論　文

自己適応島遺伝的アルゴリズムにおける非反復化と非同期化

高島　栄一[†]　　村田　佳洋[†]　　柴田　直樹[††]　　伊藤　実[†]

Techniques to Improve Exploration Efficiency of Parallel Self Adaptive Genetic Algorithms by Dispensing Iteration and Synchronization

Eiichi TAKASHIMA[†], Yoshihiro MURATA[†], Naoki SHIBATA[††], and Minoru ITO[†]

あらまし　遺伝的アルゴリズム（GA）の解の探索効率は，交叉率，突然変異率などのパラメータ値に依存する．人手によるパラメータ調整の労力を軽減するために，パラメータを自動的に調整する適応 GA が提案されている．しかし，多数のパラメータを同時に調整する適応 GA のほとんどは，反復して問題を解くために大きな計算量を必要とする．本論文では，従来の多数のパラメータを同時に調整する適応 GA より少ない計算量で多数のパラメータを同時に適応させる自己適応島 GA（SAIGA）と非同期 SAIGA を提案する．これらの手法は島 GA の一種であり，各島に異なるパラメータを与え，各島で解を探索しつつ解の探索効率を観測する．そして，複数の島から並行して得られる観測結果により新しいパラメータを生成し探索を続ける．その結果，反復することなしに，パラメータを適応させることができる．更に非同期 SAIGA は，島間で同期をとる必要性をなくすことにより，その結果，SAIGA に比べ，時間単位の解の探索効率が向上するように変更したアルゴリズムである．実験により，これらの手法の有効性を確認した．

キーワード　遺伝的アルゴリズム，並列分散処理，自己適応遺伝的アルゴリズム

1. まえがき

遺伝的アルゴリズム（Genetic Algorithm，以下 GA）は，自然界の進化を摸倣した最適化手法である．GA を用いる際，交叉率，突然変異率などのパラメータ値を設定する必要がある．ところが，問題を効率良く解くためのパラメータ値は，他のパラメータ値の設定，交叉方法や突然変異の方法，また，解く問題に依存する．そのため，人手によるパラメータ調整は，労力のかかる作業となる．

これに対し，適切なパラメータ値の設定に関する様々な研究がなされてきた．

GA の適切なパラメータ値を求めるために，De Jong は実験により，五つのテスト関数に適用する際の適切な値を求めた [1]．その値は，個体数は 50〜100，一点交叉の交叉率は 0.6，bit 単位の突然変異率は 0.001 であった．この値は，標準的パラメータと呼ばれ，GA で広く用いられている．

パラメータ設定の適切な範囲を求めるため，Goldberg らは control map を提案した [2]．control map とは，適切に解が探索されるパラメータ値の範囲を示したものである．この文献 [2] では，one max 問題に対して理論的に control map を求めて，これを実験により検証した．

しかし，これらは，あるいくつかの問題に対する分析であるために，それらと性質の異なる新たな問題に GA を適用させる際は，再度問題を分析し，適切なパラメータ値を求める必要がある．

以上のことから，パラメータを自動的に調整させつつ同時に解を探索させる適応 GA が研究されてきた．しかし，多くの既存の適応 GA では，少数のパラメータしか同時に調整できない．パラメータ特性が分かっていない問題を解くような場合，より多くのパラメータを同時に調整できる方がよい．ところが，多数のパラメータを同時に調整させることができる GA は，同じ問題を何度も反復して解く必要があったため，必要な計算量が大きかった．

[†] 奈良先端科学技術大学院大学，生駒市
Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma-shi, 630–0192 Japan
[††] 滋賀大学経済学部，彦根市
Faculty of Economics, Shiga University, Hikone-shi, 522–8522 Japan

筆者の属する研究グループでは，エージェント指向自己適応 GA（Agent-oriented Self-Adaptive GA，以下 A-SAGA）[3] を提案している．A-SAGA では，多数のパラメータを同時に適応させることができるが，問題を何度も反復して解くトレーニングが必要である．そのため，同じような多数の問題に対して連続して解く場合には有効であるが，解く問題が一つである場合には計算量が大きくなる傾向がある．

本論文では，A-SAGA での反復をなくすことにより，A-SAGA より少ない計算量で多数のパラメータを同時に適応させる自己適応島 GA（Self Adaptive Island GA，以下 SAIGA）を提案する．SAIGA では，A-SAGA と同様，メタ GA と島 GA の概念を組み合わせて用いる．ここで，各島に異なるパラメータを与え，各島で解を探索しつつ解の探索効率を観測する．そして，複数の島から並行して得られる観測結果により新しいパラメータを生成し探索を続ける．その結果，反復することなしに，パラメータを適応させることができる．

SAIGA は，1 台の計算機を用いて動作させることもできるが，複数の同じ計算機を用いて各計算機に一つの島を割り当てることで高速に動作させることができる．しかし，SAIGA では，新たなパラメータを生成する際，すべての島間で同期をとる必要があるため，島間で割り当てられた計算機の能力に差がある場合，遊休状態が発生する．本論文では，非同期 SAIGA（Asynchronous-Self Adaptive Island GA，以下非同期 SAIGA）を提案する．以後，SAIGA 及び非同期 SAIGA を提案手法と呼ぶ．非同期 SAIGA は，島間で同期をとる必要性をなくしたアルゴリズムである．その結果，遊休状態を削減でき，時間単位の解の探索効率の向上が期待できる．

提案手法の有効性を調べるため，様々な問題に対して解の探索効率を計測した．適応させたパラメータは，交叉率，突然変異率，トーナメントサイズ，個体数の四つである．実験の結果，提案手法は，多様な問題に対して適切なパラメータ値を求めることができ，人手により調整されたパラメータ値を用いた単純 GA に迫る性能を示した．更に，非同期 SAIGA は時間単位の解の探索効率において，SAIGA より，高い性能を示した．

2. 関 連 研 究

メタ GA は，GA を用いて，別の GA の適切なパラメータベクトルを探す方法である [4]．メタ GA は，任意のパラメータ値の組合せに対し適応させることができる．ここで，パラメータ値の組合せをパラメータベクトル，具体的に問題を解く GA を low level GA，low level GA のためのパラメータベクトルを適応させる GA を high level GA と呼ぶ．high level GA の一つの個体が一つのパラメータベクトルと対応する．GA では，世代ごとに各個体に対し評価を与える必要があり，high level GA では，すべての high level GA の個体に対して，評価を与える必要がある．一つの個体を評価するために，与えられた問題を low level GA を用いて一度解く必要がある．更に，この過程を毎世代繰り返さねばらならない．そのため，計算量が大きくなる．

筆者の属する研究グループで提案している手法に，エージェント指向自己適応 GA（Agent-oriented Self-Adaptive GA，以後 A-SAGA）がある [3]．A-SAGA は，メタ GA と同様の構造をもち，low level GA を島 GA として扱う点に特徴がある．ここで，島ごとに異なるパラメータベクトルを割り当て評価する．その結果，多数のパラメータベクトルの評価を同時に得るため，メタ GA よりパラメータベクトルを適応させる効率がよい．しかし，A-SAGA は，適切なパラメータベクトルを得るため何度も反復して問題を解くので，計算量が大きい．

メタ GA や A-SAGA では，ある問題に対して適切なパラメータベクトルを一度得ることができれば，それを流用することにより同じような性質をもつ問題を効率良く解くことができる．したがって，同じような多数の問題に対して連続して解く場合には有効である．しかし，解く問題が一つである場合には計算量が大きくなる傾向がある．

今回，提案するアルゴリズムでは，島 GA を非同期で動作させる．島 GA を非同期で動作させた関連研究として，次のものがある．Alba らは，同期島 GA と非同期島 GA の時間単位の解の探索効率を計測した [5]．その結果，常に非同期島 GA が同期島 GA を上回る解の探索効率を示した．Berntsson らは，非同期の島 GA における個体群の収束状況に関して研究した [6]．その結果，移民頻度を上げることにより，個体群がより早く収束することを理論的に示し，実験により確認した．ただし，これらの研究は，今回の研究とは異なり，パラメータを適応させていない．

3. 提案アルゴリズム

3.1 SAIGA

3.1.1 表 記 法

本論文では，次の表記法を用いる．

定数として次の記号を用いる．個体の一定の評価回数を **evaluation_count_per_era**，島の数を **island_max**，high level GA の個体群の個体数の最大値を **high_pop_max**，high level GA における世代の繰返し回数を **high_generation_max** とする．これらの定数の値や high level GA のパラメータの値は，人手により与えられる必要がある．しかし，メタ GA と同様，low level GA のある問題に対して，一度，適切な low level GA のパラメータの値が分かれば，それを用いて他の low level GA の問題も効率良く解くことができる．今回の実験では，適切な low level GA のパラメータの値を一つ求めて，low level GA のすべての問題を解く際にそれを流用した．

島数は，利用可能なプロセッサの数とする．

個体は，染色体と評価値により構成される．染色体とは，組合せ最適化問題の解空間の中にある一点の座標ベクトルである．

high level GA の個体 p の染色体を $p.\vec{x}$ とし，評価値を $p.fit$ で表す．l 個のパラメータを適応する場合，染色体 $p.\vec{x}$ は l 次元のベクトルとする．t 世代における high level GA の個体群を P_t で表す．また，現在の個体数を $|P_t|$ とする．high level GA のパラメータベクトルを $\vec{w}$ で表す．high level GA において，島 i への移民状況を保存する変数を $E[i]$ とし，島 i との通信状況を保存する変数を $R[i]$ とする．

島の個体 q は，染色体 $q.\vec{x}$ と評価値 $q.fit$ で構成される．島 i の g 世代における個体群を Q_g^i で表す．ただし，g は島内の局所変数である．low level GA の i 番目の島に与えるパラメータベクトルを $\vec{v}^i$ とする．他の島から島 i への移民個体を $q^i_{\mathbf{immig}}$ で表す．島 i から他の島への移民個体を $q^i_{\mathbf{emig}}$ で表す．島 i のエリート評価値を保存する変数を f^i で表す．

high level GA の個体の染色体 $p.\vec{x}$ から，low level GA の i 番目の島のパラメータベクトル $\vec{v}^i$ を与える関数を f と定義する．また，その逆関数を f^{-1} と定義する．

3.1.2 概 要

SAIGA は，A-SAGA と同様に，単純 GA の high level GA と島 GA の low level GA により構成される．SAIGA は，A-SAGA の high level GA に改良を加えたアルゴリズムであり，反復して問題を解くことなしにパラメータベクトルを適応させることができる．SAIGA が適応させることができるパラメータは，各島内で評価することができるもの（個体数，突然変異率，交叉率の種類，選択方法の種類：例えば，ルーレット方式にするかトーナメント方式にするかなど）である．ただし，島内で評価できないパラメータ（島の数，移民率など）を適応させることはできない．SAIGA における high level GA と島 GA の擬似コードを図 1 と図 2 に示す．

SAIGA では，まず，high level GA において，パラメータベクトルを生成し，low level GA の各島に割り当てる（図 1 の 5〜6 行目）．各島では，high level GA からパラメータベクトルが与えられると，一定の評価回数分，解を探索する（図 2 の 6〜8 行目）．このとき，解の探索効率を同時に観測する．その評価回数分の探索が終了後，解の探索効率からパラメータベクトルの評価値を求め，パラメータベクトル，その評価値と移民個体の組を high level GA に返す（図 2 の 8〜9 行目）．high level GA では，パラメータベクトル，その評価値と移民個体の組を受け取り，パラメータベクトルを進化させる（図 1 の 14〜25 行目）．再度，得られたパラメータベクトルを low level GA に割り当てる（図 1 の 29 行目）．各島では，個体群を初期化させずに，そのまま用いる．そして，与えられたパラメータベクトルを用いて，再度一定の評価回数分，解を探索して，パラメータベクトルの評価値を求める．これを決められた回数分繰り返す．この 1 回の繰返しを時代と呼ぶ．このことにより，適切なパラメータベクトルによる解の探索が期待できる．

SAIGA の high level GA は，A-SAGA の high level GA に比べ，次のように改良されている．A-SAGA の high level GA では，図 3 のように，時代ごとにそれぞれ別の個体群を設け，各時代に対し，解の探索効率が良いパラメータベクトルを求めていた．つまり，時代間でパラメータベクトルの探索結果を引き継いでいなかった．そのため，解の探索効率が良いパラメータベクトルを得るためには，何度も反復して問題を解くことが必要であった．SAIGA では，図 4 のように，前の時代でのパラメータベクトルとその評価値を引き継いで，次の時代のパラメータベクトルを生成する．これにより反復することなしに問題を解くことができるようになる．

```
Algorithm HighLevelGA()
1   t :=0; //t:high level GA の世代数
2   j:=0; //j:その時代の探索が終了した島数,
3   For i := 1 to island_max Do
4       Island(i) プロセスの起動;
        q^i_immig にダミーの値を代入する.
5       パラメータベクトル v^i をランダムに生成.
6       Island(i) プロセスに (v^i, q^i_immig) を送る.
7   Next
8   While t < high_generation_max Do
9       t := t + 1; P_t := ∅;
10      For i := 1 to island_max Do
11          R[i] := 0;//R[i]:島 i との通信状況を保存. その
            時代において, 島 i からパラメータベクトルとそ
            の評価値のペアを受け取っていなければ 0, 受け
            取ったならば 1 を保存する.
12      Next
13      While true Do
14          If Island(i) プロセスから (v^i, fit, q^i_emig, i)
            を受け取った and R[i] = 0 Then
            // q^i_emig: 島 i からの移民個体
15              j := j + 1;R[i] := 1;
16              p.x := f^{-1}(v^i);// パラメータベクトルを
                染色体に逆変換
17              p.fit := fit; P_t := P_t ∪ {p};
18              If i = island_max Then
19                  q^0_immig := q^i_emig;
                    // q^i_immig: 島 i への移民個体を保存す
                    る変数
20              Else q^{i+1}_immig := q^i_emig;
21              EndIf
22          EndIf
23          If j:=island_max Then Break ;EndIf
24      EndWhile
25      P'_t := HighPopGeneticOperation(P_t, w);
        //パラメータベクトル w をもとに選択, 交叉, 突然
        変異を行い, 結果を P'_t に代入する.
26      For i:=1 to island_max Do
27          P'_t の要素からランダムに一つ選び, それを p に
            代入する. P'_t := P'_t - {p};
28          v^i := f(p.x); //パラメータベクトルに変換する.
29          Island(i) プロセスに (v^i, q^i_immig) を送る.
30      Next
31  EndWhile
32  すべてのプロセスを終了させる.
End  /*HighLevelGA*/
```

図 1　SAIGA の high level GA の擬似コード
Fig. 1　Pseudo code of high level GA in SAIGA.

```
Algorithm Island(i)
1   f^i := 0; //島 i のエリート評価値
2   g:= 0; // g:島 i の世代数
3   島 i の個体群 Q^i_g 内の個体 q ∈ Q^i_g をすべて初期化;
4   While true Do
5       While true Do
6           If HighLevelGA() プロセスから (v^i, q^i_immig)
            を受け取った. Then Break ; EndIf
            // q^i_immig: 島 i への移民個体を保存する変数
7       EndWhile
8       (Q^i_{g+1}, fit, f^i, q^i_emig) :=
              ExecuteIsland( Q^i_g, v^i, f^i, q^i_immig, i);
        // 評価回数 evaluation_count_per_era 分, 個体
        群 Q^i_g を進化させる.
        // fit:パラメータベクトル v^i に対する評価値.
        // q^i_emig:他の島への移民個体.
9       HighLevelGA() プロセスに (v^i, fit, q^i_emig, i) を
        送る.
10      g:= g + 1;
11  EndWhile
End  /*Island*/
```

図 2　low level GA の各島の擬似コード
Fig. 2　Pseudo code of low level GA.

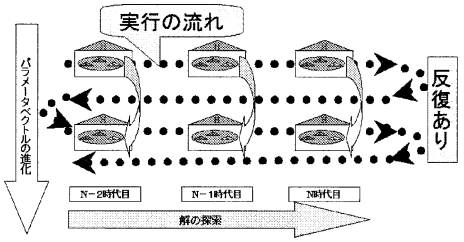

図 3　A-SAGA のパラメータベクトルの進化
Fig. 3　Evolution of parameter vector in A-SAGA.

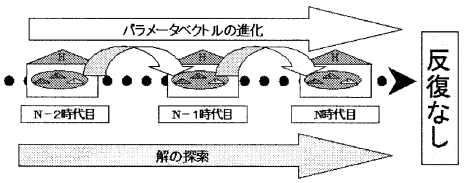

図 4　SAIGA のパラメータベクトルの進化
Fig. 4　Evolution of parameter vector in SAIGA.

3.2　A-SAIGA

3.2.1　非同期化への動機

CPU の処理能力，OS のスケジューリング管理などの様々な要因により，島間で処理能力に違いが発生する場合がある．そのような環境では，SAIGA は，島間で同期をとるため，処理能力の高い島は，処理能力の低い島の計算が終了するのを待つ必要がある．その結果，遊休状態が発生する．

そこで，本論文では，非同期化した SAIGA である非同期 SAIGA（Asynchronous-SAIGA）を提案する．非同期 SAIGA における high level GA の擬似コードを図 5 に示す．非同期 SAIGA における島 GA の擬

```
Algorithm HighLevelGA()
1   P_0 := ∅;t := 0; //t: high level GA の世代数
2   For i := 1 to island_max Do
3       Island(i) プロセスの起動;
4       q^i_immig にダミーの値を代入する.
5       パラメータベクトル v^i をランダムに生成.
6       E[i] := 0; //E[i]: 島 i への移民状況を保存する変数
7   Next
8   While t < high_generation_max Do
9       t := t + 1;
10      While true Do
11          If Island(i) プロセスから (v^i, fit, q^i_emig, i)
            を受け取った. Then Break ; EndIf
12          // q^i_emig: 島 i からの移民個体
13      EndWhile
14      If |P_{t-1}| > high_pop_max Then
15          P_{t-1} := P_{t-1} − {oldest_of(P_h)};
            // 最も古くからある要素を削除
16      EndIf
17      p.x := f^{-1}(v^i);// パラメータベクトルを染色体に
        逆変換
18      p.fit := fit; P_t := P_{t-1} ∪ {p};
19      If i = island_max Then
20          q^0_immig := q^i_emig; E[0] := 1;
            // q^i_immig: 島 i への移民個体を保存する変数
21      Else
22          q^{i+1}_immig := q^i_emig; E[i + 1] := 1;
23      EndIf
24      p^parent1 := HighIndivSelection(P_t, w);
        // 選択により親個体を得る
25      p^parent2 := HighIndivSelection(P_t, w);
26      p^offspring := HighIndivGeneticOperation(
        p^parent1 ,p^parent2, w);
        // 交叉, 突然変異により, 子個体を得る
27      v^i := f(p^offspring.x); //パラメータベクトルに変換
28      If E[i] = 1 Then E[i] := 0;
29      Else q^i_immig にダミーの値を代入する;
30      EndIf
31      Island(i) プロセスに (v^i, q^i_immig) を送る.
32  EndWhile
33  すべてのプロセスを終了させる.
End  /*HighLevelGA*/
```

図 5　非同期 SAIGA の high level GA の擬似コード
Fig. 5　Pseudo code of high level GA in A-SAIGA.

似コードは，SAIGA と同じ図 2 である．

島間で処理能力に違いがある環境で非同期 SAIGA を動作させる場合，非同期 SAIGA では，島間で同期をとらないため，処理能力の高い島は，処理能力の低い島の計算が終了することを待つ必要がなくなる．その結果，遊休状態である時間を減少させることができる．

3.2.2　定常状態 GA の導入

SAIGA では，high level GA に単純 GA の構造を用いている．単純 GA は，1 世代ごとに，個体群に属するすべての個体を新しい個体に置き換える．すべての個体に評価値が与えられて，初めて進化することができる．そのため，非同期には不向きである．

そこで，非同期で動作させるため，非同期 SAIGA では，high level GA に定常状態 GA の世代交代モデルを導入する（図 5 の 11〜31 行目）．定常状態 GA の世代交代モデルでは，1 世代ごとに，個体群に属するすべての個体を新しい個体に置き換えるのではなく，一部の個体のみを置き換える．これを導入することにより，パラメータベクトルとその評価値のペアを一部の島から受け取るだけで，進化させることができ，そして，即座に進化したパラメータベクトルを返すことができるようになる．そのため，非同期で動作させることができようになる．その結果，遊休状態である時間を減少させることが期待できる．

3.2.3　FIFO 削除法の導入

定常状態 GA の導入により，個体群の中には，最近評価された個体と昔評価された個体とが混在することになる．パラメータベクトルを最適化させる問題のような動的環境下における最適化問題では，不都合が生じる．評価値のみをもとにした選択方法だけでは，昔高い評価値が与えられた個体は，仮に環境の変化により評価関数による現在の評価値が下がっても，一度与えられた評価値をもとに，生き残り続けることになる．

この問題点に対し，本論文では，De Jong らの FIFO（First In First Out）削除法 [7] を導入する（図 5 の 15 行目）．これは，個体群の個体数が一定の数以上になると，最も古い個体から捨てる手法である．これにより，変化する環境に適応できるようになる．

3.2.4　非同期で動作させた場合の個体群の状態

探索効率が良いパラメータベクトルは，個体群の状態に依存する．島間で個体群の状態を類似させることができれば，探索効率が良いパラメータベクトルは，島間で類似させることができ，すべての島に対し同条件でパラメータベクトルを生成させることができると考える．SAIGA では，同期をとることで島間で均等に評価回数を与えることと島間で移民させることにより，島間の個体群の状態を類似させている．ところで，非同期 SAIGA では，上記の 2 条件のうち同期という条件がない．しかし，島間で与えられる評価回数が異なる非同期の島 GA でも，移民頻度が高い環境では，

個体群の状態が互いによく似てくることが知られている [6]．このため，非同期 SAIGA において，高い移民頻度を保つことができる通信環境であれば，有用であると考えられる．

4. 評価実験

4.1 実験条件

提案手法の性能を評価するため，以下の条件で実験した．

high level GA の遺伝演算子として，ガウス分布を用いた突然変異，パラメータごとの交叉，ルーレット選択を用いた．また，high level GA のパラメータ設定として，交叉率を 0.8，突然変異率を 0.6，スケーリング率を 1 対 5，**island_number** を 10 とした．提案手法の low level GA には，単純 GA を用いる．これは単純 GA がモデルが最も単純であり，分析のために適していると考えたからである．適応させるパラメータは，交叉率，突然変異率，トーナメントサイズ，個体数の四つである．これは，比較対象とする De Jong の標準的パラメータに淘汰圧に関するパラメータを加えたものである．high level GA の個体の符号化方法については，付録に示す．

評価のために扱った問題は，Rastrigin 関数の最小化問題，巡回セールスマン問題（Traveling Salesman Problem，以下 TSP）の lin105 [8]，だまし問題，ジョブショップスケジューリング問題（Job-Shop Scheduling Problem，以下 JSP）の la38 [9] である．各試行回数は 300 回である．各試行において，解の評価回数が 6.4×10^6 回に達するまで測定をした．

比較対象として，A-SAGA，De Jong の標準的パラメータの設定を用いた単純 GA，人手により調整されたパラメータの設定を用いた単純 GA，De Jong の標準的パラメータの設定を用いた島 GA と人手により調整されたパラメータの設定を用いた島 GA を用いた．ただし，標準的パラメータ設定には選択に関する情報が含まれていないため，サイズ 2 のトーナメント選択を用いた．

また，SAIGA と非同期 SAIGA において，一時代分の処理時間を実測した．その際，先ほどの条件のもと，400 MHz から 2.6 MHz のプロセッサを 10 台用いて動作させた．

4.2 評価結果

評価結果について考察する．

評価結果を表 1 に示す．以後，DeJong の標準的パラメータを用いた単純 GA を SGA（DeJong），人手により調整されたパラメータを用いた単純 GA を SGA（tuned），DeJong の標準的パラメータを用いた島 GA を IGA（DeJong），人手により調整されたパラメータを用いた島 GA を IGA（tuned）と表記する．

表 1　評価結果
Table 1　Evaluation results.

	Rastrigin 関数		だまし問題	
	最適値 0		最適値 0	
	best	mean	best	mean
SGA (Dejong)	0	0	1	3.5
SGA (tuned)	0	0	2	3.3
IGA (Dejong)	0	0	0	2.26
IGA (tuned)	0	0	0	0.25
A-SAGA	10	20.52	0	0.66
SAIGA	0	0	0	0.47
非同期 SAIGA	0	0.02	0	0.53
	TSP (lin105)		JSP (la38)	
	最適値 14379		最適値 1196	
	best	mean	best	mean
SGA (Dejong)	14434	15705	1216	1289
SGA (tuned)	14475	15175	1203	1227
IGA (Dejong)	14483	15489	1226	1261
IGA (tuned)	14401	14932	1205	1218
A-SAGA	25197	28366	1228	1249
SAIGA	14464	15391	1205	1242
非同期 SAIGA	14486	15336	1207	1234

4.2.1 提案手法と A-SAGA との比較

いずれの問題においても提案手法は，A-SAGA に比べ，良い評価結果を示した．A-SAGA は，反復の必要があるために，この実験において，初期化と 6.4×10^4 回の評価回数分の解の探索を 100 回繰り返しているのに対し，提案手法では，初期化は 1 回のみで 6.4×10^6 回の評価回数分の解の探索を続けて行う．この差が性能差となって現れたと考えられる．

4.2.2 単純 GA と島 GA の比較

TSP，JSP において，提案手法は，SGA（DeJong）と IGA（DeJong）に比べ，良い評価結果を示した．しかし，Rastrigin 関数において，SGA（DeJong）と IGA（DeJong）は，提案手法に比べ，良い評価結果を示した．これらの結果から，Rastrigin 関数のような多峰性関数を解く際には，De Jong のパラメータは有用であるが，TSP，JSP などの複雑な符号化法，遺伝演算子を用いる場合には，提案手法の方が有用であることが分かった．

だまし問題においては，IGA（tuned）の方が，SGA（tuned）に比べ，評価結果が極めて良い．このことから，だまし問題においては，島 GA の方が単純 GA に比べ優れていることが分かる．これは，IGA（tuned）

では，島構造をもつため島ごとに見つけられた正しい遺伝子構造を組み合わせることができるためであると考えられる．しかし，島構造をもつ IGA（DeJong）の解の探索結果の平均値は，0 から離れた 2 付近の値となった．これは，De Jong のパラメータがだまし問題を解くためには不向きであるためであったと考えられ，このことから不適切なパラメータが与えられらた場合，島 GA であっても，評価結果が悪くなることが分かる．提案手法の解の探索結果の平均値は 0 に近い値となった．このことから提案手法では，適切なパラメータ値を得ることができたと考えられる．

Rastrigin 関数，TSP，JSP においても提案手法は，SGA（tuned），IGA（tuned）に迫る評価結果を示した．

以上より，提案手法は，多様な問題に対して適切なパラメータ値を求めることができ，人手により調整されたパラメータ値を用いた単純 GA に迫る性能を示すことを確認した．

4.2.3 SAIGA と非同期 SAIGA の比較

TSP と JSP 以外では，提案手法は，ほぼ同等の評価結果を示した．TSP と JSP においては，非同期 SAIGA が SAIGA と比べ良い評価結果を示している．この原因として次のことが考えられる．SAIGA では，時間当りにすべての島に均等に評価回数が与えられる．しかし，非同期 SAIGA では，不均等に与えられる．そのため，多くの評価回数を偏って与えられる島が存在する．つまり，非同期 SAIGA では，処理能力の高い島は，他の島より多くの評価回数が与えられる．そのため，局所的に探索される．TSP と JSP においてこの局所的な探索が有効であったため，このような結果になったと考えられる．

4.3 処理時間の比較

SAIGA と非同期 SAIGA において，一時代分の処理時間を計測した．その結果を表 2 に示す．表 2 より，いずれの問題においても非同期 SAIGA は，SAIGA より短い時間で処理できることを示した．

これらの結果から，非同期 SAIGA において，計算機の遊休状態である時間を減らすことができたと考えられる．実験結果を見ると，非同期 SAIGA の処理時間は SAIGA のそれに比べ，1/3 程度であるが，だまし問題の場合は 2/3 であり，それらとは異なっている．これはだまし問題の場合，他の問題に比べ，評価にかかる時間が極端に短いためであると考えられる．これを確かめるため，だまし問題の評価関数の部分にダミー処理を加え，その処理の回数を増やすことで評価にかかる時間を増やし，検証を行った．その結果を表 3 に示す．

表 2 一時代分の処理時間の比較（ms）
Table 2 Comparison of processing time in one era. (ms)

	SAIGA	非同期 SAIGA
Rastrigin	6.6	2.3
だまし問題	1.4	0.9
TSP	16.4	4.1
JSP	111.6	37.2

表 3 処理の回数と処理時間の比率
Table 3 Number of function calls and ratio of processing time.

処理の回数	0	1	10	100	1000	10000	100000
比率	0.47	0.37	0.35	0.38	0.38	0.26	0.24

実験の結果，ダミー処理の負荷を増やすとともに処理時間の差が大きくなっている．このことから，処理時間の比率に対するだまし問題と他の問題との違いは，評価関数の処理時間の違いにあると考えられる．

今回の実験では，いずれの場合においても，非同期 SAIGA は，SAIGA より処理時間を短くすることができた．ただし，本文の表 1 にあるように Rastrigin 関数の一部では，解の探索効率が悪化していた．これは，時間当りに評価回数が不均等に与えられるためであると考えられるが，今後，これについて調べていきたい．

4.4 適応させるパラメータの数

Rastrigin 関数に対して，SAIGA を用いて，適応させるパラメータを六つに増やし実験を行った．

増やしたパラメータは，選択方式（ルーレット方式かトーナメント方式か）を示すパラメータ，及びルーレット方式が選ばれているときに用いる淘汰圧に関するパラメータである．以下，これを T&R6 と表記する．比較対象として，トーナメント方式に固定した場合（以下，T4）とルーレット方式に固定した場合（以下，R4）について実験を行った．これらの場合に適応させるパラメータの数はそれぞれ四つとなる．解の評価回数が 2.56×10^6 に達するまで測定をし，30 試行の平均をとった．その結果の解の探索効率は，T4 が 0.033，R4 が 0.565 で T&R6 が 0.133 となった．

T4 と R4 を比べた場合，T4 の方が良い結果を示した．これより Rastrigin 関数に対しては，トーナメント方式の方が適していると考えられる．T&R6 は R4 に比べ良い結果を示し，T4 に近い結果を示した．こ

表 4　提案手法により得たパラメータ
Table 4　Parameter obtained using proposed method.

	mu	cr	sc	po	mu	cr	sc	po
DeJong	0.001	0.6	2	50				
Grefenstette	–	0.5	-	50				
	Rastrigin 関数				TSP			
tuned	0.003	0.58	4	4	0.01	0.55	4	16
SAIGA	0.014	0.59	3	12	0.016	0.53	5	49
非同期 SAIGA	0.009	0.62	4	40	0.014	0.50	4	46
	だまし問題				JSP			
tuned	0.2	0.5	2	256	0.02	0.55	4	256
SAIGA	0.064	0.50	5	58	0.029	0.49	5	45
非同期 SAIGA	0.062	0.53	5	62	0.051	0.47	5	49

表 5　島の数と解の探索効率
Table 5　Number of islands and search efficiency.

島の数		10	20	50	100
計算時間（分）		1	2	5	10
SAIGA	Rastrigin	1.067	0.433	0.167	0.133
非同期 SAIGA	Rastrigin	0.400	0.133	0.067	0.067
SAIGA	だまし問題	1.406	0.173	0.000	0.000
非同期 SAIGA	だまし問題	0.000	0.000	0.000	0.000
SAIGA	TSP	15618	15283	14754	14627
非同期 SAIGA	TSP	15254	15092	14662	14532
SAIGA	JSP	1294	1287	1267	1256
非同期 SAIGA	JSP	1248	1242	1238	1237

れは 6 パラメータの場合において，選択に関するパラメータを適応させることができたことによると考えられる．

ただし，パラメータベクトルの組合せ空間が広がると，解の探索効率が落ちることが予想される．T&R6 が T4 と同じ結果を示さなかったのはこのためであると考えられる．

4.5　提案手法によって得られたパラメータ

SAIGA と非同期 SAIGA によって得られたパラメータの値を評価するために，それぞれ解の収束した付近のパラメータを 300 回試行の平均を計測した．

これらと De Jong の標準パラメータ，人手により調節したパラメータと文献 [10] にある Grefenstette のパラメータを表 4 に示す．表 4 において，突然変異率を mu，交叉率を cr，トーナメントサイズを sc，個体数を po で表記する．Grefenstette のパラメータは，個体数が 50 のときの交叉率を引用した．

この表 4 より，SAIGA と非同期 SAIGA によって得られたパラメータを見ると SAIGA と非同期 SAIGA はこれらの問題に対して，tuned の値に近いパラメータを見つけることができた．また，Rastrigin 関数以外の問題では，個体数が 50 付近となったが交叉率が Grefenstette の値と近い値になった．したがって，SAIGA と非同期 SAIGA はこれらの問題に対して，パラメータの適応ができたと考えられる．

4.6　島　の　数

提案手法のスケーラビリティについて検証するため，島の数を変化させ，実験を行った．実験ごとに計算時間を調整することで，島の数と同数の計算機を動作させた場合と同等の計算量を擬似的に確保した．その結果を表 5 に示す．

SAIGA と非同期 SAIGA は，島の数が増加とともに，解の高い探索効率を示した．

今回の実験では確認できなかったが，島の数が増加すると通信部分がボトルネックになることが考えられる．これに対して，high level GA にセルラ GA の構造を用いることで通信負荷を分散させることなどが考えられる．

5.　む　す　び

本論文では，A-SAGA で行っていた反復をなくすことにより，A-SAGA より少ない計算量で多数のパラメータを適応させる SAIGA を提案した．

更に，非同期で動作することにより，時間単位の探索効率が向上したアルゴリズムである非同期 SAIGA を提案した．

提案手法の有効性を調べるため，多様な問題に対して解の探索性能を計測した．交叉率，突然変異率，トーナメントサイズ，個体数の四つのパラメータを適応させた．実験の結果，提案手法は，多様な問題に対して適切なパラメータ値を求めることができ，人手により調整されたパラメータ値を用いた単純 GA に迫る性能を示した．

更に，非同期 SAIGA は，遊休状態である時間を削減することにより，時間単位の探索効率を向上することができた．

今後の課題として，二つのことが挙げられる．

一つ目は，非同期 SAIGA において，島ごとのプロセッサ資源の割当を自動的に最適化させることを考えている．今回の多くの評価実験において，非同期 SAIGA は，SAIGA より高い性能を示した．これは，島ごとのプロセッサの資源について，均等に割り当てるより不均等に割り当てた方が良い場合があることを示している．そこで，非同期 SAIGA において，島ごとのプロセッサ資源の割当が解の探索性能に与える影響を調べ，その結果をもとに，島ごとのプロセッサ資

源の割当を自動的に最適化させることを考えている．

二つ目は，FIFO 削除法の代わりに，aGA [11] の年齢モデルを導入することである．aGA の年齢モデルでは，評価値と個体の古さの両方を考慮して選択する．このことにより，より動的な環境に適応できると考えられる．

三つ目は，提案手法の low level GA に，MGG [12] モデルなど他の世代交代モデルを導入することである．今回の実験では，low level GA には単純 GA を用いたが，MGG モデルのような世代交代モデルを導入することもできる．どのような世代交代モデルを用いるのが適切か，また，適用させることができるのか，今後，これについて実験を行っていきたい．

文　献

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

[2] D.E. Goldberg, K. Deb, and D. Thierens, “Towards a better understanding of mixing in genetic algorithms,” Technical Report IlliGAL No.92009, University of Illinois, 1992.

[3] 村田佳洋，柴田直樹，伊藤　実，“エージェント指向自己適応遺伝アルゴリズム，” 情処学論，数理モデル化と応用，vol.44, SIG7(TOM8), pp.61–68, 2003.

[4] R. Weinberg, Computer simulations of a living cell, Ph. D. Thesis, University of Michigan, 1970.

[5] E. Alba and J.M. Troya, “An analysis of synchronous and asynchronous parallel distributed genetic algorithms with structured and panmictic islands,” Proc. 11th IPPS/SPDP’99, pp.248–256, 1999.

[6] J. Berntsson and M. Tang, “A convergence model for asynchronous parallel genetic algorithms,” CEC 2003, pp.2627–2634, 2003.

[7] K.A. De Jong and J. Sarma, “Generation gaps revisited,” in Foundations of Genetic Algorithms 2, pp.19–28, Morgan Kaufmann, 1993.

[8] “TSPLIB,” http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

[9] S.R. Lawrence, “Resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques (Supplement),” Working Paper, Graduate School of Industrial Administration, Carnegie-Mellon University, 1984.

[10] J. Grefenstette, “Optimization of control parameters for genetic algorithms,” IEEE Trans. Syst. Man Cybern., vol.16, no.1, pp.122–128, 1986.

[11] A. Ghosh, S. Tsutsui, and H. Tanaka, “Function optimization in nonstationary environment using steady state genetic algorithms with aging of individuals,” Proc. 1998 IEEE, ICEC, pp.666–671, 1998.

[12] 佐藤　浩，小野　功，小林重信，“遺伝的アルゴリズムにおける世代交代モデルの提案と評価，” 人工知能誌，vol.12, no.5, pp.734–744, 1997.

付　録

high level GA の個体を，次のように符号化する．今回の実験で，適応させるパラメータの数は四つであるから，high level GA の個体 a の染色体 $a.\vec{x}$ の要素数は四つである．$a.\vec{x}$ の各要素 $a.x_i$ は，0 から 1 までの 16 bit の固定小数点で表される $(0 < a.x_i < 1, a.x_i \in \Re)$．

各 $a.x_i$ は，それぞれ以下の関数を用いて，パラメータ空間を定義する．

$$n_i = \lfloor 2 \cdot \exp(8 \cdot a.x_1 \cdot \log(2)) \rfloor.$$

$$s_i = \begin{cases} \lfloor a.x_2 \cdot 8 + 2 \rfloor & \text{if } n_i > \lfloor a.x_2 \cdot 8 + 2 \rfloor, \\ 2 & \text{if } n_i \leq \lfloor a.x_2 \cdot 8 + 2 \rfloor. \end{cases}$$

$$c_i = a.x_3.$$

$$m_i = 0.00005 \cdot \exp(a.x_4 \cdot \log(1/0.0001)).$$

ただし，変換後の個体数，トーナメントサイズ，交叉率，突然変異率をそれぞれ n_i，s_i，c_i，m_i とする．

（平成 16 年 4 月 30 日受付，17 年 1 月 11 日再受付）

高島　栄一

現在，奈良先端科学技術大学院大学情報科学研究科博士後期課程に在学中．

村田　佳洋　（正員）

2003 奈良先端科学技術大学院大学情報科学研究科博士後期課程了．現在，奈良先端科学技術大学院大学情報科学研究科助手．遺伝的アルゴリズム，エージェント技術等の研究に従事．

柴田　直樹

2001 大阪大学大学院基礎工学研究科情報数理系専攻博士後期課程了．現在，滋賀大学経済学部情報管理学科助教授．分散協調，遺伝的アルゴリズム，形式的設計検証等の研究に従事．

伊藤　　実　（正員）

1977，1979，1983 にそれぞれ大阪大学基礎工学部卒，基礎工学研究科博士前期課程了，基礎工学研究科博士後期課程了．1979 より大阪大学基礎工学部助手．1986 より大阪大学基礎工学部講師．1989 より大阪大学基礎工学部助教授．1993 年 4 月より現在，奈良先端科学技術大学院大学情報科学研究科教授．関係データベース理論，オブジェクト指向のデータベースのアプリケーション，DNA プローブ等の研究に従事．ACM，IEEE 各会員．