Vol.2014-MPS-101 No.5
2014/12/9

時変環境における局所的情報共有に基づく Artificial Bee Colony アルゴリズム

髙野諒[†1]　原田智広[†1]　佐藤寛之[†1]　髙玉圭樹[†1]

本論文では，時変環境における最適化手法として個体間の情報共有を局所間のみに限定した Artificial Bee Colony アルゴリズム(ABC-lis: ABC algorithm based on local information sharing)を提案する．ABC-lis アルゴリズムは従来の ABC アルゴリズムにおいて大域的に共有されていた情報を，局所的に限定することにより探索領域全体に個体を分散させる．ABC-lis アルゴリズムの大域的な探索性能を検証するため，2 つの従来の ABC アルゴリズムと時変多峰性問題において比較する．これにより ABC-lis アルゴリズムにおいては各個体が探索領域全体に適切に分散することで全ての局所解を発見し，常に最適解を捕捉することが可能であることを示した．

Artificial Bee Colony Algorithm based on Local Information Sharing in Time-Varying Environment

RYO TAKANO[†1]　TOMOHIRO HARADA[†1]
HIROYUKI SATO[†1]　KEIKI TAKADAMA[†1]

This paper proposes a novel Artificial Bee Colony (ABC) algorithm based on local information sharing (ABC-lis) to adapt to a time-varying environmental change. ABC-lis algorithm shares only local information of solutions unlike the conventional ABC algorithms use global information. To investigates the global search ability of ABC-lis algorithm, we compare it with conventional two ABC algorithms by applying them to a multimodal problem with time-varying environmental change. The experimental results revealed that the proposed ABC-lis algorithm can find all of local minimum solutions by distributing bees to the whole search area appropriately and thus it can keep to select the best solution even its value changes as a time goes.

1. はじめに

最適化問題のための群知能アルゴリズムである人工蜂コロニー(ABC: Artificial Bee Colony)アルゴリズム[1]は，変化のない環境下での高次元空間や多峰性問題について優れた探索能力をもつ手法である[3][4].

しかし，ABC アルゴリズムは時間的な変化のない時不変環境の探索アルゴリズムであり，災害環境のような時間経過によって環境が変化する時変環境にそのまま適用することは困難である．また，ABC アルゴリズムは探索する個体間で大域的な情報共有を必要とし，これは災害環境などの十分に安定した通信が確保できない環境での探索には不適格である．これに対し，時不変環境での探索を目的として性能を失わずに時間的変化に適応されるアルゴリズムの修正が提案されている[2]．しかし，この修正では，(1)多数の局所解を持つ多峰性関数においては変化の推移に追従できず，探索性能が低下する，(2)全ての個体が大域的な情報共有を必要とするため，災害環境のような実環境へ応用することが難しいという問題がある．

この問題に対し，本研究では ABC アルゴリズムにおいて全個体間で大域的に共有されていた情報を近隣の個体間のみに局所的に限定することにより探索領域全体に個体を分散させ，最適解が時間的に変化する多峰性問題においても変化への追従性が低下しない手法として ABC アルゴリズム(ABC-lis: ABC Algorithm based on Local Information Sharing)を提案する．提案手法である ABC-lis を時間推移によって 10 個の局所解が値を変化させる関数に適用し，従来法との比較により有効性を検証する．

以降，2 章では提案手法の元となる ABC アルゴリズムについて，3 章では時変環境に適応させるための ABC アルゴリズムの修正とその問題点について説明する．4 章では提案手法である ABC-lis を説明し，5 章では検証のために用いる例題について記述する．6 章では 5 章で記述した例題における実験結果とその考察を述べ，最終章である 7 章で本論文をまとめる．

2. Artificial Bee Colony アルゴリズム

2.1 アルゴリズム

ABC アルゴリズム[1]では蜂の群れ全体をコロニーと称している．コロニーは 3 種類の探索フェーズを持つ人工蜂群で構成され，それぞれの探索フェーズを働き蜂(employed bee)，傍観蜂(onlooker bee)，斥候蜂(scout bee)と呼称する．

†1 電気通信大学
The University of Electro-Communications

このコロニーの目的は目的関数の解である餌場(food source)を探索し，最適解を見つけ出すことである．各蜂はそれぞれが 1 つの解に割り当てられるため，1 つの蜂個体と解は 1 対 1 に対応する．3 つの探索フェーズの内の 2 つである employed bee と onlooker bee のフェーズでは解の探索を担当し，探索前の解よりも良いが見つかればその解をその蜂の割り当て解として更新する．また，scout bees のフェーズではそれまで探索していた解を放棄して新しい解を探索することで探索領域全体を大局的に探索する．

ABC アルゴリズムでは，探索する目的関数の次元数をDとし，その次元は $j = \{1,2,\cdots,D\}$のようにして表わす．また，蜂の個数をN_Sとする．蜂が見出した解はそれぞれD個の要素を持つベクトルであり，それぞれの解をx_i $(i = 1,2,\cdots,N_s)$，その評価値を$fit(x_i)$と表す．ここで評価値はその値が高いほど良いものとし，最も評価値の高い解x_{best}の評価値$fit(x_{best})$を最大化させることを目的とする．ABC アルゴリズムの各ステップでの動作は以下のようになる．また，ABC アルゴリズムの各ステップにおける蜂の動きを**エラー! 参照元が見つかりません。エラー! 参照元が見つかりません。**に示す．**エラー! 参照元が見つかりません。**では長方形の枠内を探索領域とし，その色が濃い領域ほど評価値の高いことを表している．また，四角形が employed bee，円形が onlooker bee，三角形が scout bee を表現している．

- Step 0: 初期化

はじめに反復回数$C = 0$とし，探索領域内にランダムに解（蜂）を配置し，各解の探索試行回数$trial_i = 0$とする．配置された初期位置での解の評価値$fit(x_i)$を計算し，そのなかで最も評価値の高い解をx_{best}として保持する．

- Step1: employed bee フェーズ

employed bee フェーズでは現在の解 x_iの周囲から新しい解の候補v_iを選んで探索する．これは現在の解x_iについてランダムで選択された次元$j = \{1,2,\cdots,D\}$における要素であるx_{ij}について，同じくランダムに選択された別の解 x_kの要素x_{kj}との相対的な位置関係により決定される．現在の解x_iと解の候補v_iとの相違点は次元jにおけるそれぞれの要素x_{ij}とv_{ij}のみとなる．この解の候補v_iの要素v_{ij}は次の式を用いて表される．

$$v_{ij} = x_{ij} + \Phi(x_{ij} - x_{kj}) \quad (1)$$

ここで$\Phi = [-1,1]$は一様乱数である．この解の候補v_iによる解の更新は以下のように$fit(v_i) > fit(x_i)$ならば，$x_i = v_i$，$fit(x_i) = fit(v_i)$，$trial_i = 0$ とし，$fit(v_i) \le fit(x_i)$ ならば，$trial_i = trial_i + 1$とし，employed bee は現在の解に留まる．

$$x_i = \begin{cases} v_i & if\ fit(v_i) > fit(x_i) \\ x_i & otherwise \end{cases} \quad (2)$$

$$trial_i = \begin{cases} 0 & if\ fit(v_i) > fit(x_i) \\ trial_i + 1 & otherwise \end{cases} \quad (3)$$

これにより employed bee はより評価値の高い位置に解を移動させる．

- Step2: onlooker bee フェーズ

次に，onlooker bee フェーズに移る．onlooker bee フェーズでは employed bee フェーズで探索した解x_iを次式で計算される確率p_iによって選択する．この確率から高い評価値を持つ解が多く選ばれるようにルーレット選択を用いて解を選ぶ．

$$p_i = \frac{fit(x_i)}{\sum_{n=1}^{N_s} fit(x_n)} \quad (4)$$

探索には employed bee フェーズと同様に式(1)を用いて解候補を決定し，それぞれの評価値$fit(x_i)$と$fit(v_i)$を比較する．

- Step3: 最良解の記録

employed bee フェーズと onlooker bee フェーズの探索が終了した時点で，今回の処理において探索された解の中から評価値が最高となるx_{ib}について$fit(x_{ib}) > fit(x_{best})$ならば，$x_{best} = x_{ib}$として保持する．

- Step4: scout bee フェーズ

現在の解が長時間更新されず$trial_i = limit$となった解は scout bee フェーズに移行する．scout bee フェーズでは解は探索領域内にランダムに再配置される．

- Step5: 終了判定

上記処理の終了後，反復回数$C = C + 1$とし，C が上限に達するまで Step1 から一連の処理をN_{mc}回繰り返す．

3. 時変環境への適応した ABC アルゴリズム

3.1 時変環境のためのアルゴリズムの修正

2 章にて記述した ABC アルゴリズムで時変環境を探索すると解の評価値の変動を考慮することができず，過去の探索も含めて最も高い評価値の地点に留まり続けるため適切に探索を続けることができない．この問題を解決するために ABC アルゴリズムに時不変環境での探索性能を残したまま時変環境に適応させる手法が西田により提案されている[2]．時変環境での探索のため従来の ABC アルゴリズムの step1 と step3 を以下のように修正する．

- Step1’: employed bee による探索フェーズ

従来の ABC アルゴリズムにおける step 1 の(2)式では，評価値は解の候補v_iの評価値である$fit(v_i)$のみ計算し，$fit(x_i)$の算出はされていなかった．時不変環境においては解に対して評価値が変わることはないため再計算する必要はない．しかし，時変環境においては時間経過によって評価値が変化する可能性がある．そのため，探索時の評価値を再度算出することで時変変化に追従するように修正する．

- Step3’: 最良解の記録

時変環境において，全時刻を通じての最良解を保持したとしても，その解が現時刻において最も評価値の高い解であるとは限らない．そのため，step3 で記録される最良解について，全時刻を通じてではなく現時刻のみにおける最良解を記録するように修正する．具体的には，各反復回数において探索された解の中から評価値が最高となるx_{ib}につ

いて，その評価値に関わらず$x_{best} = x_{ib}$として保持する．

以上 2 点の修正により，ABC アルゴリズムは時不変環境においての探索性能を低下させることなく時変環境に適応することが可能となる．また特徴として，(1)時変環境の大局解の変化に応じて解の評価値が変化するため，これらの値が単調増加にならないこと，(2)従来の ABC アルゴリズムからパラメーターの追加がないこと，(3)修正点が簡潔なため計算量の増加が抑えられることが挙げられる[2]．また，この修正を加えた ABC アルゴリズムを以後，修正型 ABC アルゴリズムと称する．

3.2 修正型アルゴリズムの時変環境における問題点

修正型 ABC アルゴリズムには，探索時間が経過すればするほど全ての蜂が一つの局所解に集合し，時変環境における大域的探索性能を喪失するという問題がある．これは ABC アルゴリズムの step1，step2 において ，(1)式によって解の候補がランダムに選択された他の解(蜂) x_kに近づくように生成されるために，探索が進むほど全ての蜂が集合し結果的に 1 点に収束するためである．時不変環境においては最適解が移動することはないため問題になることはないが，時変環境においては全ての蜂が収束してしまった後に最適解が移動すると，その変化に追従することができなくなる．

4. ABC-lis (Artificial Bee Colony Algorithm based on Local Information Sharing)

3.2 章で記述したように修正型 ABC アルゴリズムでは，全ての蜂が一つの局所解に収束してしまうという問題がある．この問題に対し，本論文では各蜂の情報の取得に制限を加える ABC-lis (Artificial Bee Colony Algorithm based on Local Information Sharing)を提案する．具体的に ABC-lis では, step1 で各蜂が新しい解の候補v_{ij}の生成のためにランダムに選択していた他の蜂 x_k の取得に制限を加えることで，修正型 ABC アルゴリズムの持つ大局的収束性能を低下させ，時変環境での探索性能の維持を可能とする．

4.1 アルゴリズムの修正

ABC-lis では，step1 に以下のような修正を施す．

- Step1”：employed bee フェーズ

蜂x_iの探索における新しい解の候補v_{ij}の生成の際に選択される他の蜂x_kについて，従来ではランダムに選択していたのに対し，ABC-lis では蜂x_iの近傍の蜂のみに限定する．これにより蜂はある近傍のユークリッド距離dの範囲内のみ移動することになる．具体的には式(5)に示すように，蜂x_iに対して共有範囲d内にいる蜂x_nが存在した場合は, 式(1)と同様に新しい解候補 v_{ij} を計算する．一方，近傍の蜂x_nが存在しない場合は現在の解から距離dの範囲内をランダムに探索し v_{ij} を計算する．

$$v_{ij} = \begin{cases} x_{ij} + \Phi(x_{ij} - x_{nj}) & if\ \exists x_n \in X_k, \|x_i - x_k\| < d \\ x_{ij} + \Phi d & otherwise \end{cases} \quad (5)$$

ここでX_kはx_kの集合であり，Φは(1)式と同じく-1から1の一様乱数である．これにより新しい解の候補v_{ij}は必ず蜂x_iの共有範囲d内に生成される．また，step2 の探索についても同様に式(5)を式(1)の代わりに用いる．

- Step4”: scout bee フェーズ

大域的探索能力の維持するために，高い評価値を持つ餌場を保持する必要がある．そのため，情報共有により自身が局所的に最も高い評価の餌場を持つと判定した蜂については，scout bee に変化しないよう改良する．具体的には各蜂は自身の共有範囲d内の他の蜂と評価値を比較し，自身の評価値が最も高かった場合，探索回数$trial_i$の値に関わらず scout bee に変化しない．

これらの修正により，時不変環境においては大局的探索性能が制限されることになるが，時変環境での多峰性問題においては各局所解に蜂が分散して存在することになり各局所解に追従して探索することが可能となる．

5. 例題

本研究では，例題として式(6)に示すようなN個の峰を持つ多峰性関数の最小化問題について，10 個(N = 10)の局所解を持つ場合を考える．

$$g(x_1, x_2, k) = 1 - \sum_{n=0}^{N} \left[\frac{cos\left(\alpha k + n\cdot\frac{2\pi}{N}\right)+1}{2} exp\left\{ -\frac{1}{2}\left(\frac{\left(x_1 - r\ cos\ n\cdot\frac{2\pi}{N}\right)^2}{40^2} + \frac{\left(x_2 - r\ sin\ n\cdot\frac{2\pi}{N}\right)^2}{40^2} \right) \right\} \right] \quad (6)$$

この関数では原点(0,0)を中心とした半径rの円上にガウス関数で表現されたN個の局所解が等間隔に存在し，離散時刻tが増加することによりそれらの局所解が順に最小値となり大局解が変化していく．また，時刻tは反復回数の増加(C = C + 1)に合わせて値を 1 増加させる．このときの時間変化の大きさはα によってコントロールされる． N = 10，r = 350，α = 0.01のときの関数の概形を図 3 に示す．図 1 では高低差のある 9 つの谷(局所解)が確認できるが，最も低い(浅い)2 つの谷の間にも，現在は値が 1 となり確認することができないもう 1 点の局所解が存在している．これら 10 個の局所解が時間変化に応じて徐々にその値を変化させることで最小解が推移していく．

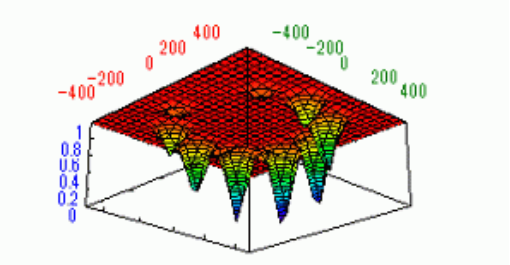


図 1 N=10, r=350, α=0.01 における$g(x_1, x_2, k)$の概形

6. 実験

6.1 実験内容

実験では，提案手法である ABC-lis と従来の ABC アルゴリズム，西田による修正を加えた修正型 ABC アルゴリズムの 3 つの手法を 5 章にて説明した関数$g(x_1, x_2, k)$において比較する．実験では 5 章での記述通り時刻tを反復回数の増加(C = C + 1)に合わせて値を 1 増加させた時変環境での比較の 2 種類を実施する．

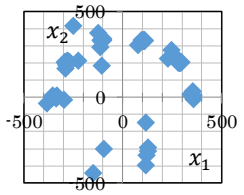
6.2 評価基準とパラメーター設定

ABC アルゴリズムと修正型 ABC アルゴリズム，ABC-lis の性能を検証するため，2 つの評価基準を用いる．1 つは関数$g(x_1, x_2, k)$における 10 個の局所解に対する蜂の分布数である．これにより時間推移に対する大域的探索性能を検証することができる．もう 1 つはその時刻tにおける最小解の座標$x_{min}(x_1, x_2)$とアルゴリズムが探索した最良解の座標$x_{best}(x_1, x_2)$のユークリッド距離$\|x_{min} - x_{best}\|$の時間推移である．これにより，アルゴリズムが探索した最良解が最小解に近似した解を探索できたか判断することができる．また，本実験は関数最小化問題のため各解の評価値$\mathrm{fit}(\mathrm{x_i})$は各解の関数値の逆数により計算する．

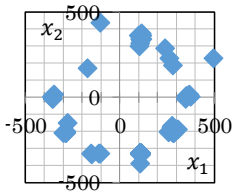
全ての手法において， ABC アルゴリズムと共通するパラメーターについては全て同一の蜂の数$\mathrm{N_S} = 100$， scout bee フェーズに移行する探索回数$\mathrm{limit} = 10$，終了までの試行回数$\mathrm{N_{mc}} = 2000$とする．ABC-lis 固有のパラメーターである探索時の移動制限のためのユークリッド距離のしきい値は$\mathrm{d} = 50$とする．また，適応する関数$g(x_1, x_2, k)$については 5 章にて記述した式(6)を用い，$\mathrm{N} = 10$， $\mathrm{r} = 350$， $\alpha = 0.01$とする．そのときの概形と各局所解の値の推移は図 3，図 4 のようになる．

6.3 結果

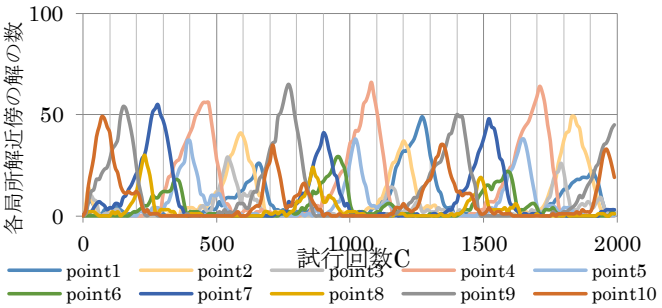
従来の ABC アルゴリズムによる結果を図 2(a)~(d)に示す．図 2(a)と(b)はそれぞれ，反復回数$\mathrm{C} = 500$と$\mathrm{C} = 2000$における探索空間内の蜂の分布を表しており横軸は探索空間の$\mathrm{x_1}$軸，縦軸は$\mathrm{x_2}$軸である．図 2 (c)は 10 個の局所解近傍に収束した解の個数を示し，横軸は試行回数を縦軸は各局所解の近傍に集まった解の数を表している．図 2(d)は関数の最小値と$\mathrm{x_{best}}$のユークリッド距離の時間推移を示し，横軸は試行回数，縦軸は最小値と$\mathrm{x_{best}}$のユークリッド距離を表している．まず，図 2(a)，(b)より各蜂の分布は反復回数 500 回と 2000 回では，その分布が 500 回目において探索領域の上部に大きく分布していたが，2000 回目において探索領域の下部にその分布は大幅に推移している．また，両方の試行回数においても全ての局所解を捕捉できていない．．また，図 2(c)から局所解への解の収束性は低く，安定していない．図 2(d)より最適解とのユークリッド距離が一部を除き大きく離れていることから，時変環境を全く追従できていないことがわかる．これは$\mathrm{x_{best}}$が 1 つの局所解からほとんど離れていないためである．



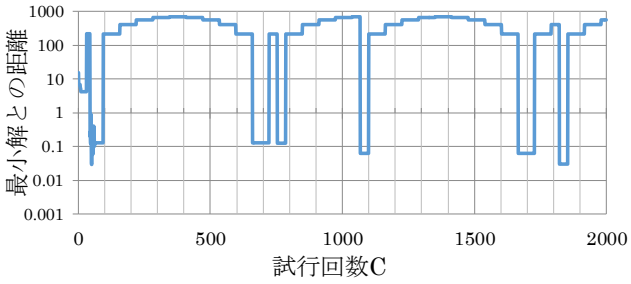
(a) C=500 での解の分布



(b) C=2000 での解の分布



(c) 10 個の局所解近傍に収束した解の個数

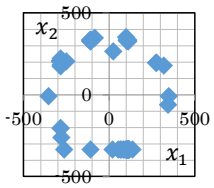


(d) 最小解と$\mathrm{x_{best}}$のユークリッド距離の推移

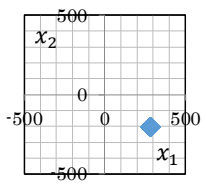
図 2 ABC アルゴリズムにおける探索領域内での解の分布と最小解とのユークリッド距離

西田による修正型 ABC アルゴリズムによる結果を図 3(a)~(d)に示す．図 3(a)と(b)はそれぞれ，反復回数$\mathrm{C} = 500$と$\mathrm{C} = 2000$における探索空間における分布を表しており，図 3(c)は 10 個の局所解近傍に収束した解の個数を示し，図 3(d)は関数の最小値と$\mathrm{x_{best}}$のユークリッド距離の時間推移となる．まず，図 3(a)，(b)より試行回数が 500 回では大まかに各局所解に蜂が集まっているが，試行回数が 2000 回では 1 つの局所解に収束していることがわかる．また，図 3(c)から各蜂が試行回数 1300 回前後である一点の局所解に集まり始め，約 1500 回で全ての蜂が収束している．このことが修正型 ABC アルゴリズム自体の時変環境における探索性能の低下に関係しており，図 3(d)に示すように試行回数 1400 回程までは断続的ながら維持していた探索能力が喪失し，全蜂が収束した局所解に$\mathrm{x_{best}}$が留まっている．

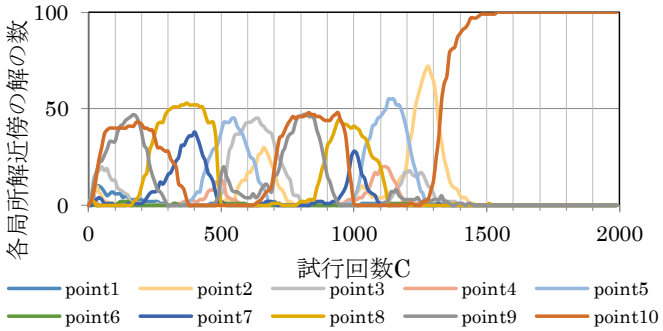
Vol.2014-MPS-101 No.5
2014/12/9



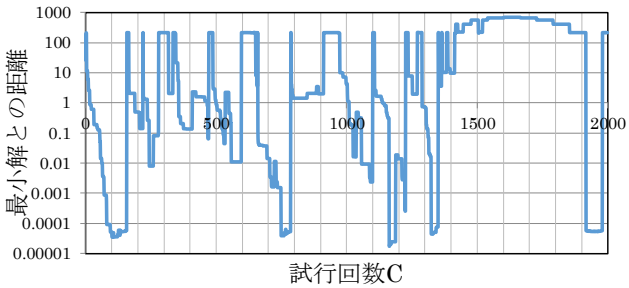
(a) C=500 での解の分布



(b) C=2000 での解の分布



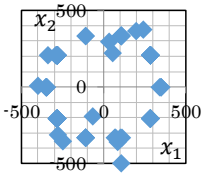
(c) 10 個の局所解近傍に収束した解の個数



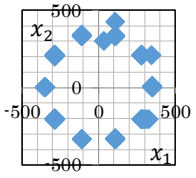
(d) 最小解とx_{best}のユークリッド距離の推移

図 3 修正型 ABC アルゴリズムにおける探索領域内での解の分布と最小解とのユークリッド距離

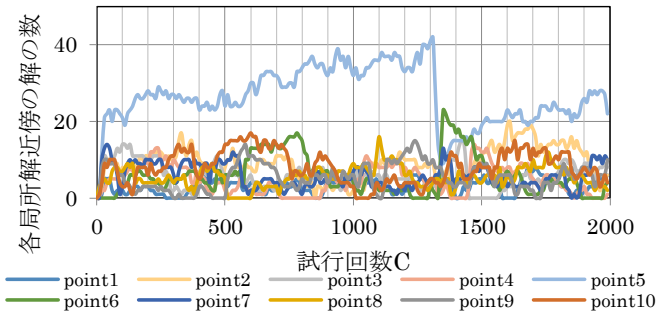
この結果を図 4(a)~(d)に示す．図 4(a)と(b)はそれぞれ，反復回数C = 500とC = 2000における探索空間における分布を表しており，図 4(c)は 10 個の局所解近傍に収束した解の個数を示し，図 4(d)は関数の最小値とx_{best}のユークリッド距離の時間推移となる．図 4(a)(b)により試行回数 500 回と 2000 回の双方で全ての局所解を捕捉できている．また，図 4(c)により 1 点を除く全ての局所解に 10 体ほどの蜂が集合しており，残りの一点についても試行回数 1300 回ほどで多く集合していた解が急激に減少している．これは，各局所解にそれぞれの解が適切に探索しており，また，過度な収束が発生しても全ての蜂が集合する前に蜂の探索や scout bee の働きにより集合した蜂が他の探索に移動していることを示している．これは ABC-lis の改良が機能した結果，蜂の収束を抑制したためである．加えて，全ての局所解に蜂が存在するため，時間経過に応じて最小解が移動しても，移動した先の適切な解が最適解x_{best}として選択されるため，図 4(d)で示すように常にその探索性能を維持することができている．



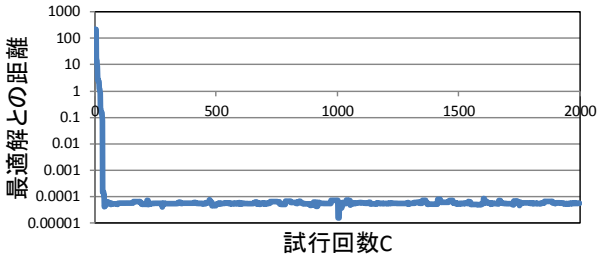
(a) C=500 での解の分布



(b) C=2000 での解の分布



(c) 10 個の局所解近傍に収束した解の個数



(c) 最小解とx_{best}のユークリッド距離の推移

図 4 ABC-lis における探索領域内での解の分布と最小解とのユークリッド距離

6.4 考察

図 5~図 7 の結果から，改良を加えていない ABC アルゴリズムから修正型 ABC アルゴリズム，ABC-lis と時変環境の探索性能が向上している．これは修正型 ABC アルゴリズムと ABC-lis の改良が有効に作用していることを示している．特に分散型の修正では，全ての局所解に蜂が集まり，安定して試行回数が 500 回からはほとんど変化していない．これは，step0 でランダムに探索領域全体に一様に分布した蜂が，離れた場所に移動せずに近隣の蜂とのみ集合したためである．これにより全ての局所解に蜂が分散し，大局解の推移に追従することが可能となっている．

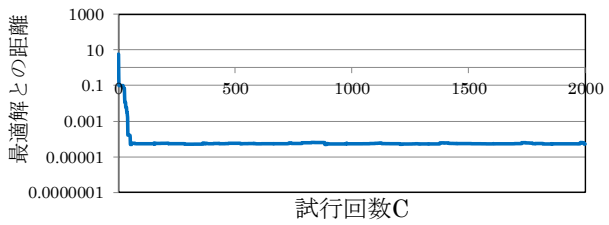
6.4.1 環境変化の速度について

前節までの検証に用いた問題においては，その環境変化の速度は(6)式の変化の度合いを示すパラメーターαについてα = 0.01として設定していた．本節ではこのパラメーターαを変更し，ABC-lis の問題の変化速度に対する探索性能について検証する．検証にはα = 0.01を基準として 5 分の 1 の変化速度であるα = 0.002と，5 倍の変化速度であるα = 0.05の 2 種類の環境設定を用いる．図 5 と図 6 にそれぞれの変化速度における結果として関数の最小値とx_{best}のユークリッド距離の時間推移を示す．

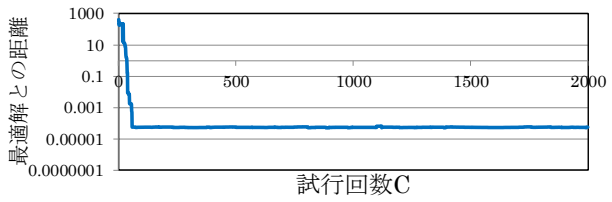
図 5(a)と(b)では変化速度を 5 分の 1 に遅くしたα＝0.002における結果である．ここで図 5(a)は ABC-lis の共有半径をd＝200と大きく設定した場合の結果を，図 5(b)は共有半径をd＝10と小さく設定した場合の結果をそれぞれ示している．これら 2 つの結果から ABC-lis は速度変化が遅い場合に関しては共有半径dの設定に関わらずα＝0.01での環境と同様の性能を発揮することがわかる．

次に，図 6(a)と(b)では変化速度を 5 倍に速くしたα＝0.05における結果である．また，図 6(a)は図 5(a)と同様に共有半径をd＝200と大きく設定し，図 6 (b)では図 5(b)と同様に共有半径をd＝10と小さく設定した場合である．図 6(a)では探索が進むにつれて探索性能を維持することができず細かい矩形の波が表れている．これは複数の局所解において，探索する蜂が移動してしまいその局所解の探索がされないためである．これにより，探索する蜂がいない局所解に問題の最小解が推移したときその解に隣り合う局所解にいる蜂が最適解として選ばれる．このため，最適解との距離が大きく離れてしまい矩形波のような波が表れる．一方，図 6(b)では矩形波の大きさは小さく最小解と最適解との距離は 0.1 ほどしか離れておらず，その頻度も少なくなっている．このことから全ての局所解を蜂が捕捉できていることを示している．これは共有半径を小さく設定したことにより，一度発見した局所解から蜂が大きく移動することを防いでいるためである．しかし，反復回数の増加に従って矩形波の表れる頻度が上昇しているため，探索が進むにつれて 1 つの局所解から蜂が減少してしまい局所的な探索性能の低下が発生していると考えられる．

以上の結果から ABC-lis が遅い変化速度に対してはその探索性能を維持し，速い変化速度についてはその探索性能を低下させることがわかった．しかし，共有半径を小さく設定することで変化速度が速い場合においても探索性能を維持することが可能となる．

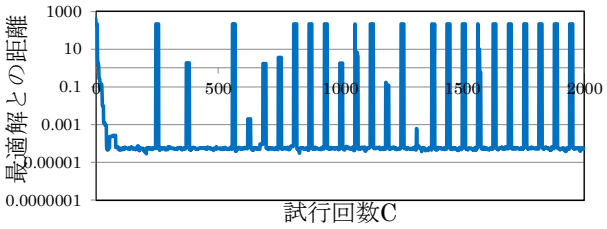


(a)　α＝0.002，d＝200

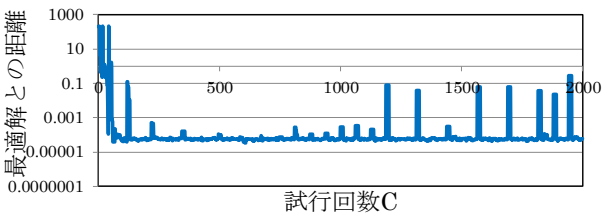


(b)　α＝0.002，d＝10

図 5　遅い変化速度における探索性能の推移



(c)　α＝0.05，d＝200



(d)　α＝0.05，d＝10

図 6　速い速度変化における探索性能の推移

7. おわりに

本稿では，まず時変環境への適応のための ABC アルゴリズムを修正する従来手法について，多峰性関数における時間経過によって大域的探索能力が低下することを明らかにした．次にこの問題を解決するために，各蜂が新しい解の候補の生成においてランダムに選択していた他の蜂の情報の取得について自身とのユークリッド距離に基づいて制限を加える改良手法である ABC-lis を提案した．

提案手法の有効性を検証するため，時間経過によって最適解が変化する多峰性関数を用いて計算機実験を実施し，従来の 2 種類の ABC アルゴリズムと比較する実験を行った結果，多峰性問題において時間経過により探索能力が低下せず，大域的探索性能を維持できるという知見を得た．また，ABC-lis の固有のパラメーターである共有範囲dの設定については，極力小さく設定することで速い環境変化についても対応が可能であることという知見を得た．

提案した ABC-lis は探索する個体間で大域的な情報共有を必要としないため，災害環境の探索などの実問題への応用が期待できる．また，今後の課題として，(1)問題に依存しない共有範囲dの自動調整法の提案，(2) 更に局所解の位置が一定でないなどの複雑な環境における探索性能の検証が挙げられる．

参考文献

1)　D. Karaboga, B. Bastur: A powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm, Journal of Global Optimization Volume39 pp.459-471,2007
2)　T. Nishida: Modification of ABC Algorithm for Adaptation to Time-Varying Functions, Electronics and Communications in Japan, 2012.
3) I. Iimura, S. Nakayama: Search Performance Evaluation of Artificial Bee Colony Algorithm on High-Dimensional Function Optimization, ISCIE, Vol. 24 (2011) No. 4 P 97-99, 2011
4) D. Karadoga, B. Basturk: On the performance of artificial bee colony (ABC) algorithm, Ap-plied Soft Computing, Vol8, pp.687-697, 2007