集団分割型非同期並列遺伝的アルゴリズムにおける個体交換アルゴリズムの改良と評価

棟朝雅晴† 高井昌彰† 佐藤義治†

本論文では集団分割に基づく並列遺伝的アルゴリズムにおいて，効率的な個体交換を行う交換アルゴリズムを提案する．集団分割による並列遺伝的アルゴリズムは，個体からなる集団をいくつかの部分集団に分割し，それぞれを並列計算機のプロセッサに割り当てて遺伝的アルゴリズムを実行することにより，中粒度の並列処理を実現するものである．この手法においては集団の一様化による探索効率の減少を防ぐために部分集団間で通信ネットワークを介した個体交換を行う必要がある．マルチプロセッサシステムにおいてプロセッサ間通信量を減少させることがその性能を向上させる上で重要であるが，並列遺伝的アルゴリズムに関する従来の研究では，個体の交換がその必要性とは関わりなく一定世代ごとまたは一定確率で行われており，並列処理の効率が悪いと考えられる．本論文で提案する個体交換アルゴリズム Sigma-Exchange は各部分集団内の適合度分布を観測し，適合度分布の標準偏差の値が一定割合減少した場合にのみ交換の手続きを起動することにより，少ないプロセッサ間通信でより精度の高い解を速く得ることを目的としている．提案する手法の有効性を示すために，非同期のメッセージ受渡しによる中粒度並列計算機であるマルチコンピュータネットワークを前提としたシミュレーション実験を行った．その結果，代表的な組合せ最適化問題について，提案する手法が有効であることが示された．

An Efficient String Exchange Algorithm for a Subpopulation-Based Asynchronously Parallel Genetic Algorithm and Its Evaluation

MASAHARU MUNETOMO,† YOSHIAKI TAKAI† and YOSHIHARU SATO†

We present an efficient string exchange scheme on subpopulation-based parallel genetic algorithms. The subpopulation-based parallel genetic algorithm divides a population into subpopulations in which genetic operations are executed simultaneously. In this scheme, exchanging strings between subpopulations through communicating network is essential to avoiding performance degradation of genetic search due to uniformity of the subpopulation. To reduce unnecessary inter-processor communications is an important issue to realize efficient parallel computation. In conventional subpopulation-based parallel genetic algorithms, however, inefficient parallel computations are taken place because string exchanges are executed at a constant interval or fully at random. The sigma-exchange algorithm we propose observes a fitness distribution of each subpopulation and starts on exchanging strings only when the standard deviation of the fitness distribution of some subpopulation decreases to some ratio. The purpose of our algorithm is to obtain more precise solutions with less inter-processor communications. We show the effectiveness of our scheme through simulation experiments on a multicomputer network in which communications are realized via asynchronous message passing.

1. はじめに

遺伝的アルゴリズムは生物個体の集団における遺伝システムをもとにした近似最適化手法であり，複数の個体を用いて最適解を探索する確率的な多点探索の一手法である．このアルゴリズムは従来の最適化手法に比べ，問題に対して要請される条件が緩やかであるにもかかわらず，比較的良い近似解を得られるという特徴があるため，これまで関数最適化，組合せ最適化問題，ガスパイプラインの制御，機械学習など数多くの問題に対して適用されてきた．特に近年は，その応用分野の拡大に伴い良質の近似解をより高速に得るために，遺伝的アルゴリズムの並列化に関する研究が注目されている．

遺伝的アルゴリズムを並列化する最も自然な方法

† 北海道大学工学部情報図形科学講座
Information and Graphics Sciences, Faculty of Engineering, Hokkaido University

は，個体の集団を分割する手法である．この手法は複数個体からなる集団をいくつかの部分集団に分割し，それらを各プロセッサに割り当てることで空間並列的に遺伝的アルゴリズムを実行するものである．このとき，部分集団の一様化による急速な局所解への収束を防ぐために部分集団間での個体の相互交換が必要となる．従来，集団分割に基づく並列化の研究においては，この交換が一定確率，または一定単位時間ごとに起動されている．並列アルゴリズムをマルチプロセッサシステムに実現する場合に，そのボトルネックとなるのはプロセッサ間通信であるが，従来の手法では必要かどうかを観測することなしに個体の交換を起動しているので，並列処理の効率が悪いと考えられる．

本論文では集団分割による並列遺伝的アルゴリズムにおいて，できる限り単純な仕組みで効率的なプロセッサ間通信を実現する個体交換アルゴリズム Sigma-Exchange を提案する．このアルゴリズムは各部分集団の適合度の分布を観測し，集団内の個体の多様性が減少した場合にのみ個体交換を実行することにより，少ないプロセッサ間通信で高速な探索を実現する．このアルゴリズムを非同期通信による中粒度並列計算機のモデルであるマルチコンピュータネットワークに実現した場合を想定したシミュレーションを行い，従来の交換アルゴリズムと比較した場合の優位性を確認した．

2. 並列遺伝的アルゴリズム

遺伝的アルゴリズム (Genetic Algorithms，以下 GA と略す)[1] は，生物の遺伝システムの働きをもとに考案された多点探索手法であり，文字列として符号化された個体からなる集団に対して，基本3操作である Selection, Crossover, Mutation を繰り返し用いることで近似最適解を求める．GA は 1960 年代に J. H. Holland らによって提唱されて以来，関数最適化問題，巡回セールスマン問題に代表される組合せ最適化問題，ガスパイプラインの制御問題，機械学習など多くの問題に適用されてきた[2]．本章では GA の並列化について述べる．

GA の並列化に関するこれまでの研究を大きく分類すると，個体を格子などの構造上に配置することで細粒度の超並列計算を行うもの[3)~5)]，集団をいくつかの部分集団に分割し，その間で個体の交換を行うことで中粒度の並列計算を達成するもの[6)~9)]，局所探索や疑似焼きなまし法と組み合わせたもの[10),11)] に大別される．

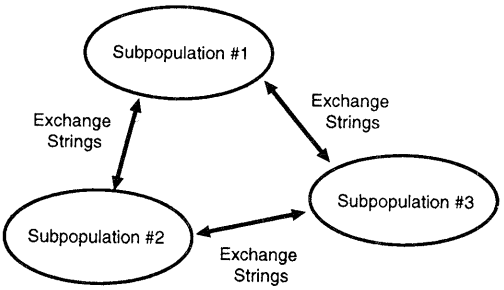


図 1 集団分割型の並列 GA の概念図
Fig. 1 Subpopulation-based parallel GA.

集団分割による方法は最も自然な並列化であり，その実現が容易であることから多くの研究がなされている．集団分割型の並列 GA について，その概念図を**図 1** に示す．この方法では，集団をいくつかの部分集団 (Subpopulation) に分割し，各部分集団にプロセッサを割り当てる．各プロセッサではそれぞれの部分集団に対して独立に GA を実行するとともに，一様化を防ぐために部分集団間で一定割合の個体を交換 (Exchange) する．Tanese[8] は一定世代ごとに交換を起動する集団分割型の並列 GA を，Hypercube 構造を持つ中粒度並列の並列計算機である NCUBE 上に実現し，Walsh 関数の最適化を行っている．同様の並列 GA を，K-Partition Problem に適用した例についても報告されている[6]．また，Pettey[7] は理論的解析を行うために，交換を一定確率でランダムに起動するアルゴリズムを示している．以上の研究では交換が起動されると，ランダムに交換相手となる部分集団を選択し，一定割合の個体がランダムに選択される．以下，本論文では一定世代ごとまたは一定確率で交換を行うアルゴリズムを Random-Exchange と呼び，提案する手法の比較対象とする．

3. 交換アルゴリズムの設計

集団分割型の並列 GA において，部分集団間で個体の交換を行う場合の基準として以下の3つがあげられる．

1. いつ，交換の手続きを起動するか．
2. どの部分集団を交換の相手とするか．
3. どの個体を交換対象とするか．

本章では，この3つの基準をもとに従来の手法の問題点を指摘し，部分集団における個体の多様性の維持と並列処理の観点から効率的な交換アルゴリズム

(Sigma-Exchange と呼ぶ) を設計する．GA の改良を行う場合に，Goldberg[12] が言及しているように，アルゴリズムの複雑化が必ずしも性能の向上につながるとは限らず，過度の制御によりかえってGA本来の頑健性を損なう場合が多い．そこで本研究では実現の容易性も考慮し，できる限り単純なアプローチをとることとした．

3.1 集団の一様化と交換の起動条件

各部分集団において，集団の一様化を検出し，交換を能動的に起動するというのが提案する手法の要点である．従来行われてきた集団分割型の並列 GA では，交換の必要性を全く観測せずに，一定世代ごともしくは一定確率で交換が起動されていた．交換が必要なのは部分集団に含まれる個体が一様化した場合だけであるので，何らかの手法で多様性の減少を観測しこれを交換の起動条件とすることが効率的な交換すなわちプロセッサ間通信を実現することにつながる．

GA はスキーマの組合せにより探索を行うアルゴリズムである[1]．集団内に含まれる相異なるスキーマの数が減少すると，組合せによって新しいスキーマが生まれる可能性が少なくなり，Crossover による探索の効率が低下する．集団分割型の並列 GA では，部分集団に含まれる個体数が相対的に少なくなるため，特に局所解への収束が速くなる傾向がある．集団の一様化の指標として相異なるスキーマの数を直接数え上げることは計算量の観点から非常に効率が悪い．また，Lost alleles, Percent involvement, エントロピーなど集団の多様性を測る指標がいくつか提案されているが，これらの指標は非常に多くの計算量を必要とする[13]．

そこで，集団の一様化と集団内に存在する個体の適合度分布に関する標準偏差の減少が同時に起こるという仮説 Sigma-Hypothesis[14),15)] を採用し，集団の一様化を知るための指標として適合度の分布の標準偏差を用いることとする．この仮説の理論的証明に関しては，適合度関数や文字列への符号化の方法（コーディング）に強く依存するため困難であると考えられるが，Selection のみの場合について証明がなされている[15]．また，標準偏差は変化した部分に関して差分として計算できるので，指標の計算量が少ないという利点がある．

理想的には，基準となる標準偏差を，落ち込んでしまった局所解ごとに定めることも考えられるが，そのためには解空間の様子など問題の性質があらかじめ十分に知られていることが必要である．そこで提案する手法では，標準偏差の初期値に比べて現在の値がある一定割合減少した場合に，その部分集団が一様化したと判断して他の部分集団と個体の交換を行う．さらに，交換終了後に標準偏差の初期値を再計算する．この方法により，標準偏差値の相対的な減少割合を検出し，それを用いて交換を起動している．

3.2 部分集団の選択

交換相手となる部分集団の選択に関しては，集団の多様性を回復するという交換の目的から，できるかぎり異なった分布を持つ部分集団を選択することが望ましい．分布間の離れ具合（距離）を求める場合，厳密にはそれぞれの分布の差を全体について計算する必要が生じるが，そのためには適合度分布すべての情報を部分集団間で交換する必要があり，非常に多くのプロセッサ間通信が全体の効率を低下させる．つまり，交換の相手となる部分集団を選択する場合に，(1)分布間の距離の計算の正確さと(2)必要とする通信量の間にトレード・オフが存在する．そこで，分布を代表するいくつかの指標を用いるのが妥当であると考えられる．本論文では指標として適合度分布の平均 $\bar{f}$ と標準偏差 σ を用いた．適合度の分布に関して正規分布などを仮定することは困難であるが，平均と標準偏差はおおまかな分布をしめす指標として用いることが可能である．

それぞれの部分集団は，その適合度の分布に関して，平均と標準偏差の値を内部状態として持っている．この情報をプロセッサ（部分集団）間でいつ伝達するかについては，以下の3つの方針が考えられる．

1. 交換を行う時，すなわち他の内部状態の情報が必要になった時に要求を伝達する．
2. 定期的にそれぞれの内部状態の情報を伝達する．
3. 内部状態が変更になった時，すなわち分布が変化した時に新しい内部状態の情報を伝達する．

GA では集団に基本3操作を加えるたびに適合度分布が変化しうるので，方式3で必要な通信量は膨大である．また，方式2では通信の周期が長いと内部状態の情報が不正確となる．したがって，ある部分集団で交換が起動され，平均と標準偏差の値が必要になった時点でのみ，それらの値を要求する方式1が妥当である．また，情報の要求先は全部分集団が理想的であるが，通信のオーバーヘッドを考慮し，ある限られた近傍の部分集団のみとする．

交換を起動した部分集団（適合度の平均 $\bar{f}$，標準偏

差 σ) は，近傍の部分集団すべてから，平均 $\bar{f}'$ と標準偏差 σ' の値を得た後，$d=(\bar{f}-\bar{f}')^2+(\sigma-\sigma')^2$ をそれぞれについて計算し，d の値が最も大きな部分集団を交換相手として選択する．

3.3 個体の選択

部分集団の中で，交換対象となる個体はランダムに選択される．適合度の高い個体はその値に応じて多く存在するので，ランダムに選択することはもう一度 Selection をかけることに相当する．

4. インプリメンテーション

本章では提案する個体交換アルゴリズムである Sigma-Exchange および従来法の Random-Exchange のマルチコンピュータネットワークへの実現について述べる．

4.1 マルチコンピュータネットワーク

マルチコンピュータネットワークは，あるトポロジーに従って結合された多数のノードからなる疎結合の並列計算機である．各ノードは，ローカルバスにプロセッサとローカルメモリ，通信コントローラが接続された構造を持つ．通信コントローラは，他のノードの通信コントローラとつながっており，内部にはバッファが存在する．プロセッサ間通信は非同期のメッセージ受渡しで実現され，通信遅延が比較的大きい通信ネットワークを想定している．

また，通信の手段としてはユニキャストを用い，個別に近傍のプロセッサに対してメッセージを送っている．他の通信手段としては，すべてのプロセッサに対して同時にメッセージを送るブロードキャストやいくつかのプロセッサに対して同時に送るマルチキャストが考えられるが，それらの手法が実際のシステム上に実現されている場合でも，それぞれのプロセッサが一定時間後に厳密な意味で同時にメッセージを受けとるという保証はない．本論文ではメッセージの送出数と通信負荷を明確に対応づけるため，アルゴリズムの評価を行う上でユニキャストを前提とした．

4.2 Sigma-Exchange

このアルゴリズムは Sigma-Exchange と Message-Handling-Sigma の2つの手続きにより構成されている．各プロセッサでは Sigma-Exchange が並列に実行される．他のプロセッサからメッセージが送られてくると，割り込みにより Message-Handling-Sigma が起動され，受信メッセージ処理を行う．

Sigma-Exchange を**図2**に示す．はじめに部分集団を初期化し，適合度分布の標準偏差の初期値 σ_0 を計算する．Selection を実行した後，標準偏差の値が初期値に比べてある一定割合減少した場合，すなわち $\sigma<\lambda\cdot\sigma_0$ $(0<\lambda<1)$ を満たした時に個体交換を起動する．条件が成立すると，まずパラメータ（部分集団の適合度分布の平均値と標準偏差の値）の要求メッセージをネットワークにおける近傍のすべてのプロセッサに送り，パラメータが送られてくるのを待つ．ここで近傍とは，あるプロセッサから中継なしに直接接続されているプロセッサの集合である．パラメータがすべて揃ったなら，交換を行う部分集団を決定し，その部分集合へ交換の要求メッセージを送る．もし，相手の部分集団が交換可能，すなわち他のプロセッサと交換を行っている最中でなければ，交換される個体群

```
procedure Sigma-Exchange
constant λ ;

begin
 Initialize;
 Calculate σ0 ;
 while not terminate
  begin
   Selection;
   if σ < λ·σ0 then
    begin
     Send Parameter_requests;
     Wait for Parameters;
     Select a subpopulation to exchange;
     Send Status_request to the subpopulation;
     Wait for Status;
     if status = Accept then
      begin
       Get Migrants;
       Send Migrants;
       Recalculate σ0 ;
      end;
    end;
    Crossover;
    Mutation;
  end;
end.
```

図2 Sigma-Exchange アルゴリズム
Fig. 2 Sigma-Exchange algorithm.

(Migrants) が送られてくるのでそれを受けとり，さらにこちらからも個体群を送り返し，標準偏差の初期値 σ_0 を再計算する．交換が終了したなら，Crossover と Mutation を行う．以上を終了条件が満たされるまで繰り返す．

次に，Message-Handling-Sigma を**図3**に示す．受けとったメッセージの内容が Parameter_Request の場合には部分集団の現在の適合度分布の平均値と標準偏差の値を送る．また，Status_Request の場合，個体交換を行うことが可能であれば Accept に続いて交換対象となる個体群を送り，交換を行えない場合（他の集団に対してすでに個体交換を要求している場合）には交換不可能であるという Reject メッセージを送り返す．Migrants の場合には，送られてきた個体群を受けとり，標準偏差の初期値 σ_0 を再計算する．

```
procedure Message-Handling-Sigma

begin
 if message_type = Parameter_Request then
  begin
   Send_Parameter;
  end;
 else if message_type = Status_Request then
  begin
   if migrating then Send Reject
   else Send Accept with Migrants;
  end;
 else if message_type = Migrants then
  begin
   Get_Migrants;
   Recalculate σ0 ;
  end;
end.
```

図3 Sigma-Exchange における受信メッセージの処理
Fig. 3 Message handling for Sigma-Exchange.

通信の手順を**図4**に示す．斜線の入った黒丸は個体交換を起動したプロセッサ，白丸はその近傍のプロセッサである．近傍のプロセッサでは Message-Handling-Sigma の手続きにより受信メッセージに対する処理が実行される．ここで，提案するアルゴリズムが正常に動作するためには，Parm_send が Parm_request の送出順に戻らなくても良いが，送ったメッセージに関する返答はすべて戻ってくる必要がある．

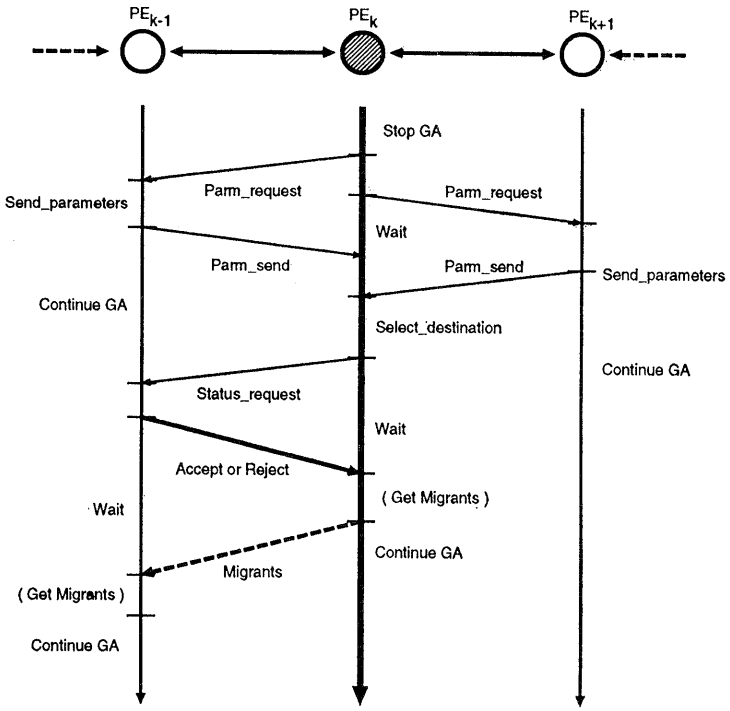


図4 Sigma-Exchange における個体交換の手順 (PE$_k$ が交換を起動した場合)
Fig. 4 Exchanging process in Sigma-Exchange (where PE$_k$ invokes an exchange).

4.3 Random-Exchange

比較対象として，従来の個体交換アルゴリズム Random-Exchange を**図5**に示す．このアルゴリズムでは，あらかじめ定められた確率 p_exchange に従って個体交換がランダムに起動される．受信メッセージの処理 Message-Handling-Random については**図6**，個体交換の手順に関しては**図7**に示されるとおりとなる．交換が起動されると，交換相手となる集団をランダムに選び，Status_request メッセージを送る．それを受けとった部分集団では交換が可能である場合には Accept メッセージを交換対

象となる個体群とともに送る．また，他の集団との交換が進行中で，新たな交換の受け入れが不可能な場合には Reject メッセージを送り返す．Accept メッセージを受けとった場合には，個体群を受けとり，一定割合の個体をランダムに選択し送り返す．

5. シミュレーション実験

本章では，集団分割型の並列 GA のシミュレーションによる実験結果について述べる．UNIX ワークステーション上にマルチコンピュータネットワークのシミュレータを開発し，これを用いて実験を行った．評価の対象となる問題として，単峰性関数である De Jong のテスト関数 F1，多峰性関数である F5，だまし関数である Trap function[16] の最大化問題，代表的な組合せ最適化問題である巡回セールスマン問題 (Traveling Salesman Problem, TSP) とナップザック問題を採用した．

De Jong のテスト関数 F1 は以下の式で定義される関数である[2]．

$$f_1(x_1, x_2, x_3) = \sum_{i=1}^{3} x_i^2, \qquad (1)$$

$$(-5.12 \leq x_i \leq 5.12).$$

F5 は以下に示される関数である[2]．

$$f_5(x_1, x_2) = 0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}, \qquad (2)$$

$$(-65.536 \leq x_i \leq 65.536).$$

これらの関数については，{0,1} の 2 つの文字からなる 2 進数による符号化を行った．F1 についてはそれぞれの x_i を 10 ビットの 2 進数として，F5 については 17 ビットの 2 進数として符号化した．また，Crossover の方式としては One-point Crossover[2] を用いた．

Trap function は以下で定義される関数である[16]．

$$f(u) = \begin{cases} a(z-u)/z & \text{if } u \leq z, \\ b(u-z)/(l-z) & \text{otherwise.} \end{cases} \qquad (3)$$

u は {0,1} からなる長さ l の文字列として符号化された個体に含まれる 1 の数である．ここでは，$l=40$, $z=30$, $a=80$, $b=100$ とした．この関数の概形を示したのが図 8 である．GA を用いた場合に，最適解 (最大値) である $f(40)=100$ ではなく $f(0)=80$ なる局所解に陥ることが知られている．この問題に関しては Uniform Crossover[17] を用いた．

TSP の符号化は訪れる都市の順序を表す順列をそのまま用い，Crossover の方式としては PMX (Partially Mapped Crossover)[18] を用いている．今回の実験では，10 都市，20 都市，40 都市の問題を用い

```
procedure Random-Exchange
constant p_exchange;

begin
 Initialize;
 while not terminate
  begin
   Selection;
   if Random[0,1) < p_exchange then
    begin
     Select a subpopulation to exchange;
     Send Status_request to the subpopulation;
     Wait for Status;
     if status = Accept then
      begin
       Get Migrants;
       Send Migrants;
      end;
    end;
    Crossover;
    Mutation;
  end;
end.
```

図 5 Random-Exchange アルゴリズム
Fig. 5 Random-Exchange algorithm.

```
procedure Message-Handling-Random

begin
 if message_type = Status_Request then
  begin
   if migrating then Send Reject
   else Send Accept with Migrants;
  end;
 else if message_type = Migrants then
  begin
   Get_Migrants;
  end;
end.
```

図 6 Random-Exchange における受信メッセージの処理
Fig. 6 Message handling for Random-Exchange.

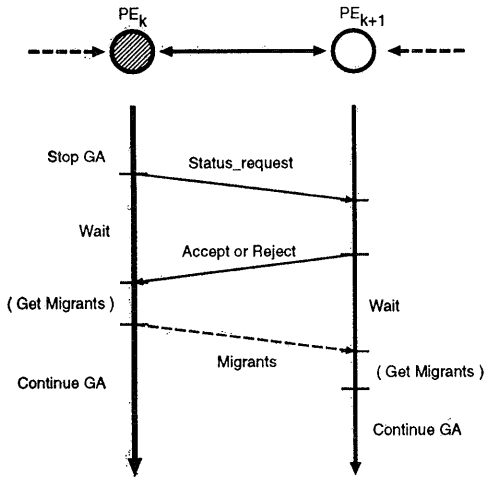


図 7 Random-Exchange における個体交換の手順 (PE$_k$ が交換を起動した場合)
Fig. 7 Exchanging process in Random-Exchange (where PE$_k$ invokes an exchange).

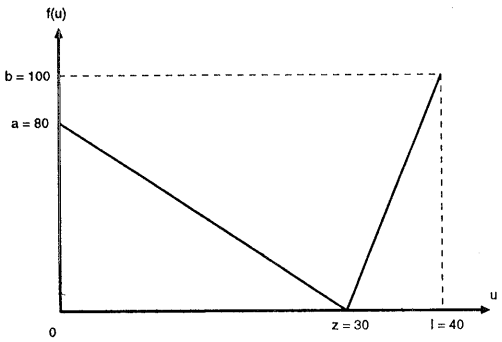


図 8 だまし関数 (trap function)
Fig. 8 A deceptive function (trap function).

た．ナップザック問題は，あらかじめ決められた制限重量を持つナップザックに様々な重量と価値をもつ荷物を詰め込み，詰め込むものの価値の和を最大にする問題である．この問題における適合度値は，詰め込まれたものの価値の和としてあらわされ，制限重量を上回った場合には値を 0 とする．この問題の場合，それぞれの荷物について，詰め込む場合には 1，詰め込まない場合には 0 とするビット列を用いて符号化している．今回，荷物数 100 の問題を用いたので，個体の文字列の長さは 100 である．Crossover の方式としては Uniform Crossover を採用した．

Selection の方式としてはすべての問題について Proportionate reproduction を用いている．TSP に関しては巡回路の長さを最小化する問題であるので，巡回路の長さを l とすると，$f=-l$ なる変換を行って最大化問題としている．また，すべての問題に関して，負の適合度の値を許すためと適切な収束圧力を維持するために $g=(f-f_{min})/(f_{max}-f_{min})$ なる変換を行った g の値を適合度としている．ここで，f_{min}, f_{max} はそれぞれ集団内の個体の持つ適合度の最小値，最大値である．

GA に関する基本的なパラメータとしては，Crossover する個体の割合を 0.6，Mutation の確率を 0.01 とし，交換対象となる個体の割合を 0.2 とした．また，最も適合度の高い個体を保存する手法である，De Jong の Elitist scheme を用いている．シミュレーションにおける計算時間に関する条件を**表 1** に示す．計算時間については，ワークステーション上での実測データをもとに，1 回の条件判断に必要な時間を 1 単位時間として他の計算時間を表現した．ここに示されているのは GA に関する処理時間や Sigma-Exchange において余分の負荷となる標準偏差の計算に要する処理時間，個体の交換に要する通信遅延時間である．

提案するアルゴリズムは非同期並列の並列 GA を前提としているために，通信遅延はモデルから分離できない本質的なものであり，アルゴリズムの評価を行うための必須パラメータであると考えられる．また，アルゴリズムを実際的なシステム内で評価するのが本論文の立場である．ただし，通信遅延について処理の細部に関しては考慮せずに，送信から受信まで全体として要する時間をまとめて遅延時間とした．

個体交換に関するパラメータである，p_exchange および λ に関しては，それぞれの実験のそれぞれの条件について実験的に最適化を行った値を用いている．また，以下すべての実験では，ランダムに発生させた異なる初期個体集団を用いてそれぞれ 10 回の実験を行い，得られた解（全集団内での最良値）に関して算

表 1 共通のシミュレーション条件
Table 1 Simulation conditions.

パラメータ名	値 (単位時間)
t_cond(条件判断一回に要する時間)	1
t_copy(一文字のコピーに要する時間)	2
t_select(一個体の選択に要する時間)	5
t_cross(一文字の交叉に要する時間)	1
t_mutate(一文字の突然変異に要する時間)	1
t_pairing(一つのペアを作るのに要する時間)	2
t_parm(パラメータ計算に要する時間)	2
t_send_message(メッセージ作成に要する時間)	10
t_evaluation(適合度関数の評価に要する時間)	100

術平均を求めている．

5.1 問題の性質と解との関係

ここでは，テスト関数として単峰・多峰性関数，だまし関数，TSP，ナップザック問題を取り上げ，それぞれの問題の性質と，Random-Exchange と Sigma-Exchange で得られる解との関係について論じる．

De Jong の F1 と F5，だまし関数の1つである Trap function について Random-Exchange と Sigma-Exchange の比較実験を行った結果を**表 2** に示す．それぞれ 10^5 単位時間経過後に得られた解を示している．その時間内に実行した世代数の平均は，F1 で約 130 世代，F5 で約 220 世代，Trap function で約 100 世代である．これによると，単峰性・多峰性関数による違いは見られず，だまし関数においてどちらの場合にも局所解に陥っているという結果が得られた．精度の良い近似最適解が求められるという意味で，F1 と F5 はいずれも GA にとって比較的易しい問題であるので，それぞれ Random-Exchange と Sigma-Exchange で結果に差が出なかったものと考えられる．また，GA にとって本質的に難しいだまし関数については，改良を加えた Sigma-Exchange でも難しい問題であることに変わりはない．

10 都市，20 都市，40 都市の TSP を用いて解空間の大きさを変化させて実験を行った結果を**表 3** に示す．10 都市 TSP の場合には 10^5 単位時間（約 13 世代），20 都市 TSP の場合には 1.3×10^6 単位時間（約 116 世代），40 都市 TSP の場合には 4×10^6 単位時間（約 240 世代）経過後に得られた解を表示している．この結果によると 10 都市の場合のように解空間の大きさがそれほど大きくない場合にはどちらの場合にも最適解が得られている．しかし，20 都市や 40 都市の場合のように解空間が大きくなると，与えられた同じ計算時間内では Random-Exchange による最適解が得られていないのに比べ，Sigma-Exchange では最適解を得ることができている．

ナップザック問題において，制約条件である制限重量を変化させて実験を行った結果を**表 4** に示す．それぞれ 10^6 単位時間経過に得られた解を示している．その時間内に実行した世代数は約 24 世代である．この結果によると，制約条件が非常に強い場合（制限重量 1000）には Random-Exchange と Sigma-Exchange のどちらも制約条件を満たす解を得ることはできない．しかしながら，制約条件が比較的緩やかな場合（制限重量 5000）には Sigma-Exchange が Random-Exchange に比べて与えられた同じ計算時間内で良い解を得ることができている．

以上の結果により，Sigma-Exchange は組合せ最適化問題など解空間が大きく，良い解を得るのに多数の世代数を要する問題について有効であることがわかった．GA にとって非常に易しい問題に関しては，少ない世代数で良い解が得られるため，個体交換を効率化する必要性が少なく，従来の手法との差は見られない．また，だまし関数や制約条件が強い問題など GA にとって本質的に難しい問題を解く場合には，従来手法と同様良い解を得ることはできない．

表 2 単峰・多峰性関数，だまし関数に関する結果（全個体数 32×32，通信遅延 100，5次元 hypercube）

Table 2 Results on unimodal, multimodal and deceptive functions (32 subpopulations with 32 strings, communication latency is 100, 5 D-hypercube).

テスト関数	最適値	Random-Exchange		Sigma-Exchange	
		平均	標準偏差	平均	標準偏差
F1	78.6	78.09	0.22	78.12	0.12
F5	500	500	0	500	0
Trap	100	80	0	80	0

表 3 解空間の大きさと得られた解に関する結果（TSP，全個体数 32×32，通信遅延 100，5次元 hypercube）

Table 3 Results on the relation between the size of solution space and the obtained solutions (TSP, 32 subpopulations with 32 strings, communication latency is 100, 5 D-hypercube).

都市数	解空間の大きさ	最適値	Random-Exchange		Sigma-Exchange	
			平均	標準偏差	平均	標準偏差
10	3.6×10^5	967.784	967.784	0	967.784	0
20	1.2×10^{17}	1133.62	1195.59	105.54	1133.62	0
40	2.0×10^{46}	1025.61	1078.08	157.41	1025.61	0

表 4 制約条件の強さと得られた解に関する結果（ナップザック問題，全個体数 32×32，通信遅延 100，5次元 hypercube）

Table 4 Results on the relation between the constraints and the obtained solutions (Knapsack problem, 32 subpopulations with 32 strings, communication latency is 100, 5 D-hypercube).

制限重量	最適値	Random-Exchange		Sigma-Exchange	
		平均	標準偏差	平均	標準偏差
5000	4968	4665.7	47.8	4763	31.6
3000	4053	3450.9	89.1	3481.5	54.7
1000	2070	0	0	0	0

5.2 解と交換回数の推移

得られた解の最適解からの誤差と交換回数の推移について比較結果を示す．全体の個体数を1024とし，これを32分割してそれぞれのプロセッサに割り当てる．ネットワークトポロジーは5次元の Hypercube で，隣接するプロセッサ間の通信遅延は100単位時間である．40都市の TSP（最短経路長 1025.61）について，得られた解の最適解からの誤差の推移を**図 9** に示す．この場合，Random-Exchange の p_exchange は 0.1，Sigma-Exchange の λ は 0.3 である．ここで，4×10^6 単位時間後に得られた解は，Sigma-Exchange で 1025.61（最適解からの誤差 0%），Random-Exchange で 1078.08（最適解からの誤差 5.1%）と，Sigma-Exchange が良い解を得ている．また，Sigma-Exchange では 3678000 単位時間で最適解を得ているが，Random-Exchange では最適解を得るまでに Sigma-Exchange に比べて約2倍の 7256000 単位時間を要している．交換回数の推移を**図 10** に示す．ここで，グラフの縦軸は 10^5 単位時間内に起動された交換回数を示している．Random-Exchange に比べて Sigma-Exchange は 75% 以上も交換回数が少ないという結果を得た．すなわち，この問題について Sigma-Exchange はより少ない個体交換で，より良い解を速く得ていることになる．

5.3 プロセッサの台数効果

プロセッサの台数効果は，並列計算機のプロセッサ数を増やした場合に，どの程度の性能向上が得られるかを示すもので，並列アルゴリズムの性能について議論する上で重要な要素である．ここでは Hypercube ネットワークおよび Ring ネットワーク上で，全個体数 1024 の集団を 2, 4, 8, 16, 32, 64, 128, 256 分割としてそれぞれのプロセッサに割り当てた場合に，最適解に対して誤差1% 以内の解が得られるまでの計算時間が，プロセッサ1台の場合に比較して何倍向上するかを観測した．ここで，すべての部分集団についての個体数の和が1024であるので，例えば16分割の場合には各部分集団は 64 個の個体から構成される．問題に応じて最適な全個体数を定めるのは困難であるが，台数効果の評価に際しては，現実的な分割の数において各部分集団が極端に小さくならないように全個体数を十分大きく定めることが必要であり，ここでは全個体数を1024とした．

Hypercube ネットワークに関する結果を**図 11** に，Ring ネットワークに関する結果を**図 12** にそれぞれ示す．ここで用いた問題は 20 都市の TSP（最短経路長 1133.62）である．ネットワークの通信遅延は 100

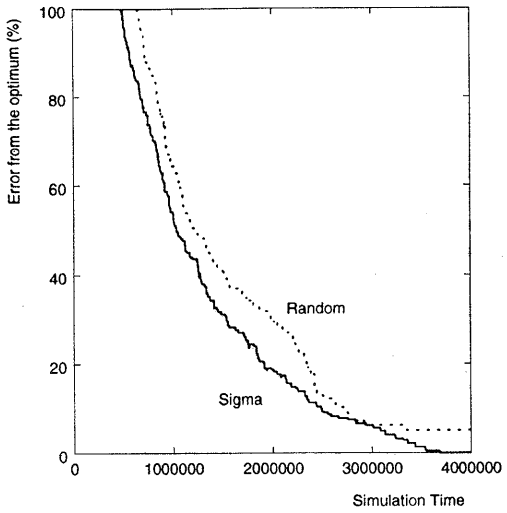


図 9 最適解からの誤差の推移（40都市 TSP，最適値 1025.61，全個体数 32×32，通信遅延 100，5次元 hypercube）

Fig. 9 Error from the optimal solution (40-city TSP, optimum 1025.61, 32 subpopulations with 32 strings, communication latency is 100, 5D-hypercube).

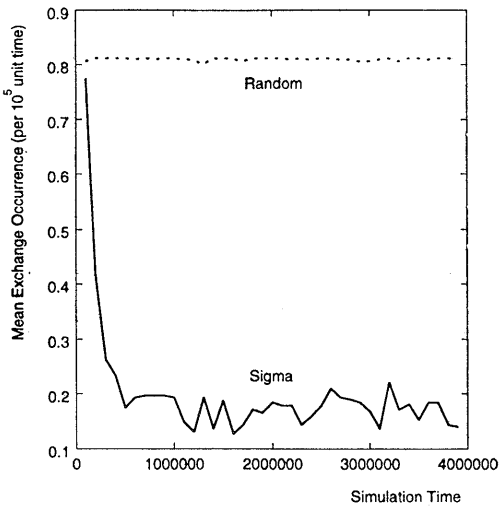


図 10 10^5 単位時間内に起動された交換回数の推移（40都市 TSP，全個体数 32×32，通信遅延 100，5次元 hypercube）

Fig. 10 Mean exchange occurrence in 10^5 unit time (40-city TSP, 32 subpopulations with 32 strings, communication latency is 100, 5D-hypercube).

単位時間である．この結果によると，分割数が 32 以下では有意な差は見られないが，どちらのトポロジーでも分割数が 64〜128 と大きくなるにつれて Sigma-Exchange が Random-Exchange よりも優れた台数効果を示している．しかし，分割数が 256 の場合には，それぞれの部分集団の個体数が 4 と非常に少なくなるので局所解に陥りやすくなり，いずれの方式でも性能が落ちている．すなわち，一定の初期集団を等分割してプロセッサに割り当てる場合，台数効果が最大となる分割数が存在する．

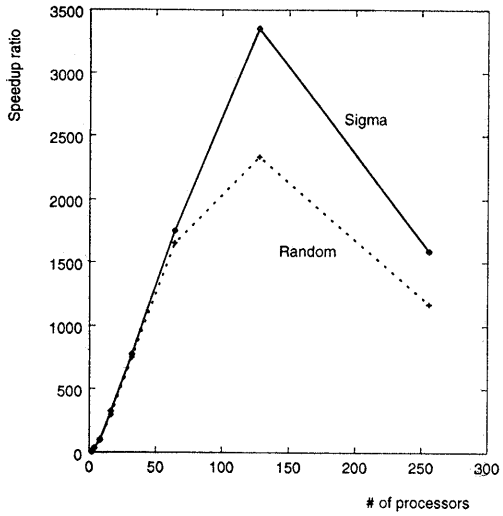


図 11 プロセッサの台数効果（20 都市 TSP，全個体数 1024 一定，通信遅延 100, hypercube）
Fig. 11 Speedup (20-city TSP, total population is 1024, communication latency is 100, hypercube).

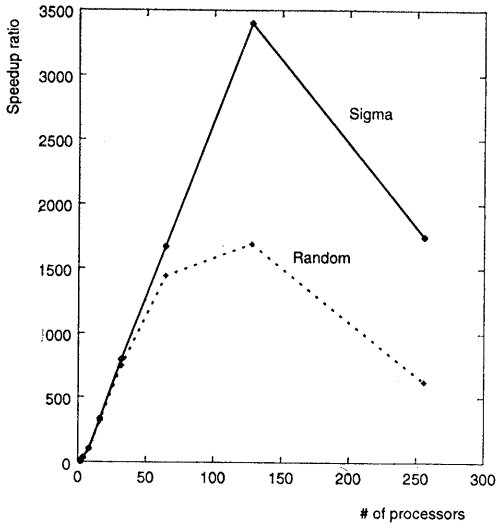


図 12 プロセッサの台数効果（20 都市 TSP，全個体数 1024 一定，通信遅延 100, ring）
Fig. 12 Speedup (20-city TSP, total population is 1024, communication latency is 100, ring).

また，Hypercube に比べて Ring の場合の方が Sigma-Exchange と Random-Exchange の差が大きく現れている．Ring ネットワークの場合には，ノードの次数（通信リンクの数）が Hypercube に比べて少なく，個体が離れた部分集団に到達するために多くの通信量を必要とする．そのためプロセッサ間通信を Sigma-Exchange で効率化した効果がより強く作用し，それが Ring ネットワークにおける大きな差となって現れていると考えられる．

以上の結果では性能の向上割合が 1 を超えていて，プロセッサ台数の増加以上の性能向上が見られる．Selection の計算量は，ここで採用している Proportionate reproduction の場合，個体数 n に対して，1 回のルーレットによる選択に $O(n)$ を要し，これを個体数だけ繰り返すため全体として $O(n^2)$ である[19]．そのため，個体数が $1/n$ になった場合，Selection が n^2 倍速くなる．また，個体数が少なくなると Selection による部分集団の収束が速くなることも理由の 1 つとしてあげられる．

5.4 プロセッサ間通信遅延の影響

20 都市 TSP（最短経路長 1133.62）に関して，隣接するプロセッサ間の通信遅延と，一定の計算時間（10^6 単位時間）の後に得られた解の最適解からの誤差の関係を**図 13** に示す．この場合の全個体数は 1024 であり，これを 32 個の部分集団に等分割した．ネットワークトポロジーは 5 次元の Hypercube ネットワークである．それぞれの通信遅延に対して，個体交換に関するパラメータ（p_exchange, λ）を最適化した．また，同じ条件下で，ナップザック問題（制限重量 3000，最適解 4053）に関して通信遅延と 5×10^6 単位時間後に得られた解の関係を**図 14** に示す．

実際的なシステムでの通信遅延を計算すると，システムによって異なるが，シミュレーションにおける 100 から 1000 単位時間程度に相当している．実験の結果によると，いずれの問題でも現実的な通信遅延の範囲で Sigma-Exchange は Random-Exchange よりも優れた解を得ていることがわかる．例えば，通信遅延が 800 単位時間の場合を考えると，各部分集団の個体数が 32 であるので，1 回の通信は 0.25 世代分の適合度評価に要する時間に対応する．したがって，

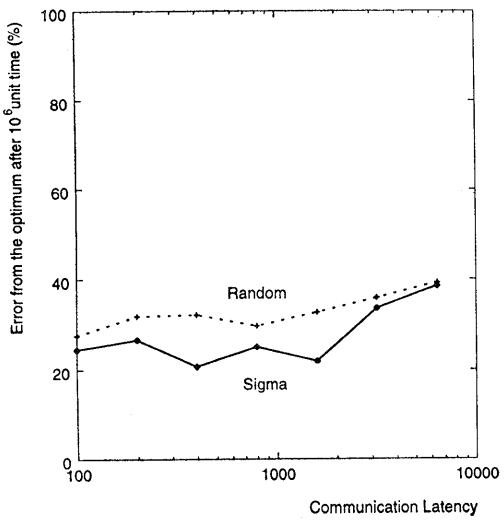


図 13 通信遅延と 10^6 単位時間後に得られた解の最適解からの誤差との関係（20 都市 TSP，最適値 1133.62，全個体数 32×32，5 次元 hypercube）
Fig. 13 Communication latency vs. error from the optimal solution after 10^6 unit time (20-city TSP, optimum 1133.62, 32 subpopulations with 32 strings each, 5D-hypercube).

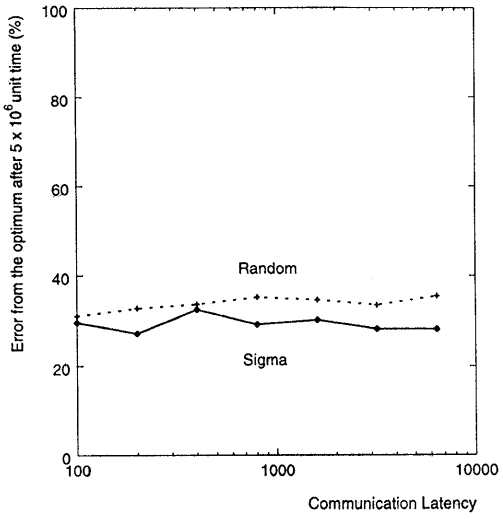


図 14 通信遅延と 5×10^6 単位時間後に得られた解の最適解からの誤差との関係（ナップザック問題，最適値 4053，全個体数 32×32，5 次元 hypercube）
Fig. 14 Communication latency vs. error from the optimal solution after 5×10^6 unit time (Knapsack problem, optimum 4053, 32 subpopulations with 32 strings each, 5 D-hypercube).

Sigma-Exchange において個体交換を起動したプロセッサが，交換を終了するまでに要する時間は約 1 世代分の適合度評価に要する時間と等しく，このような場合に Sigma-Exchange が有効であることがわかる．

5.5 ネットワークトポロジーとの関係

20 都市 TSP（最短経路長 1133.62）に関してネットワークトポロジーを Ring, Mesh, Hypercube, Perfect（完全結合）とした場合に，最適解に対して誤差 1% 以内の解を得るまでの時間と，その時間内にそれぞれの部分集団で実行した世代数の平均値を**表 5** に示す．実験条件としては，全個体数を 1024 とし，これを 32 等分割しそれぞれ並列計算機のプロセッサに割り当てた．従って，近傍のプロセッサ数（ノードの次数）はそれぞれ，2(Ring), 4(Mesh), 5(Hypercube), 31(Perfect) である．ネットワークの通信遅延は 100 単位時間とした．

この結果によると，Perfect の場合のみ Sigma-Exchange が解を得るのに Random-Exchange よりも多くの計算時間を要しているが，それ以外の場合には少ない時間で解を得ていることがわかる．Sigma-Exchange では，近傍のすべてのプロセッサに対してパラメータを要求するため，ノードの次数の高いネットワークでは，この点が負荷となるためと考えられる．また，交換を全く行わない場合（No-Exchange）との比較も行ったが，交換を行った場合に比べて多くの計算時間，世代数を要している．

表 5 各ネットワークトロジーにおける 1% 誤差の解が得られるまでの時間（単位時間）と世代数（20 都市 TSP，全個体数 32×32，通信遅延 100）
Table 5 Computation time (unit time) and generations to converge within 1% error in different network topologies (20-city TSP, 32 subpopulations with 32 strings each, communication latency is 100).

トポロジー	ノードの次数	Random-Exchange		Sigma-Exchange		No-Exchange	
		時間	世代数	時間	世代数	時間	世代数
ring	2	1,304,220	116.3	1,218,770	116.3	1,797,440	181.0
mesh	4	1,498,260	135.0	1,351,040	125.3		
hypercube	5	1,295,430	117.0	1,258,850	108.6		
perfect	31	1,346,190	121.2	1,386,500	129.2		

6. おわりに

本論文では，集団分割型の並列 GA において部分集団内の適合度分布の標準偏差を用いて個体交換の起

動条件とし，分布の平均と標準偏差の観点からできる限り分布の異なる集団と交換を行う手法 Sigma-Exchange を提案し，交換をランダムに起動する従来手法の一般化である Random-Exchange との比較検討をシミュレーション実験により行った．

解空間が非常に大きく，精度の良い解を得るのに多くの世代数を要する問題に対して，Sigma-Exchange は有効である．また，完全結合のようにノードの次数が高いネットワークトポロジーの場合には分布の情報を近傍に要求することが負荷となり効率が悪いが，それ以外の場合，特に実際的な通信遅延の範囲において Random-Exchange より精度の高い近似解を速く得ている．さらに，並列計算の台数効果においても優れている．

今後，実際の並列分散システムへのアルゴリズムの実現とその評価を行う予定である．

参考文献

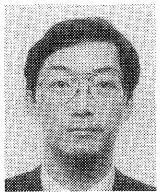
1) Holland, J. H.: *Adaptation in Natural and Artificial Systems*, University of Michigan Press (1975).
2) Goldberg, D. E.: *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley (1989).
3) Collins, R. J. and Jefferson, D. R.: Selection in Massively Parallel Genetic Algorithms, *Proceedings of the Fourth International Conference on Genetic Algorithms* (ed. by Belew, R. K.), pp. 249-256, Morgan Kaufmann Publishers (1991).
4) Manderick, B. and Spiessens, P.: Fine-grained Parallel Genetic Algorithms, *Proceedings of the Third International Conference on Genetic Algorithms* (ed. by Schaffer, J. D.), pp. 428-433, Morgan Kaufmann Publishers (1989).
5) Spiessens, P. and Manderick, B.: A Massively Parallel Genetic Algorithm, Implementation and First Analysis, *Proceedings of the Fourth International Conference on Genetic Algorithms* (ed. by Belew, R. K.), pp. 279-285, Morgan Kaufmann Publishers (1991).
6) Cohoon, J. P., Martin, W. N. and Richards, D. S.: A Multi-population Genetic Algorithm for Solving the k-Partition Problem on Hypercubes, *Proceedings of the Fourth International Conference on Gentic Algorithms* (ed. by Belew, R. K.), pp. 244-248, Morgan Kaufmann Publishers (1991).
7) Pettey, C. C. and Leuze, M. R.: Theoretical Investigation of a Parallel Genetic Algorithm, *Proceedings of the Third International Conference on Genetic Algorithms* (ed. by Schaffer, J. D.), pp. 398-405, Morgan Kaufmann Publishers (1989).
8) Tanese, R.: Parallel Genetic Algorithm for a Hypercube, *Proceedigs of the Second Intertional Conference on Genetic Algorithms* (ed. by Grefenstette, J. J.), pp. 177-183 (1987).
9) Tanese, R.: Didtributed Genetic Algorithms, *Proceedings of the Third International Conference on Genetic Algorithms* (ed. by Schaffer, J. D.), pp. 434-439, Morgan Kaufmann Publishers (1989).
10) Brown, D. E., Huntley, C. L. and Spillane, A. R.: A Parallel Genetic Heuristic for the Quadratic Assignment Problem, *Proceedings of the Third International Conference on Genetic Algorithms* (ed. by Schaffer, J. D.), pp. 406-415, Morgan Kaufmann Publishers (1989).
11) Gorges-Schleuter, M.: ASPARAGOS: An Asynchronous Parallel Genetic Optimization Strategy, *Proceedings of the Third International Conference on Genetic Algorithms* (ed. by Schaffer, J. D.), pp. 422-427, Morgan Kaufmann Publishers (1989).
12) Goldberg, D. E.: Zen and the Art of Genetic Algorithms, *Proceedings of the Third International Conference on Genetic Algorithms* (ed. by Shaffer, J. D.), pp. 80-85, Morgan Kaufmann Publishers (1989).
13) Mansour, N. and Fox, G. C.: A Hybrid Genetic Algorithm for Task Allocation in Multicomputers, *Proceedings of the Fourth International Conference on Genetic Algorithms* (ed. by Belew, R. K.), pp. 466-473, Morgan Kaufmann Publishers (1991).
14) Munetomo, M., Takai, Y. and Sato, Y.: An Efficient Migration Scheme for Subpopulation-based Asynchronously Parallel Genetic Algorithms, *Proceedings of the Fifth International Conference on Genetic Algorithms* (ed. by Forrest, S.), p. 649 (1993).
15) 棟朝雅晴，高井昌彰，佐藤義治：並列 GA における交換アルゴリズムの改良とその評価，情報処理学会研究報告，Vol. 92, No. 58, pp. 41-48 (1992).
16) Deb, K. and Goldberg, D. E.: Analyzing Deception in Trap Functions, *Foundations of Gentic Algorithms 2* (ed. by Whitley, L. D.), pp. 93-108, Morgan Kaufmann Publishers (1993).
17) Syswerda, G.: Uniform Crossover in Genetic Algorithms, *Proceedings of the Third International Conference on Genetic Algorithms* (ed. by Schaffer, J. D.), pp. 2-9, Morgan

Kaufmann Publishers (1989).
18) Goldberg, D.E.: Alleles, Loci, and the Traveling Salesman Problem, *Proceedings of the First International Conference on Genetic Algorithms* (ed. by Grefenstette, J. J.), pp. 154-159, Lawrence Erlbaum Associations, Publishers (1985).
19) Goldberg, D.E. and Deb, K.: A Comparative Analysis of Selection Schemes Used in Genetic Algorithms, *Foundations of Genetic Algorithms* (ed. by Rawlins, G. J.), pp. 69-93, Morgan Kaufmann Publishers (1991).

(平成5年8月16日受付)
(平成6年5月12日採録)

棟朝 雅晴（学生会員）

昭和43年生．平成3年北海道大学工学部電気工学科卒業．平成5年同大大学院工学研究科情報工学専攻修士課程修了．現在同博士後期課程在学中．遺伝的アルゴリズムおよび並列分散処理に興味を持つ．IEEE 会員．



高井 昌彰（正会員）

昭和35年生．昭和58年東北大学工学部電子工学科卒業．昭和63年同大大学院工学研究科情報工学専攻博士課程修了．工学博士．同年東京大学理学部情報科学科助手．平成元年北海道大学工学部講師．平成4年同助教授，現在に至る．並列分散処理，計算機アーキテクチャ，コンピュータグラフィクスの研究に従事．電子情報通信学会，IEEE 各会員．

佐藤 義治（正会員）

昭和21年生．昭和50年北海道大学大学院工学研究科情報工学専攻修士課程修了．同年，北海道大学工学部助手．同助教授を経て現在，北海道大学工学部教授（情報図形科学講座）．工学博士．多次元データ解析，多変量解析，ニューラルネットワーク理論，ファジィデータ解析の研究に従事．日本統計学会，日本応用統計学会，日本計算機統計学会，日本行動計量学会，日本ファジィ学会，国際計算機統計学会，国際分類学会，アメリカ統計学会各会員．