第1章　Meta-heuristics

1.1　Firefly algorithms

ホタルは生物発光を利用して発光し、空中を飛び回る。この発光はメスを誘引するためである。それぞれのホタルが発光する明るさは個体により異なり、以下の原則に従って他のホタルを引き寄せるとされている。

- 魅力の強さは光の強さに比例する。
- より明るく光っているオスホタルにメスホタルはより引き寄せられる。
- 光の強さは距離により減少する。

ホタルの点滅に基づく探索手法がFirefly algorithms(FA)である[6]。このアルゴリズムでは、性別の区別をしない。つまりすべてのホタルは他のホタルに引き寄せられる。この際、ホタルの光の強さは目的関数によって決まる。最小化問題を解く場合、低い関数値（より良い適合度）にいるホタルの方が強い光を放つ。さらに最も光っているホタルはランダムに動く。

FA の概要を 2頁に示す。ホタル i がホタル j に引き寄せられるときの移動式は次のようになる。

$$\boldsymbol{x}_i^{\boldsymbol{new}} = \boldsymbol{x}_i^{\boldsymbol{old}} + \beta_{i,j}(\boldsymbol{x_j} - \boldsymbol{x}_i^{\boldsymbol{old}}) + \alpha(rand(0,1) - \frac{1}{2}) \tag{1.1}$$

ただし $rand(0,1)$ は $0 \sim 1$ の一様乱数である。α はランダム値の大きさを決めるパラメータであり、$\beta_{i,j}$ はホタル i に対するホタル j の魅力の強さを

Algorithm 1 Firefly algorithm

Initialize a population of fireflies $\boldsymbol{x}_i$ $(i = 1, 2, \cdots, n)$ ▷ Minimizing objective function $f(\boldsymbol{x})$, $\boldsymbol{x} = (x_1, \cdots, x_d)^T$.
Define light absorption coefficient γ
$t = 1$ ▷ Generation count.
while $t < MaxGeneration$ and the stop criterion is not satisfied **do**
  for $i = 1$ to n **do** ▷ for all n fireflies
    for $j = 1$ to n **do** ▷ for all n fireflies
      Light intensity I_i, I_j at $\boldsymbol{x}_i, \boldsymbol{x}_j$ is determined by f
      if $I_i > I_j$ **then**
        Move firefly i towards j in all d dimensions
      end if
      Attractiveness varies with distance r via $e^{-\gamma r}$
      Evaluate new solutions and update light intensity
    end for
  end for
  Rank the fireflies and find the current best
  $t = t + 1$
end while
Postprocess results and visualization

表す。

$$\beta_{i,j} = \beta_0 e^{-\gamma r_{i,j}^2} \tag{1.2}$$

β_0 は $r_{i,j} = 0$ のとき、つまり 2 匹のホタルが同じ位置にあるときの魅力の強さである。$r_{i,j}$ はホタル i とホタル j とのユークリッド距離を表すため、魅力の強さはホタル間の距離によって変化する。

最も光強度の強いホタルは次式に従ってランダムに動く。

$$\boldsymbol{x_k}(t+1) = \boldsymbol{x_k}(t) + \alpha(rand(0,1) - \frac{1}{2}) \tag{1.3}$$

この理由は、そうしなければ初期配置における最良の解に全体が収束してしまうからである。

魅力は距離が大きい程弱くなるため、Firefly algorithms ではホタル全てが一箇所に集合するのではなく、それぞれ離れた場所で群れを形成するようになる。

Firefly algorithms は多峰性の最適化問題に向いており、PSO よりも良い結果が出るとされている。ホタルを二つのグループに分け、影響されるホタルを同じグループに属すホタルに限る拡張もある。これにより大域解と局所解をともに探索することが可能となる。

1.2 Cuckoo search

Cuckoo Search（CS）[4] は、カッコウの托卵行動に基づくメタヒューリスティクスである。托卵とは卵の世話を他種の個体に托す（その卵を育てさせる）動物の習性のことであり、托卵するカッコウは何種類か知られている。カッコウは、オオヨシキリ、ホオジロ、モズ、オナガ等の他種の巣に托卵する[1]。その際に巣にあった卵を一つ持ち去って数を合わせ、宿主（仮親）の卵の模様に似せた卵を産む (卵擬態と呼ばれる) という興味深い生態

[1] ただし托卵する相手の種はメスごとにほぼ決まっている。

行動を示す[2]。宿主の鳥は自身のものではない卵を発見するとその卵を捨ててしまうからである。

カッコウのヒナはとりわけ大きくて色鮮やかな口ばしを有し、仮親から過剰に餌をもらう。これは「超正常刺激」と呼ばれている。さらにヒナの翼の裏側には皮膚が露出している部分があり、口ばしと同じ色をしている。仮親が餌を運んできたとき、ヒナは翼をひろげてこの部分を見せつける。仮親は翼の色部分をヒナと勘違いしてしまう。つまり実際よりも多くのヒナがいると思い込み、より多くの餌を運ぶことになる。成長したカッコウは宿主よりも何倍も大きく、それに見合うような大量の餌をもらうために進化した戦略であると考えられる。

CS では、以下の 3 つのルールでカッコウの托卵行動をモデル化する。

- カッコウ は一度に 1 つ卵を産み、無作為に選んだ巣に托卵をする。
- 最も高品質な（宿主の鳥に見破られくい）卵は次の世代に受け継がれる。
- 巣の数は固定されており、托卵された卵はある一定の確率で宿主の鳥に見破られる。この場合、宿主の鳥は卵を破棄するか新しく巣を作り直す。

CS のアルゴリズムを 5頁に示す。このアルゴリズムでは、ランダムに選んだ巣の卵から Lévy flight によって新しい卵を産む。Lévy flight では、規則性のない短距離の random walk がほとんどであるが、ときどき長距離の移動をする。この動きはいくつかの動物や昆虫に観察されている。飛行パターンや採餌行動など，様々な自然現象や物理現象における確率的変動を表現できるとされている。

[2] さらに、うまれたばかりのカッコウのヒナは宿主の卵をすべて巣から放逐してしまう。そのためにカッコウのヒナの背中には窪みがあり、そこに宿主の卵を載せて巣の内側からよじ登って卵を巣の外に捨てる。この行動を発見したのは、種痘の考案で有名なエドワード・ジェンナーであった。

Algorithm 2 Cuckoo search

Initialize a population of n host nests ▷ Minimizing objective function $f(x),\ x = (x_1, \cdots, x_d)^T$.
Produce one egg x_i^0 in each host $i = 1, \cdots, n$
$t = 1$ ▷ Generation count.
while $t < MaxGeneration$ and the stop criterion is not satisfied **do**
  Choose a nest i randomly
  Produce a new egg x_i^t by performing Lèvy flights ▷ Brood parasite of the cuckoo.
  Choose a nest j randomly and let its egg be x_j^{t-1}
  if $x_j^{t-1} > x_i^t$ **then** ▷ The new egg is better.
    Replace j's egg by the new egg, i.e., x_i^t
  end if
  Sort the nests according to their eggs' performance
  A fraction (p_a) of the worse nests are abandoned and new ones are built by performing Lèvy flights
  $t = t + 1$
end while
Postprocess results and visualization

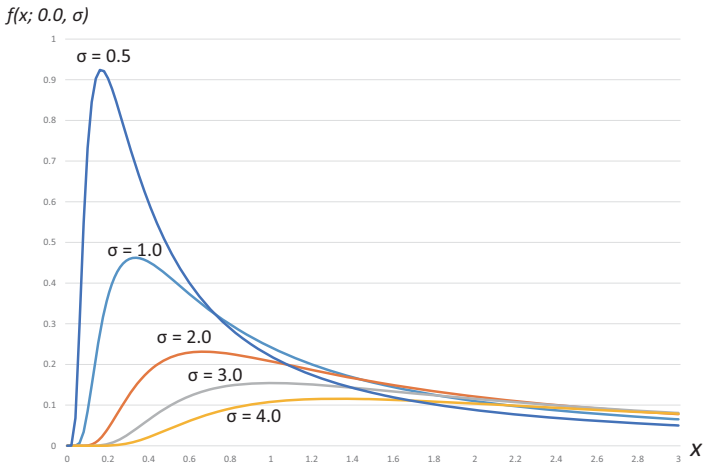

図 1.1: Levy 分布

より詳しく言えば、Lévy 分布は以下の確率密度関数で表される。

$$f(x;\mu,\sigma) = \begin{cases} \sqrt{\frac{\sigma}{2\pi}} \exp\left[-\frac{\sigma}{2(x-\mu)}\right](x-\mu)^{-3/2} & (\mu < x) \\ 0 & (otherwise) \end{cases} \tag{1.4}$$

ここで μ は位置パラメータ σ は尺度パラメータである。Lévy flight は、この分布に基づいて、ほとんどが短距離の移動だが、微小な一定確率で長距離の移動を行うランダムウォークである。最適化においては、正規分布に従うランダムウォーク (Gauss flight) を用いる場合に比べ、効率的な探索を行うことができる [4]。

たとえば、目的関数を $f(\boldsymbol{x})$, $\boldsymbol{x} = (x_1, \cdots, x_d)^T$ としよう。このとき、カッコウは巣 i について次式にしたがって新たな解候補（solution candidate) を

生み出す。

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \alpha \otimes \text{Lévy}(\lambda) \tag{1.5}$$

ここで $\alpha(>0)$ は問題のスケールに関係する。ほとんどの場合には $\alpha = 1$ である。$\otimes$ は要素ごとに α を掛けることを表している。Lévy(λ) は各要素が Lévy 分布に従う乱数ベクトルであり、以下のように実現する。

$$\text{Lévy}(\lambda) \sim rand(0,1) = t^{-\lambda} \quad 1 < \lambda \leq 3 \tag{1.6}$$

ただし $rand(0,1)$ は $0 \sim 1$ の一様乱数である。この式は、本質的にはランダムウォークであるが、重い裾野のある（heavy tail）べき乗の歩幅の分布となっている。そのため無限の平均と無限の分散を持つ。このように、通常 Lévy flight の長距離移動には、指数が $-1 \sim -3$ の指数分布を用いる。

p_a は switching probability と呼ばれるパラメータで、この確率で最もできの悪い卵（＝解）が宿主の鳥によって取り除かれる。この確率は exploration（探索）と exploitation（活用）のバランスを調整する。

CS は PSO や ACO と比べて頑健である（robust）とされている [5]。

1.3 Harmony Search (HS)

Harmony Search（HS）[1] はジャズのセッション (人間の即興演奏の生成過程) に基づいて作られたメタヒューリスティクスである。ミュージシャンは、主に以下のいずれかの方法で即興を行うとされる。

- 既知の（記憶の中にある）フレーズ（音階）をそのまま用いる
- 既知のフレーズの一部を変更・修飾して用いる。記憶中にある一つの音階に隣接する音階を演奏する。
- 新しいフレーズを作成する。演奏可能領域の中のランダムな音階を演奏する。

ミュージシャンが作曲するとき、記憶にある様々な音階の組み合わせを考える過程を一種の最適化と見なす。多くのメタヒューリスティックが魚や虫などの生物による群知能に基づいているのに対し、HS はある美的基準にしたがって調和のとれたハーモニーを探索する音楽的過程から着想を得ている点で大きく異なっている。

Harmony Search アルゴリズム（以下 HS）はミュージシャンが行なっている過程を模倣し、以下の三つの規則に従って最適解を探索する。

- HS 記憶から任意の値を選択する。
- HS 記憶から任意の値に隣接した値を選択する。
- 選択可能範囲からランダムな値を選択する。

HS では、解候補 (solution candidate) のベクトルをハーモニー、解候補の集合をハーモニーメモリ（HM: Harmony memory) と呼ぶ。ハーモニーメモリ内の解候補を一定の手順で入れ替えてゆく。一定の回数（または、終了条件を満たすまで）これを繰り返し、最後にハーモニーメモリ内に残っているハーモニーのうち、最良のものを最終解とする。

Harmony Search アルゴリズムを 10頁に示す。ここで、$HMCR$(Harmony Memory Considering Rate) がハーモニーメモリ内のハーモニーを選択する確率、PAR(Pitch Adjust Rate) はハーモニーメモリから選択したハーモニーに手を加える確率である。HMS はハーモニーの数（集団数）であり、通常 50 から 100 の間に設定される。

新たな解候補（ハーモニー）はハーモニーメモリから $HMCR$ に基づいて生成される。$HMCR$ は現在の HM から構成要素[3]を選択するときの確率であり、$1 - HMCR$ は新たな構成要素をランダムに生成することを示している。この後でさらに PAR の確率で突然変異する。bw(Bandwidth) は突然変異の最大サイズである。新しく生成された解候補（ハーモニー）が HM の最悪解よりも良い場合、最悪解を入れ替える。

[3] GA では遺伝子型 (genotype) における各遺伝子座 (allele) に相当する。

この手法は遺伝的アルゴリズム (GA) にも類似しているが、GA では一つもしくは二つの現存する染色体の連なり（親個体）のみを用いて子供の染色体を生成するのに対し、HS では HM の全てのメンバーが親候補となる点で異なっている。

1.4 Cat swarm optimization(CSO)

Cat swarm optimization(CSO) は、猫の行動にヒントを得た最適化法である。猫はほとんどの時間を安静に過ごしているが、自分の周りの環境に興味を持ち、かつ非常に警戒している。ネコはエネルギーを節約するために、休んでいる時間に比べて、獲物を追いかける時間は非常に短い。

これに基づいて、CSO には 2 つのモードがある。

- **seeking mode** 休憩時間（休息しており周囲を観察している状態）を表す。猫は獲物や危険を感知すると動くことを決断する。
- **chasing mode** 獲物を追いかけている時間を表す。seeking mode で獲物を見つけたあとに、追いかけるスピードと方向を決めて移動する。

CSO では、複数の猫が探索空間に生成され、これらがそれぞれ解候補となる。それらの猫が seeking mode と chasing mode の二つのグループに分けられる。このモードの比率が混合比（MR：chasing mode 数/seeking mode 数）である。ネコはほとんどの時間を安静にして周囲を観察することに費やすので、通常 MR は小さく設定される。

seeking mode には以下の 4 つのパラメータがある。

- SMP:seeking mode の猫のいくつを取り扱うか。ネコのメモリサイズを定義するために使われる。
- SRD:各パラメータをどのくらい変えるか。選択した次元における突然変異の幅である。変更する場合、新しい値と古い値の差が SRD の範囲外にならないようにする。

Algorithm 3 Harmony search

for $i = 1$ to HMS **do** ▷ HM initialization.
  for $j = 1$ to n **do** ▷ n: harmony length.
    Randomly initialize x_j^i in HM ▷ x_j^i: j-th position of i-th harmony.
  end for
end for
while the stop criterion is not satisfied **do** ▷ Generate a new solution candidate $\boldsymbol{x}$.
  for $j = 1$ to n **do**
    if rand(0,1)$<$ $HMCR$ **then**
      Let x_j in $\boldsymbol{x}$ be the j-th dimension of a randomly selected HM member
      if rand(0,1)$<$ PAR **then**
        Apply pitch adjustment distance bw to mutate x_j
        $x_j = x_j \pm \text{rand}(0,1) \times bw$
      end if
    else
      Let x_j in $\boldsymbol{x}$ be a random value
    end if
  end for
  Evaluate the fitness of $\boldsymbol{x}$ by $f(\boldsymbol{x})$
  if $f(\boldsymbol{x})$ is better than the fitness of the worst HM member **then**
    Replace the worst HM member with $\boldsymbol{x}$ ▷ HM update
  else
    Disregard $\boldsymbol{x}$
  end if
end while
Postprocess results and visualization

- CDC:パラメータのうちいくつを変えるか。変化させる要素数を規定する。
- SPC:移動の候補かどうか。ネコがすでにいる場所が移動先の候補となるかを示す。

k 番目の猫 cat_k に対する seeking mode は以下のようになる。ただし探索空間の次元数を M とする。

Step1 j を以下のように設定する。

$$j = \begin{cases} SMP - 1 & \text{if } SPC \text{ is } TRUE \\ SMP & \text{otherwise} \end{cases} \tag{1.7}$$

Step2 k 番目の猫 cat_k の現在位置を j 個コピーする。

Step3 各々のコピーに対して、その位置を次式にしたがって置き換える。

$$x_{k,d} = (1 \pm SRD \times rand(0,1)) \times x_{k,d} \tag{1.8}$$

ただし $d \in \{1, 2, \cdots, M\}$ であり、変更する要素数 (ことなる d の数) は CDC を上限として選ばれる。

Step4 全ての候補点の適合度 (FS_i) を計算する。ただし $1 \leq i \leq j$ である。

Step5 次式に従って各候補点 (cat_i) の選択確率 P_i を計算する。

$$P_i = \frac{\mid FS_i - FS_b \mid}{FS_{max} - FS_{min}}, \quad 1 \leq i \leq j \tag{1.9}$$

ただし FS_i は cat_i の関数値（適合度）であり、FS_{max}, FS_{min} は関数の最大、最小値である。目的関数がが最小値探索である場合は $FS_b = FS_{max}$、最大値探索なら $FS_b = FS_{min}$ とする。

Step6 選択確率 P_i を用いたルーレット選択に基づいて、ランダムに現在の候補点から移動する地点を選んで、cat_k の位置を置き換える。

Tracing mode になると、各々のネコはそれぞれの速度に従って動く。このモードは、以下の 3 つのステップにより表される。

Step1 次式にしたがって、それぞれの次元 d における速度 $(v_{k,d})$ を更新する。

$$v_{k,d} = v_{k,d} + rand(0,1) \times c_1 \times (x_{best,d} - x_{k,d}) \tag{1.10}$$

ただし、$d \in \{1, 2, \cdots, M\}$ であり、$x_{best,d}$ は最も適合度の良いネコの位置である。c_1 はユーザの定義するスケーリングパラメータである。

Step2 速度が最高速度を超えていないかチェックする。最高速度を超えてしまった場合は、最高速度に修正する。

Step3 各々のネコの位置を次式にしたがって更新する。

$$x_{k,d} = x_{k,d} + v_{k,d} \tag{1.11}$$

Algorithm 4 Cat swarm optimization

Initialize a population of cats cat_i $(i = 1, 2, \cdots, n)$ with random positions $\boldsymbol{x}_i$ in M-dimensional space ▷ n: the number of cats.
Randomly assign each cat a velocity in range of the maximum velocity value (i.e., $\boldsymbol{v}_i; i = 1, 2, \cdots, n$).
while the stop criterion is not satisfied **do**
    According to MR, assign each cat a flag of the seeking or tracing mode
    Calculate the fitness function values for all cats and sort them
    X_g = the best cat ▷ find the cat with the best solution.
    for $i = 1$ to n **do**
        if cat_i's mode is in seeking mode **then**
            Start seeking mode
        else
            Start tracing mode
        end if
    end for
end while
Postprocess results and visualization

関連図書

[1] Geem,Z.W., Kim,J.H., and Loganathan,G.V., "A new heuristic optimization algorithm: harmony search," *Simulation*, vol.76, no.2, pp.60–68, 2001.

[2] Weyland,D., "A critical analysis of the harmony search algorithm? How not to solve sudoku," *Operations Research Perspectives*, vol.2, pp.97–105, 2015.

[3] Weyland,D., "A Rigorous Analysis of the Harmony Search Algorithm - How the Research Community can be misled by a "novel" Methodology," *International Journal of Applied Metaheuristic Computing*, vol.1, no.2, pp.50–60, 2010.

[4] Yang,X.-S. and Deb,S., 2009. "Cuckoo search via Levy flights," in *Proc. World Congress on Nature & Biologically Inspired Computing (NaBIC 2009)*, pp.210–214, IEEE Publications, 2009.

[5] Civicioglu,P. and Besdok,E., "A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms," *Artificial Intelligence Review*, vol.39, no.4, pp.315–346, 2013.

[6] Yang,X., *Nature-Inspired Metaheuristic Algorithms*, 2nd ed., Luniver Press, 2010.

[7] Chu,S.-C., Tsai,P.-W., Pan,J.-S., "Cat Swarm Optimization," in *Proc. Pacific Rim International Conference on Artificial Intelligence (PRICAI 2006)*, pp.854-858, 2006.