A1-2 ABC アルゴリズムを用いたハイブリッド探索手法に関する研究

福井大学 大学院工学研究科 知能システム工学専攻 進化ロボット研究室
加藤 達郎 (指導教員：前田 陽一郎、高橋 泰岳)

1. 緒 言

近年、ミツバチの群れの採餌行動をモデルとして考え出された群知能アルゴリズムである人工蜂コロニー (Artificial Bee Colony: ABC) アルゴリズム [1] が注目されている。ABC アルゴリズムは、関数最適化を目的として開発された最適化アルゴリズムであり、高次元問題に対して特に有効であることが検証されている [2]。

しかし、ABC アルゴリズムには、個体の多様性を重視した探索を行うために、優良解に収束するまでに多くの世代数を必要とするなど、特性上の問題が存在する。この問題を改善するために、複数の探索手法を組み合わせることで従来の問題点を克服し、探索性能を高めたハイブリッド進化的計算手法に関する研究が数多く報告されている [4,5]。

そこで本研究では、ABC アルゴリズムの探索処理に実数値 GA で用いられている交叉手法の一つである算術交叉を組み込んだ算術交叉適用型 ABC アルゴリズム (Arithmetic Crossover based ABC algorithm: AC-ABC) [3] と、ABC アルゴリズムの探索オペレータに GA で用いられる確率的探索処理を組み合わせることで、局所解に陥りにくいという性質を損なわずに探索性能の向上を実現した大域探索型 ABC アルゴリズム (Global Search type ABC algorithm: GS-ABC) を提案する。また提案手法の有効性を検証するために、様々なベンチマーク関数を用いて関数近似シミュレーションを行ったので、この結果についても報告する。

2. ハイブリッド ABC アルゴリズム

ABC アルゴリズムおよび本研究で提案したハイブリッド手法である AC-ABC アルゴリズムと GS-ABC アルゴリズムの 2 つの処理手順について説明する。

2·1 ABC アルゴリズム

ABC アルゴリズムは D. Karaboga らにより 2005 年に提唱された群知能アルゴリズムであり、蜜蜂を模擬した 3 種類の探索を行う [1]。ABC アルゴリズムの探索手順を以下に示す。

Step0 初期化

世代数 g=1 として、問題の定義域内に初期個体をランダムに生成する。個体数を N とする時、働き蜂の数 N_e と見物蜂の数 N_o は $N_e = N_o = N$ となる。各個体の $TC_i = 0$(i は個体番号) に設定する。TC_i は各個体が働き蜂と見物蜂の探索によって更新されなかった回数を表わし、偵察蜂の探索で用いられる。

Step1 employed bee(働き蜂) による探索

各個体につき 1 回ずつ式 (1) を用いて、全個体 $\boldsymbol{x_i}$ ($i = 1,2,\ldots,N$) に対して、ランダムに選択された一つの変数成分を更新した更新候補個体 $\boldsymbol{v_i}$ ($i = 1,2,\ldots,N$) を生成する。この時、$\boldsymbol{x_i}$=$\{x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{iD}\}$ と $\boldsymbol{v_i}$=$\{v_{i1}, v_{i2}, \ldots, v_{ij}, \ldots, v_{iD}\}$ は共に問題の次元数 D の数の変数成分を持つ実数値ベクトルであり、x_{ij} は個体番号 i の第 j 番目の変数成分を表している。次に (2) 式より、$\boldsymbol{x_i}$と$\boldsymbol{v_i}$でグリーディ選択 (適応度が高い一方を次世代に残す選択方法) を行い、個体を更新する。

$$v_{ik}^g = x_{ik}^g + \phi_{ik}^g (x_{ik}^g - x_{mk}^g) \quad (1)$$

$$(i = 1,2,\ldots,N;\ m = 1,2,\ldots,N)$$

$$\boldsymbol{x_i}^g = \begin{cases} \boldsymbol{v_i^g} & if\ f_{v_i^g} > f_{x_i^g}\ then\ \ TC_i = 0 \\ \boldsymbol{x_i^g} & others\ \ then\ \ TC_i = TC_i + 1 \end{cases} \quad (2)$$

式 (1)、(2) において、i は個体番号、k はランダムに選ばれた次元の番号、g は世代数、m は i 以外の個体番号、ϕ は - 1〜1 までの一様乱数、f_i は個体 i の適応度を表す。

Step2 onlooker bee(見物蜂) による探索

式 (3) を用いて計算した各個体の相対価値確率 P_i^g に基づくルーレット選択により 1 つの個体 i^g を選択し、式 (1) を用いて探索し、式 (2) によって位置の更新を行う。N は個体の総数を表す。

$$P_i^g = f_i^g / \sum_{n=1}^{N} f_n^g \quad (3)$$

Step3 scout bee(偵察蜂) による探索

$TC \geq limit$ を満たす個体を式 (4) を用いて置き換える。

$$x_{ij} = x_{min} + r_{ij}^g (x_{max} - x_{min}) \quad if\ \ TC_i \geq limit \quad (4)$$

$$(i = 1,2,\ldots,N;\ j = 1,2,\ldots,D)$$

x_{max} は探索空間の定義域の最大値、x_{min} は定義域の最小値、r は 0〜1 までの一様乱数を表す。置き換えられた場合、その個体の TC の値は初期化される。

Step4 最良個体の更新

その世代の探索により得られた最良個体の適応度が、その前世代までの最良個体の適応度を上回った場合、最良個体を以下のように更新する。

$$\boldsymbol{x_{best}} = \begin{cases} \boldsymbol{x_{best}^g} & if\ f_{x_{best}^g} > f_{x_{best}} \\ \boldsymbol{x_{best}} & others \end{cases} \quad (5)$$

x_{best} は前世代までの最良個体、x_{best}^g は世代 g での最良個体を表す。

Step5 終了条件の判定

終了条件を満たしている場合、探索を終了する。満たしていない場合、g=g+1 として Step1 に戻る。

2·2 AC-ABC アルゴリズム

AC-ABC では、onlooker bee による探索の代わりに実数値 GA で用いられている算術交叉を用いることで、複数の次元に渡る探索を行い、探索を加速させる。onlooker bee の探索における (1) 式の代わりに本手法で用いる算術交叉を (6) 式に示す。

算術交叉は、実数値であることを生かした交叉の一手法である。子個体を親個体同士を結んだ線分上に生成することで、一点交叉とは異なり、実数値空間の構造に適した探索を行うことができる。

$$v_{ij}^g = \begin{cases} \lambda x_{ij}^g + (1-\lambda) y_{ij}^g & if\ r_{ij}^g \leq CR \\ x_{ij}^g & others \end{cases} \quad (6)$$
$$(i = 1, 2, \ldots, N;\ j = 1, 2, \ldots, D)$$

y_{ij}^g は x_{ij}^g と共にルーレット選択により選ばれた個体、λ は内分パラメータ ($0 \leq \lambda \leq 1$)、r は 0〜1 までの一様乱数、CR は交叉率を表す。個体の位置の更新には、(2) 式を用いる。

2·3 GS-ABC アルゴリズム

GS-ABC では探索速度を高めるため、一個体に対して一度に複数の次元に渡って探索を行う。AC-ABC のように算術交叉を用いるのではなく、GA の突然変異率に類似する働きを持つパラメータ α を働き蜂と見物蜂の探索に導入する。GS-ABC の働き蜂と見物蜂の探索では、(1) 式の代わりに (7) 式を用いる。

ABC アルゴリズムでは一度に一つの変数をランダムに選択し、その変数に対してのみ探索を行うが、GS-ABC では個体の全ての変数に対して (7) 式と (2) 式を適用して探索および個体の置き換えを行う。この時、各変数は確率 α で探索処理を適用する変数に選ばれることになる。

$$v_{ij}^g = \begin{cases} x_{ij}^g + \phi_{ij}^g (x_{ij}^g - x_{mj}^g) & if\ r_{ij}^g \leq \alpha \\ x_{ij}^g & others \end{cases} \quad (7)$$
$$(i = 1, 2, \ldots, N;\ j = 1, 2, \ldots, D)$$

(7) 式において、すべてのパラメータは ABC アルゴリズムと同様で、r は 0〜1 までの一様乱数、α は 0.0〜1.0 の探索パラメータである。

GS-ABC が従来の ABC アルゴリズムと異なる点は、パラメータである確率 α の大きさに従って、探索対象となる個体の複数の変数を選択して探索を行う点のみである。このため優良解を獲得しやすいという ABC アルゴリズムの探索の長所は失われにくいと考えられる。

2·4 提案手法のアルゴリズムフロー

AC-ABC および GS-ABC のアルゴリズムフローを図 1、図 2 に示す。両提案手法は共に初期個体の生成後、employed bee の探索を行う。AC-ABC の場合は従来の ABC アルゴリズムと同様の処理であるが、GS-ABC では、個体を構成している各変数のうち、(7) 式を用いて、確率 α で選択された変数に対して探索を行い、(2) 式により個体を置き換える。この処理を全個体に適用する。

次に各個体の適応度を計算し、相対価値確率 P_i^g を求める。この確率に基づきルーレット選択を行う。そして、ルーレット選択で選択された N_o 個体に対して、onlooker bee の探索を行う。AC-ABC ではこの探索が算術交叉で置き換えられる。各変数に対して、交叉率の確率 CR で (6) 式による探索を行い、(2) 式によりグリーディ選択で個体を更新する。GS-ABC の場合は、employed bee の探索と同様に、ルーレット選択で選ばれた全個体の各変数に対して、(7)、(2) 式を用いて探索、更新処理が行われる。

その後、ABC アルゴリズムと同じく、TC の値が $limit$ 以上の個体を再初期化する scout bee の探索を行う。最後に終了条件判定が行われ、条件が満たされている場合には探索を終了する。

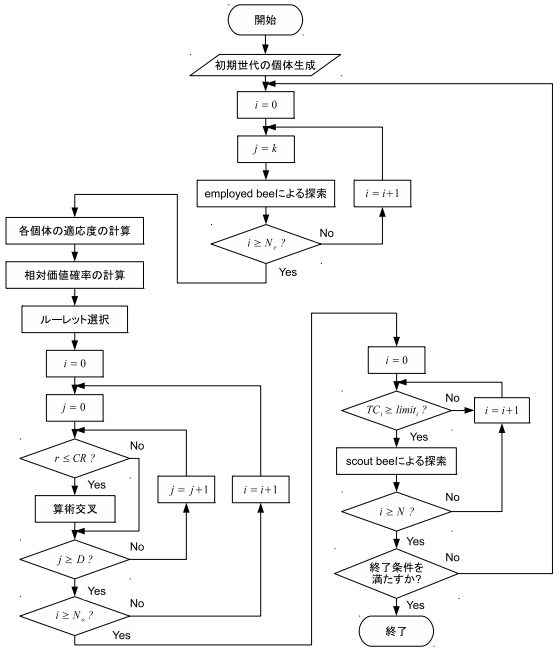

図 1: AC-ABC のアルゴリズムフロー

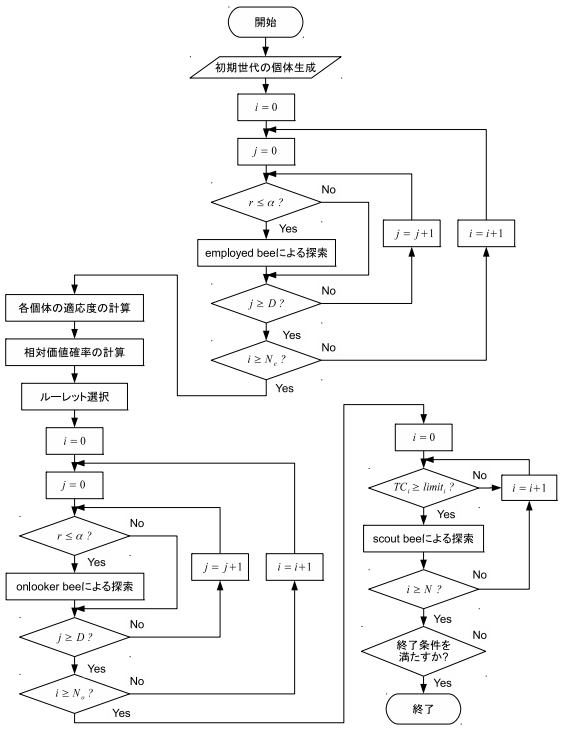

図 2: GS-ABC のアルゴリズムフロー

3. シミュレーション

本研究の提案手法である AC-ABC と GS-ABC の有効性を関数近似シミュレーションによって検証したので、その結果について報告する。

3·1 シミュレーション条件

対象問題として、関数最適化問題のベンチマークとして知られている 6 種類の関数を用いて、遺伝的アルゴリズム (GA)、差分進化法 (DE)、粒子群最適化 (PSO)、ABC、AC-ABC、GS-ABC の比較シミュレーションを行った。関数近似シミュレーションに用いた各関数のの式を (8)〜(13) に、それぞれの特徴を表 3.1 に示す。また、シミュレーションで設定した各手法のパラメータを表 3.2 に示す。

Sphere 関数

$$F_{Sphere}(x) = \sum_{i=1}^{n}(x_i^2) \tag{8}$$
$$(-5.0 \leq x_i < 5.0)$$
$$\text{最適解} : min(F_{Sphere}(x)) = F(0, 0, \ldots, 0) = 0$$

Rastrigin 関数

$$F_{Rastrigin}(x) = 10n + \sum_{i=1}^{n}\left(x_i^2 - 10\cos(2\pi x_i)\right) \tag{9}$$
$$(-5.0 \leq x_i < 5.0)$$
$$\text{最適解} : min(F_{Rastrigin}(x)) = F(0, 0, \ldots, 0) = 0$$

Rosenbrock 関数

$$F_{Rosenbrock}(x) = \sum_{i=1}^{n-1}\left(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right) \tag{10}$$
$$(-5.0 \leq x_i < 10.0)$$
$$\text{最適解} : min(F_{Rosenbrock}(x)) = F(1, 1, \ldots, 1) = 0$$

Griewank 関数

$$F_{Griewank}(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n}(x_i^2) - \prod_{i=1}^{n}\left(\frac{x_i}{\sqrt{i}}\right) \tag{11}$$
$$(-600.0 \leq x_i < 600.0)$$
$$\text{最適解} : min(F_{Griewank}(x)) = F(0, 0, \ldots, 0) = 0$$

Alpine 関数

$$F_{Alpine}(x) = \sum_{i=1}^{n}\left|x_i \sin(x_i) + 0.1x_i\right| \tag{12}$$
$$(-10.0 \leq x_i < 10.0)$$
$$\text{最適解} : min(F_{Alpine}(x)) = F(0, 0, \ldots, 0) = 0$$

2^nminima 関数

$$F_{2^n minima}(x) = \sum_{i=1}^{n}\left(x_i^4 - 16x_i^2 + 5x_i\right) \tag{13}$$
$$(-5.0 \leq x_i < 5.0)$$
$$\text{最適解} : min(F_{2^n minima}(x)) \approx F(-2.90, \ldots, -2.90) \approx -78n$$

表 3.1: ベンチマーク関数の特徴

関数名	単峰性/多峰性	設定変数間の依存関係
Sphere 関数	単峰性	なし
Rastrigin 関数	強い多峰性	なし
Rosenbrock 関数	単峰性	あり
Griewank 関数	強い多峰性	あり
Alpine 関数	弱い多峰性	なし
$2^n minima$ 関数	弱い多峰性	なし

表 3.2: シミュレーションのパラメータ設定値

	GA	DE	PSO	ABC	AC-ABC	GS-ABC
世代数	500	500	500	500	500	500
個体数 N	100	100	100	100	100	100
次元数 D	40	40	40	40	40	40
遺伝子長	800	-	-	-	-	-
選択方式	roulette	-	-	-	-	-
交叉率	0.6	0.6	-	-	0.1	-
突然変異率	0.001	-	-	-	-	-
F	-	0.3	-	-	-	-
減衰係数 ω	-	-	0.6	-	-	-
C_1	-	-	0.8	-	-	-
C_2	-	-	1.0	-	-	-
$limit$	-	-	-	$N{\cdot}D$	$N{\cdot}D$	$N{\cdot}D$
内分パラメータ λ	-	-	-	-	0.1	-
α	-	-	-	-	-	0.7

3·2 シミュレーション結果および考察

GA、DE、PSO、ABC、AC-ABC アルゴリズム、GS-ABC アルゴリズムを用いて行った関数の最適化シミュレーションの結果を図 3〜8 に示す。Rosenbrock 関数のみ、20 次元でのシミュレーションの結果を示した。グラフの横軸は世代数、縦軸は乱数シードを変更して行った 20 試行の最大適応度の平均値である。また、グラフの最大適応度の値は、0.0〜1.0 に正規化したものであり、1.0 が最適解の適応度を表している。

図 3(Sphere 関数) では、GS-ABC は探索開始から数世代で最適解に到達し、AC-ABC も 100 世代ほどで最適解に至る結果となった。図 4(Rastrigin 関数) では、Sphere 関数の結果と同様に、GS-ABC が探索初期から大幅に適応度を伸ばし、20 世代ほどで最適解を獲得した。AC-ABC は GS-ABC と比較して適応度の立ち上がりは大きく遅れたものの、500 世代前でこちらも最適解に到達した。図 5(Rosenbrock 関数) では、GS-ABC が Rosenbrock 関数に対する探索性能に優れた ABC アルゴリズムを大幅に上回る結果を示したのに対し、AC-ABC は ABC アルゴリズムを下回る結果となり、GA の交叉手法を組み合わせたことで、逆に性能が下がる場合があることを示す結果となった。図 6(Griewank 関数) では、GS-ABC の結果は、Rastirigin 関数と同じく 20 世代付近で最適解に到達し、AC-ABC は ABC アルゴリズムよりもわずかに早い世代で最適解に収束した。図 7(Alpine 関数) では、GS-ABC は Griewank 関数の場合とほぼ同じ結果となり、20 世代ほどで最適解に収束した。AC-ABC は、探索初期では PSO に適応度の立ち上がりの早さで下回ったが、100 世代前で追い抜き、その後に最適解に到達する結果となった。図 8(2^nminima 関数) では、GS-ABC が他の探索手法を大幅に上回る適応度の立ち上がりの早さを示し、10 世代

付近で最適解に収束する結果となった。AC-ABC の結果も 100 世代あたりから適応度を大きく伸ばし、ABC アルゴリズムよりも早い世代で最適解に到達した。

シミュレーション結果から、GS-ABC は、問題の性質を問わず非常に高い収束性能を持っていることが分かる。GS-ABC が ABC アルゴリズムと比較して大幅に収束性能が高まった要因としては、探索パラメータによって一回に多数の次元を探索対象に選択しており、一回に一つの次元のみに絞った探索を行う ABC アルゴリズムの数十世代分に相当する探索処理を一世代に圧縮して行っていることが考えられる。AC-ABC では、GS-ABC に次ぐ高い探索性能を示し、多峰性関数でより良好な結果が得られることが分かった一方で、Rosenbrock 関数に対しての性能は ABC アルゴリズムに比べて下がる結果となった。これは、GA の交叉手法を組み込んだことで、GA が持つ探索特性の欠点も内包してしまったことが原因と考えられる。

4. 結 言

本研究では、ABC アルゴリズムの探索性能を向上させるためのハイブリッド探索手法として、ABC アルゴリズムの onlooker bee の探索処理を実数値 GA で用いられる算術交叉にすることで収束性能を高めた AC-ABC、GA の突然変異率に類似する確率的パラメータを ABC アルゴリズムの employed bee と onlooker bee の探索に適用し、一度に個体の複数の次元を確率的に選択し大域探索を行うことで性能を向上させた GS-ABC を提案した。GS-ABC は、6 種類のベンチマーク問題の全てにおいて従来手法を大幅に上回る結果を出した。同様に AC-ABC も、優良解への収束性能を ABC アルゴリズムやその他の従来手法と比較して高めることができたことから、ABC アルゴリズムのハイブリッド手法は、最適化問題に対する優れた解法として十分に期待できる。

今後の課題として、他の関数近似のベンチマーク問題を対象としたシミュレーションによる提案手法の有効性のさらなる検証が必要であると考えられる。

参考文献

[1] D.Karaboga and B.Basturk, “A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm,” *J. Global Optimization*, Vol.39, pp.459-471 (2007)

[2] 飯村伊智郎, 中山茂, “関数最適化問題を対象とした Artificial Bee Colony アルゴリズムのロバスト性に関する研究,” 進化計算シンポジウム *2010*, pp.42-46 (2010)

[3] 加藤達郎, 前田陽一郎, 高橋泰岳, “算術交叉を用いた改良型 Artificial Bee Colony アルゴリズム,” 第 28 回ファジィシステムシンポジウム, CD-ROM, pp.430-435 (2012)

[4] G.Zhu and S.Kwong, “Gbest-guided artificial bee colony algorithm for numerical function optimization,” *Applied Mathematics and Computation*, Vol.217, pp.3166-3173 (2010)

[5] A.Abraham, R.K.Jatoth and A.Rajasekhar, “Hybrid Differential Artificial Bee Colony Algorithm,” *Journal of Computational and Theoretical Nanoscience*, Vol.9, No.2, pp.1-9 (2012)

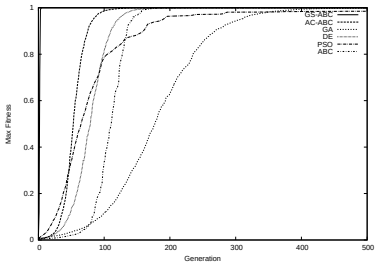

図 3: シミュレーション結果 1 (Sphere 関数、40 次元)

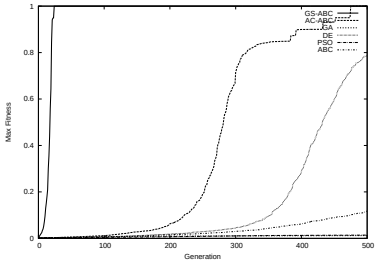

図 4: シミュレーション結果 2 (Rastrigin 関数、40 次元)

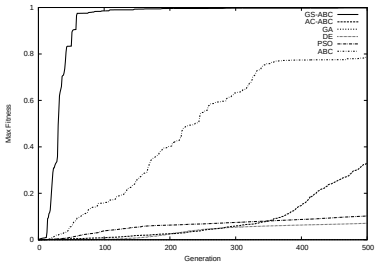

図 5: シミュレーション結果 3 (Rosenbrock 関数、20 次元)

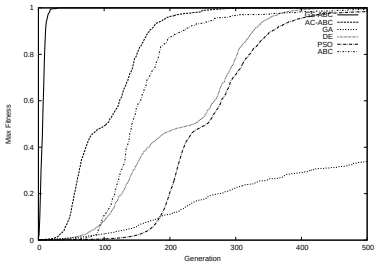

図 6: シミュレーション結果 4 (Griewank 関数、40 次元)

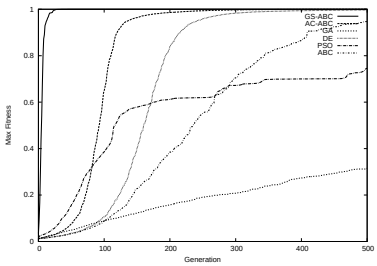

図 7: シミュレーション結果 5 (Alpine 関数、40 次元)

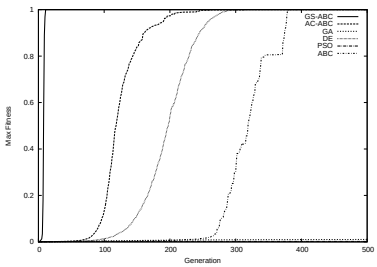

図 8: シミュレーション結果 6 (2^nminima 関数、40 次元)