

9. モンテカルロ法による最適化 Optimization by Monte Carlo method

乱数を使った数値計算の方法は、全て「モンテカルロ法 Monte Carlo method」と呼ばれます。賭博場として有名なモナコ公国のモンテカルロ Monte Carlo 地区 ([Fig. 9.1](#)) にちなんでこのように呼ばれるようです。



Fig. 9.1 モンテカルロ・ルーレット ([補足 9.A](#))

モンテカルロ法は**最適化**以外の目的、例えば積分を計算したり方程式を解く目的で用いられる場合もあります。

モンテカルロ法を用いる計算手法には、以下の特徴があります。

- (1) プログラミング（コーディング）が容易である。
- (2) 計算の効率は良くなく、高い精度も期待できない場合が多い。
- (3) 繰り返しの回数を増やせば増やすほど正解に近づくことは期待できる。
- (4) 多次元（多変数函数^{かんすう}）の最適化に特に適している。
- (5) 並列化（分散）処理に適している。

モンテカルロ法には、計算資源を浪費しがちな傾向があることに注意は必要と思いますが、計算システムのコストパフォーマンスは向上し続けているので、将来さらに有望な方法となることは間違いありません。

この章ではモンテカルロ法を用いた大域的な最適化 global optimization に内容を限定します。モンテカルロ法の一種の「擬似焼鈍」^{しょうどん} simulated annealing と「遺伝的アルゴリズム」 genetic algorithm の人気は高く、実際によく使われるものです（[補足 9.B](#)）。この章では、モンテカルロ法を用いた最適化計算の考え方の基本的な骨格、擬似焼鈍、進化論的アルゴリズムについて紹介します。

9-1 ランダムサンプリングとインポートランスサンプリング Random sampling and importance sampling

モンテカルロ法による最適化の最も素朴な方法では、「ランダムなパラメータの組み合わせの生成」「結果の評価」を繰り返す手順を取ります。

ランダムと言っても、あらゆる「可能性」を同じ確率で選択する「**ランダム・サンプリング random sampling**」だけでなく、「重要そうなところ」を重点的に探索する「**インポートランス・サンプリング importance sampling**」と呼ばれる方針の取られる例が実際には目立つかもしれません。

インポートランス・サンプリングのために用いられる代表的な方法が**ランダム・ウォーク random walk** です。**マルコフ連鎖モンテカルロ Markov chain Monte Carlo (MCMC)** 法とよばれる場合もあります。「マルコフ連鎖」とは、とりうる状態が離散的である「離散状態マルコフ過程」と同じ意味で、「位置」が離散的に表現される場合も含むとされますが、実際には「位置」は連続的な値をとるものとして「時間（時刻）」が離散的な値をとる過程を指すことが多いようです。事実上「離散時間マルコフ過程」のことが「マルコフ連鎖」と呼ばれます。

「マルコフ過程」は、「未来の状態が現在の状態だけで決まり、過去の状態にはよらない」という性質を持つ確率過程とも言えます。マルコフ過程を進行させるために必要な記憶 (memory) 領域は、「その時点までに見つかった『最小であった値』と『最小であった時の位置を表すパラメータの組』、『現時点での位置パラメータ』の組」だけなので、ごくわずかです。

9-2 棄却サンプリング Rejection sampling

現在位置からランダムな方向に、ランダムな距離移動すること（ランダム・ステップ）を試み、「現在の値より小さい値になれば進み、そうでなければ進まない」という「選択」を繰り返すことで、最小位置付近の重点的な探索を実現することができます。この方法は**棄却サンプリング rejection sampling** と呼ばれます。棄却サンプリングでは、**偽最小 false minima** に陥るか、正しい最小を見つけられるかが、「初期位置」の影響を強く受けます。

9-3 メトロポリス・ヘイスティングスのアルゴリズム

Metropolis-Hastings method

過去にはメトロポリスのアルゴリズムと呼ばれていましたが、最近ではメトロポリス・ヘイスティングスのアルゴリズム Metropolis-Hastings (M-H) algorithmと呼ばれる例が多くなっています。

ある温度 T を仮定します。位置 (状態) を \mathbf{r} から $\mathbf{r} + \Delta\mathbf{r}$ にランダムに変化させたときにエネルギーが E から $E + \Delta E$ に変化したとします。

メトロポリス・ヘイスティングスの方法では、 $\Delta E \leq 0$ なら確率 1 で、 $0 < \Delta E$ なら確率 $\exp(-\Delta E/k_B T)$ で、変化後の位置 $\mathbf{r} + \Delta\mathbf{r}$ を採択します。ここで、 k_B はボルツマン定数を表します。この方法は「有限な温度での物質の挙動」を模擬するものと言える面があり、(偽最小ではなく) 正しい最小に到達する可能性が高くなると期待できる場合があります。

実際には以下のようなアルゴリズムが用いられます。

【メトロポリス・ヘイスティングスのアルゴリズム】

位置をランダムな方向、ランダムな距離に動かすこと (提案) を考え、新しい位置のポテンシャルと前の位置のポテンシャルの差 ΔE を計算する。

(1) $\Delta E \leq 0$ (エネルギーが下がる) なら、新しい提案を採択する。

(2) $0 < \Delta E$ の (エネルギーが上がる) 場合、0 から 1 の一様乱数 t を発生して

$\exp(-\Delta E/k_B T) < t$ なら新しい提案は棄却し、

$t \leq \exp(-\Delta E/k_B T)$ なら新しい提案を採択する。

ここで、このアルゴリズムでは「 $\Delta E \leq 0$ なら常に採択し、 $0 < \Delta E$ なら確率 $\exp(-\Delta E/k_B T)$ で採択する」ことになっていることを確認します。

(i) $0 < \Delta E$ のとき、 $0 < \exp(-\Delta E/k_B T) < 1$ の関係が成立します。

(ii) $0 \leq t \leq 1$ の一様乱数 t に対して、 $\exp(-\Delta E/k_B T) < t$ となる (棄却される) 確率は $1 - \exp(-\Delta E/k_B T)$ であり、 $t \leq \exp(-\Delta E/k_B T)$ となる (採択される) 確率は $\exp(-\Delta E/k_B T)$ で表されます。

メトロポリス・ヘイスティングスのアルゴリズムでの採択・棄却のルールを「新しい提案を採択する確率 (遷移確率) を $\min(1, e^{-\Delta E/k_B T})$ とする」という表現のしかたをする場合もあります。 $\min(a, b)$ は a と b の小さい方の値を返す函数です。この表現でも上記

(1) と (2) の場合の両方に対応できます。

「棄却サンプリング」では、 $0 < \Delta E$ の場合、必ず引き返す（棄却する）ので、これを「メトロポリス・ヘイスティングスのアルゴリズム」にあてはめると、「絶対零度 $T=0$ の場合」に対応します。

メトロポリス・ヘイスティングスの方法で有限の T を用いる場合、 $0 < \Delta E$ の場合であっても（少し悪くなっても）一定の割合でそのまま進みます。 T の値が大きいほど進む確率が高くなり、ランダム・サンプリングに近くなります。

メトロポリス・アルゴリズムで十分な試行回数を繰り返すと、エネルギーの分布（確率密度） $P(E)$ が「ボルツマン分布 Boltzmann distribution に比例する状態になる」とされています。つまり、

$$P(E) \propto \exp\left(-\frac{E}{k_B T}\right) \quad (9.3.1)$$

という数式で表されるような状態になり、これは「**詳細釣り合い (detailed balance)**」が成立するからと説明されます。

「詳細釣り合い」とは、「**順過程 forward process** と **逆過程 reverse process** が同じ頻度（確率）で起こること」を意味します。

エネルギー E の状態から $E' = E + \Delta E$ の状態への変化（順過程）の起こる確率 $P(E \rightarrow E + \Delta E)$ と、 $E' = E + \Delta E$ の状態から E の状態への変化（逆過程）の起こる確率 $P(E + \Delta E \rightarrow E)$ について考えます。

もし順過程 $E \rightarrow E + \Delta E$ の頻度が逆過程 $E + \Delta E \rightarrow E$ の頻度より高ければ、エネルギー E の状態をとる確率 $P(E)$ が低くなり、エネルギー $E + \Delta E$ の状態をとる確率 $P(E + \Delta E)$ が高くなります。逆に逆過程の頻度が順過程の頻度より高ければ、 $P(E)$ が高くなり $P(E + \Delta E)$ は低くなります。

はじめに、「エネルギー E をとる確率 $P(E)$ が不釣り合いに高い」状況であったとします。「 $P(E)$ が不釣り合いに高い」ことは、「順過程の頻度が逆過程の頻度より高い」と同じことです。このとき順過程の頻度が高いので $P(E)$ は低下し、 $P(E + \Delta E)$ は高くなります。 $P(E)$ が低下し、 $P(E + \Delta E)$ が高くなれば、順過程の頻度は低く、逆過程の頻度が高くなります。

はじめに「不釣り合い」な状態だったとしても、十分に時間をかけて「 $P(E)$ に比例した頻度の順過程」と「 $P(E + \Delta E)$ に比例した頻度の逆過程」をランダムに繰り返せば、最終的には順過程と逆過程の頻度が等しくなる「釣り合い」のとれた状態になると期待できることとなります。

メトロポリスのアルゴリズムでは、エネルギー E から $E + \Delta E$ への遷移確率 transition probability（エネルギー E の状態にあるときにエネルギー $E + \Delta E$ の状態に変化する確

率) $A(E \rightarrow E + \Delta E)$ と, エネルギー $E + \Delta E$ から E への遷移確率 $A(E + \Delta E \rightarrow E)$ がそれぞれ

$$A(E \rightarrow E + \Delta E) = \min(1, e^{-\Delta E/k_B T}) \quad (9.3.2)$$

$$A(E + \Delta E \rightarrow E) = \min(1, e^{\Delta E/k_B T}) \quad (9.3.3)$$

と表されます。このことから, エネルギー E をとる確率を $P(E)$ と表すことにすれば,

$$\begin{aligned} \frac{P(E + \Delta E \rightarrow E)}{P(E \rightarrow E + \Delta E)} &= \frac{P(E + \Delta E) A(E + \Delta E \rightarrow E)}{P(E) A(E \rightarrow E + \Delta E)} \\ &= \frac{P(E + \Delta E) \min(1, e^{\Delta E/k_B T})}{P(E) \min(1, e^{-\Delta E/k_B T})} = \frac{P(E + \Delta E)}{P(E)} e^{\Delta E/k_B T} \end{aligned} \quad (9.3.4)$$

の関係が成立します。

式 (9.3.4) の中の 3 番目の等式は ΔE の符号によらず成立します。「詳細釣り合い」が成立する場合,

$$\frac{P(E + \Delta E \rightarrow E)}{P(E \rightarrow E + \Delta E)} = 1 \quad (9.3.5)$$

なので, 式 (9.3.4) から

$$\frac{P(E + \Delta E)}{P(E)} = e^{-\Delta E/k_B T} \quad (9.3.6)$$

となります。

式 (9.3.1) の関係が成立するならば式 (9.3.6) の関係は成立します。また, 式 (9.3.6) の関係が成立するのは, 式 (9.3.1) が成立する場合には限られます (補足 9.3.A)。

9-4 擬似焼鈍 Simulated annealing

前節のメトロポリス・ヘイスティングスのアルゴリズムを用いれば, 「一定の温度 T でのボルツマン分布に従うような状態」を模擬できます。

高い温度設定から始めて, 徐々に温度を下げていくと, 「はじめは広い範囲・ランダムな配置をくまなく探索する状態であったのが, 温度を下げるにつれて, 最小位置付近を重点的に探索するような挙動になる」ことも期待できます。

この挙動は現実の金属材料の焼鈍 (やきなまし) のプロセスで, 「高い温度で保持することにより歪みを解消し, ゆっくり温度を下げることで自然に安定な原子配置になる」と似ています。そのようにする数値計算手法が「擬似焼鈍」「シミュレーテッド・アニーリング simulated annealing」と呼ばれます。略して SA 法と呼ばれる場合もあります。

「現実の『金属材料の焼き鈍し』と似ているので、うまく働くに違いない」と必要以上に思われがちな面のある印象を受けますが、確かに実際にうまく働くように見える場合も少なくありません。

9-5 進化論的アルゴリズム Evolutionary algorithm

9-5-1 遺伝的アルゴリズム Genetic algorithm

遺伝的アルゴリズム genetic algorithm は、**ダーウィン Darwin の進化論**をもとにした方法と言われ、特に**自然選択**（自然淘汰）**natural selection** と、**有性生殖** sexual reproduction で生殖細胞が減数分裂 (meiosis) をする際に起こる**染色体交叉**（**クロスオーバー**, chromosomal crossover）とを真似する性格を持つものです。また普通は**突然変異** mutation が取り入れられます。略して **GA 法**と呼ばれる場合もあります。

この方法は「現実の有性生殖による『生物の進化』と似ているので、うまく働くに違いない」と（前節の疑似焼鈍と比較してさらに）必要以上に思われがちな面が強いように思いますが、確かに実際にうまく働くように見える場合が多いのも事実です。

セントラル・ドグマ central dogma によれば、ヒトを含む DNA 型の生命体の遺伝情報は、デオキシリボ核酸 deoxyribonucleic acid; DNA 高分子の中の塩基（アデニン adenine, グアニン guanine, シトシン cytosine, チミン thymine）の配列として記録されています。これはコンピュータで「数値」あるいは「数値の組み合わせ」が、連続する“1”か“0”かの配列（ビット列）で表されることと似ています。遺伝的アルゴリズムと呼ばれる方法では「最適化すべきパラメータの組み合わせ」について遺伝子と同じような扱い方をします。

また、遺伝的アルゴリズムでは、異なる遺伝子を持った個体（異なるパラメータの組み合わせ）を複数保持し、**適応性** adaptivity あるいは**適応度** fitness の低い個体を**淘汰**（排除）するプロセス（自然選択 natural selection）を適用します（[補足 9.5.1.A](#)）。

また、生存した個体のうち一定の割合で対（ペア）を作り、有性生殖と同じように遺伝子を交叉させ、子を作らせます。このプロセスは**交叉** crossover と呼ばれます。交配 mating と呼ばれる場合もあります。どのようにペアを作るかには色々な流儀がありますが、例えば適応度ランキングの上位から順にペアを組ませる方法（エリート選択）があります。

また、それとは別に、一定の割合で、親から受け継いだ遺伝子と無関係に遺伝子の一部をランダムに変更する（**突然変異** mutation）を施します。突然変異の起こし方にもバリエーションがありますが、例えば複数のパラメータのうち、どれか一つのパラメータを「乱数」で置き換える方法があります。

ビット列の交叉の仕方にもいろいろな方法が提案されましたが、もっとも単純な「一点交叉」と呼ばれる方法では、交叉点の位置をランダムに選択し、例えば交叉点より上位のビット配列に親 A のビット配列、下位のビット配列に親 B のビット配列を用いて、つなぎ合わせたビット配列を持つ子 C を作ります。結果的に C のビット配列は、親 A と B の中間的な性格を持つものになります。

ビット配列を直接処理することは「自然言語 natural language」から「機械語 machine language」に翻訳するシステム（コンピュータ言語）を使う場合には、扱いにくくなる傾向があります。

例えば、親 A の持つパラメータ x_A, y_A, z_A, \dots と、親 B の持つパラメータ x_B, y_B, z_B, \dots とから、 $0 < r < 1, 0 < r' < 1, 0 < r'' < 1, \dots$ の一様乱数を使って、

$x_C = (1 - r)x_A + rx_B, y_C = (1 - r')y_A + r'y_B, z_C = (1 - r'')z_A + r''z_B, \dots$ とするようにランダムな位置への「ないそう内挿 (interpolation) いんたーぽれいしょん」によって、子 C のパラメータを決めれば、遺伝子を複数点で交叉させたのと同じような結果が得られ、むしろその方が「一点交叉」より「コンピュータ言語」を利用した実装は容易です（[補足 9.5.1.B](#)）。

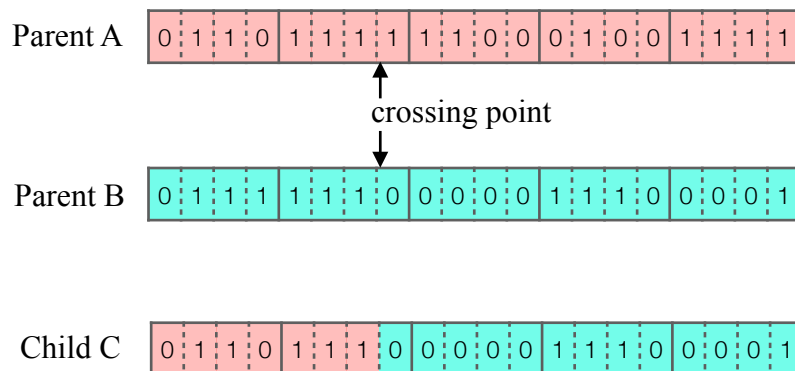


Fig. 9.5.1 遺伝的アルゴリズムで有性生殖（遺伝子交叉）を模擬する例。親 A, B から子 C を生成するとき、ランダムな交叉点でビット列を切り換える。

遺伝的アルゴリズムを使って複数の変数を含む函数 $f(x, y, z, \dots)$ の最小値を求めるためには、例えば以下のようにします。

- (I) 初期世代の生成：100 体の個体 $(x, y, z, \dots) = (x_0, y_0, z_0, \dots), \dots, (x_{99}, y_{99}, z_{99}, \dots)$ をランダムに生成します。
- (II) 自然選択：かんすう函数 $f(x, y, z, \dots)$ の値の大きい（適応度の低い）40 体の個体を淘汰（排除）します。

(III) 交叉：残った上位 60 体から、30 ペアを作成し、それぞれのペアごとに中間的な性格を持つ個体を 1 体ずつ、合計 30 個体を生成します。

(IV) 突然変異：10 個体をランダムに生成し、個体数を 100 体に戻します。

(V) 世代更新：(II) の自然選択に戻ります。

上記の手順は通常使われる「遺伝的アルゴリズム」の簡略版とも言えるもので、一般的にはもう少し複雑な手順がとられますが、世代交代を繰り返すうちに、適応度の高い個体の割合が増えてきて、最小位置近傍に集団 (population) の濃集する挙動が現れます。

突然変異を入れなくても最適化はできますが、複雑に山や谷が入り組んだ地形に対して個体数が十分に多くなければ、モンテカルロ法のように「初期配置」をランダムに選択するタイプのアルゴリズムを使う場合に、モンテカルロ法の「棄却サンプリング」と同じように偽最小に陥る危険が高くなります。一方で、突然変異体 ミュータント mutant を素朴に「ランダムな遺伝子を持った個体」として発生させると、最適化が進行した時点では、生存する個体より適応度の低い確率が高く、すぐに淘汰 (排除) されます。突然変異の出現の割合を大きくすることは、多くの場合に「無駄な計算」を増やすこととなります。

遺伝的アルゴリズムには多様なバリエーションがあり「遺伝的アルゴリズム」と言うだけでは、具体的にはどのような手法をとったのか特定できません。どのようなバリエーションを選択するべきか、どのようにパラメータを決めるべきかも不明確であるという難点は残っています。

9-5-2 差分進化アルゴリズム Differential evolution algorithm

進化論的アルゴリズム evolutionary algorithm と呼ばれる計算手法のうちで、現時点では**差分進化アルゴリズム** differential evolution algorithm と呼ばれる手法の使われる例が比較的多くなる傾向があるようです。差分進化アルゴリズムは、1990 年代後半に Storn と Price によって提案されました (Storn, [1996](#); Storn & Price, [1997](#))。

差分進化アルゴリズムでは、以下の手順をとります。

変数が n 個 (n 次元) の関数 $f(x_1, x_2, \dots, x_n) = f(\mathbf{x})$ を最小化する問題に用いるとします。

(i) 以下に示すように差分進化アルゴリズムの挙動を決定づける 3 つのパラメータ、 N_p, P_c, F の値を決める。

(1) **個体数** (number of population) N_p を決める。最低でも N_p は 4 以上の整数 ($N_p \in [4, 5, \dots]$) でなければいけない。例えば $N_p = 10n$ とする。

(2) **交叉確率** (crossover probability) P_c を決める。 P_c は 0 以上 1 以下 ($0 \leq P_c \leq 1$) の数であり、例えば $P_c = 0.9$ と決める。

(3) **差分荷重 (differential weight) F** を決める。 F は0以上2以下 ($0 \leq F \leq 2$) の数とする。例えば $F = 0.8$ とする。

(ii) すべての個体 (エージェント agent) $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{N_p}\}$ の**初期配置**を探索空間の中でのランダム配置として決定する。

(iii) 終了条件を満たす (例えば繰り返しが指定上限回数に達するか, $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{N_p}\}$ の値の取る範囲が十分に小さくなるような場合) まで, それぞれの個体 \mathbf{x}_j ($j = 1, \dots, N_p$) について, 以下の操作を繰り返す。

(1) **突然変異体 (mutant)** を作成する。 $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{N_p}\}$ の中から, \mathbf{x}_j と一致しない3つの異なる $\mathbf{x}_k, \mathbf{x}_l, \mathbf{x}_m$ をランダムに選び出し,

$$\mathbf{v}_j = \mathbf{x}_k + F(\mathbf{x}_l - \mathbf{x}_m) \quad (9.5.2.1)$$

として計算される \mathbf{v}_j を「突然変異体」とする ([補足 9.5.2.A](#)) 。

(2) オリジナルの個体 $\mathbf{x}_j = (x_{1j}, \dots, x_{nj})$ と, 対応する突然変異体 $\mathbf{v}_j = (v_{1j}, \dots, v_{nj})$ との間で**遺伝子交叉 (crossover)** を起こさせて, **候補個体 (trial) $\mathbf{u}_j = (u_{1j}, \dots, u_{nj})$** を作成する。 n 個の $r_i \in [0, 1]$ の関係を満たす連続一様乱数 r_i を発生させる。また $i_R \in \{1, \dots, n\}$ の範囲で離散一様乱数 i_R を一つ発生させる (乱数 i_R は, 候補個体 \mathbf{u}_j の遺伝子の少なくとも一つに, 突然変異体 \mathbf{v}_j の遺伝子を引き継がせるために用いる)。二つの場合にわけて処理をする。(a) $i = i_R$ または $r_i \leq P_C$ のとき, $u_{ij} = v_{ij}$ とする (突然変異体の遺伝子を採用する)。(b) $i \neq i_R$ かつ $P_C < r_i$ のとき, $u_{ij} = x_{ij}$ とする (オリジナル個体の遺伝子を採用する) ([補足 9.5.2.B](#)) 。

(3) **選択 (selection)** を行う。ここでの「選択」は, 現存するオリジナル個体 \mathbf{x}_j と新しい候補個体 \mathbf{u}_j のどちらを次世代に残すかという意味での「選択」である。

$f(\mathbf{x}_j) < f(\mathbf{u}_j)$ であればオリジナル個体 \mathbf{x}_j を採用し, そうでなければ新しい候補個体 (trial) \mathbf{u}_j を採用する。

(iv) **世代 (generation)** を交代する。[\(iii\)](#) のプロセスに戻る。

差分進化アルゴリズム (differential evolution algorithm) は, **遺伝的アルゴリズム (genetic algorithm)** のように「生物学的な進化のプロセス」を模倣する^{もほう}というより, むしろ[8章2節](#)に示した**ネルダー・ミード法 (Nelder-Mead method)** にやや近い性格を持ちます。擬似焼鈍法や遺伝的アルゴリズムと同様に, **準ニュートン法 (quasi Newton method)** を適用できないか, 適用しても極端に非効率になるような「微分不可能」あるいは「微分困難」な性格を持つ関数 $f(\mathbf{x})$ にも適用できる汎用性の高さを持つと思われています。安定して動作しますが, 最適解を得る効率が良いというわけではありません。個体数 N_p と交叉確率 P_C , 差分荷重 F の3つのパラメータのみで挙動を特定できるので, 実装は比較的容易です。

補足

(補足 9.A) モンテカルロ・ルーレット (↔)

欧州型のモンテカルロ・ルーレット Monte Carlo roulette は赤か黒に塗り分けられた 1 から 36 までの数字と他の色に塗られた 0 と合わせた 37 のスロット slots を持ち、米国型のルーレットは赤か黒に塗り分けられた 1 から 36 までの数字と緑色に塗られた 0 と 00 とを合わせた 38 のスロットを持つと言われます。

ルーレットが賭博^{とばく}に用いられる場合、欧州型モンテカルロ・ルーレットでは胴元^{どうもと}が賭け金の $1/37 \approx 2.7\%$ 儲かり、米国型ルーレットの場合では胴元^{どうもと}が賭け金の $2/38 \approx 5.3\%$ 儲かるしくみになっています。

(↔)

(補足 9.B) 「遺伝的アルゴリズム」 (↔)

「遺伝的アルゴリズム」 genetic algorithm が提案された当初の人気は高く、実際にうまく動作して見える場合も多いことから、この手法を使おうとする人が多かったことは事実です。「遺伝的アルゴリズム」と呼ばれる手法は、現在は「進化論的アルゴリズム」 evolutionary algorithm のうちのひとつと位置付けられており、「遺伝的アルゴリズム」と異なる「差分進化アルゴリズム」と呼ばれる手法の用いられる割合が多くなっています。 (↔)

(補足 9.3.A) 詳細釣り合いからのボルツマン分布の導出 (↔)

式 (9.3.6) のように

$$\frac{P(E + \Delta E)}{P(E)} = e^{-\Delta E/k_B T} \quad (9.3.A.1)$$

のような「詳細釣り合い (detailed balance)」が成立する場合、式 (9.3.A.1) の両辺の対数 (logarithm) をとれば、

$$\ln P(E + \Delta E) - \ln P(E) = -\frac{\Delta E}{k_B T} \quad (9.3.A.2)$$

とも書けて、

$$\frac{\ln P(E + \Delta E) - \ln P(E)}{\Delta E} = -\frac{1}{k_B T} \quad (9.3.A.3)$$

とも書けます。

かりに $\Delta E \rightarrow 0$ の極限をとれば、式 (9.3.A.3) の数学的な形式は

$$\frac{d \ln P(E)}{dE} = -\frac{1}{k_B T} \quad (9.3.A.4)$$

と同じことになり、式 (9.3.A.4) の微分方程式を解けば

$$\ln P(E) = -\frac{E}{k_B T} + (\text{定数}) \quad (9.3.A.5)$$

$$\Rightarrow P(E) = (\text{定数}) \times \exp\left(-\frac{E}{k_B T}\right) \quad (9.3.A.6)$$

となり、式 (9.3.1) のように

$$P(E) \propto \exp\left(-\frac{E}{k_B T}\right) \quad (9.3.A.7)$$

と表現できるという関係が自然に導かれます。ただし、ここでは $\Delta E \rightarrow 0$ の極限での話ではなく、実際には ΔE が有限の値をとるので、それほど単純なことではないかもしれません。 (↔)

(補足 9.5.1.A) 遺伝的アルゴリズムでの自然淘汰 (↔)

遺伝的アルゴリズムで用いられる「自然選択」 natural selection のプロセスは、日本では「自然淘汰」と翻訳されました。実際には農作物や家畜の品種改良のような「人為選択」 artificial selection に近いものです。

(↔)

(補足 9.5.1.B) 遺伝的アルゴリズムでの一点交叉 (↔)

遺伝的アルゴリズムを「比較的多くの変数 (パラメータ) を持つ関数の最適化」に用いる時の「一点交叉」は、実質的には「パラメータを一行に並べ、1 番目から何番目までのパラメータの組を父親からそのまま引き継がせるかをランダムに決め、残りのパラメータの組は母親からそのまま引き継がせる」ことです。一点交叉は、たとえば「目・耳・鼻・口・顎」の順に並べて、「目と耳は父親そっくり、鼻・口・顎は母親そっくりな子」を作るようなことです。しかし、このように文字通りに「ビット列の切り替え」をするタイプの一点交叉では、任意性のある「パラメータの並べ方」によって最適化のプロセスが変わることになってしまいます。

ランダム性を入れてあるとしても、「遺伝子交叉 cross over」と呼ばれることでは、子が両親の中間の性格を持つこと自体は変わりません。遺伝的アルゴリズムは、基本的には近親交配を回避する仕組みを持たないので、例えば母親が自分より低いランクの配偶者と子を作り、子ができたら古い配偶者を捨てて、子を新しい配偶者として、その子を作るようなことが繰り返され、結局自分自身の複製を作るだけのことになってしまう傾向もあります。「遺伝子交叉」と言う呼び方はされるのですが、一般的には「現実に行なわれている遺伝子交叉」とは少し違う意味合いもあります。 (↔)

(補足 9.5.2.A) 差分進化アルゴリズムでの突然変異 (↔)

「差分進化アルゴリズム」の文脈の中での「突然変異」は、現存する遺伝子から派生させる意味を持つ点では本文中に示した「遺伝的アルゴリズム」の簡略版でのまったくランダムな「突然変異」より現実的な面があるかもしれません。

オリジナル個体 \mathbf{x}_i に対して式 (9.5.2.1) :

$$\mathbf{v}_i = \mathbf{x}_j + F(\mathbf{x}_k - \mathbf{x}_l) \quad (i \neq j, k, l) \quad (9.5.2.1)$$

の関係で表される \mathbf{v}_i を突然変異体として扱えば、世代交代が進行するにつれて $\{\mathbf{x}_j\}$ のとりうる空間の範囲が収縮することから、突然変異体の出現できる許容範囲を制限していき、収束を容易にさせる効果が期待できます。 (↔)

(補足 9.5.2.B) 差分進化アルゴリズムでの交叉 (↔)

「差分進化アルゴリズム」の文脈の中での「交叉」(cross-over)では、オリジナル個体 \mathbf{x}_j と突然変異体 \mathbf{v}_j の遺伝子をランダムに引き継がせた候補個体 \mathbf{u}_j を作成するので、一点交叉とは異なり複数の場所で遺伝子を交叉させるのと同じような意味になります。 (↔)

参考文献 9

Storn, R. (1996). “On the usage of differential evolution for function optimization”. Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS). pp. 519–523. [doi:[10.1109/NAFIPS.1996.534789](https://doi.org/10.1109/NAFIPS.1996.534789)]

Storn, R. & Price, K. (1997). “Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces,” *Journal of Global Optimization*. **11**, 341–359. [doi:[10.1023/A:1008202821328](https://doi.org/10.1023/A:1008202821328)]