ＧＡについて

立川 一芳*

平成11年10月21日

1　はじめに

コンピュータが我々の生活に幅広く浸透し、欠かせない存在になっている現状で「コンピュータは何でもできる」ような考えを持ってもおかしくない。しかし、高性能のコンピュータを利用したとしても

- データのサイズが非常に大きい問題
- 試行錯誤を繰り返す必要がある問題

などの中には時間的コストがかかり過ぎるため実際には解けないというものもある。逆に、その問題を解かせることにより、コンピュータの性能を競うことがある。そのような問題に対して、真の解である最適解（厳密解）を求めるのではなく、それを含む部分集合 (近似解) を求めることでコストを削減する[1]。いろいろな方法が研究されているが、ここでは遺伝という自然の仕組みをモデルにシステムを構築する遺伝的アルゴリズム (Genetic Algorithms、以下ＧＡ) を取り上げ、巡回セールスマン問題 (Traveling Salesman Problem、以下ＴＳＰ) を解く。

2　遺伝とＧＡ

- メンデルの法則について (優性の法則、分離の法則)
- エンドウの種子の形 (丸型としわ型)
- 優性と劣性、対立遺伝子
- 遺伝子型と表現型

*新発田南高等学校 豊浦分校
[1]近似アルゴリズムという

長い年月の中、生物が環境に適応する過程で様々な新しい種が誕生した。そこで先祖より優れた子孫が残り、遺伝によりその形質が伝えられていくことが広く知られている[2]。この遺伝の仕組みをアルゴリズムで定式化し、情報処理のシステムに応用しようとするのがＧＡである。

ＧＡは遺伝子の一群の組としての染色体 (遺伝子型：genotype、以下 g-type) とそれにより個体に現れている形質 (表現型：phenotype、以下 p-type) を仮定し、遺伝子が各個体の特徴を決定する関数のパラメータと考え、環境に応じて p-type から適応性 (適合度：fitness value、以下 fit) が決まる。次の表は生物とＧＡの用語の比較である。

生物	**ＧＡ**
遺伝子型 →ＡＡ、Ａａ、ａａ	g-type →000、001、010
表現型 →丸型、しわ型	p-type →０、１、２
適合度 →生き残りやすさ	適合度 →評価関数の値
遺伝子 →Ａ、ａ	遺伝子 →0、1

始めに、ランダムな遺伝子を持つ染色体の集団 (初期世代、世代０) を作り、その中の各個体が条件に対してどれだけ適応しているかを評価し、成績上位の個体を残す。次に残った個体で交配し、遺伝子の交叉、突然変異をある確率のもとで行い、新しい子孫 (世代１) を作って再び適合度を評価する。このサイクルを繰り返すことにより、世代の進行に従って集団全体の適合度が高まっていくことが期待される。これがＧＡの基本的な考え方である。

3　ＧＡの機能

ＧＡが適応される範囲として次の場合が考えられている。

1. 大量のデータを処理する
2. データから状態を表す数式を決定する
3. 試行錯誤しながら最適な条件を導く

[2]Darwin 「種の起源」(1859)

また、ＧＡのオペレータとして次の３つを挙げる。

- 突然変異 (mutation)
- 交叉[3](crossover)
- 逆位 (inversion)

これらは生物の遺伝とほぼ同じ意味を持つ。ここでは突然変異と交叉を主に用いてＧＡを実現する。次の関数[4]を用いて実験を行い、ＧＡの仕組みを見ることにする。

$$F = F(x_1, x_2, x_3) = -\sum_{i=1}^{3} x_i^2$$
$$-5.11 \leq x_i \leq 5.12, \quad \Delta x_i = 0.01$$

この関数Ｆにおいてx_iの定義域とΔx_iの値から

$$\frac{5.12 \times 2}{0.01} = 1,024 = 2^{10}$$

となり、g-typeは各x_iに対して10bitsが必要である。さらに関数Ｆは3つのパラメータを持つので、

$$(2^{10})^3 = 2^{30} = 10^{9.03\cdots}$$

約10^9が探索空間になる。関数Ｆの最適値を最大値０とおくと、適合度fitは１から０の間の値をとり、最適値が１で、０に近いほど評価が低くなるように次の式で正規化する。

$$\text{fit} = \frac{1}{1 - F}$$

資料１[5]は世代０と世代１の様子を示す。採用したＧＡパラメータは次の通りである。だたし、$\mathrm{P}_{\mathrm{cross}}$と$\mathrm{P}_{\mathrm{mut}}$はそれぞれ交叉率、突然変異率である。

集団の数	10
g-type長	30
$\mathrm{P}_{\mathrm{cross}}$	0.6
$\mathrm{P}_{\mathrm{mut}}$	0.033

g-typeはx_1, x_2, x_3の順にバイナリ表示したものであり、p-typeをその下に表示してある。例えば、世代０の１）の$x_1 = 1111100101$をp-typeに変換してみる。0000000000が-5.11であり、1bit増えるごとに$\Delta x_i = 0.01$だけp-typeが増えるようにする。x_1を10進数に直すと、

$$2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^2 + 2^0 = 997$$

[3]乗換えともいう
[4]DeJongにより導入された標準関数の１つ、ＧＡのbenchmark testや他の手法との比較に使われる
[5]参考文献１）より抜粋

となり、x_1のp-typeは

$$-5.11 + 997 \times \Delta x_i = 4.86$$

になる。また、Fの計算は

$$\begin{aligned} F &= F(4.87, 4.33, 3.93) \\ &= -(4.87^2 + 4, 33^2 + 3, 93^2) \\ &= -57.9107 \end{aligned}$$

となり、fitは

$$\text{fit} = \frac{1}{1 - (-57.9107)} = 0.016974844\cdots$$

で約0.017となる。

世代１の子は、世代０の親の交叉と突然変異から生じている。parentsとxsiteは交叉の時の親の番号と交叉位置を示す。例えば、世代１の子２）は世代０の１）と５）を親に持ち、４の位置（左から５番目）の右側で交叉が起きている。また、20の位置（左から21番目）で突然変異が起きて、０→１になっている。

交叉は一定の確率$\mathrm{P}_{\mathrm{cross}} = 0.6$で起こる。ここでは２回の交叉(ncross=2)が起きている。世代１の４）～９）までの子は交叉が起こらず、親の遺伝子コードをそのまま受け継いでいる。また、突然変異はg-typeの各bitsに対して$\mathrm{P}_{\mathrm{mut}} = 0.0333$の確率で起こり、ここでは14回の変異[6](nmutation=14)があった。

世代０と世代１の適合度fitの平均(avg)、適合度の最大値(maxn)、最小値(minn)を次に示す。

	avg	maxn	minn
世代０	0.0566	0.1493	0.0170
世代１	0.0738	0.1858	0.0170

4　ＴＳＰについて

問題によっては、それを解く効率のよいアルゴリズムがわからないため、非常に多くの解の候補を一つ一つ調べなければならないものがある。

問題の大きさNに対して、計算時間が2^Nに比例するアルゴリズムを考える。$N = 10$の問題を1秒で解けるとすると、$N = 20$では、1,024秒=約17分もかかる。$N = 100$ともなれば、現実的な計算時間と費用で解くことはできない。このようなタイプの問題で有名なＴＳＰ(巡回セールスマン問題)がある。この問題は

[6]∧印をつけたbits

会社を出発したセールスマンが、N個の都市を丁度１回ずつ訪問して
会社に戻るとき、移動コストの総和が最小になるような経路を求める

というものである[7]。この問題の難しさは多くの経路の長さを全部調べなくてはならないことに原因がある。少し抵抗して適切でない経路をできるだけ早く発見し、枝刈りを行って探索空間を小さくしても本質的な解決にはならない。それだけ計算量が指数的に増えることは困難さを伴うのである。

5　ＴＳＰを解く

ＴＳＰの例として$N = 5$の場合を挙げる。５つの都市に０～４の番号をつけ、始点と終点を都市０に固定する。各都市間の移動コストを次の隣接行列で表現する[8]。

$$\begin{pmatrix} 0 & 3 & 0 & 6 & 2 \\ 3 & 0 & 7 & 0 & 3 \\ 0 & 7 & 0 & 4 & 1 \\ 6 & 0 & 4 & 0 & 5 \\ 2 & 3 & 1 & 5 & 0 \end{pmatrix}$$

例えば都市２から都市３への移動コストは４である[9]。全都市を訪問する経路を

$$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 0$$

のように表す。単純に１～４の数字をランダムに生成させると、全部で$4^4 = 256$パターンある。しかし、ほとんどが全都市を訪問できない経路である。実際に全都市が登場する経路は次に示す$4! = 24$パターン[10]である。この24パターンの経路を解の候補(探索空間)とし、それぞれの経路がもつ移動コストが評価の対象になる。ただし、この中には次のように全都市を訪問できない経路(移動コストが×印)が含まれている。

- 終点(都市０)に戻らない場合：４番目に都市２を訪問する
- 接続していない都市へ移動する場合：０→２、３→１など

このような経路に対し、移動コストに大きな数値(PENALTY)を加え評価を低くする。この問題の厳密解は１７で経路は

$$0 \rightarrow 1 \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 0$$

である。

[7]類似問題としてハミルトン閉路問題、中国人郵便配達問題がある
[8]必要であれば往路、復路の移動コストを別々に設定できる
[9]接続していない都市間の移動コストは０
[10]対称な経路を含む

経路	移動コスト
1→2→3→4	2 1
1→2→4→3	2 2
1→3→2→4	×
1→3→4→2	×
1→4→2→3	1 7
1→4→3→2	×
2→1→3→4	×
2→1→4→3	×
2→3→1→4	×
2→3→4→1	×
2→4→1→3	×
2→4→3→1	×
3→1→2→4	×
3→1→4→2	×
3→2→1→4	2 2
3→2→4→1	1 7
3→4→1→2	×
3→4→2→1	2 2
4→1→2→3	2 2
4→1→3→2	×
4→2→1→3	×
4→2→3→1	×
4→3→1→2	×
4→3→2→1	2 1

6　ＧＡを利用したプログラム

ＴＳＰを解くＧＡのプログラム[11]の outline を次に示す。
○プログラム中の主な変数

TOWN_COUNT	：都市の数
POPULATION_COUNT	：集団を形成する個体数
CUT_LINE	：足切りライン (これ以下は淘汰の対象)
MUTATION_ODDS	：突然変異率
STOP_COST	：しきい値 (これより小さい解が得られたら終了)
MAX_COST	：各都市間の移動コストの上限
PENALTY	：都市が接続されていない場合、通過しない都市がある場合、移動コストに加える大きな値
map	：各都市間の移動コスト一覧 (TOWN_COUNT × TOWN_COUNT の行列)
population	：個体データ (経路 pathway と移動コストの総和 cost からなる)

$N = 5$ の例の厳密解の経路と移動コストは次のようになる。

pathway= {１、４、３、０、２} [12]，cost=17

○プログラムの流れ (各関数の働き)

1) ga_init()	・各個体を格納するメモリを確保する ・各個体の pathway にランダムに生成した０～１４までのデータを入れる
2) map を表示	
3) ga_ranking()	・cost により個体データを順位付けする
・3)-1 calculate_cost()	・各個体の pathway より cost を計算する 行き先の都市が接続されていない場合や通過しない都市がある場合は cost に PENALTY を加える 繋がっていない都市が j 個ある場合は PENALTY × j を cost に加える
・3)-2 個体を sort	・cost の値で sort する (cost 小が上位)
4) 最良の cost を表示	・その世代の最良の個体の cost を表示する cost<STOP_COST ならば 6) へ進む cost>=STOP_COST ならば 5) へ進む

[11]参考文献３）
[12]０→１、１→４、２→３、３→０、４→２のようにみる

5) ga_next_genelation()　・次世代を生成し、3) へ戻る
・5)-1 crossover()　・順位が CUT_LINE 以下の個体は淘汰され
残った上位の個体を交叉してできる個体に
置き換えられる
・5)-2 mutation()　・全個体に対して MUTATION_ODDS の確率で
突然変異を起こす
6) 近似解と時間を表示　経路と計算時間を表示し、終了する

このプログラムにおいて、都市数 (TOWN_COUNT) を 10、15、20 としたときの結果を次に示す[13]。上段が移動コストで下段が計算時間 (秒) である。
注意：個体の数、map の状態、STOP_COST、乱数種などにより結果が異なる。

都市数	ＧＡ	厳密解
１０	１０２ (100)	１０２ (0)
１５	９０ (159)	３０ (1)
２０	１７１ (279)	５８ (12499)

この結果からも訪問する都市数が増えると厳密解を求めることは実用的な意味で困難である。そこでＧＡの手法が有効になってくる。

7　ＧＡの課題

他の最適値探索 (総当り探索、ランダム探索、山登り法) の手法を比較すると、次のようになる。

総当り探索　：探索空間が大きくなると計算時間が爆発する
ランダム　：しきい値の設定で計算量が増えることがある
山登り法　：局所最適解に陥る可能性がある

ＧＡは総当り探索が難しい問題に対し、交叉を行って適合度の高い領域を探索し、突然変異で局所解に陥らないように一様な探索を実行する。これはランダム探索と山登り法の良い点を取り入れているといえる。
しかし、ＧＡも弱点を持っている。一般的にどのような戦略が最適解に行きやすいかわかっていない。そのため、

[13]使用機種：NEC PC-9821(i486DX2/50MHz)

- 適合度のスケーリング(探索効率の劣化を防ぐ)
- ＧＡオペレータの組合せ(問題に応じて使い分ける)

などの工夫も研究されているが、本質的な解決にはいたらない。一番の問題はＧＡパラメータの設定である。対象に対して適当でない設定は探索が収束しないなどの悪影響を呼ぶ。これについてはＧＡパラメータを探索する「メタＧＡ」というアイディアがあり、元のＧＡの最適値探索の結果がメタＧＡの適合度になる。

「ＧＡは本当に有効なのか」という疑問に対し、理論的な根拠を明示することは難しい。今後、この課題に関する研究が進んで、わかりやすい形で我々に示されることを期待する。

8　おわりに ～ｍｏｒｅ ＧＡ！～

ここでは紹介できなかったが、ＴＳＰの他に

- Ｎクイーン問題
- 囚人のジレンマ
- 充足可能性問題

などの興味深い例がある[14]。また、ＧＡの手法をプログラミングに用いた遺伝的プログラミング[15]もＡＩ[16]の問題として盛んに研究されている。まだまだＧＡの世界には新しい可能性が期待できそうである。

ＧＡを知ったのは書店で偶然見た雑誌からだった。最初は半信半疑で選んだ題材であったが読み進むうちにすっかり夢中になっていた。今後はＡＩやＡＬ[17]への応用であることを意識しながら取り組みを続けていきたい。

○引用・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）服部桂 「人工生命の世界」 オーム社 1994
３）長久勝 「最適解を模索する遺伝的アルゴリズム」 C MAGAZINE 1999 9月号

[14]参考文献１）
[15]ＧＰ：Genetic Programming John Kozaが提唱
[16]Artificial Intelligence：人工知能
[17]Artificial Life：人工生命