0/(10.0 + 65.0 + 20.0))$
f_3 57.1% $(20.0/(10.0 + 5.0 + 20.0))$

となる．これは 1 回の選択によって各個体が選ばれる割合を示している．この結果から明らかなように，f_3 は他の個体と比較して選択されやすくなっている．このことを，f_3 には**淘汰圧 (selection pressure)** が強いという．

さて，最大値 100 の問題に対して，探索が進み，各個体の適合度が次のようになったとする．

f_1 $80.0 \cdots 32.7\%$
f_2 $75.0 \cdots 30.6\%$
f_3 $90.0 \cdots 36.7\%$

適合度に対して，選択される割合を上記に示した．最大値である 100 に最も近い f_3 への淘汰圧は他の f_1, f_2 と比較してそれほど強いとはいえない．

この例のように，適合度の差が淘汰圧にうまく反映されないことが起こる．特に探索の後半で効率が劣化することがある．こうした場合にも適当な淘汰圧を実現しようとする手法がスケーリング技法である．スケーリング技法には主に次の 3 つが存在する．Table 1 で簡単に示しておく．どの手法においても，適合度が負になった場合には，適合度 $= 0$ とする．

Table 1　代表的なスケーリング技法

名称	変換式	概説
線形スケーリング	$f' = af + b$	スケーリング前後で平均適合度が変化しないように設定する必要がある．
シグマ (σ) スケーリング	$f' = f - (f_{avg} - c \times \sigma)$	個体群の大部分が高い適合度を持ち，残りのわずかの個体が非常に小さな値を持つ場合に有効．
べきスケーリング	$f' = f^k$	k は問題の構造に依存し，世代の進化に応じて変更する必要もあるので，一般には適切に使用するのは困難．

3.2　選択

選択 (selection) は，適合度の高い個体が次の世代により多い子孫を残すという自然淘汰 (natural selection) の考えに基づくもので，いくつかの方法が提案されている．

GA においては，選択は n 個体の集合から重複を許して n 個体を選ぶということである．このとき，適合度の高いものほどたくさん選ぶようにする．この選び出された n 個体に対して，交叉や突然変異を行って次世代の子を作り出す．以下で代表的な選択の手法について述べる．

3.2.1　ルーレット方式

ルーレット方式 (roulette wheel selection) は，個体を適合度に比例した割合で選択する方法である．これは，適合度に比例した領域を持つルーレットを回し，ルーレットの玉が入った領域の個体を選び出すというものである．乱

数で個体を選ぶため，適合度の低い個体も選ばれる可能性が残る．

この方式を形式的に記述すると次のようになる．

> $f_1, f_2 \cdots f_5$ の n 個の個体とおのおのの適合度が与えられたとき，i 番目の個体が選択される確率は，
> $p_i = f_i / \sum f_i$ となる．

次に簡単な例を示す．

f_1	5.0	⇒	0.0〜5.0 のとき選択
f_2	2.5	⇒	5.0〜7.5 のとき選択
f_3	1.5	⇒	7.5〜9.0 のとき選択
f_4	3.0	⇒	9.0〜12.0 のとき選択
f_5	2.0	⇒	12.0〜14.0 のとき選択
合計	14.0		

上記のような場合を考える．このとき重み付けのルーレットでの選択を次のように実現する．適合度の合計が 14.0 であるので，0 から 14 までの乱数を一様に発生させて，上記の規則で選択する．たとえば，乱数が次のようなものであったとする．

$$8.5, 3.0, 4.5, 11.5, 10.0$$

この乱数に対して，上記の法則を適用すると，

$$f_3, \quad f_1, \quad f_1, \quad f_4, \quad f_4$$

という個体が選ばれることになる．

3.2.2 ランク方式

ランク方式は，各個体を適合度の大きいものから順に並べ，その順位に応じた関数により子の数を決める方式である．この方式では，ランクの決定にソートが必要となることや，関数の決定，さらに端数 (小数点以下の数) の割り当てなどの問題が残る．

3.2.3 トーナメント方式

トーナメント方式は集団の中からある個体数 (S_t, トーナメントサイズ) をランダムに選び出して，そのなかで一番適合度の高いものを選択，この過程を集団数が得られるまで繰り返す手法である．この手法では，トーナメントサイズによって様々のバリエーションがある．

3.2.4 選択の機能

ここまで，代表的な選択方式を述べた．ここで選択の機能をまとめておく．n 個体の親から n 個体の子の候補を取り出す．そしてそれらに GA オペレータ (交叉，突然変異など) を適用して n 個体の子を作り出す．これが最も基本的な構成であるが，必ずしも子を n 個体つくり出す必要はない．ここで世代間のギャップ (generation gap) という変数 G を導入し，$n \times G$ を子供の個体数とする．すなわち，

$G = 1.0$	⇒	n 個体すべてが子供と入れ替わる．
0.0<G<1.0	⇒	n 個体中 $n \times G$ 個体が子と入れ替わる．

とする．このように選択の方式を拡張すると，親の個体の特性をある程度保持した GA の世代交代が実現できる．$n \times (1 - G)$ 個体の親は GA オペレータの適用を受けずに次の世代にコピーされるからである．

さて，ここで問題となるのは，$n \times (1 - G)$ 個体にコピーされる親をどのように選ぶのか，ということである．その手段として，**エリート戦略**がよく知られている．

エリート戦略

エリート戦略では，適合度の最も高い親を無条件にそのまま残す．これによって，最良の個体は破壊されずに次の世代でも保持される．この方式は一般に探索能力が優れているとされているが，親の成績をソートする手間がかかること，探索が局所解に陥る場合があるなどの問題がある．

3.3 GA オペレータ

GA オペレータには，さまざまなものがあるが，ここでは，一般によく用いられる GA オペレータである，交叉と突然変異について以下で説明する．

3.3.1 交叉オペレータ

選択によって適合度の高い個体が母集団内に増えるが，それだけでは初期分布のなかで評価値の高い個体が母集団内に広がるだけで，進化を進めることは不可能である．そこで，生物の有性生殖を参考にして，染色体の一部の組み替えを行う．この操作のことを，**交叉 (crossover)** と呼ぶ．交叉は常に行われるわけではなく，**交叉率 (crossover rate)** で定められた確率によって行われる．交叉においては，(1) どのように 2 個体を選択するか，(2) どのように交叉するか，が重要となる．(1) については，ランダムに 2 個体を選び出す方法が用いられることが多い．また，(2) については，現在までに次のようなバリエーションが提案されている．

1. **一点交叉 (one-point crossover, 1X)**
2. **多点交叉 (n-point crossover, nX)**
3. **一様交叉 (uniform crossover, UX)**

それぞれについて，簡単に説明する．

一点交叉

2 個体を選び，GTYPE 上のある点 (**交叉点**) を境に，両者の情報の入れ替えを行う．

多点交叉

交叉点が n 点あるもので，$n = 1$ の場合が一点交叉に相当する．この方法では，交叉点の間で交互に片方の親から情報を受け継ぐ．

一様交叉

任意個の交叉点を取れるような交叉法で，0, 1 からなるビット列のマスクを用いて実現する．まず，ランダムに 0, 1 の文字列を発生させる．子 a は対応するマスクが 0 のときに親 A から受け継ぎ，マスクが 1 のときには親 B から受け継ぐ．子 b についてはその逆となる．

3 つの手法の適用例を以下で示す．ここでは各個体の染色体を，2 進数 9 ビットとしている．

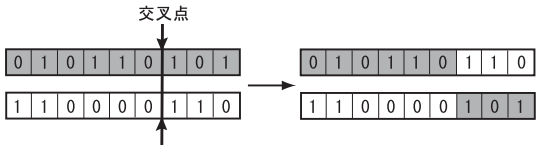


Fig. 2　一点交叉の例

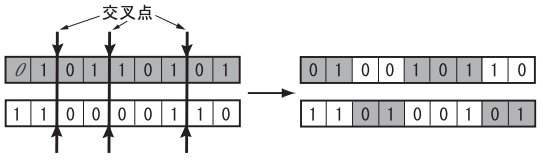


Fig. 3　多点交叉 ($n = 3$) の例

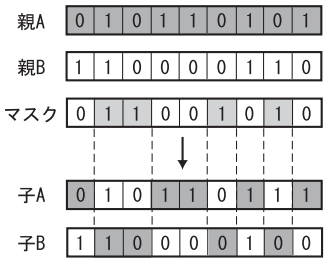


Fig. 4　一様交叉の例

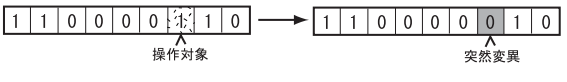


Fig. 5　突然変異の例

3.3.2 突然変異オペレータ

多くの場合，交叉だけでは十分な進化が得られない．それは，交叉により生ずる個体が必ず親の個体の一部のコピーであるからである．そのため，一般的にはあまり望ましくない解 (局所解など) に収束することが多くなる．

このような事態を避けるために，世代交代の過程に置いて，母集団内に新しい遺伝子を持つ個体を発生させる必要がある．これを自然界にならい，**突然変異 (mutation)** と呼ぶ．突然変異では，染色体上のある遺伝子を他の対立遺伝子に置き換えることによって，交叉だけでは得られない染色体を得ることができる．交叉同様，突然変異も常に起こるわけではなく，**突然変異率 (mutation rate)** で定められた確率で発生する．一般に，突然変異率は染色体長分の 1 をとる．

4 問題への GA の適用

これまで説明してきた GA の手法を実際に適用する．組み合わせ問題，関数最適化問題の 2 つについて述べる．

4.1 組み合わせ問題への GA の適用

ここで，組み合わせ問題として巡回セールスマン問題 (Travelling Salesman Problem, TSP) を取り上げ，GA で解く方法について述べる．ここでは，解法の流れのみを説明する．

まず，TSP に対する GTYPE と PTYPE を設計する．巡回路をそのまま GTYPE として定義すると，交叉によって巡回路以外のものが生じてしまう．例えば，abcde という 5 つの都市の巡回を考えることにする．これを順に 12345 と番号付け，次のような二つの巡回路をとるとし，2 番目と 3 番目の間で交叉が生じたとする．

親 A	13|542	a → c → e → d → b
親 B	12|354	a → b → c → e → d
		⇓
子 A	12542	a → b → e → d → b
子 B	13354	a → c → c → e → d

この交叉では，子 A と子 B ともに同じ都市を回る巡回のために，TSP の解とはなり得ない．このような GTYPE を**致死遺伝子**と呼ぶ．効果的な探索のためには致死遺伝子の発生を押さえる必要がある．

そこで以下のような方法を考える．巡回すべき都市 abcde を 12345 まで順序づける．ただしこれは相対的な順番で，PTYPE が acedb という経路は GTYPE では次のように構成することにする．

Table 2　GTYPE の設定

都市	順序	遺伝子コード	説明
a	abcde 12345	1	都市 a は順序の中の 1 番であるので，遺伝子コードを 1 とする．
c	bcde 1234	2	順序は選ばれた a を消す．都市 c はその中で 2 番目なのでコードを 2 とする．
e	bde 123	3	同様にして順序を作る．都市 e は 3 番目なので，コードを 3 にとる．
d	bd 12	2	以上と同様に進めると遺伝子コード =2 となる．
b	b 1	1	最後は候補が 1 つしかないので，1 となる．

Table 2 の表現を用いれば，通常の交叉によって得られた GTYPE が致死遺伝子とはならず，先ほどの交叉の例の GTYPE も，以下のように都市の巡回を表現するようになっている．

親 A	12|321	a → c → e → d → b
親 B	11|121	a → b → c → e → d
		⇓
子 A	12121	a → c → b → e → b
子 B	11321	a → b → e → d → c

GA を用いて TSP を解く場合の適合度は次のように定義される．

$$Fitness(\text{PTYPE}) = \frac{Minimum}{Length(\text{PTYPE})}$$

ここで，$Length$(PTYPE) は PTYPE の閉路長，$Minimum$ は最短閉路長である．したがって，$Fitness$ が 1.0 となる PTYPE が最短経路である．ただし，この場合あらかじめ最短経路長が既知であることが必要である．

GA オペレータには，突然変異と交叉を用いる．いま，GTYPE はビット表現ではなく，一般の文字列表現であるために，突然変異では変異する遺伝子を適切な文字集合から選択する必要がある．例えば，親 A の GTYPE12321 で 1 番目の遺伝子「1」が突然変異する場合，変異後の文字の候補は {2,3,4,5} となる．2 番目の遺伝子「2」ならば {1,3,4} が候補となる．

4.2 関数最適化問題への GA の適用 [2)]

ここでは，次の 1 変数関数を用意した．

$$f(x) = x \cdot \sin(10\pi \cdot x) + 2.0$$

この関数の定義域を閉空間 $[-1, 2]$ とし，その中で関数値が最大になるような x を求めるという関数の最適化問題について考えてみる．まず関数の図を Fig. 6 に示しておく． 図からわかるように，この関数は局所的最大値と局所的最

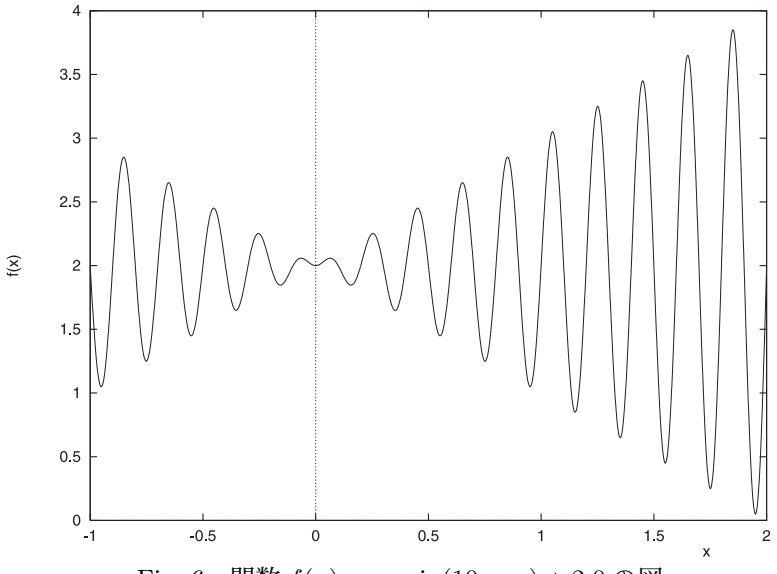


Fig. 6 関数 $f(x) = x \cdot \sin(10\pi \cdot x) + 2.0$ の図

小値が繰り返されている．この関数は，$x \cong 1.850542$ のとき，最大値 $f(x) \cong 3.850274$ をとる．この問題に対して，GA をどのように適用するかを説明する．

4.2.1 コード化

解の候補である個体を GA で扱う染色体とするため，個体を 0,1 の 2 進文字列で表現 する．変数 x を小数点以下第 6 桁までで近似するとする．変数 x の定義域の長さは 3 であるので，閉区間 [-1,2] を 3×10^6 個に分割しなければならない．

まず，染色体として何ビット必要かを考える．ここで，$2^{21} = 2097152, 2^{22} = 4194304$ なので，

$$2^{21} < 3 \times 10^6 < 2^{22}$$

となる．したがって，22 ビット必要である．

次に，2 進文字列から実数値へのデコード化を考える．これは，次の 2 つの手順により容易に行える．例として，文字列 $s_1 =< 1000101110110101000111 >$ について考える．デコードには，(1)10 進への変換と，(2) 対応する実数値の計算の 2 段階が必要となる．

(1)2 進数から 10 進数への変換

$$x' = (1000101110110101000111)_2 = 2288967$$

実数値のコード化[3)]

染色体を実数値にコード化する方法としては，今回の例で用いている，2 進数の**バイナリコード化 (binary coding)** が最も理解しやすい．しかし，バイナリコードでは，隣り合う値に対応するコードのハミング距離が一定でないため，最適解に収束しにくいという問題が現れる．

実数値をコーディングする方法において，バイナリコード化のほかに**グレイコード化 (Gray-coding)** というものがある．グレイコードとは，隣り合う符号間のハミング距離が常に 1 になるようにコード化する方法である．次の表に 10 進数の 0〜7 の値について，バイナリとグレイコードの場合の対応を示す．

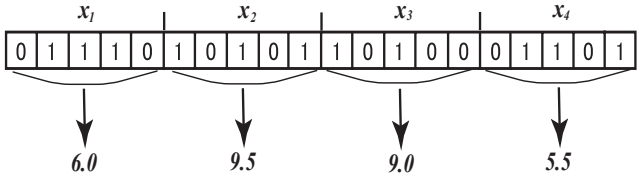
10 進数	binary code	距離	Gray code	距離
0	000	-	000	-
1	001	1	001	1
2	010	2	011	1
3	011	1	010	1
4	100	3	110	1
5	101	1	111	1
6	110	2	101	1
7	111	1	100	1

(2) 対応する実数値の計算

$$x = -1.0 + \frac{x'}{2^{22}-1} \cdot 3 = -1.0 + \frac{2288967}{4194303} = 0.63719$$

設計変数が 2 つ以上の場合のコーディング

ここで取り扱っている例では，設計変数が 1 つの場合のコード化について述べている．設計変数が 2 つ以上の場合には，それぞれの設計変数ごとのコードを連続して並べることで対処する．具体的には，次のような形になる．



この例では，設計変数が 4 つである問題に対して各設計変数ごとに 5 ビットを割り当て，20 ビットで表現している．デコードは図のように，それぞれの設計変数に対応するビット列ごとに行う．設計変数が 1 つの場合と同様，この結果求まった実数値を適合度の計算に用いることになる．

4.2.2 初期個体群の生成

初期の個体群は，22 ビットの 2 進文字列で表される個体を，各文字列に対して 22 ビットすべてを ランダムに初期化する ことにより生成すればよい．

4.2.3 適合度関数

最大化する関数 $f(x)$ をそのまま適合度関数 $f(s)$ と考えればよい．例として，さきほどの文字列 s_1 について考える．これは，$x = 0.637197$ に対応しているので，適合度の値は，$f(s_1) = f(x) = 2.586345$ となる．

4.2.4 交叉オペレータ

ここでは，交叉として 1 点交叉を考える．例えば，2 つの文字列

$s_2 =< 00000|01110000000010000 >$
$s_3 =< 11100|00000111111000101 >$

に対して交叉を行うとする．これらの適合度はそれぞれ，

$$f(s_2) = 1.078878$$
$$f(s_3) = 3.250650$$

である．ランダムに選ばれた交叉点が 5 番目と 6 番目の遺伝子の間であったとする．このとき，交叉によって

$$s'_2 =< 00000|00000111111000101 >$$
$$s'_3 =< 11100|01100000000010000 >$$

となる子が生成される．適合度はそれぞれ，

$$f(s'_2) = 1.940865$$
$$f(s'_3) = 3.459245$$

となり，子 s'_3 は両方の親よりも適合度が高くなっている．

4.2.5 突然変異オペレータ

上記の例で，突然変異が s_3 の文字列の 5 番目の遺伝子で起こるとする．すると，s_3 の 5 番目の遺伝子が 0 から 1 に反転されるため，

$$s_3 =< 11100000000111111000101 >$$
$$s''_3 =< 11101000000111111000101 >$$

変換後は s''_3 となる．適合度は，

$$f(s''_3) = 0.917743$$

となり，突然変異によって s_3 の文字列の適合度は著しく減少している．一方，文字列 s_3 で 10 番目の遺伝子で突然変異が起こったとすると，ビット列と適合度は次のようになる．

$$s'''_3 =< 11100000011111111000101 > \qquad f(s'''_3) = 3.343555$$

となり，s_3 よりも適合度が高くなっている．

4.2.6 シミュレーション結果

個体群サイズ 50，交叉率 0.25，突然変異率 0.01 と設定して，SGA(Simple GA: 単純 GA) を適用すれば，131 世代で準最適な文字列が得られた．

$$s_{max} =< 11110011010000010001010 >$$
$$x_{max} = 1.850685, f(x_{max}) = 3.850258$$

文字列に対応する値，関数値から，準最適解が得られていることがわかる．

5 遺伝的交叉を用いた並列シミュレーテッド アニーリング

この手法は，本研究室の小掠真貴さんが考案したものである．小掠さんの研究は，タンパク質の構造同定を対象としている．タンパク質の立体構造はエネルギーの最小状態に対応しており，エネルギーを最小とするような構造を最適化手法を用いて求めることで，構造同定が可能である．

しかし，タンパク質のエネルギー関数は大局的にも，局所的にも極小値を持っている．そこで本手法では局所的探索が得意な SA と，大局的探索が得意でかつ部分解の組み合わせで最適解が得られる問題に有効である GA のオペレータを取り入れ，タンパク質の構造同定に有効な手法を提案している．

5.1 遺伝的交叉を用いた並列 SA のアルゴリズム

この手法では，図 7 に示すように並列に実行している各 SA の解の伝達時に，GA のオペレータである遺伝的交叉を用いたものである．GA のオペレータを用いた SA であるため，SA の探索点の総数 (SA の並列数) を個体数，GA の世代数のことをアニーリングステップ (計算繰り返し回数) と呼ぶ．

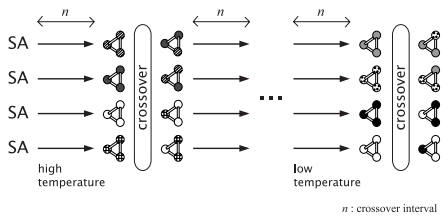


Fig. 7　遺伝的交叉を用いた SA

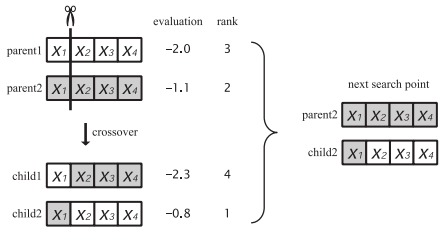


Fig. 8　遺伝交叉

このモデルでは，解の伝達時に並列に実行している SA からランダムに親として 2 個体を選択し，設計変数間交叉を行う．設計変数間交叉とは各設計変数の間でのみ交叉を行うことをいう．図 8 に示すように，もとの親と生成した子との 4 個体間のうち評価値の高い 2 個体を選択して，この 2 個体から次の探索を続けるという方法である．

ある設計変数の最適値が決まっている場合，遺伝的交叉によってその設計変数の最適値を他の SA 探索に伝達することができるため，アニーリングの収束を早めることができる．連続問題をあまり得意としない SA に GA オペレータを適用することにで，タンパク質の構造同定の問題に対して有効な手法となることが期待できる．

6　GA の研究応用例と実用例

6.1　研究応用例

- **配合飼料の最適設計**：実用の肉豚肥育用飼料の配合設計問題
- **分子系統樹作成への応用**：生物の系統分類に用いられる系統樹を, 種々の生物から得られた DNA 塩基配列またはアミノ酸配列などの配列データどうしを比較することにより作成する新しい手法として GA を応用
- **DNA 配列の機能部位のシグナルパターン抽出問題への適用**(隠れマルコフ＋ GA)：DNA-蛋白質間の相互作用に関する考察や遺伝子同定における機能部位の予測などに用いられる

6.2　実用例

- エアコン室温制御の最適設計
- バス仕様ダイヤ作成
- エレベータ群管理システム
- ボーリング位置配置 (地盤液状化などの問題に際して)

7　課題

配布するファイルの「function」の部分だけを書き換えてコーディングしてください．

参考文献

1) 伊庭斉志“ 遺伝的アルゴリズムの基礎 ”, オーム社, 1994
2) 金子美華，渡邉真也“ GA 基礎講座 (2) ”, 1998
3) 畠中一幸“ 卒業論文　遺伝的アルゴリズムの分散並列化 ”, 1998