階層サーバ・クライアント型モデルによる 非同期並列分散遺伝的アルゴリズム*

小嶋 和徳[†]・石亀 昌明[†]・牧野 正三[‡]

Asynchronous Parallel Distributed Genetic Algorithm by Layered Server-Client Model*

Kazunori KOJIMA[†], Masaaki ISHIGAME[†] and Shozo MAKINO[‡]

The most popular researches about Parallel GAs are implemented as; Population is divided into some subpopulations, each subpopulation executes GA independently and some individuals are migrated in fixed intervals or fixed probability. On the other hand, Grid Computing has been noticed and a research that implements Parallel GA by using Master-Worker model on Grid Computing has been reported. However, on the huge search space problems, Parallel GA by using Master-Worker model needs a lot of worker to get better solution quality. If there are a lot of workers, the traffic loads to the master.

In this paper, we propose Asynchronous Parallel Distributed GA by using Layered Server-Client model. This model is based on Elite Migration on Server-Client model we proposed before. In this model, an Elite Server manages some Subpopulation Clients, and a Master Server manages some Elite Servers. From this structure, the number of Subpopulation Clients that a Elite Server manages is able to be reduced and the traffic on an Elite Server is also able to be reduced. To evaluate our proposed model, we apply to some problems. As the results, we confirm that the fitness is as well as that of current methods and the traffic is less than that of current methods. We also confirm that the migration time is able to be reduced especially in large search space problems.

1. はじめに

遺伝的アルゴリズム (Genetic Algorithm; GA) [1] は、生物の遺伝機構を模倣した多点探索手法であり、問題の解やパラメータを符号化した染色体をもつ個体を集めた集団に対し、選択、交叉、突然変異の操作を繰り返すことにより近似解を求めることができるというものである。

GA は 1970 年代初頭に J. H. Holland らによって提唱されて以来、簡素な手法であるにもかかわらず比較的良質な近似解が得られるという特徴から、関数最適化問題、組合せ最適化問題、制御問題、機械学習問題などに適用されてきた [2]。とくに近年、より大規模でより複雑な最

適化問題に対し、より高速にかつより効率よく近似解を得ることが求められるようになってきている。しかし、このような問題に対し GA を適用する場合、単純 GA のみでは限界があり、より良い近似解を得るためには多くの資源と時間が必要になるという問題がある。

このような問題を解決するために、複数のプロセッサで並列的に GA を実行する、並列分散 GA の研究 [3] が行われている。並列分散 GA に関する研究では、集団をいくつかの部分集団に分割し、その間で個体交換を行う手法 [3–9, 17–20] が多くを占める。この手法は、一定世代ごとに個体交換 (移民) を行う同期的な手法 [5, 6] と、一定確率により移民を行う非同期的な手法 [7] があるが、前者の場合、移民を行う際に相手の応答を待たなければならず、一つでも処理の遅いプロセッサが存在すればすべてに影響することになり、CPU の使用効率にむだが出る。後者の場合、非同期的な要求に対応する手法を実装するのは一般に難しいという問題がある。また、いずれの手法の場合でも移民が必要の如何に関わらず行われるため、効率のよい移民手法であるとはいいいがたい。マスタ・ワーカー型で実装している研究 [17–20] もあるが、

* 原稿受付 2007 年 11 月 12 日

[†] 岩手県立大学 ソフトウェア情報学部 Faculty of Software and Information Science, Iwate Prefectural University; Takizawa, Iwate 020-0193, JAPAN

[‡] 東北大学 大学院工学研究科 Graduate School of Engineering, Tohoku University; 05 Aoba, Aramaki, Aoba, Sendai, Miyagi 980-8579, JAPAN

Key Words: genetic algorithm, asynchronous, parallel distributed GA, elite migration, layered server-client model.

基本的には同期的な手法の性質を抜け出していない．それらに対し，各部分集団の標準偏差の変化から移民を行うタイミングを決定する手法が提案されている [9] が，そのタイミング決定のためのパラメータ設定が非常に難しいという問題がある．

最近では，グリッドコンピューティング上でマスタ・ワーカ型の並列分散 GA を実装した例 [20] も報告されている．しかし，大規模な問題にマスタ・ワーカ型モデルを適用する場合，膨大な数のワーカが必要となる．ワーカの数が増大になると，マスタへの負荷も膨大なものとなり，マスタとの通信がボトルネックとなる．

本研究では以前，サーバ・クライアント型モデルを用いて非同期に効率的な移民を行うエリート移民法を提案し，その有効性について検討を行ってきた [23]．しかし，サーバ・クライアント型モデルを用いているため，問題の規模が大きくなり，多数の部分集団が必要となる場合，上述のマスタ・ワーカ型モデルと同様にサーバとの通信がボトルネックとなってしまう．

そこで本論文では，さらに大規模な問題を考慮した，階層サーバ・クライアント型モデルによる非同期並列分散 GA を提案する．本研究で用いるモデルでは，サーバが複数のクライアントを管理し，さらにマスタサーバが複数のサーバを管理する．全体に必要なクライアント数に対し，サーバが管理するクライアントの数を減らし，複数のサーバを用意することで個々のクライアントを管理する．さらに，サーバを管理するマスタサーバを置くことで全体を管理する．

提案法を評価するため，まず TSP の 30 都市および 100 都市に適用し，以前提案したサーバ・クライアント型モデルによるエリート移民法との比較実験を行ったところ，サーバの負荷を分散させることができることが確認できた．

また，大規模な問題として TSP の 2392 都市，およびより実用的な問題として画像高解像度化問題に適用し，従来モデルとの比較実験を行ったところ，探索性能の面においても通信時間の面においても，従来モデルよりも優れていることが確認できた．

2. 並列分散 GA

並列分散 GA に関して網羅的に調査を行った Erickらの文献 [3] によると，これまでの研究は，集団を一つのプロセッサに割り当て，評価計算や遺伝的操作を複数のプロセッサに割り当てる大域並列化 (Global Parallelization) に関する研究 [10,11]，集団をいくつかの部分集団に分割し，その間で個体交換を行うことで並列計算を実現する粗粒度並列化 (Coarse-grained Parallelization) に関する研究 [4-7,9]，個体を格子状に配置し，近傍の個体のみと遺伝的操作を行うことで並列計算を行う細粒度並列化 (Fine-grained Parallelization) に関する研究 [12-14]，そしてこれら複数の手法を組み合わせた研究 [15,16] に

Fig. 1 Island model

大別できる．これらの中で，集団分割による並列化は最も自然な手法であり，実現が容易とされている．

集団分割による並列分散 GA では，一般的に Fig. 1 に示すような島モデルが使用される．このモデルでは集団をいくつかの部分集団に分割し，各部分集団が独立して GA を実行する．また，各部分集団の均一化を防ぐため，適当な世代間隔または適当な確率で部分集団間で一定割合の移民を行う．多くの場合，このモデルは Fig. 1 のように，一定の世代間隔でリング状に配置された島の隣へランダムに選択された一定割合の染色体情報を送信し，別の隣から送られた染色体情報を受信することで移民が行われる．このとき，すべての島で移民を行うタイミングが同じなので同期的である．以降，本論文ではこの手法を同期モデルとよぶ．同期モデルの場合，あるプロセッサでの実行が速い場合でも，移民時に他の遅いプロセッサが移民要求に応えるまで待ち状態となる．このため，プロセッサに処理速度の差がある場合は CPU の使用効率が悪くなるという問題がある．通信相手がそのつどランダムに決定される踏み石モデルという手法 [8] もあるが，その場合でもリング状の配置をランダムにしないという手法であるため，CPU 使用効率に関しては同様に問題が残る．一方，一定世代間隔ではなく一定確率により，ランダムに選択された島との移民を行うようにする手法 [5] も存在する．このとき，各部分集団はランダムなタイミングで移民要求を出すため不定期かつ非同期的である．以降，この手法を Random-Exchange とよぶ．しかし，これら同期モデルおよび Random-Exchange の二つの手法に関して，一定世代間隔にまたはランダムなタイミングで移民要求を行うことは，移民の必要性の如何に関わらず行われることから，効率よく情報交換が行えるとはいえず，不要な通信が行われている可能性が高い．

この問題に対し，棟朝らは各部分集団の適合度の標準偏差を監視し，ある程度部分集団が収束した時，つまり部分集団内にある程度同じ個体が存在するようになった時に移民を行う手法を提案した [9]．以降，この手法を Sigma-Exchange とよぶ．具体的には，あらかじめ求め

ている部分集団の適合度の標準偏差 σ_0 と、世代更新ごとに求められる部分集団の適合度の標準偏差 σ とを比較し、 $\sigma < \lambda\sigma_0$ を満たしたときに移民を行うというものである。ここで、 λ は移民タイミングを決定するためのパラメータとなる。しかし、集団の収束のしやすさは適用する問題の性質や集団サイズ、使用する遺伝的オペレータで変わる。たとえば、単峰性の性質を持つ問題は収束しやすく、逆に多峰性の性質を持つ問題は収束しにくい。また、集団サイズが小さければ収束しやすく、集団サイズが大きければ収束しにくい。さらに、どの選択法あるいは交叉法を使用するかでも収束のしやすさが変わる。このように、様々な要素によって値を変更する必要がある。かつ最適な値を探すのは難しい。そのため、最適な値を設定するためには、多回数の予備実験を行う必要がある。

また、Random-Exchange と Sigma-Exchange の実装について、棟朝らが [9] で示したものでは、GA を処理するルーチンとメッセージを処理するルーチンの二つで構成されている。通常は GA を処理するルーチンがプロセッサごとに実行され、移民要求などのメッセージを受け取ったとき、割り込みによりメッセージを処理するルーチンがそのプロセッサで起動され、メッセージが処理される。棟朝らはこれらの手法を独自のシステム上で実装しており、また必要条件として応答に必ず答えなければならないことをあげている。とくに Sigma-Exchange では、移民交換相手を決めるための近隣プロセッサとのパラメータの送受信、移民可能かどうかの確認のための送受信、移民のための送受信と、一度の移民処理で複数回の通信を行う必要がある。そのため、これらの処理の途中で通信が途切れるとそれ以降の処理が行えなくなってしまう、デッドロック状態に陥ってしまう。しかし、一般的な分散環境、とくに大規模な環境になると通信が途中で途切れる可能性があり、すべてのメッセージ処理を保証するのは事実上不可能である。その条件を満たすためには棟朝らが独自のシステムを構築したように、すべてのメッセージを管理するシステムを構築しなければならず、通信を専門とせず、GA をツールとして利用したい研究者にとって非常に大きな負担となってしまう。

最近の研究では、Erick ら [3] が並列 GA について網羅的に調査や実験的研究を行っているが、通信量や通信時間についてはほとんど議論されていない。また、パラメータフリー遺伝的アルゴリズム [17,18] や 2 個体分散遺伝的アルゴリズム [19,20] などの研究で、並列実装としてマスタ・ワーカ型モデルで実装する試みが行われているが、移民が定期的に行われるなど同期型の性質から抜け出しておらず、またやはりこれらの文献でも通信量や通信時間については議論されていない。

現在、グリッドコンピューティング [21] が注目される中、一般のオフィスなどの、普段能力を使いきれていない PC を計算資源として利用することが注目されている。

この中で、グリッドコンピューティング環境に並列分散 GA を実装した例 [20] も報告されている。しかし、大規模な問題にマスタ・ワーカ型モデルを適用する場合、膨大な数のワーカが必要となる。ワーカの数が増大になると、マスタへの負荷も膨大なものとなり、マスタとの通信がボトルネックとなる。

本研究では以前、マスタ・ワーカ型モデルと同様のサーバ・クライアント型モデルを用いたエリート移民法を提案し、その有効性について検討を行ってきた [23]。このモデルでは、一つのサーバと複数のクライアントで構成され、部分集団である各クライアントのエリートの更新状況により移民を行うモデルである。しかし、大規模な問題に適用し、クライアント数が増大すると、上述のマスタ・ワーカ型モデルと同様にサーバへの通信負荷が大きくなり、ボトルネックになってしまう。

これらの点を考慮し、本論文ではサーバへの負荷の分散を目的とした、階層サーバ・クライアント型モデルによる非同期並列分散 GA を提案する。

3. 階層サーバ・クライアント型モデルによる非同期並列分散 GA

3.1 サーバ・クライアント型モデルによるエリート移民法

本論文で提案する階層サーバ・クライアント型モデルの基本モデルは、Fig. 2 のようにサーバ・クライアント型モデルの形式を取り、一つのサーバプログラムと一つ以上のクライアントプログラムで構成される。

各クライアントプログラムはそれぞれ部分集団となり、独立に GA を実行する。また、サーバプログラムは各部分集団のエリートを管理する。以降、本論文ではクライアントプログラムを部分集団クライアント (Subpopulation Client) とよび、サーバプログラムをエリートサーバ (Elite Server) とよぶこととする。

Fig. 2 Server-client model

サーバ・クライアント型モデルの形式を取ることで、エリートサーバにはメッセージを処理するだけの働きをもたせ、部分集団クライアントはエリートサーバとのみの通信を行わせることができる。これにより、個々の部分集団クライアントは他の部分集団クライアントと通信

を行う必要がなくなる．個々の部分集団クライアントがメッセージ処理ルーチンを持つ必要がなくなることで、実装の簡素化および通信制御の容易化ができるだけでなく、メッセージが途切れたり、計算途中に部分集団クライアント用のプロセッサがダウンしても全体に与える影響はほとんどない．さらに、経過を確認しながら新たに部分集団クライアントを追加したり、不要な部分集団クライアントを削除したりといった使い方も可能となる．

エリートサーバ・部分集団クライアント間の通信方法は以下の通りである．以降、本論文ではこの通信方法をエリート移民法とよぶ．

- 部分集団クライアントはエリートが更新された場合、エリートの染色体情報をエリートサーバに送信する．
- 各部分集団クライアントは寿命 (longevity) パラメータ (γ) をもっており、部分集団クライアントにおける探索がうまく進行していない場合、つまりエリートが更新されない場合に寿命パラメータ (γ) を減少させる．
- 寿命パラメータ (γ) が 0 になったとき、部分集団クライアントはエリートサーバに移民要求を送り、エリートサーバが管理している染色体情報を受け取ることで移民を行う．
- エリートサーバは、各部分集団クライアントから送られてくるエリート情報を管理する．部分集団クライアントから移民要求が送られた場合、管理している個体の上位 μ 個体分の染色体情報を送り返す．

このように、エリート移民法の通信は、各部分集団クライアントのエリート更新によるその染色体情報のエリートサーバへの送信、各部分集団クライアントのエリートサーバへの移民要求の送信、およびエリートサーバからの移民受信であり、それぞれの部分集団クライアントからの通信は別々のタイミングで発生し、非同期的である．しかし、上述のようにサーバ・クライアント型モデルにより実装し、エリートサーバにメッセージ処理だけを行わせることで実装が容易となる．また、通信は各部分集団クライアントのエリートの更新状況により発生するため、不要な通信を減らすことができ、全体的な通信量を減らすことができる．

3.1.1 パラメータ

エリート移民法には、三つのパラメータが存在する．エリート送信間隔 (ν) 部分集団クライアントのエリートが更新された場合、毎回エリートサーバにその情報を送信するのではなく、 ν 回に 1 度の間隔で送信する．これにより、部分集団クライアントからエリートサーバへの通信頻度を減らすことができる．

寿命パラメータ (γ) 寿命パラメータ γ は部分集団クライアントの移民タイミングを制御するパラメータである．エリート移民法では、実装の簡素化という観点から、部分集団クライアントのエリートが更新されなかった場合に γ を 1 減少させる．そして γ が 0

になったときに移民を行う．エリートが更新された場合は γ を初期値にリセットする． γ の初期値は $1 \sim \infty$ の値を取ることができ、1 の場合はエリートが更新されない時は毎回移民が行われ、 ∞ の場合は移民が行われない．このパラメータを使用することにより、探索が進んでいる場合は移民を行わないようにすることができ、通信頻度を抑えることができる．移民数 (μ) 移民は、部分集団クライアントの寿命パラメータ γ が 0 になったとき、部分集団クライアントは移民要求をエリートサーバに送り、エリートサーバは管理しているエリート情報から μ 個体を選択し、部分集団クライアントに送り返すことにより行われる．部分集団クライアントは、受け取ったエリート情報を部分集団内のランダムな位置にコピーする．ここでも、移民するエリート情報の数を μ 個体分に抑えることで通信量を削減することができる．

3.1.2 結果概要

サーバ・クライアント型モデルによるエリート移民法の効果を検証するため、先行研究 [23] では、単峰性の問題や多峰性の問題など、様々な種類の代表的な 8 種類の関数問題を使用し、従来法である同期モデル、Random-Exchange および Sigma-Exchange との比較を行った．その結果、適合度に関しては従来法とエリート移民法との間に大きな差は認められなかったが、従来法に比べエリート移民法では通信量が少なく抑えることができることが確認でき、その有効性が示された．

また 3 種類のパラメータに関し、いずれのパラメータにおいても通信量が多くなるように設定すれば適合度が高くなり、通信量が少なくなるように設定すれば適合度が低くなる傾向が見られた．これらの傾向は、問題に依存せず、いずれの問題においても同様の傾向を示していることから、問題によってパラメータの設定を変更する必要がないことが確認できた．また、これらのパラメータによる適合度の変動が大きなものではなかったことから、ある程度の幅をもってパラメータを設定しても、適合度に大きな影響を与えないことも確認でき、パラメータ設定の柔軟さも示すことができた．

3.2 階層サーバ・クライアント型モデル

探索空間が比較的小規模な問題では、ある程度の集団サイズあるいは部分集団数があれば比較的良質な解が得られるが、探索空間が非常に大きな問題では、やはり集団サイズの増加や部分集団数の増加が必要となってくる．しかし、単純に集団サイズを増加しても、適合度の低い個体の影響から必ずしも探索性能の向上にはつながらない．そのため、並列分散 GA における部分集団数の増加による解決が必要になってくる．前節で使用したサーバ・クライアント型モデルの場合、部分集団クライアントの数を増加させることになるが、部分集団クライアントの数が多すぎるとエリートサーバへの負荷も大きなものと

Fig. 3 Layered server-client topology

なり，サーバとの通信がボトルネックとなってしまう．

そこで本論文では，エリートサーバの負荷を分散させる目的で階層サーバ・クライアント型モデルを提案する．Fig. 3 に，階層サーバクライアント型モデルの概要を示す．

階層サーバ・クライアント型モデルでは，前述のサーバ・クライアント型モデルが複数存在し，さらにそれらを管理するマスタサーバが存在する構造となる．つまり，複数の部分集団クライアントを管理するエリートサーバが存在し，さらに複数のエリートサーバを管理するマスタサーバが存在する階層構造となる．この構造により，並列分散 GA における部分集団数，つまり部分集団クライアント数が増加してもエリートサーバにかかる負荷を分散させることができ，通信パフォーマンスの向上につながる．

構造が大規模になることで，部分集団クライアント，エリートサーバ，マスタサーバのいずれかが動作不能になってしまう場合も考えられる．階層サーバ・クライアント型モデルでは，部分集団クライアントが動作不能になっても，サーバクライアント型モデルの場合と同様に全体の動作に対してはほとんど影響はない．逆に，各部分集団クライアントの進化状況を確認しながら，不要な部分集団クライアントを削除したり，新しく部分集団クライアントを追加したりという使い方も可能となる．

エリートサーバが動作不能となった場合，エリートサーバ以下の部分集団クライアントも動作不能になってしまうが，マスタサーバが全体を管理しているため，部分集団クライアントと同様に全体に対する影響はそれほど小さくなく，ここでも探索途中での追加や削除が可能である．

マスタサーバが動作不能になった場合は，全体が動作不能になってしまうためそれ以上の全体の探索は期待できなくなってしまう．しかしこの点に関しては，各部分集団クライアントの個体情報を定期的に保存することで探索途中から容易に再開することが可能である．これは，

上述の部分集団クライアントおよびエリートサーバに関しても同様にいえ，全体としてロバストなシステムとして機能させることができる．

3.2.1 通信ルール

階層サーバ・クライアント型モデルにおいて，部分集団クライアントとエリートサーバ間の通信は，前述のサーバ・クライアント型モデルと同様のルールで行う．つまり，部分集団クライアントのエリート個体が更新されると，その染色体情報をエリートサーバに送信し，ある一定期間部分集団クライアントのエリートが更新されなければエリートサーバが管理している染色体情報を部分集団クライアントが受け取ることでより移民を行う．

階層サーバ・クライアント型モデルでは，さらにエリートサーバとマスタサーバ間の通信ルールも必要となる．ここでは，エリートサーバ・部分集団クライアント間の通信ルールと同様のルールを使用することとする．つまり，エリートサーバ内の最良個体が更新されると，その染色体情報をマスタサーバに送信し，ある一定の期間，エリートサーバ内の最良個体が更新されなければ，マスタサーバが管理している最良個体を受け取るというルールとなる．

3.2.2 追加パラメータ

階層サーバ・クライアント型モデルでは，サーバ・クライアント型モデルでのパラメータに加え，さらに同様の三つのパラメータが存在する．

最良個体送信間隔 (ν_s) エリートサーバが管理している最良個体情報が更新された場合，毎回マスタサーバにその情報を送信するのではなく， ν_s 回に 1 度の間隔で送信する．

寿命パラメータ (γ_s) 部分集団クライアントの寿命パラメータ γ と同様に，エリートサーバにも寿命パラメータ γ_s を設定する．エリートサーバでは，部分集団クライアントからのエリート情報の受信回数をカウントし，エリート情報を受信した時に，管理している個体の最良個体が更新されない場合に γ_s を 1 減少させる．そして， γ_s が 0 になったとき，マスタサーバが管理している個体情報を受け取る．最良個体が更新された場合， γ_s を初期値にリセットする．

移民数 (μ_s) 移民は，エリートサーバの寿命パラメータ γ_s が 0 になったとき，エリートサーバは移民要求をマスタサーバに送り，マスタサーバは管理している最良個体情報から μ_s 個体を選択し，エリートサーバに送り返すことにより行われる．

4. 評価実験

4.1 TSP 30 および 100 都市での比較

3.1.2 で概要を示した通り，われわれの先行研究 [23] により，サーバ・クライアント型モデルによるエリート移民法は，同期モデル，Random-Exchange および Sigma-Exchange と探索性能は同等であることを確認し

ている．そこでここでは，サーバ・クライアント型モデルおよび提案手法である階層サーバ・クライアント型モデルの比較を行う．

本節では，巡回セールスマン問題 (Traveling Salesperson Problem; 以下，TSP とする) の 30 都市および 100 都市問題に適用した場合についての比較を行うとともに，大規模問題に適用するための指針を得るために，階層サーバ・クライアント型モデルの構造およびパラメータと，適合度，処理時間，通信量との関係について調査することで，有効な構造やパラメータ範囲について考察する．

4.1.1 実験条件

本実験では，SGI 製の Origin2000 (32 CPUs) 上で，C 言語および MPI (Message Passing Interface) を用いてそれぞれの手法を実装した．

ここでの実験条件は以下の通りである．ここでは，GA パラメータについては一般的に使用されているものを，サーバ・クライアント型モデルのパラメータについては先行研究を参考に決定したもの，階層サーバ・クライアント型モデルのパラメータについては，ここではサーバ・クライアント型モデルのパラメータを参考に決定したものである．なお，階層サーバ・クライアント型モデルのパラメータ考察の詳細については後述している．

- 部分集団数：32
- 集団サイズ：1 部分集団当たり 32
- 選択法：ルーレット選択
- 交叉：確率 0.6，部分一致交叉
- 突然変異：確率 0.03，逆位
- 終了条件：500 世代更新後 (30 都市)，10,000 世代更新後 (100 都市)
- 試行回数：20 回
- サーバ・クライアント型モデルパラメータ
 - － エリート送信間隔 (ν)：5
 - － 寿命初期値 (γ)：10
 - － 移民数 (μ)：2
- 階層サーバ・クライアント型モデルパラメータ
 - － 最良個体送信間隔 (ν_s)：2
 - － エリートサーバ寿命初期値 (γ_s)：5
 - － 移民数 (μ_s)：1

4.1.2 適合度および処理時間

Table 1 および Table 2 は最大適合度の試行回平均値と 1 島当たりの処理時間の試行回平均値を示す．ここで，適合度は最適値を 1.0 として正規化している値であり，大きい値ほど良い探索性能であることを示している．またこの表で，Server-Client はサーバ・クライアント型モデルを示し，Layered は階層サーバ・クライアント型モデルを示している．さらに，Layered の欄の括弧内は，階層サーバ・クライアント型モデルの構造を示している．たとえば「(1-4-8)」は，マスタ数 1，エリートサーバ数 4，エリートサーバごとの部分集団クライアント数 8 で

Table 1 Fitness and processing time on TSP with 30 cities

項目 モデル	適合度	処理時間 [秒]
Server-Client	0.909	6.91
Layered (1-8-4)	0.863	8.46
Layered (1-4-8)	0.885	7.75
Layered (1-2-16)	0.900	7.39
Layered (1-1-32)	0.917	7.08

Table 2 Fitness and processing time on TSP with 100 cities

項目 モデル	適合度	処理時間 [秒]
Server-Client	0.914	274.14
Layered (1-8-4)	0.893	349.54
Layered (1-4-8)	0.905	313.85
Layered (1-2-16)	0.909	293.95
Layered (1-1-32)	0.913	283.75

あることを示しており，全部分集団クライアント数はすべての構造で 32 となるように調整している．

これらの表から，階層サーバ・クライアント型モデルの適合度は，(1-1-32) の構造ではサーバ・クライアント型モデルとほぼ同等かそれ以上の適合度が得られているが，その他の構造 (1-2-16)，(1-4-8)，(1-8-4) ではサーバ・クライアント型モデルよりも低い値となっていることが確認できる．また，30 都市，100 都市いずれの場合においても，(1-8-4) 構造の適合度がやや低く，(1-4-8)，(1-2-16)，(1-1-32) となるにしたがい徐々に適合度が高くなる．

このような結果となる原因としてまず，一つのエリートサーバが管理する部分集団クライアントの数が少ないと，規模の小さなサーバ・クライアント型モデルが複数個存在することになることが挙げられる．1 部分集団クライアント当たりの集団サイズは同一としているため，規模の小さなサーバ・クライアント型モデルでは，そこで管理される集団サイズも小さくなる．結果的に一つサーバ・クライアント型モデルでの探索範囲が狭くなり，それだけでは探索性能の向上は望めないことになる．またそれに加え，一つのエリートサーバが管理しているエリート情報が他のエリートサーバに伝わりづらくなっていることもあげられる．結果的に優良な情報が全体に伝わりづらくなり，全体の探索性能の向上につながらないものであると考えられる．

また，これらの表から，階層サーバ・クライアント型モデルでは，(1-8-4)，(1-4-8)，(1-2-16) において，サーバ・クライアント型モデルよりも多くの時間を必要としている．この中でも，(1-8-4) が一番多くの時間を必要と

し、(1-4-8)、(1-2-16)となるにしたがって必要な時間が少なくなる．これはまず、エリートサーバとマスタサーバ間の通信に時間が必要となるためであると考えられ、加えてエリートサーバが増えることによってマスタサーバとの通信が多くなることから、結果的に(1-8-4)では多くの時間が必要となっているものと考えられる．以上のことから、ここではエリートサーバの数と処理時間にはトレードオフが存在することが確認できる．

しかし、サーバ・クライアント型モデルと比較しても、それぞれの構造を比較しても、(1-8-4)の構造以外では最大適合度および処理時間ともにそれほど大きな差はなっていない．このことから、上述の傾向を念頭に1エリートサーバ当たりの部分集団クライアント数が少なすぎない構造とすることで、ある程度同等の適合度が得られるともいえ、柔軟性をもっているともいえる．

4.1.3 通信量

階層サーバ・クライアント型モデルは、サーバ・クライアント型モデルのエリートサーバにかかる負荷を分散させる目的で考案したモデルであることから、ここでは、サーバ・クライアント型モデルと階層サーバ・クライアント型モデルのエリートサーバ上での通信量を比較する．

Table 3 および Table 4 は、それぞれ TSP 30 都市および 100 都市におけるサーバ・クライアント型モデルのエリートサーバの通信量と、階層サーバ・クライアント型モデルの前節での実験で使用した四つの構造モデルの一つのエリートサーバあたりの平均通信量を示している．これらの表で「送信量」は一つのエリートサーバが部分集団クライアントおよびマスタサーバに送った通信量 (bytes) を示し、「受信量」は一つのエリートサーバが部分集団クライアントおよびマスタサーバから受け取った通信量 (bytes) を示している．またこの表で、Layered の欄の括弧内は、前節の表と同様に、階層サーバ・クライアント型モデルの構造を示している．

送信量を見ると、階層サーバ・クライアント型モデルの(1-8-4)、(1-4-8)および(1-2-16)構造において、エリートサーバの通信量が減少することが確認できる．また、(1-8-4)構造では通信量が少なく、(1-4-8)、(1-2-16)となるにしたがって通信量が増加することも確認できる．(1-1-32)構造ではサーバ・クライアント型モデルよりも同等あるいは若干多い通信量となっている．この結果は、受信量に関しても同様にみられる．

このような結果になる原因は、一つのエリートサーバが管理する部分集団クライアント数が少ないためで、たとえば(1-1-32)構造では一つのエリートサーバが管理する部分集団クライアント数が32であるのに対して(1-2-16)ではその数が16となり、部分集団クライアントとの通信が半分で済むようになるためである．(1-4-8)、(1-8-4)構造の場合でも同様の理由から通信量が少なくなる．

以上のことから、階層サーバ・クライアント型モデル

Table 3 Traffic on an elite server on TSP with 30 cities [bytes]

項目 モデル	送信量	受信量
Server-Client	16886.40	33260.64
Layered (1-8-4)	2896.79	4228.95
Layered (1-4-8)	5268.90	8899.92
Layered (1-2-16)	10051.48	18340.16
Layered (1-1-32)	19601.27	37279.88

Table 4 Traffic on an elite server on TSP with 100 cities [bytes]

項目 モデル	送信量	受信量
Server-Client	4045952.00	548133.76
Layered (1-8-4)	445769.41	87934.11
Layered (1-4-8)	948043.56	164044.28
Layered (1-2-16)	1961455.76	306774.43
Layered (1-1-32)	3941969.23	580938.62

においてはその構造を調節することでエリートサーバの通信負荷を分散および軽減できることが確認できた．

4.1.4 パラメータ

前章で示した通り、階層サーバ・クライアント型モデルでは三つの追加パラメータが存在する．本節では、これらのパラメータと適合度および通信量との関係について考察する．ただし、マスタサーバからエリートサーバに送られる移民数 μ_s は、本実験では1と固定しているため、ここでは最良個体送信間隔 ν_s およびエリートサーバの寿命パラメータ γ_s を考察対象とする．

Table 5 は、TSP 30 都市における、各パラメータの値による適合度と通信量を示す．このときの階層サーバ・クライアント型モデルの構造は(1-2-16)である．

Table 5 Effect of parameters in layered server-client model on TSP with 30 cities

項目 パラメータ	適合度	送信量	受信量
Server-Client	0.909	16886.40	33260.64
$\nu_s = 1, \gamma_s = 1$	0.917	12641.63	29419.76
$\nu_s = 2, \gamma_s = 1$	0.908	10638.07	29733.51
$\nu_s = 3, \gamma_s = 1$	0.904	10076.84	29935.12
$\nu_s = 1, \gamma_s = 2$	0.908	12062.33	22440.68
$\nu_s = 2, \gamma_s = 2$	0.905	10254.77	22558.73
$\nu_s = 3, \gamma_s = 2$	0.904	9738.81	22701.29
$\nu_s = 1, \gamma_s = 3$	0.905	11862.55	20138.25
$\nu_s = 2, \gamma_s = 3$	0.903	10421.95	20180.60
$\nu_s = 3, \gamma_s = 3$	0.900	9734.39	20176.32

この表から、最良個体送信間隔 ν_s が増加するにしたがい適合度が減少し、合わせて送信量が減少することが確認できる。エリートサーバからマスタサーバへの通信頻度を規定するパラメータであるので、このパラメータが増加するとエリートサーバからマスタサーバへの通信頻度が低くなり、送信量が少なくなる。このため、エリート情報がマスタサーバへ伝わる頻度が低くなり、結果的にエリート情報が全体に伝わりにくい原因となり、適合度が若干下がる結果となる。

エリートサーバの寿命パラメータ γ_s に関しては、この値が増加するにしたがい適合度が減少し、合わせて受信量が減少する。このパラメータは、マスタサーバからの通信頻度を規定するパラメータであるので、値が増加するとマスタサーバからエリートサーバへの通信頻度が低くなり、受信量が少なくなる。結果的に、ここでも通信頻度が低くなることでエリート情報が全体に伝わりにくい原因となり適合度が若干下がる結果となる。

これらの結果から、若干ではあるが適合度と通信量にはトレードオフが存在することがわかる。つまり、通信量が多ければエリート情報が全体に伝わりやすいため適合度が高くなり、逆に少なければ適合度が低くなるということである。

またこの表から、最良個体送信間隔 ν_s が1、エリートサーバの寿命パラメータ γ_s が2のとき、階層サーバ・クライアント型モデルはサーバ・クライアント型モデルと同等の適合度が得られていることが確認できる。このとき、送信量、受信量ともに減少していることも確認できる。さらに同様の結果が、TSP 100都市問題でも確認することができた。つまり、階層サーバ・クライアント型モデルでは、パラメータを調整することで探索性能を維持しながらサーバ・クライアント型モデルよりも通信量が少なくて済むということである。

これらのパラメータに関しては、 $\nu_s = 1, \gamma_s = 1$ と比較して $\nu_s = 3, \gamma_s = 3$ のとき、および4.1.2で使用したパラメータ $\nu_s = 2, \gamma_s = 5$ では若干適合度が下がっているものの、それ以外の値については大きな差は見られない。この点を考慮し、筆者らは $\nu_s = 1 \sim 2, \gamma_s = 1 \sim 2$ 程度が適当な値であると考えた。

4.2 TSP 2392都市での比較

階層サーバ・クライアント型モデルは、多くの部分集団クライアントが必要となる場合を想定したモデルであることから、ここでは大規模な問題としてまずTSPの2392都市に適用した場合について考察する。

4.2.1 実験条件

本実験では、SGI製のAltix350 (16 CPUs) 上でC言語およびMPI (Message Passing Interface) にてそれぞれを実装した。このときの実験条件は以下の通りである。ここでは、前節での実験と同様に、GAパラメータについては一般的に使用されているものを、サーバ・クライ

アント型モデルのパラメータについては先行研究を参考に決定したものを使用した。階層サーバ・クライアント型モデルのパラメータについては、前節での考察を参考に決定している。

- 部分集団クライアント数：256
- 部分集団サイズ：1部分集団当たり32
- 選択法：ルーレット選択
- 交叉：確率0.6、部分一致交叉
- 突然変異：確率0.03、逆位
- 終了条件：300,000世代更新後
- 試行回数：1回
- サーバ・クライアント型モデルパラメータ
 - － エリート送信間隔 (ν)：3
 - － 寿命初期値 (γ)：10
 - － 移民数 (μ)：2
- 階層サーバ・クライアント型モデルパラメータ
 - － 最良個体送信間隔 (ν_s)：1
 - － エリートサーバ寿命初期値 (γ_s)：1
 - － 移民数 (μ_s)：1

また本実験では、比較対象として同期リング型モデル (Ring (Sync))、非同期リング型モデル (Ring (Async))、非同期トーラス型モデル (Torus (Async)) を追加した。

4.2.2 実験結果

Table 6は、それぞれの手法における最大適合度と処理時間を示している。ここで、適合度は最適値を1.0として正規化している値であり、大きい値ほど良い探索性能であることを示している。また、ここで示している処理時間は、一つの部分集団クライアントあたりの処理時間、巡回路の計算時間、移民時間である。さらにこの表で、時間に関しては時:分:秒を示している。

同期リング型モデル (Ring (Sync)) では、適合度は低い値となっている。これは、部分集団数が多くなると、移民が隣同士の部分集団としか行われないため、局所的な情報交換しか行われず、優良個体の情報が全体に伝わりにくいためであることが原因として考えられる。

非同期リング型モデル (Ring (Async)) および非同期トーラス型モデル (Torus (Async)) では、進捗がほとんど進んでいない。これらのモデルでは、移民はランダムなタイミングで行われていることから、移民が集中的に行われる場合がある。それが逆に悪影響を及ぼし、有用な情報を破壊してしまっている可能性があることから、このような結果になったのではないかと推測できる。

それらに比べ、サーバ・クライアント型モデルおよび階層サーバ・クライアント型モデルは比較的高い適合度を得られている。この結果から、とくに大規模な問題では他のモデルでは良い解を得ることが難しいことが確認でき、逆にサーバ・クライアント型モデルおよび階層サーバ・クライアント型モデルでは、エリートサーバやマスタサーバを介して優良個体の情報が全体に伝わっていることで、結果として比較的良好な解を得ることができ

Table 6 Fitness and processing time on TSP 2392 cities

項目 モデル	適合度	処理時間	計算時間	移民時間
Ring (Sync)	0.582	7:37:45	1:02:00	1:26:11
Ring (Async)	0.076	5:24:47	0:53:30	0:00:38
Torus (Async)	0.075	5:23:56	0:53:17	0:00:50
Server-Client	0.841	7:46:49	0:59:55	1:41:28
Layered (1-2-128)	0.862	6:28:46	0:59:53	0:23:27
Layered (1-4-64)	0.833	6:34:01	1:00:23	0:23:53
Layered (1-8-32)	0.834	6:31:30	0:59:59	0:25:48

ている．つまり、規模の大きな問題では、優良な情報を有効活用することが必要であり、サーバ・クライアント型モデルおよび階層サーバ・クライアント型モデルではそれが有効に働いていることが確認できた．

処理時間に関して見ると、まず巡回路の計算時間はいずれの手法でもほぼ一致している．これは、各部分集団の集団サイズおよび世代更新数が一定になっているため、ここでは各モデルによる計算時間の揺らぎがないことが確認できる．また処理時間に関しては、移民時間の差がほぼそのまま差として表れていることから、遺伝的操作の部分に関してもモデルによる差がないことが確認でき、差が現れるのは移民時間の差によるものであることも確認できる．

移民時間に関して見ると、同期リング型モデルでは多くの時間が必要となっている．これは、同期処理により他のプロセスの応答を待たなければいけないことが原因である．部分集団数が多い場合には特に、その待ち時間が顕著に表れる．

非同期リング型モデルおよび非同期トラス型モデルでは、その待ち時間がほとんどないため、移民時間もほとんどないに等しい結果となっている．ただし、上述のように適合度は悪い結果となる．

サーバ・クライアント型モデルでは、通信が非同期に行われるため遅いプロセスを待つ必要はないが、部分集団クライアント数が増えるとその通信がサーバに集中してしまい、エリートサーバとの通信がボトルネックとなり移民時間に多くを必要としてしまう結果となる．

階層サーバ・クライアント型モデルでは、(1-2-128)、(1-4-64)、(1-8-32)いずれの構造においても同期リング型モデルおよびサーバ・クライアント型モデルよりかなり少ない移民時間で済んでいる．この結果から、エリートサーバにかかる負荷を分散させ、ボトルネックを解消できていることが確認できる．さらに、適合度もサーバ・クライアント型モデルと同等の結果が得られている．

以上により、階層サーバ・クライアント型モデルは、少ない移民時間でサーバ・クライアント型モデルと同等の探索性能を得られる優れた手法であることが確認できた．

4.3 画像高解像度化問題での比較

ここでは、より実用的な問題での比較として、画像高解像度化問題へ適用した場合について考察する．

4.3.1 画像高解像度化問題

ここで適用する画像高解像度化問題は、ビデオカメラから撮影された、ビデオカメラ自身または対象物が微少移動している時系列画像を利用し、その時系列画像から抽出された複数枚の連続した画像をもとに高解像度化された画像を推定するものである．一般的には、抽出された n 枚の画像から n 倍に高解像度化された画像を推定する．その推定する手法の一つとして、遺伝的アルゴリズムを利用した例 [24] が存在する．

本研究では、標準画像から擬似的に、ビデオカメラが横方向に微少移動したという想定で作成した連続時系列画像 4 枚 (Fig. 4 参照) を利用し、横方向に 4 倍に高解像度化した画像を推定する．

このとき、染色体は推定する高解像度化画像の画素の濃度値列となる．また、作成した推定高解像度化画像から n 枚 (この例では 4 枚) の縮小画像を 1 画素ずつずらしながら作成し、それらの画像と Fig. 4 の画像を比較して濃度差の合計が少ないものを高い適合度を持つものとする．

Fig. 4 Sequential image for evaluation on image high-resolutionization problem

4.3.2 実験条件

ここでは、前節での実験と同様に、SGI 製の Altix350 (16 CPUs) 上で C 言語および MPI にてそれぞれを実装した。このときの実験条件は以下の通りである。

- 部分集団クライアント数：256
- 部分集団サイズ：1 部分集団当たり 32
- 選択法：ルーレット選択
- 交叉：確率 0.6，部分一致交叉
- 突然変異：確率 0.0000148，ランダムな値に変更
- 終了条件：75,000 世代更新後
- 試行回数：1 回
- サーバ・クライアント型モデルパラメータ
 - エリート送信間隔 (ν)：3
 - 寿命初期値 (γ)：10
 - 移民数 (μ)：2
- 階層サーバ・クライアント型モデルパラメータ
 - 最良個体送信間隔 (ν_s)：1
 - エリートサーバ寿命初期値 (γ_s)：1
 - 移民数 (μ_s)：1

ここでのパラメータは、前節の TSP 2392 都市での条件を参考に設定している。なお、突然変異は、1 回の突然変異につき推定高解像度画像の 1 画素の値が変わる確率となっている。

4.3.3 実験結果

Table 7 は、同期リング型モデル (Sync (Ring))，サーバ・クライアント型モデル (Server-Client)，そして 1-4-64 構造，つまりマスタサーバ数 1，エリートサーバ数 4，エリートサーバごとの部分集団クライアント数 64 という構造の階層サーバ・クライアント型モデル (Layered (1-4-64)) それぞれの手法における最大適合度と処理時間の比較を示している。ここで、適合度は最適値を 1.0 として正規化している値であり、大きい値ほど良い探索性能であることを示している。また、ここで示している処理時間は、一つの部分集団クライアントあたりの処理時間、評価計算の時間および移民時間である。さらにこの表で、時間に関しては時:分:秒を示している。

結果的には、先に示した TSP 2392 都市問題での結果と同様の傾向が見られた。

たとえば、同期リング型モデルでは初期世代である第 0 世代での最大適合度 0.963 と比較してそれほど進化が進んでいない。それに対し、サーバ・クライアント型モデルおよび階層サーバ・クライアント型モデルでは比較的高い最大適合度を得ることができた。

また時間に関しても、同期リング型モデルとサーバ・クライアント型モデルでは多くの移民時間および処理時間を必要としており、それらと比較して階層サーバ・クライアント型モデルではかなりの移民時間の短縮、ひいては処理時間の短縮に成功している。

つぎに、結果として得られた画像を示す。Fig. 5 は初期世代に作成される画像で、評価用画像を単純に横方向

Fig. 5 Initial high-resolutionized image

Fig. 6 High-resolutionized image by synchronous ring topology

に引き伸ばした画像である。Fig. 6 は同期リング型モデルを使用した場合に作成された画像、そして Fig. 7 は階層サーバ・クライアント型モデルを使用した場合に作成された画像である。なおここでは、サーバ・クライアント型モデルを使用して作成された画像は、階層サーバ・クライアント型モデルで作成されたものと大きな違いは見られなかったため、省略している。

初期世代での画像と比較して、同期リング型モデルで作成された画像は、ある程度高解像度化されていることは確認できるが、まだ多くの部分で粗さが残っていることがわかる。

それに対し階層サーバ・クライアント型モデルで作成された画像は、同期リング型モデルで作成された画像と比較して高解像度化が進んだ画像が得られており、とくに中央の文字の部分でその差が明確に確認できる。

以上のことから、ここでも階層サーバ・クライアント

Table 7 Fitness and processing time on image high-resolutionalization problem

項目 モデル	適合度	処理時間	計算時間	移民時間
Ring (Sync)	0.970	10:43:22	1:13:49	2:16:37
Server-Client	0.982	11:27:12	1:11:38	3:18:51
Layered (1-4-64)	0.981	8:11:56	1:12:14	0:00:10

Fig. 7 High resolutionalized image by layered server-client topology

型モデルは、少ない移民時間でサーバ・クライアント型モデルと同等の探索性能を得られる優れた手法であることが確認できた。

5. おわりに

並列分散 GA では、粗粒度並列化が一般的に行われており、その一般的な手法は同期モデル、Random-Exchange、Sigma-Exchange である。しかし、それらは以下の問題がある。同期モデルや Random-Exchange では、効果的な移民が行われていない。Random-Exchange と Sigma-Exchange では、非同期イベントによる実装が難しい。そして、Sigma-Exchange では、移民のタイミングを決定するパラメータの設定が難しいという問題があった。

それに対して、以前われわれはサーバ・クライアント型モデルを用いたエリート移民法を提案した。これは、一つのサーバプログラムと複数のクライアントプログラムから構成されている。以前のわれわれの実験より、サーバ・クライアント型モデルによるエリート移民法は、他の手法に比べ同等の探索性能を維持しながら通信量の削減に成功していることが確認できている。

しかし、大規模な問題にサーバ・クライアント型モデルを使用した場合、エリートサーバに通信負荷が集中してしまうことが問題点としてあげられる。

この問題に対処するため、本論文では階層サーバ・クラ

イアント型モデルを提案した。このモデルは、一つのマスタサーバ、複数のエリートサーバ、そして複数の部分集団クライアントから構成される。一つのエリートサーバは複数の部分集団クライアントを管理し、その構造が複数個存在する。そして、さらに複数のエリートサーバを管理するマスタサーバが存在する、という構造になる。

提案法の評価のため、まず TSP の 30 都市および 100 都市にそれぞれの手法を適用し、比較を行った。その結果、パラメータの調整により、階層サーバ・クライアント型モデルではエリートサーバにかかる負荷を分散させながらサーバ・クライアント型モデルと同等の探索性能が得られることが確認できた。

また本研究では、大規模な問題での効果を検証するため TSP の 2392 都市に、またより実用的な問題として画像高解像度化問題に適用した場合についても検討した。その結果、同期リング型モデルでは移民に非常に時間がかかり探索性能もそれほど良くなかったこと、非同期リング型モデルおよび非同期トーラス型モデルでは移民時間は少なかったものの探索性能が悪かったこと、サーバ・クライアント型モデルでは探索性能は良かったがエリートサーバに通信が集中するため、大きな問題では移民に多くの時間が必要となってしまうことがわかった。

それらに対して階層サーバ・クライアント型モデルでは、サーバ・クライアント型モデルと同等の探索性能を維持しながら移民時間の減少に成功しており、エリートサーバにかかる負荷を探索性能の劣化なしに分散させることができたことが確認できた。

以上のことから、階層サーバ・クライアント型モデルは、探索性能を維持しながら移民時間が少ない非常に優れた手法であることが確認できた。

謝 辞

本研究を遂行するにあたり、並列計算機 SGI Origin2000 および SGI Altix350 の利用を許可していただいた岩手県立大学メディアセンターのスタッフ皆様に感謝します。

参 考 文 献

- [1] J. H. Holland: *Adaption in Natural and Artificial Systems*, University of Michigan Press (1975)
- [2] D. E. Goldberg: *Genetic Algorithm in Search Optimization and Machine Learning*, Addison Wesley

- (1989)
- [3] E. Cantú-Paz: *Efficient and Accurate Parallel Genetic Algorithms*, Kluwer Academic publishers (2000)
 - [4] J. P. Cohoon, W. N. Martin and D. S. Richards: A multi-population genetic algorithm for solving the k-partition problem on hypercubes; *Proc. of ICGA-91*, pp. 244–248 (1991)
 - [5] R. Tanese: Distributed genetic algorithms; *Proc. of ICGA-89*, pp. 434–439 (1989)
 - [6] R. Tanese: Parallel genetic algorithm for a hypercube; *Proc. of ICGA-87*, pp. 177–183 (1987)
 - [7] C. C. Petty and M. R. Leuze: Theoretical investigation of a parallel genetic algorithm; *Proc. of ICGA-89*, pp. 398–405 (1989)
 - [8] 三木, 廣安, 中村: 遺伝的アルゴリズムの分散並列化に関する研究 (踏み石モデルによる分散遺伝的アルゴリズムの検討); 日本機械学会論文集 (A 編), Vol. 65, No. 638, pp. 2177–2182 (1999)
 - [9] 棟朝, 高井, 佐藤: 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; 情報処理学会論文誌, Vol. 35, No. 9, pp. 1815–1827 (1994)
 - [10] T. C. Fogarty and R. Huang: Implementing the genetic algorithm on transputer based parallel processing systems; *Parallel Problem Solving from Nature*, pp. 145–149 (1991)
 - [11] R. Hauser and R. Männer: Implementation of standard genetic algorithm on MIMD machines; *Parallel Problem Solving from Nature, PPSN III*, pp. 504–513 (1994)
 - [12] R. J. Collins and D. R. Jefferson: Selection in massively parallel genetic algorithms; *Proc. of ICGA-91*, pp. 249–256 (1991)
 - [13] B. Manderick and P. Spiessens: Fine-grained parallel genetic algorithms; *Proc. of ICGA-89*, pp. 428–433 (1989)
 - [14] P. Spiessens and B. Manderick: A massively parallel genetic algorithm, implementation and first analysis; *Proc. of ICGA-91*, pp. 279–285 (1991)
 - [15] D. E. Brown, C. L. Huntley and A. R. Spillane: A parallel genetic heuristics for the quadratic assignment problem; *Proc. of ICGA-89*, pp. 406–415 (1989)
 - [16] M. Gorges-Schleuter: ASPARAGOS: An Asynchronous Parallel Genetic Optimization Strategy; *Proc. of ICGA-89*, pp. 422–427 (1989)
 - [17] 木津, 澤井, 足立: 可変な局所集団の適応的探索を用いたパラメータフリー遺伝的アルゴリズムとその並列分散処理への拡張; 電子情報通信学会論文誌 (D-II), Vol. J82-D-II, No. 3, pp. 512–521 (1999)
 - [18] 足立, 澤井: 並列分散パラメータフリー遺伝的アルゴリズムにおける移民選択法の効果; 電子情報通信学会論文誌 (D-I), Vol. J83-D-I, No. 8, pp. 834–843 (2000)
 - [19] 廣安, 三木, 佐野, 谷村: 2個体分散遺伝的アルゴリズム; 情報処理学会研究報告, Vol. 2000, No. 85, pp. 37–40 (2000)
 - [20] 谷村, 廣安, 三木: Grid 計算環境における遺伝的アルゴリズムの提案; 情報処理学会論文誌, Vol. 43, No. 4, pp. 986–994 (2002)
 - [21] 中田: グローバルコンピューティングにおけるセキュリティ; *Computer Today*, Vol. 17, No. 99 (2000)
 - [22] P. パチェコ: MPI 並列プログラミング; 培風館 (2001)
 - [23] K. Kojima, M. Ishigame, G. Chakraborty, H. Matsuo and S. Makino: Asynchronous parallel distributed genetic algorithm with elite migration; *International Journal of Computational Intelligence*, Vol. 4, No. 2, pp. 105–111 ISSN:1304-2386 (2007)
 - [24] 森, 南元, 淵田, 村島: 遺伝的アルゴリズムによる時系列画像の1次元高解像度化; 電子情報通信学会論文誌 (D-II), Vol. J83-D-II, No. 5, pp. 1224–1331 (2002)

著者略歴

こじま かずのり
小嶋 和徳 (正会員)

1970年生。1993年3月秋田大学鉱山学部電子工学科卒業, 1995年3月同大学大学院鉱山学研究科電子工学専攻修士課程修了。1998年4月岩手県立大学ソフトウェア情報学部助手。2008年3月東北大学大学院工学研究科博士後期課程修了。2008年10月岩手県立大学ソフトウェア情報学部講師, 現在に至る。遺伝的アルゴリズムに関する研究に従事。博士(工学)。情報処理学会, 電子情報通信学会, 人工知能学会各会員。

いしがめ まさあき
石電 昌明

1944年生。1968年東北大学工学部電子工学科卒業, 1974年同大学大学院工学研究科博士課程修了。同年同大学応用情報学研究センター助手, 1978年松下電送(株)入社, 1988年秋田大学鉱山学部情報工学科助教授, 1998年岩手県立大学ソフトウェア情報学部教授, 現在に至る。信号処理, 画像処理, 知識工学の研究に従事。工学博士。電子情報通信学会, 画像電子学会, 日本音響学会各会員。

まさの りょうぞう
牧野 正三

1969年東北大学工学部電子工学科卒業, 1974年同大学大学院博士課程修了。同年同大学電気通信研究所助手, 1981年同大学応用情報学研究センター助手, 同助教授, 同大学大型計算機センター教授, 情報シナジーセンター教授を経て, 現在同大学大学院工学研究科教授。1984~1986年アメリカ合衆国 STL 客員研究員。言語情報を利用した音声認識・理解の研究, 音声データベース, 音声信号処理, 音声 CALL システム, 画像情報処理, 文字認識の研究に従事。工学博士。電子情報通信学会, 日本音響学会, 情報処理学会, 人工知能学会, 自然言語処理学会, IEEE, ESCA 各会員。