Vol.2017-MPS-112 No.2
2017/2/27

移住を低減するエリート保存方式をもつ分散遺伝的アルゴリズムの検討

内田 健[1,2,a)]　井口 寧[3,b)]

概要：従来より少ない移住量で解探索を可能とする分散遺伝的アルゴリズムを提案する．移住の少ない分散遺伝的アルゴリズムは，並列計算の際に計算ノード間の情報交換や同期によるオーバーヘッドを抑えることを可能とし，並列システムでの効率良い並列化を期待できる．提案法は，移住の頻度が低く頻繁に遺伝子情報を交換できず解探索が停滞する状況において，従来法より多くのエリート個体を個体群に残し解探索に関わらせる特徴を持ち，移住量の少ない環境での解探索の悪化を改善する．提案手法をベンチマーク問題に適用した結果，各島の個体数が適切に設定されている状況において，提案法は半分程度の移住量で従来と同等の解探索を可能とすることがわかった．

キーワード：島モデル型遺伝的アルゴリズム, エリート保存, 移住低減, 高並列処理

1. はじめに

分散遺伝的アルゴリズムは島と呼ばれる複数の母集団をもち，島と島の間で移住と呼ばれる情報交換をしつつ各島で同時に解を探索する．移住は，その効果により大域的な解探索を行なっていると言われており，良解探索において重要な要因となっている．分散遺伝的アルゴリズムが発見的に解を探索する手法として優れた手法であることが広く知られるようになると，移住量を制御する各種パラメータに関して数値実験にもとづく研究成果が数多くみられるようになった．廣安らは移住率をランダムに変更することでより短時間に満足いく解を発見できることを明らかにした [1]．棟朝らの研究成果 [2] や小嶋らの研究成果 [3] では，同期式移住に代わり非同期移住が提案され，同期式移住における不必要な移住を削減できることを示した．仲村らは定期的な移住の必要性を明らかにするための数値実験を試みている [4], [5]．さらに，Skolicki らは解探索において移住率よりも移住間隔を適切に設定することの方が重要であることを数値実験により示した [6], [7]．そこでは，移住頻度が極端に低い場合と極端に高い場合において解探索性能の悪化がみられた．

これらの数多くの研究成果においても，同期移住における定期的な移住の必要性については検討されているものの，移住そのものを削減する手法については検討されていない．また，分散遺伝的アルゴリズムの効率の良い並列化を目指した非同期式の移住も提案されているが，実装の容易な同期式の移住については検討されていない．

そこで，本稿では分散遺伝的アルゴリズムにおける同期式移住の頻度を低減し効率良い並列化を目的に，従来の De Jong によるエリート保存方式 [8] を改良した新たなエリート保存方式を持つ分散遺伝的アルゴリズムを提案する．さらに，数値実験において提案するエリート保存方式が解探索の悪化を改善することを示す．

2. 分散遺伝的アルゴリズム

ここでは，本稿で用いる分散遺伝的アルゴリズムについて説明する．本稿では，島モデル型遺伝的アルゴリズムに代表される典型的な分散遺伝的アルゴリズムを用いる．**図 1** に本稿で用いる分散遺伝的アルゴリズムにおける遺伝的操作と処理手順を示す．本稿では次の表記を用いる．

t	世代 t
N_{is}	島数
$N_{\mathrm{pop}}(i)$	島 i の個体数
N_{pop}	個体の総数 $N_{\mathrm{pop}} = \sum_{i=1}^{N_{\mathrm{is}}} N_{\mathrm{pop}}(i)$
L	遺伝子長
p_{c}	交叉率

[1] 北陸先端科学技術大学院大学 情報科学研究科 JAIST
[2] サレジオ工業高等専門学校 情報工学科 Salesian Polytechnic
[3] 北陸先端科学技術大学院大学 情報社会基盤研究センター JAIST
[a)] uchida@jaist.ac.jp
[b)] inoguchi@jaist.ac.jp

p_{m}　突然変異率
$P_i(t)$　世代 t における島 i の個体群
M_{rate}　移住率
M_{interval}　移住間隔

```
1: procedure Dga()
2:     t := 0
3:     for i := 1 to N_is
4:         P_i(0) := InitializePopulation(N_pop(i), L)
5:         Evaluate(P_i(0))
6:         e_i(0) := GetElite(P_i(0))
7:     end for
8:     repeat
9:         for i := 1 to N_is
10:            P'_i(t) := Reproduction(P_i(t), e_i(t))
11:        end for
12:        if t/M_interval = 0 then
13:            G_mt := MakeMigrationTopology(N_is)
14:            for (i, j) in G_mt
15:                SendMigrant(P'_i(t), j)
16:                ReceiveMigrant(P'_j(t), i)
17:            end for
18:        end if
19:        for i := 1 to N_is
20:            Crossover(P'_i(t), p_c)
21:            Mutation(P'_i(t), p_m)
22:            Evaluate(P'_i(t))
23:            P_i(t + 1) := P'_i(t)
24:            e_i(t + 1) := GetElite(P_i(t + 1) ∪ {e_i(t)})
25:            t := t + 1
26:        end for
27:    until the termination conditions are met
28: end procedure
```

図 1　本稿で用いる分散遺伝的アルゴリズムの処理手順

分散遺伝的アルゴリズムは，N_{pop} 個の個体を複数の集団（N_{is} 個の島）へ分割して解を探索する．各島 $P_i(t)$ は $N_{\mathrm{pop}}(i)$ 個の個体を持ち，各個体は長さ L のビット列からなる遺伝子とそれに対応する目的関数値を適応度として持つ．各島 $P_i(0)$ の個体の遺伝子はランダムに生成されたビット列である（図 1:4 行目）．これらの遺伝子に対応する目的関数値を求め各個体に格納している（図 1:5 行目）．また，各島 $P_i(0)$ の中で最も目的関数値の優れた個体をエリート個体 $e_i(0)$ として，各島 $P_i(0)$ とは別に記憶しておく（図 1:6 行目）．分散遺伝的アルゴリズムは，自然界における遺伝子に対する交叉や突然変異に類似した確率的な操作を個体に適用し，以下の手順で新たな個体群を生成する（図 1:8〜27 行目）．

- 再生（Reproduction）において，各島 $P_i(t)$ から目的関数値の優れた個体を選び次世代の島 $P'_i(t)$ を各々生成する（図 1:9〜11 行目）．この時，エリート個体 $e_i(t)$ が適切に次世代の島に各々残るようエリート保存戦略を採用している．
- 移住間隔 M_{interval} 世代毎に島から島へ個体を移住させ遺伝子情報を交換する（図 1: 12〜18 行目）．まず，島番号のペアからなる移住トポロジー G_{mt} を生成する（図 1:13 行目）．次に，島 $P'_i(t)$ から ランダムに選んだ $M_{\mathrm{rate}} \times N_{\mathrm{pop}}(i)$ 個の個体のコピー（移民）を島 $P'_j(t)$ へ移動し（図 1:15 行目），島 $P'_i(t)$ からの移民を島 $P'_j(t)$ の個体群に追加する（図 1:16 行目）．移民を島 $P'_j(t)$ に追加する際，島 $P'_j(t)$ からランダムに選んだ移民と同数の個体を削除し，島の個体数が変化しないようにしている．
- 島 $P'_i(t)$ 内の個体ペアに対して交叉率 p_c で交叉する（図 1:20 行目）．
- 島 $P'_i(t)$ 内の個体に対して突然変異率 p_m で突然変異する（図 1:21 行目）．
- 島 $P'_i(t)$ 内の個体の遺伝子に対応する目的関数値を求め，各個体に適応度を保存する（図 1:22 行目）．
- 次世代の島 $P_i(t+1)$ を生成し（図 1:23 行目），島 $P_i(t+1)$ の個体と世代 t までのエリート個体 $e_i(t)$ の中で最も目的関数値の優れた個体を次世代のエリート個体 $e_i(t+1)$ とする（図 1:24 行目）．

これらの一連の処理を繰り返し適用し，あらかじめ設定した条件を満足するまで解を探索する．

図 1 より明らかなように，分散遺伝的アルゴリズムにおける多くの処理が N_{is} 個の島で各々並行して実施できる（図 1:3〜11 行目,19〜26 行目）．しかし，移住トポロジーの決定や移民の送受信のため，移住はすべての島の同期を必要とする（図 1:12〜18 行目）．そのため，分散遺伝的アルゴリズムを島毎に並列処理する場合，この移住による島の同期が並列効果をあげるためのボトルネックとなっている．しかし，安易に移住間隔を長くすると解探索の悪化が見られることを，Skolicki らは数値実験にもとづく研究 [6] で明らかにしている．

そこで，本稿の以下の節では高並列向きの分散遺伝的アルゴリズムのために，移住間隔が長い場合に解探索性能を悪化させない新たなエリート保存方式を含む再生処理を提案する．

3. エリート保存方式の改良

エリート保存方式は，一般的に典型的な遺伝的アルゴリズムの選択処理の中で用いられるものである．ここでは，本稿で用いる 2 種類のエリート保存方式を紹介する．一つ目は De Jong の論文 [8] で提案された方式で，遺伝的アルゴリズムの多くの実装の中で用いられてきたものである．二つ目は，解探索性能の向上を目的に本稿で提案する方式で，分散遺伝的アルゴリズムにおける各島の個体群へエリート個体を戻すタイミングを改良したもの [9] である．

3.1 De Jong のエリート保存方式

De Jong のエリート保存方式は多くの文献で参照されているが，各島の個体群へエリート個体を戻す際の個体群サイズの扱いについては複数の実装がある．本稿では，文献 [10] の実装を De Jong のエリート保存方式として用いる．

エリート保存は，解探索の途中で得られた各島で最も良い目的関数値を持つ個体（エリート個体）を一つ，各島の個体とは別に記憶しておく．**図 2** は De Jong のエリート保存方式を採用した典型的な再生処理の手順，**図 3** はその概念図である．関数 Reproduction_DeJong() は，世代 t における島 i $P_i(t)$ とエリート個体 $e_i(t)$ に対して De Jong のエリート保存方式とルーレット選択にもとづいて個体を選択し次世代の個体候補からなる島 $P'_i(t)$ を返す．

```
1: procedure Reproduction_DeJong(P_i(t), e_i(t))
2:     P'_i(t) := φ
3:     for j := 1 to |P_i(t)|
4:         P'_i(t) := P'_i(t) ∪ RouletteSelection(P_i(t))
5:     end for
6:     if e_i(t) ∉ P'_i(t) then
7:         w := GetIndividualRandomly(P'_i(t))
8:         P'_i(t) := (P'_i(t) − {w}) ∪ {e_i(t)}
9:     end if
10:    return P'_i(t)
11: end procedure
```

図 2 De Jong のエリート保存方式を採用した再生処理の手順

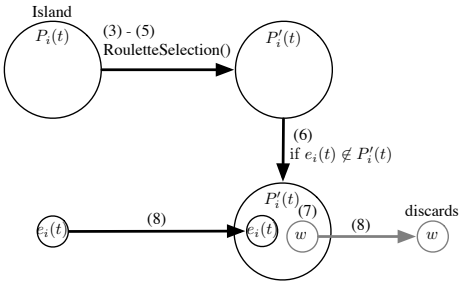

図 3 De Jong のエリート保存方式を採用した再生処理の概念図

De Jong のエリート保存方式は，まず島 $P_i(t)$ から個体を選択し次世代の個体候補群 $P'_i(t)$ を生成する（図 2:3〜5 行目，図 3:(3)-(5)）．本稿では，個体の選択にルーレット選択方式 [10] を採用する．ルーレット選択関数 RouletteSelection() は，島 $P_i(t)$ の個体の持つ適応度に比例した確率で個体を選び，選ばれた個体一つを返す．このルーレット選択を島の個体群の数 $|P_i(t)|$ だけ繰り返し，次世代の個体候補群 $P'_i(t)$ を得る．

この個体群 $P'_i(t)$ からエリート個体 $e_i(t)$ が失われたとき，島 $P_i(t)$ とは別に記憶しておいたエリート個体 $e_i(t)$ を個体群 $P'_i(t)$ へ戻す（図 2:6〜9 行目，図 3:(6)-(8)）．エリート個体 $e_i(t)$ を個体群 $P'_i(t)$ へ戻すとき，$P'_i(t)$ からランダムに選ばれた個体 w とエリート個体 $e_i(t)$ を入れ替えることで，母集団サイズを一定に保ちながらエリート個体を個体群へ戻す．これにより，関数 Reproduction_DeJong() は，$P_i(t)$ と同じ個体数を持つ $P'_i(t)$ を返す．

3.2 エリート先戻し法

本稿で提案するエリート先戻し法は，解探索性能の向上を目的に，De Jong のエリート保存方式を改良したものである．**図 4** はエリート先戻し法を採用した選択処理の手順，**図 5** はその概念図である．関数 Reproduction_Proposed() は，島 $P_i(t)$ に対してエリート先戻し法とルーレット選択にもとづいて個体を選択し，個体群 $P'_i(t)$ を返す．

```
1: procedure Reproduction_Proposed(P_i(t), e_i(t))
2:     P_i(t) := P_i(t) ∪ {e_i(t)}
3:     P'_i(t) := φ
4:     for j := 1 to |P_i(t)|
5:         P'_i(t) := P'_i(t) ∪ RouletteSelection(P_i(t))
6:     end for
7:     return P'_i(t)
8: end procedure
```

図 4 提案するエリート保存方式を採用した再生処理の手順

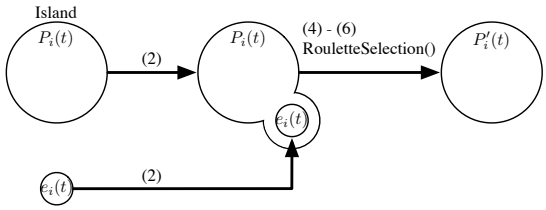

図 5 提案するエリート保存方式を採用した再生処理の概念図

本稿で提案するエリート先戻し法は De Jong のエリート保存方式と異なり，あらかじめ島 $P_i(t)$ とは別に記憶しておいたエリート個体 $e_i(t)$ を用意し，次世代の個体群 $P'_i(t)$ を生成する前に島 $P_i(t)$ へエリート個体 $e_i(t)$ を無条件で戻す（図 4:2 行目，図 5:(2)）．これにより，島 $P_i(t)$ は一時的にサイズが $|P_i(t)|+1$ となるが，ルーレット選択を $|P_i(t)|$ 回だけ実施することで島 $P_i(t)$ の個体数と同じ数だけ個体を選択し，$P_i(t)$ と同サイズの次世代個体群 $P'_i(t)$ を得る（図 4:4〜6 行目，図 5:(4)-(6)）．

3.3 エリートの保存に関する考察

分散遺伝的アルゴリズムは，解探索の途中で各島に各々エリート個体 $e_i(t)$ を持つ．解探索が停滞している一つの状況として，すべての島のエリート個体の中で適応度の最も良いエリート個体が更新されないことが考えられる．以下では，このような状況における提案するエリート先戻し法の特徴を，島の個体群に残るエリート個体数に着目して

検討する．

まず，De Jong のエリート保存方式に関して，分散遺伝的アルゴリズムの解探索が停滞している状況を考える．解探索が停滞している状況では，他の島より適応度の良いエリート個体をもつ島においてエリート個体が更新されない状況が生まれている．今，島 i に他の島より適応度の良いエリート個体 $e_i(t)$ が存在すると考える．このとき，次世代の島 i の個体群 $P_i(t+1)$ には一世代前のエリート個体 $e_i(t)$ より適応度の良い個体が存在せず，図 1:24 行目の処理で，エリート個体 $e_i(t+1)$ は一世代前のエリート個体 $e_i(t)$ となる．当然，図 2:4 行目の次世代個体候補群 $P'_j(t)$ にはエリート個体が存在しないため，図 2:8 行目の処理により個体候補群 $P'_i(t)$ にはエリート個体 $e_i(t)$ （一世代前のエリート個体 $e_i(t-1)$ と同じもの）を 1 つ持つことになる．

次に，提案するエリート先戻し法に関して，分散遺伝的アルゴリズムの解探索が停滞している状況を考える．De Jong のエリート保存方式と同様に，解探索が停滞している状況では次世代の島 i の個体群 $P_i(t+1)$ には一世代前のエリート個体 $e_i(t)$ より適応度の良い個体が存在せず，図 1:24 行目の処理で，エリート個体 $e_i(t+1)$ は一世代前のエリート個体 $e_i(t)$ となる．しかし，提案するエリート先戻し法ではルーレット選択の前に無条件でエリート個体 $e_i(t)$ を個体群 $P_i(t)$ へ戻している（図 4:2 行目）．その結果，ルーレット選択後の個体群 $P'_i(t)$ に存在するエリート個体数の期待値は

$$\frac{N_{\mathrm{pop}}(i) f_{\mathrm{elite}}}{N_{\mathrm{pop}}(i)\overline{f} + f_{\mathrm{elite}}} \tag{1}$$

となる．ただし，f_{elite} はエリート個体 $e_i(t)$ の適応度，$\overline{f}$ は個体群 $P'_i(t)$ のすべての個体の適応度の平均値である．

ここで，解探索が停滞している状況で提案法が従来法より多くのエリート個体を島へ戻す条件は，式 (1) より

$$\frac{N_{\mathrm{pop}}(i) f_{\mathrm{elite}}}{N_{\mathrm{pop}}(i)\overline{f} + f_{\mathrm{elite}}} > 1 \tag{2}$$

で与えられ，

$$\frac{f_{\mathrm{elite}}}{\overline{f}} > \frac{N_{\mathrm{pop}}(i)}{N_{\mathrm{pop}}(i) - 1} \tag{3}$$

を得る．一般的に，エリート個体の適応度 f_{elite} は島のすべての個体の適応度の平均より十分大きいため，多くの場合で式 (3) が成り立つ．すなわち，解探索が停滞している状況で提案法が従来法より多くのエリート個体を島へ戻すことが予想される．さらに $N_{\mathrm{pop}}(i)$ が大きい場合，提案法はエリート個体の適応度が島のすべての個体の適応度の平均に近いものでも，従来法より多くのエリート個体を島へ戻す能力を持つと考えられる．

このエリート個体数の違いが，移住間隔が長く解探索が停滞する環境でも，各島の母集団を刺激し解探索を促進すると考えている．

4. 数値実験

ここでは，移住頻度が低く移住量を適切に設定していない場合に，提案するエリート先戻し法を適用した分散遺伝的アルゴリズム（提案法）が，De Jong のエリート保存方式を適用したもの（従来法）に比べ，解探索性能の改善に効果があることを数値実験の結果によって確認する．

4.1 テスト関数

式 (4)，式 (5) と式 (6) で定義される 3 つの最適化問題を用いて数値実験を実施する．これらの関数はすべて最適解への収束を確認するために用いられる，よく知られたベンチマーク問題である [11]．

$$f_{\mathrm{rastrigin}}(\vec{x}) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i)\right) \tag{4}$$
$$(-5.12 \le x_i < 5.12)$$

$$f_{\mathrm{griewank}}(\vec{x}) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) \tag{5}$$
$$(-512 \le x_i < 512)$$

$$f_{\mathrm{ridge}}(\vec{x}) = \sum_{j=1}^{n} \left(\sum_{i=1}^{j} x_i\right)^2 \tag{6}$$
$$(-64 \le x_i < 64)$$

Rastrigin 関数と Griewank 関数は n 次元の多峰性関数である．他方，Ridge 関数は n 次元の単峰性関数である．数値実験ではすべての関数で n を 10 に設定している．

4.2 実験条件

数値実験では，提案法にとって極端に問題が簡単にならないように問題規模等のパラメータを設定している．**表 1** は数値実験で用いる分散遺伝的アルゴリズムに関するパラメータをまとめたものである．

表 1 数値実験で用いる各種パラメータの設定値

パラメータ	値
遺伝子長: L	100
個体の総数: N_{pop}	512
交叉率: p_{c}	1.0
突然変異率: p_{m}	$1\ /\ L$
島数: N_{is}	4, 16, 64
移住間隔: M_{interval}	5, 25, 50
移住率： M_{rate}	0.25
移住トポロジー	ランダムリング

個体の遺伝子は L ビットのグレイコードで表現されている．実際，ベンチマーク関数の各設計変数 x_i を 10 ビットのグレイコードで符号化し，10 変数（$n = 10$）を 100 ビットの遺伝子で表現している．また，数値実験では交叉

率 1.0 の一点交叉と突然変異率 $1/L$ の突然変異を用いている．この突然変異は，遺伝子の一部のビットを対立遺伝子に変更するものである．

数値実験では，移住間隔 M_{interval} を 5，25，50 へ変化させる．移住間隔が短いほど移住の頻度が高く移住量が多くなる．実際，一回の移住でおよそ $N_{\mathrm{pop}} \times M_{\mathrm{rate}}$ 個の移民が島と島の間を移動する．今回，個体の総数 N_{pop} と移住率 M_{rate} は一定値のため移住一回あたりの移民はほぼ一定となるため，数値実験では移住間隔 M_{interval} の長短で移住量を制御している．また，移住トポロジーは多くの先行研究で用いられているランダムリングを用いている．ランダムリングトポロジーは，移住の際にすべての島をノードとする一方向リングを構成する．ただし，移住毎にこのリングを構成する島の順序がランダムに変化する．

以上の条件で分散遺伝的アルゴリズムは最大 10^6 世代まで解を探索する．ただし，10^6 世代より前に最適解を発見した場合には，その世代で解探索を終了する．乱数による解探索のばらつきを確認するために，島数と移住間隔の設定値の各組合せに対して乱数のシード値を変更しながら解探索を 10 回試行する．ただし，島数と移住間隔の異なる組合せに対する数値実験の結果を比較できるように，島数と移住間隔の各組合せに対して同じシード値集合を用いて実験を行い，最適解発見世代を記録する．

4.3 解探索の比較

Rastrigin 関数，Griewank 関数と Ridge 関数に対する数値実験の結果を，各々**図 6**，**図 7** と**図 8** に示す．グラフの横軸は移住間隔を，グラフの縦軸は最適解発見世代を表している．グラフでは，従来法と提案法で島数を 4，16，64 に変化させたときの結果をプロットしている．各図では，10 回の解探索で得られた最適解発見世代の平均値をプロットしている．

図 6 の Rastrigin 関数の結果より以下のことがわかる．

- 島数と移住間隔のすべての組合せについて，提案法は従来法より早く最適解を発見している．また，移住間隔を大きくし移住量を少なくしていくと，すべての島数について従来法は最適解の発見が遅くなる傾向にある．
- 島数 64 の場合を除き，移住間隔 5 の従来法に比べ移住間隔 25，50 の提案法の方が早く最適解を発見している．すなわち，島数 4 や 16 のように各島の個体数を適切に設定できていれば，移住量を 10 分の 1 に減らしても提案法は従来法と同等またはそれ以上の解探索を実施できる．
- 島数 64 の場合，移住間隔 25 の提案法より移住間隔 5 の従来法の方が早く最適解を発見している．移住間隔 50 の提案法と移住間隔 25 の従来法についても同様のことが言える．すなわち，島が多くなり島の個体が少なくなると，提案法の優位性が低くなる．

図 7 の Griewank 関数の結果より以下のことがわかる．

- Rastrigin 関数の場合と異なり，従来法の解探索は島数によって異なる特徴を持つ．島数 64 の場合，Rastrigin 関数のときと同様に，移住間隔を大きくし移住量を少なくしていくと，従来法は最適解の発見が遅くなる傾向にある．他方島数 4 や 16 の場合，移住間隔を大きくしても小さくしても，従来法は最適解の発見が遅くなり，移住間隔 25 ときに一番早く最適解を発見している．以上より，Griewank 関数の結果については島数 4 や 16 の場合と島数 64 の場合に分けて結果を検討する．
- 島数 64 ではすべての移住間隔において，提案法は従来法より早く最適解を発見している．しかし，移住間隔 5 の従来法に比べ，移住間隔 25 や 50 の提案法は最適解の発見が遅くなる．すなわち，島が多くなり島の個体が少なくなると，提案法は安易に移住量を減らせない．
- 島数 4 や 16 の場合，移住間隔 5 や 50 では提案法は従来法より早く最適解を発見している．他方，移住間隔 25 では提案法は従来法より早く最適解を発見できない．また，移住間隔を 25 から 50 へ変えた際に，従来法では解探索が遅くなっているのに対し，提案法では解探索が早くなっている．さらに，島数 4 の場合に限ると，移住間隔 50 の提案法は移住間隔 25 の従来法より早く最適解を発見できている．すなわち，提案法は従来法とは異なる移住間隔の適値をもち，島数 4 や 16 のように各島の個体数を適切に設定できていれば，移住量を 2 分の 1 に減らしても従来法と同等またはそれ以上の解探索を実施できる．

図 8 の Ridge 関数の結果より以下のことがわかる．

- 島数と移住間隔のすべての組合せについて，提案法は従来法より早く最適解を発見している．しかし，Rastrigin 関数の場合と異なり，従来法の解探索は島数によって異なる特徴を持つ．島数 16 や 64 の場合，Rastrigin 関数のときと同様に，移住間隔を大きくし移住量を少なくしていくと，従来法は最適解の発見が遅くなる傾向にある．他方島数 4 の場合，移住間隔を大きくしても小さくしても，従来法は最適解の発見が遅くなり，移住間隔 25 ときに一番早く最適解を発見している．以上より，Ridge 関数の結果については島数 4 の場合と島数 16 や 64 の場合に分けて結果を検討する．
- 島数 4 の場合，提案法と従来法はともに移住間隔 25 のときに最も早く最適解を発見しており，同様の移住間隔の適値をもつ．また，移住間隔 50 の提案法は移住間隔 25 の従来法より早く最適解を発見している．島数 16 の場合，移住間隔 25 や 50 の提案法は，移住間

隔 5 の従来法とおおむね同様の世代数で最適解を発見できる．他方島数 64 の場合，移住間隔 25 や 50 の提案法は，移住間隔 5 の従来法に比べ解探索が遅くなっており，安易に移住量を減らせない．すなわち，島に十分な個体数を用意している場合に提案法は従来法に比べ移住量を減らせる．しかし，その効果は他のテスト関数ほど顕著ではない．

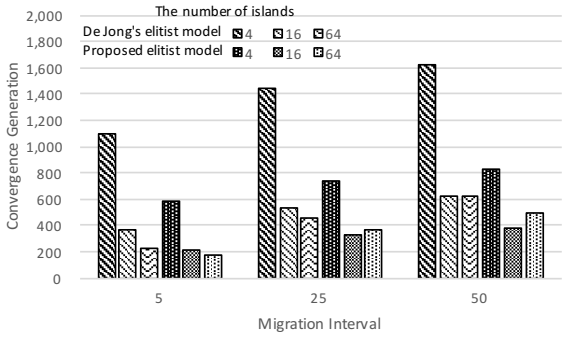

図 6 Rastrigin 関数での最適解発見世代の平均値（10 回試行）

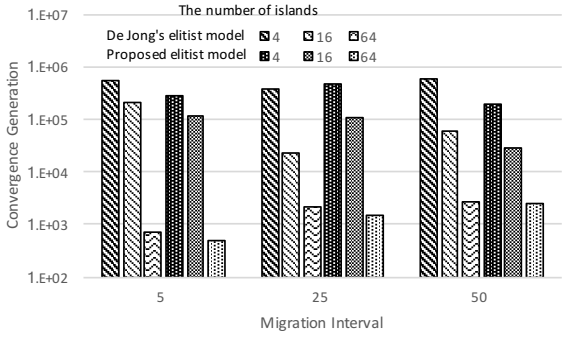

図 7 Griewank 関数での最適解発見世代の平均値（10 回試行）

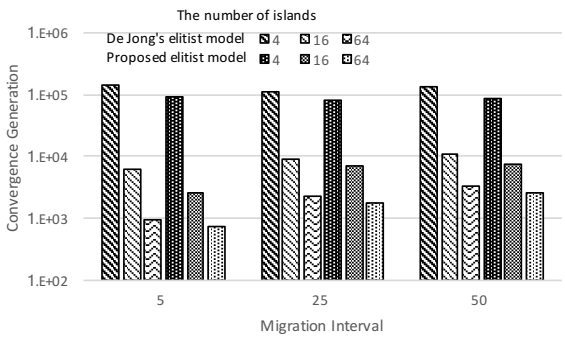

図 8 Ridge 関数での最適解発見世代の平均値（10 回試行）

5. おわりに

本稿では，De Jong のエリート保存方式を改良した新たなエリート保存方式を持つ分散遺伝的アルゴリズムを提案した．提案法は，移住の頻度が低く頻繁に遺伝子情報を交換できず解探索が停滞する状況において，従来法より多くのエリート個体を個体群に残し解探索に関わらせる特徴を持つ．数値実験の結果より，提案法は従来の 2 分の 1〜10 分の 1 程度の移住頻度で従来法と同様に最適解を発見できることを確認した．また，この傾向は分散遺伝的アルゴリズムにおける各島の個体数が多いほど強くなることもわかった．

参考文献

[1] Hiroyasu,T., Miki,M., Negami,M.: Distributed Genetic Algorithms with Randomized Migration Rate. In: Proceedings of 1999 IEEE International Conference on Systems, Man, and Cybernetics Conference, vol.1, pp.689–694, IEEE, Tokyo (1999)

[2] Munetomo,M., Takai,Y., Sato,Y.: An Efficient String Exchange Algorithm for a Subpopulation-Based Asynchronously Parallel Genetic Algorithm and Its Evaluation (in Japanese). Transactions of IPSJ, 35, 9, pp.1815–1827 (1994)

[3] Kojima,K., Ishigame,M., Chakraborty,G., Hatsuo,H., Makino,S.: Asynchronous Parallel Distributed Genetic Algorithm with Elite Migration. International Journal of Computational Intelligence, 4, 2, pp.105–111 (2008)

[4] Gong,Y., Guan,S., Nakamura,M.: Migration Effects of Parallel Genetic Algorithms on Line Topologies of Heterogeneous Computing Resources. IEICE Transactions, 91-A, 4, pp.1121–1128 (2008)

[5] Miyagi,H., Tengan,T., Mohamed,S., Nakamura,M.: Migration Effects on Tree Topology of Parallel Evolutionary Computation. In: Proceedings of TENCON 2010 - 2010 IEEE Region 10 Conference, pp.1601–1606, IEEE, Fukuoka (2010)

[6] Skolicki,Z., De Jong,K.A.: The influence of migration sizes and intervals on island models. In: Beyer,H., O'Reilly,U.(eds.) Proceedings of the 2005 conference on Genetic and evolutionary computation - GECCO '05, pp.1295–1302, ACM, Washington DC (2005)

[7] Skolicki,Z.: An analysis of island models in evolutionary computation. In: Beyer,H., O'Reilly,U.(eds.) Proceedings of the 2005 conference on Genetic and evolutionary computation - GECCO '05, pp.386–389, ACM, Washington DC (2005)

[8] De Jong,K.A.: An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Ph.D Thesis, University of Michigan (1975)

[9] Takeshi,U., Teruo,M., Yasushi,I.: The Influence of Elitism Strategy on Migration Intervals of a Distributed Genetic Algorithm. In: Handa,H., et al.(eds.) Proceedings of IES 2014, vol.2, pp.363–374, Springer, Switzerland (2014)

[10] 坂和正敏, 田中雅博: 遺伝的アルゴリズム. 日本ファジィ学会, 朝倉書店 (1995)

[11] Simon,D.: Evolutionary Optimization Algorithms - Biologically Inspired and Population-Based Approaches to Computer Intelligence-. John Wiley & Sons, New Jersey (2013)