

遺伝的アルゴリズムと機械の進化

和田 健之介

1 遺伝的アルゴリズムとはなにか？

これから紹介する“遺伝的アルゴリズム (genetic algorithms, GA)”はシステムの最適化手法の一つです。しかし、従来の最急降下法や準ニュートン法などの最適化アルゴリズムと多くの点で異なります。簡単にその手続きを述べると、

- システムを文字列にコーディングする。
- システム（文字列）の集団を扱う。つまり、1点探索ではなく、多点探索を行なう。
- 文字列をデコードし、実現されたシステムの挙動を評価する。
- 淘汰により、次の世代の親となる集団を選ぶ。
- 選ばれた集団から、オペレータ (genetic operator) を施すことにより新しいシステム（文字列）を生成する。
- このサイクルを繰り返すことによりシステム集団のパフォーマンスを向上させる。

このように言わば“プログラム・ミューテーション”とでも言った手法を取ります。これらの具体的なオペレーションについては、後にいくつかの例を通じて説明します。

人工的なシステムの進化の研究の歴史は意外と古く、L. Fogel, A. Owen, M. Walsh^[1]は、オートマトンの集団をさまざまな突然変異によって進化させる実験を行なっています。この他に F. Warburton の最適適応形質の進化のモデル^[2]などがあります。また最近、捕食行動を遺伝子によってプログラムされた虫の集団がバクテリアを食べながら這い回り、これらが淘汰 (selection) によって進化していくプログラムが紹介されました^[3]。このプログラムはグラフィックスルーチンも含めて 300 行ほどの小さなプログラムなのです

が、しばらくみてみると、単純なコーディングにもかかわらず、虫の行動パターンがかなり変わり、突然変異と淘汰による進化過程を観察することができます。さらに染色体（設計図）の交叉（文字別の組み換え）を取り入れたシステムも考えることができます。これにより、部分的に実現された機能をさらに統合して、より高いパフォーマンスを獲得することができます。また、多峰性の問題の場合、交叉によって、それぞれの低い山に登りつめた親どうしからもっと高い山に登ることのできる遺伝子空間に配置された子供が生まれる可能性も出てきます。このモデルとしては、J. Reed らの有性生殖のモデル^[4]などがあります。J. Holland は、これらの概念を形式化し、遺伝子のコーディング方法、各オペレーションの定義、“Schemata”定理、“building block”仮説などを体系化しました^[5]。これにより多くの研究者がさまざまな分野に GA の応用を試みています^[6]。ではいくつかの例を通じて、具体的な適用方法を説明していきましょう。

2 単純 GA の例題

まず手始めに、図 1 のような多峰性の関数 $f(x)$ の最大値探索を考えてみます。例えば、このときの各遺伝子のコーディング（遺伝子型, genotype）を、固定長（注：単純 GA では異種交雑を考えません）の 2 進数で表現してみます（この場合は、各桁の遺伝子座 (locus) に対立遺伝子 0 または 1 が存在します）。このような表現をとると、上位桁で大まかな位置を決め、下位桁で細かい位置の微調整を行なうことになります。1 番目の個体は $X_1 = 11010010$ 、2 番目の個体は $X_2 = 00111011$ 、…といった具合に最初はランダムに N 個体生成します（注：一般的には $f(x)$ を $[0, 1]$ 区間

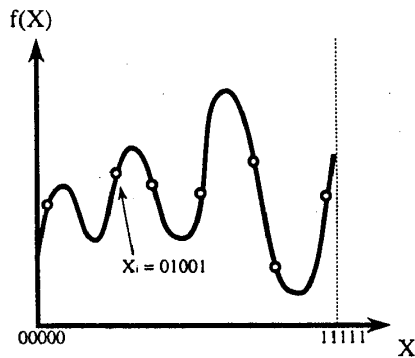


図 1

に正規化し, X を 2 進小数で表現します). この遺伝子によって実現された個体の特性 (表現型, phenotype) の評価 (pay off) は関数 $f(x)$ (objective function) によって与えられるものとします.

次に各個体の適応度 (fitness) を計算します. つまり各個体が次の世代にどれくらいの子孫を残せるかを考えます. 例えば, 集団の数 N を一定に保つ場合は, i 番目の個体が次世代に残す子孫の数の期待値を,

$$N \cdot f(X_i) / \sum f(X_i)$$

とします. すると評価の高い個体は多く子孫を残しやすく, 逆に評価の低い個体は集団から取り除かれやすくなります (注: 確率的に生成するため, 最も評価の低い個体の子孫を残すこともあります. これにより集団が収束進化して局所解に落ち込むことを防ぐ役目をすることがあります).

これだけでは, 最初に生成した個体 (2 進数の文字列) 群のなかで最も大きな関数値を持つものが増えていくだけで, 違った特性を持った個体が生まれてきません. そこで, これらの選ばれた個体の中からあらかじめ決められた交配率で交配を行ないます (注: 評価の高いものほど交配率を高くするなどいろいろなバリエーションがあります). このとき, それぞれの配偶子が合体 (2 個の配偶子が接着して 1 個の接合子を生じる) もしくは接合 (2 個体が接着したあと, 分離して 2 個体となる) します. このとき 2 本の染色体 (各機能を持った遺伝子の連鎖群. 複雑な多機能システムの構築には多数の遺伝子が必要です) の間である確率 (交叉価) で交叉 (crossing-over) が生じます. 生物の世界では不等交叉や, 交叉の生じる場所 (キアズマ) と遺伝子組成に関係があったりしますが, 人工システムでは簡単のため, あらかじめきまった一定確率で交叉を生じて, 遺伝子間の部分交換が行なわれるものとします. 例えば次に示すように i 番目の個体と j 番目の

個体が交配して, 3 桁目と 4 桁目の間に交叉が生じたとすると X_i' と X_j' の遺伝子を持つ子孫が生まれます (: が交叉の位置を示します).

$$X_i = 11011 : 001 \quad X_i' = 11011 : 101$$

$$X_j = 01100 : 101 \quad X_j' = 01100 : 001$$

この後, 遺伝子に突然変異 (mutation) が生じます. 生物では突然変異の種類も多く, トランジション, トランスバージョン, フレームシフト変異, 欠失突然変異, 重複突然変異, 転座突然変異などがあり, また突然変異率も一定ではなくホットスポットがあることが知られています. 単純 GA では, あらかじめ決められた突然変異率に従った点突然変異のみを考えます. また突然変異は機能単位である遺伝子 (文字列) ではなく, 遺伝子を構成する塩基 (文字) に作用します. しかし, 単純 GA では形式的に文字の置換操作と考えて, 例えば k 番目の個体の 5 桁目の文字に変異が生じたとすると, 次のように変化します.

$$X_k = 01001101 \quad X_k' = 01011101$$

このように単純 GA では, 生殖 (reproduction) 過程が, 増殖 (multiplication), 交叉 (crossing-over), 突然変異 (mutation), 淘汰 (selection) によって構成されています.

このサイクルを繰り返すことによって進化していきます. といっても, 交叉や点突然変異などの一見乱暴な操作を行なうことが, システムの適応度を増すことになるのでしょうか? では, 単純 GA の進化のメカニズムをもう少し詳しく調べてみましょう.

3 単純 GA の基本定理

J. Holland は, 遺伝子の構成要素がどのように変化していくか観察するために, “スキーマタ (schemata)” という文字列の部分集合を考えました. これはアルファベットの集合にドント・ケア記号 * (任意のアルファベット) を加えた集合からなる文字列で, 例えば {0, 1} アルファベット上のスキーマタ $S = 01 * 110 * 1$ は, * に 0 または 1 を代入した次の 4 つの文字列の集合を指定します.

$$S = 01 * 110 * 1$$

$$\rightarrow \{01011001, 01011011, 01111001, 01111011\}$$

また, スキーマタ S の次数 (order) $O(S)$ を,

$$(\text{全長 } L) - (* \text{ の数})$$

で定義します. 上記のスキーマタの次数は,

$$O(01*110*1) = 8 - 2 = 6$$

です。次に、スキーマ S の構成長 (defining length) $\delta(S)$ を最も左の 0 または 1 と最も右の 0 または 1 の間の距離で定義します。例えば、

$$\delta(*110*10*) = 7 - 2 = 5,$$

$$\delta(* * 01 * 1 * *) = 6 - 3 = 3.$$

時刻 t において、スキーマ S に属する遺伝子を持つ個体数の期待値を $P(S, t)$ とします。交叉も突然変異もなく、総個体数 N が保存されるときには、スキーマ S の平均増加率は $N \cdot f(S) / \sum f(X_i)$ となります。ここで $f(S)$ は、スキーマ S に属する遺伝子を持つ個体の評価値の平均を表します。例えば、上記の例では、次のようになります。

$$f(s) = f(01*110*1)$$

$$= \{f(01011001) + f(01011011)$$

$$+ f(01111001) + f(01111011)\} / 4$$

交叉によってスキーマが分断される確率は $p_c \cdot \delta(S) / (L - 1)$ となります。ここで p_c は交配率を表します。突然変異率が p_m のとき、スキーマが突然変異を免れる確率は、 $(1 - p_m)^{O(S)}$ となります。交叉と突然変異が独立の確率過程なので次式が成り立ちます。

$$P(S, t+1) \geq P(S, t)$$

$$\cdot \frac{f(s)}{\bar{f}} \left[1 - p_c \frac{\delta(S)}{L-1} \right] (1 - p_m)^{O(S)}$$

$$\approx P(S, t) \cdot \frac{f(s)}{\bar{f}} \left[1 - p_c \frac{\delta(S)}{L-1} - O(S)p_m \right],$$

$$\bar{f} = \sum f(X_i) / N$$

ここで不等号になっているのは、交叉や突然変異によって生じる他のスキーマからの流入分などの高次の項を省略したためです。

この方程式は、

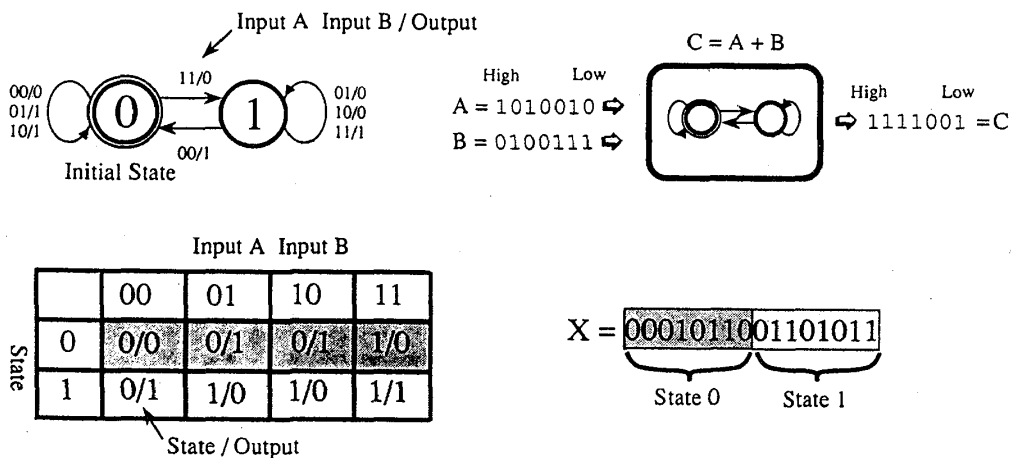


図 2

“構成長 $\delta(S)$ が短く、次数 $O(S)$ が低く、平均評価 $\bar{f}$ より高いパフォーマンスを示すスキーマ S が、次の世代で生き残る”

ということを示しています。これは“スキーマ定理 (Schemata Theorem)”または、“遺伝的アルゴリズムの基本定理”と呼ばれています^{[5][7]}。

生存に不可欠なハウス・キーピング遺伝子が分断されてしまうようなコード体系を持ったマシンは安定に生存できないわけですから、ランダム交叉ではなく、機能単位 (building block) で交叉するほうが適応度が上がります。単純 GA では最初のコーディング体系によって機能遺伝子の遺伝子座が固定されてしまうので、機能単位で交叉を生じるような進化は生じません。しかし、システムが冗長なコーディングによって記述されていて、最初に述べたような欠失突然変異、重複突然変異、転座突然変異などのオペレーションが導入されれば、機能をコードした領域と機能に関係ない領域を持つことにより、交叉に対して、よりロバストなシステムを構成することができます。

4 GA の応用

上記の例のように、単純でかつ、直接的な関数表現ができる場合は、かえって GA の有効性がわかりにくいかも知れません。

そこで次に、従来の解析的なシステムの最適化手法では、評価関数とシステムパラメータとの対応が困難な例として、記号処理を考えてみましょう。図 2 に、2 進数の加算を行なうオートマトンの状態遷移図と状態遷移表、およびそれを 1 本のテープにしたものを示しました。このようなオートマトンを生成する問題も上記の例題と同じように考えることができます。最初

にランダムに生成したオートマトンに、1桁の2進数の加算、2桁の2進数の加算、…の例題を与え、各桁ごとのヒット率によって評価を行ないます。

また環境から文字列を与え、次にくる文字を予測するオートマトン集団の進化は先に述べたL. Fogelら^[1]が詳細に調べています。彼らのシステムには交叉のオペレーションが入っていませんでしたが、A. Dewdney^[6]は突然変異と交叉を取り入れたオートマトン集団のプログラムを解説しています。このプログラムと先程紹介した、遺伝的に決められた虫の捕食行動の進化のプログラムを組み合わせ、周囲の環境（例えば虫の近傍に存在する餌の方向や仲間の方向など）や自分の状態（年齢、栄養状態、方向など）に応じて次の行動（捕食、増殖、休息、逃避など）を決めることのできる虫（オートマトン）の進化プログラムを作ることとはそれほど難しいことはありません。興味を持たれた方は是非プログラムされて御自分のマシンワールドの進化の有様を観察されてはいかがでしょうか？ここでは、紙数の関係で詳しく説明できませんが、J. Holland^[5]はプロダクション・システムのルール生成規則として遺伝的アルゴリズムを用い、各ルールの信頼値分配則としてバケツリレー・アルゴリズムを採用した分類システム(classifier system)を提案しています。このシステムのパスカル・プログラムと解説が文献^[7]に載っていますので、興味がおありの方はそちらを参照してください。

5 システム・コーディングの規範

もっと複雑な多機能システムを構築するときは、コーディングを塩基（文字）レベルで行ない、機能単位である遺伝子（単語）群からなる染色体（文章）を考えなければならないでしょう。このとき遺伝子間の相互作用(epistasisなど)も考慮する必要があるでしょう。このとき、システムをコーディングする際の、何らかの規範があるのでしょうか？例えば、有名なトラベリング・セールスマン問題(TSP)を考えてみましょう。“N個の都市をただ一度だけ訪れて、すべての都市を最短距離で回れ”という、最短一筆書きの問題です。ニューラル・ネットワークのリバイバルのきっかけを与え、エネルギー関数、シミュレーティッド・アニーリングなどいろいろな手法をみだしました。最急勾配法では局所的な解に落ち込んでしまうので、熱的揺らぎを与えて脱出確率を大きくし、徐々に温度

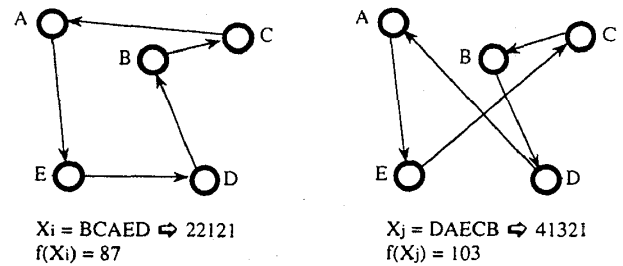


図3

を下げていくのですが、ある意味ではGAも定向性（淘汰と増殖）と揺らぎ（交叉と突然変異）を用いて、集団として並列に最適化処理（進化）を行なっているプロセスと考えられなくはありません。

では、具体的にA, B, C, D, Eの5つの都市を考えてTSPをGAでコーディングしてみましょう。最も簡単な方法は、訪れた順に都市のアルファベットを並べる方法です。例えば2つの個体がそれぞれ、図3のように巡回したとしましょう。

$$X_i = \text{BCAED}, f(X_i) = 87$$

$$X_j = \text{DAECB}, f(X_j) = 103$$

評価関数 $f(X)$ は遺伝子 X が表す道順の合計距離です。ところが、すぐわかるように、このようなコーディング体系では、交叉や突然変異を行なうと、ルールに違反した致死遺伝子ばかりができてしまいます。したがってすぐ絶滅してしまいます。これに対し、J. Grefenstetteら^[9]は実に巧妙なコーディング方法を考えました。最初に都市をアルファベット順に並べた標準リストをつくり、巡回路のリストのアルファベットがこの標準アルファベットのなかで左から数えて何番目にあるかを数え、その数字に置き換えます。そして、標準リストからそのアルファベットを取り除き、次の巡回都市についても同じことを繰り返します。例えば上記の例をこの方法でコーディングしなおすと、

$$X_i = \text{BCAED} \rightarrow X_i = 22121$$

$$X_j = \text{DAECB} \rightarrow X_j = 41321$$

この方法だと任意の位置で交叉を行なってもルールを満たした遺伝子が作られ、評価に従って子孫を残すことができます。例えば、3桁目と4桁目の間で交叉が生じたとすると以下のような遺伝子が生成されます。

$$X'_i = 22:321 \rightarrow X'_i = \text{BCEDA}$$

$$X'_j = 41:121 \rightarrow X'_j = \text{DABEC}$$

この他にも、I. OliverらのPMX交叉^[10]などのアルゴリズムがあります。また、J. Muhlenbeinら^[11]は、多

数のトランスピュータを連結した専用の並列マシンを使って TSP や学習進化の問題に GA を適用しています。

このようにコーディンの善し悪しによって GA の有効性が大きく左右されてしまいます。何か良い規範はないのでしょうか？ D. Goldberg は次のような提案をしています。

- 1) “できるだけ少ないアルファベットで構成すること (principle of minimal alphabets)”
- 2) “与えられた問題が、低い次数のスキーマに関係するようにコーディングすること”

彼はこれを “principle of meaningful building blocks” と呼んでいます。しかし、簡単な問題は Walsh 関数で解析できますが、実際の問題への適用は難しく、これは GA プログラマーの “art” に依存すると言っています。また、問題の並列性の見極めが重要なことは、他のアルゴリズムと変わりません。

6 GA の限界

システムに要求されるタスクが複雑になったときに、アルファベット数はいくつにするのか？ 各遺伝子の長さや遺伝子間の相互作用、階層構造をどのように決めるのか？ といった、さらに複雑な問題が生じます。このようにシステムのアーキテクチャや機能を直接コーディングする手法では、システムの仕様を決める段階で多くの難問が発生し、プログラマーの “art” に依存する部分が大きくなります。生物の遺伝子にコーディングされているのは、システム・アーキテクチャ（形やネットワーク構造）そのものではなく、システムを生成するためのルールのパラメータ（酵素の活性や細胞の接着性）です。これらが生成システムのなかで解釈されて、種に固有のルールが派生し、これがデコードされて初めてシステムのアーキテクチャが実現されます。この具現化されたシステムが、環境も含めた集団のなかで機能する過程で、システムの評価が行なわれます。このようなメタ・ルールをどのような枠組みで考えたらいいいのかについては、以前にも論じたことがありますので御参照ください^[12]。また、進化の原動力はなにか？ といった微妙な問題はあえてさげましたが、興味がおありの方は文献 [13] などを御覧下さい。

参考文献

- [1] L. J. Fogel, A. J. Owens & M. J. Walsh: Artificial Intelligence Through Simulated Evolution, John Wiley & Sons, (1967)
- [2] F. E. Warburton: “A Model of Natural Selection based on a Theory of Guessing Games”, *J. Theoret. Biol.* 16, 78-96 (1967)
- [3] A. K. デュードニー: “進化のシミュレーション: 虫がバクテリアを食べるようになるまで”, サイエンス, 1989-7
- [4] J. Reed, R. Toombs & N. A. Barricelli: “Simulation of Biological Evolution and Machine Learning”, *J. Theoret. Biol.* 17, 319-342 (1967)
- [5] J. H. Holland: Adaptation in Natural and Artificial Systems, Univ. Michigan Press (1975)
- [6] J. J. Grefenstette (Ed.): Proceedings of an International Conference on Genetic Algorithms and Their Applications, Hillsdale, NJ: Lawrence Erlbaum Associates (1985)
J. J. Grefenstette (Ed.): Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, Hillsdale, NJ: Lawrence Erlbaum Associates (1987)
L. Davis: Genetic Algorithms and Simulated Annealing, Pitman (1987)
- [7] D. E. Goldberg: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley (1989)
- [8] A. K. デュードニー: “fib がすむ原始のコンピューターの海で遺伝的アルゴリズムを探索する”, サイエンス, 1986-1
- [9] J. J. Grefenstette & R. Gopal: “Genetic Algorithms for the Traveling Salesman Problem”, Proceedings of an International Conference on Genetic Algorithms and Their Applications, 160-168
- [10] I. M. Oliver, D. J. Smith & J. R. C. Holland: “A Study for Permutation Crossover Operators on the Traveling Salesman Problem”, Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, 224-230
- [11] H. Muhlenbein & J. Kindermann: “The Dynamics of Evolution and Learning—Towards Genetic Neural Networks”, Connectionism in Perspectives, R. Pfeifer (Ed.), Elsevier (1990)
- [12] 和田健之介: “発生オートマトンとニューロ・コンピュータ”, 数理科学, 1988-6
- [13] R. Dawkins: The Blind Watchmaker, Longman Scientific & Technical (1986)
G. M. Edelman: Neural Darwinism, Basic Books (1987)
根井正利: 分子進化遺伝学, 培風館 (1990)
E-mail: wada @ sccs, chukyo-u. ac. jp
(わだ・けんすけ, 中京大学・情報科学部)