PC クラスタシステムにおける 並列遺伝的アルゴリズムのモデルの検討 *

廣安 知之 *1,
三木 光範 *1,
谷村 勇輔 *2

Characteristics of Models of Parallel Genetic Algorithms on PC Cluster System *

Tomoyuki HIROYASU*1 , Mitsunori MIKI*1 and Yusuke TANIMURA*2

*1 Faculty of Engineering, Doshisya University *2 Graduate Student, Doshisya University

In this paper, the characteristics of the typical two models of parallel genetic algorithms; the coarse grained model and the fine grained model, are compared. Especially, the parallel efficiency on PC clusters that have no more than 10 CPUs is discussed. The cluster used in this study has two kinds of network architectures; FastEthernet and Myrinet. Through the numerical examples, these models are examined and discussed. The followings were made clarified. In the fine grained model, the master slave model, the ideal parallel efficiency is not 100 %. It is concluded that the fine grained model is not suitable for this kind of cluster. On the other hand, in the coarse grained model, the population is divided into sub populations. The communication does not happen frequently. The coarse grained model is suited for this type of clusters. However, even in the coarse grained model, the data transfer schedule is needed because of the data stacking.

Key Words : Optimization, Parallel Processing, Genetic Algorithms, PC Clusters

1. は じ め に

近年，コモディティハードウエアと呼ばれる一般に市販されているハードウエアの性能が飛躍的に進歩している．これまで並列計算機というとスーパーコンピュータに代表されるような通常の計算では処理できないような数値計算を行う特別なものであった．それに対して，先に述べたような一般に市場に流通しているハードウエア，特にネットワークの性能向上を背景にいわゆるコモディティアーキテクチャによる並列計算機の構築が盛んに行われている．ロスアラモス研究所の Avalon [(1)]や RWCP のクラスタ型並列計算機 [(2)]がその代表例である．

最適化問題 [(3)]は制約条件内で目的関数を最小化もしくは最大化するような設計変数を探索する問題である．工学的問題の中には非常に数多くこの最適化問題が見られ，特に各分野での設計問題では最適化技術を利用する場合が非常に多い．よって最適化問題を解決するアプリケーションは工学的に非常に重要なアプリケーションの一つであると言えその並列化の意義は高い．

遺伝的アルゴリズム (Genetic Algorithm: GA) は生物の遺伝と進化の仕組みを模擬した最適化手法の一つである [(7)]．GA では多点による確率探索を特徴としておりこれにより，局所解が多く存在する多峰性の高い問題や設計空間が離散的な問題にも対応できる．このように GA は強力な最適化手法の一つである反面，繰り返し計算を非常に多く必要とするという欠点を有する．この問題の解決方法の一つが GA の並列処理である．

GA の並列化に関連した研究はいくつか見られる [(9)~(11)]．そこでは，GA の並列処理においてはいくつかのモデルが存在しそれぞれのモデルにおいて長所・短所があることが報告されている．しかしながら，それらの研究の多くが超並列計算機上での GA の並列化に関するものであり，近年盛んに実用的に使われるようになっているプロセッサ数が１０を切るような小規模 PC クラスタに関するものはほとんど見られない．

そこで本研究ではプロセッサ数が 10 を超えない PC クラスタシステム上での GA の特性について検討する．特に GA を並列処理する際に使用されるモデルとして代表的な粗粒度モデルと細粒度モデルに焦点をあてる．次章ではこれら二つのモデルについて説明を行い，続く章ではテスト関数にこれらのモデルによる GA を実装して各モデルにおける GA の長所および短所を検討

* 原稿受付 平成 13 年 5 月 21 日
*1 同志社大学工学部知識工学科 (〒 610-0321 京都府京田辺市多々羅都谷 1-3)
*2 同志社大学大学院
Email: tomo@is.doshisha.ac.jp

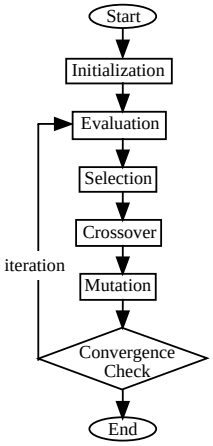

Fig. 1　Flow of genetic algorithms

する．

2. 並列遺伝的アルゴリズムのモデル

2·1　遺伝的アルゴリズム　遺伝的アルゴリズム (Genetic Algorithms: GAs) は生物の遺伝と進化の仕組みを模倣した最適化手法である．図 1 に典型的な GA の流れを示す．これらの操作は遺伝的操作と呼ばれる．各遺伝的操作にはいくつもの手法が考えられている．本研究では一点交叉およびルーレット選択およびエリート保存戦略といったその中で最も基本的な方法を選択している．

GA は多点探索手法であり，潜在的に並列性を含んでいるアルゴリズムであると言われている [7]．それ故に GA には様々な並列化手法が存在するがそれらは 3 種類に大別できる [9]．それらをまとめて表 1 に示す．

表 1 に示す手法の中で Method3 は実際には効率的な実装が非常に難しい．プロセッサ間の通信量も多いために並列効率も悪い．よって，実際には Method 1 と Method 2 が GA の並列化の手法となる．通常，通信量の観点から，Method1 を細粒度モデル (fine grained model)，Method2 を粗粒度モデル (coarse grained model) と呼んでいる．

次節ではこれらの細粒度および粗粒度モデルについて説明する．

2·2　粗粒度モデル　粗粒度モデル (coarse grained model) は別名，島モデルと呼ばれている．このモデルにおいては GA での個体群が複数のサブ個体群に分割される．このサブ個体群が島と呼ばれる．島内では通常の遺伝的操作が行われる．通常の GA と異なるのは数世代後に移住と呼ばれる操作が行われる点である．移住では各島内ではいくつかの個体が選択され他の島へ移動もしくは他の島で選択された個体とで選択される．この移住という操作により個体の多様性が維持される．このモデルでの典型的な GA の流れを図 2 に示す．

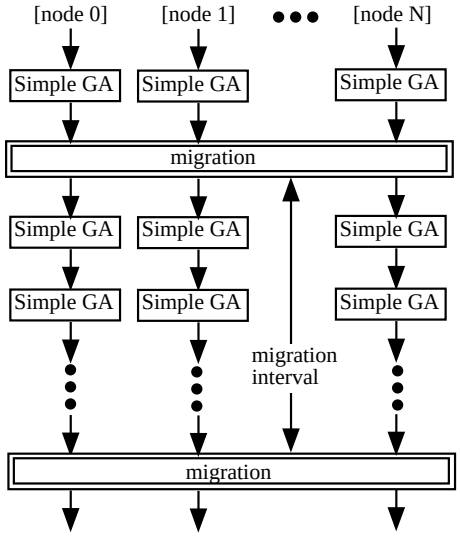

Fig. 2　Flow of coarse-grained model

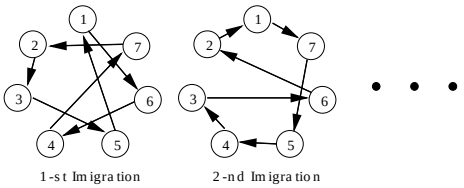

Fig. 3　Migration method

移住個体はランダムに選択され，選択される個体数は島内の個体数と移住率との積で決定される．移住方法にはいくつかの方法が存在する．Nang and Matusuo [9] はこれらの方法についてまとめを行っている．粗粒度モデルにおいては島間の通信が移住およびそれに伴う通信以外には起こらない．そのため，このモデルでは高い並列化効率が得られることが期待できる．

本研究では移住の際に個体が移住する島の順序は固定せずその都度ランダムに決定される．その様子を図 3 に示す．それ故に通信の際にロック現象が起こる場合が考えられる．一つのプロセッサは同時にはデータを送受信できないために起こる現象である．例えば偶数個の島が存在する場合には実際には移住には 2 段階以上の手続きが必要となり，奇数の島が存在した場合には実際には移住は 3 段階以上の手続きが必要となる．このロック現象を図 4 に示す．このロック現象は計算時間を消費し並列化効率を下げてしまう．このロック現象は次章で数値計算例により確認する．

このモデルの並列化の評価は非常に難しい．なぜならば，単一の母集団モデルに対して，複数母集団モデルに変更するだけで必要な計算量自体が減少するからである．すなわち，単一のプロセッサにおいても本モデルに変更することで 100%以上の速度向上が得られ

Table 1　Parallel methods of GAs

Method 1	Divide the population into sub populations and perform GA in each sub population. (Coarse grained model)
Method 2	Perform the evaluation part in parallel. (Fine grained model)
Method 3	Perform the genetic operation such as selection, crossover and mutaion in parallel.

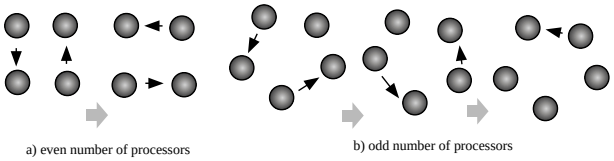

Fig. 4　Lock situation of data transfer

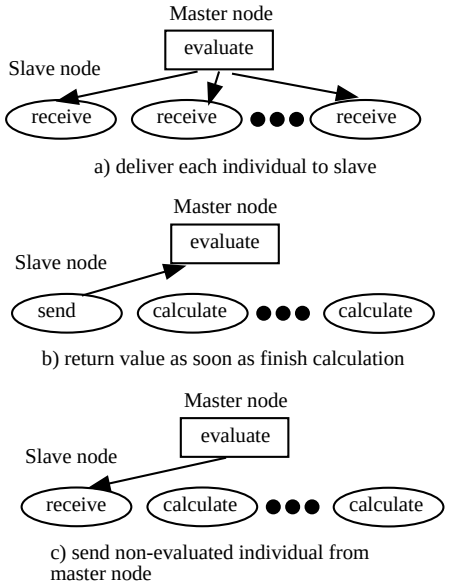

Fig. 5　Flow of fine-grained model

る．それ故に本モデルにより並列処理を行った場合に，得られた計算速度の向上が，並列にしたことにより得られたものなのかモデルの変更による効果なのか明確にならない場合が多い．

2·3　細粒度モデル　　GA においては適合度の値を求める部分に最も計算時間を要する．よって，この部分を並列に処理することは最も基本的な方法である．細粒度モデルはその構成方法からマスタ・スレーブモデルと呼ばれる場合がある．このモデルにおいてはマスタプロセッサが適合度を求める部分以外の遺伝的操作を行い，適合度を求める部分でデータを各スレーブに送り各スレーブは適合度計算を行って，そのデータをマスタに返送するのである．データを返送されたマスタはそのスレーブに新たな個体のデータを送付する．このモデルでの典型的な GA の流れを図 5 に示す．

このモデルによる GA に関する研究は多数行われている．例えば Forgaty ら [6]はこのモデルが個体数が非常に多い場合に非常に効率が良いことを報告している．

PGA pack [4]は Argonne National Loboratory で開発されたこのモデルによる汎用ソフトウエアパッケージで広く使用されている．このパッケージでは通信ライブラリとして MPICH [5]が使用されており C および Fortran で使用可能である．本研究では数値計算例の粗粒度モデルの実装として PGA pack を使用している．ただし，本パッケージでは厳密に言えば粗粒度モデルであるのは 3 プロセッサ以上の場合であり，2 プロセッサの場合には両方のプロセッサにおいて適合度計算を行うモデルとなっている．

このモデルの優位な点は実装の容易さであろう．個体の数が膨大であっても単一母集団 GA の実装をあまり変化させることなく適応することが可能である．また本モデルではロードバランスが取りやすいという点も長所であると言われている．

しかしながら本モデルでは１０プロセッサ程度では並列化効率の向上は期待できない．なぜならば，1 つのプロセッサがマスタとして占有されてしまうからである．例えば，8 プロセッサある場合であれば，最高でも 7/8 しかこのモデルでは並列化効率がえられないこととなる．

また，粗粒度モデルと比較して通信のデータのサイズが小さくかつ通信が頻繁に行われるため，並列化効率の向上は通信時間の大きさによることとなる．例えばここで 1 データをマスタからクラスタへ送受信するのに c_{time} 必要であり，それらを評価するのに $n * c_{time}$ 必要であると仮定する．その場合にはマスタは c_{time} ごとにスレーブにデータを送信するわけだが $(n+1) * c_{time}$ 後にはスレーブからデータが返送され，マスタはその返送されたスレーブに対してデータを送信することとなる．すなわち高々 $(n+2)$ 個のスレーブが使用されるだけなのである． $n < 1.0$ の際のプロセッサ数と総計算時間との関係を図 6 に示す．ただし

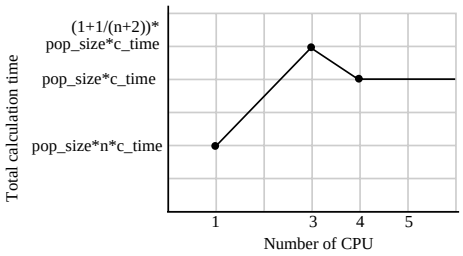

Fig. 6　Total calculation time (n 1.0)

Table 2　Spec of 8 PC cluster system

CPU	Pentium II (Deschutes, 400MHz)× 2
Memory	128MB
OS	Linux2.2.10
Network	FastEthernet TCP/IP　Myrinet GM1.02
Communica-tion library	MPICH1.1.2

ここでは２プロセッサの場合は省略している．この図からもわかるように， $n < 1.0$ の際には 3 スレーブ以上は使用されないため，いくらスレーブの数を増加させても速度向上は得られない．

3. 遺伝的アルゴリズムにおける細粒度モデルと粗粒度モデルとの比較

本章では遺伝的アルゴリズムを並列処理にて行う際の一般的なモデルである細粒度モデルと粗粒度モデルとの比較を数値計算例を通じて行う．

3·1　使用したクラスタシステムの概要　本研究におけるシミュレーションでは表 2 に示すような８プロセッサからなる PC クラスタシステムを使用している．各ノードは FastEthernet と Myrinet によって接続されている．このようなタイプの PC クラスタは今後広く活用されていくものと考えられる．本システムでは FastEthernet はスイッチングハブに接続され Myrinet は Myrinet スイッチにそれぞれ接続されている．Myrinet は Myricom 社の開発したギガビットネットーワークの一つであり，ドライバを用意することにより０コピー通信を行うことができる．並列通信ライブラリには MPICH [(5)]を利用している．

図 7 および 8 にはそれぞれバンド幅とレイテンシを示している．これらは FastEthernet においては MPICH 1.1.2 を Myrinet においては GM 1.0.2 に MPICH 1.1.2 を被せたものの結果である．これらの結果から Myrinet は FastEthernet と比較して 3 倍の性能が得られていることがわかる．しかしながら，FastEthernet ではほとんどハードウエアの性能の上限が得られているのに対して Myrinet では得られていない．これは各ネットワークにおけるドライバの性能に依存している．よって FastEthernet ではドライバが十分開発されているのに対して，Myrinet ではまだ十分であるとは言えない．

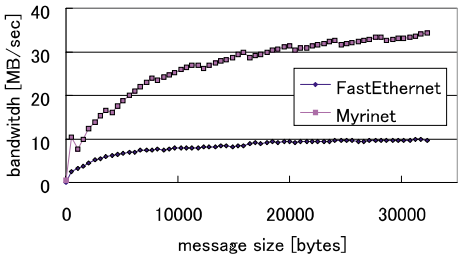

Fig. 7　Band width (20000 bytes)

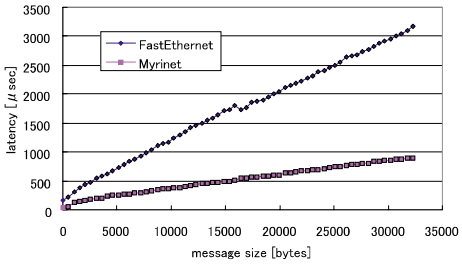

Fig. 8　Latency (20000 bytes)

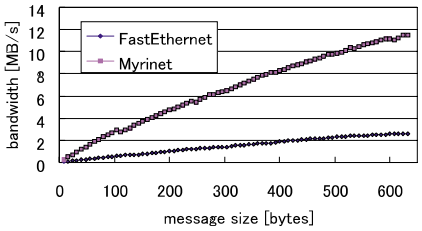

Fig. 9　Band width (400 bytes)

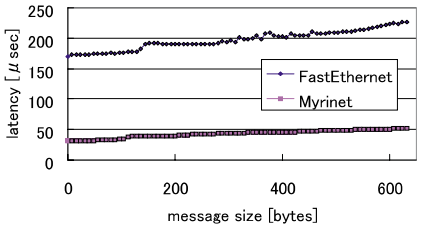

Fig. 10　Latency (400 bytes)

本研究では各遺伝子のビット表現として unsighned interger を使用して実装を行っている．よって例えば 50 ビットの遺伝子の場合，メッセージサイズは 200 バイトとなる．同様に 150 ビットの場合には 600 バイトのサイズとなる．図 9 および 10 には 400 バイト付近のバンド幅とレイテンシを示している．これらの図からも分かる通り FastEthernet と Myrinet の差は 20000 バイト付近よりも 400 バイト付近の方が大きくなっている．よって，GA などの計算を行うような場合には 400 バイト付近の通信が多くなるものと考えられ，Myrinet の利用が非常に有効であるものと思われる．

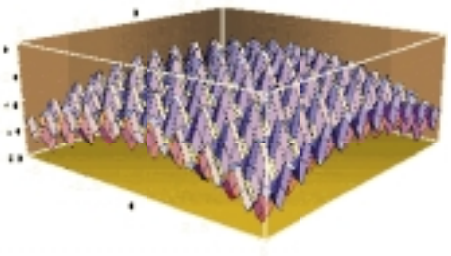

Fig. 11　Rastrigin function

3·2　テスト関数　並列 GA における粗粒度モデルと細粒度モデルの特性の比較を行うために式 1 に示す Rastrigin 関数 [8]をテスト関数としてシミュレーションを行う．

$$f = -10n - \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i)). \qquad (1)$$

式 1 では f が適合度であり最適値は原点 O にありそのときの適合度 f は 0 である．２設計変数問題における Rastrigin 関数の景観を図 11 に示す．

Rastrigin 関数は設計領域全体にわたって局所解が多く存在するのに対して，設計変数間に依存がないため，GA によって比較的容易に大局的最適解が探索できる．そのため GA の特性を知るために適したテスト関数であると言える．

本研究では５および１５設計変数の Rastrigin 関数最適化問題に対して各並列 GA のモデルを適応している．各設計変数の値を表すために１０ビットを使用するため，１遺伝子長は５０および１５０となる．交叉率は 0.6 とし突然変異率は１遺伝子あたり突然変異が１度起こるような確率とするため５設計変数に対して 1/50，１０設計変数に対して 1/150 としている．粗粒度モデルにおいては移住率を 0.15 とし移住間隔を 5 としている．個体数は５設計変数問題に対しては 960 を，１５設計変数問題に対しては 4800 を設定した．細粒度モデルにおいては個体数は 1000 と設定している．解を求めるために必要な総計算時間はこの個体数に依存している．

GA においてはほとんどの計算時間を適合度計算に費やす．しかしながら適合度計算にあまり時間を要しないような問題においては相対的に通信コストが増大するために並列化効率が減少してしまうものと考えられる．よって，この適合度計算に要する時間によって得られる特性も異なることとなる．本研究では適合度計算に要する時間を調整するために適合度計算部分に待機時間を設定しこれをパラメータとしている．その待機時間を本研究では 0.0，0.0001，0.001，0.01[s] としたこの待機時間が短い場合には通信コストは高くなり，逆に長い場合には通信コストは小さくなる．これらの計算コストは表 3 にまとめた図 12 のようなトラス構造を FEM 解析した際のものに対応している．

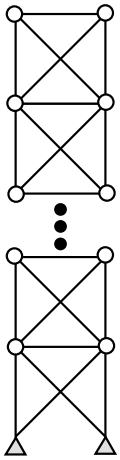

Fig. 12　Truss structure

Table 3　Equivalent calculation cost

Calculation Time [s]	Truss structure		
	# of nodes	# of elements	# of stages
0.0001	8	15	3
0.0010	66	160	32
0.0100	622	1550	310

3·3　粗粒度モデル　本節では粗粒度のモデルを Rastrigin 関数に適応している．島数はプロセッサ数と同一とした．図 13 から 16 には５および１５設計変数の問題のプロセッサ数と総計算時間との関係を表している．図 13 および 14 は Ethernet による結果であり，図 15 および 16 は myrinet による結果である．

図 13 および 14 から待機時間が短い場合には総計算時間が 3 プロセッサの場合よりも 2 プロセッサの方が短いことが明らかである．この傾向は図 14 に示されるようなデータサイズが大きな場合にはより顕著である．2·2 で説明したロック現象がこの原因であると考えられる．そのため粗粒度モデルにおいてはこのロック現象が最も重要な問題であると考えられる．この問題を解決するには適切なスケジューリングが必要であろう．図 15 および 16 には Myrinet による結果が示されているが待機時間が短くデータが大きくなっても通信コストは非常に小さいため総計算時間はプロセッサを多数使用することによって減少する．

3·4　細粒度モデル　本節では Rastrigin 関数を細粒度モデルによる GA を適応している．５および１５設計変数の問題のプロセッサ数と総計算時間との関係を図 17，18，19，20 に示す．図 17 および 18 は Ethernet によるものの結果であり，図 19 および 20 は Myrinet によるものの結果である．

待機時間が短い場合には総計算時間は Myrinet の場合でもそれほど短縮されない．待機時間が短い場合の

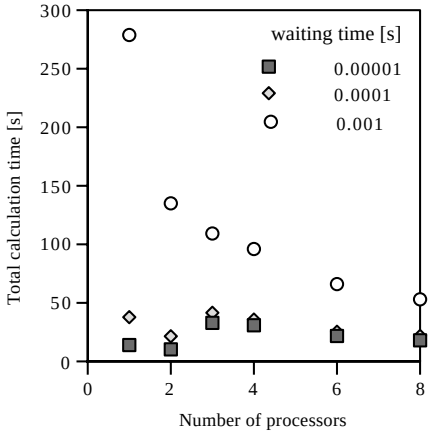

Fig. 13 Total calculation time and number of processors (Ethernet 50 bits)

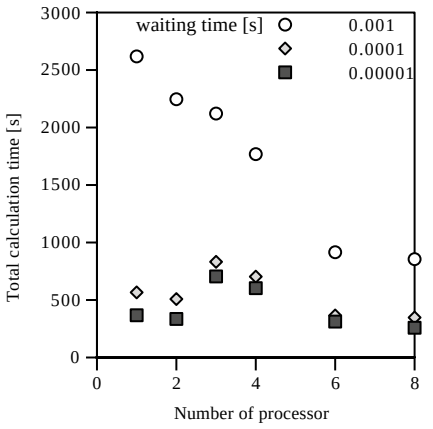

Fig. 14 Total calculation time and number of processors (Ethernet 150 bits)

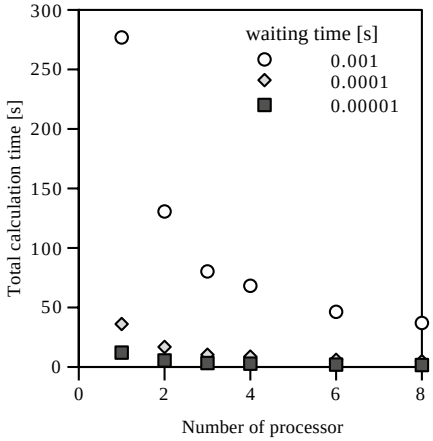

Fig. 15 Total calculation time and number of processors (Ethernet 50 bits)

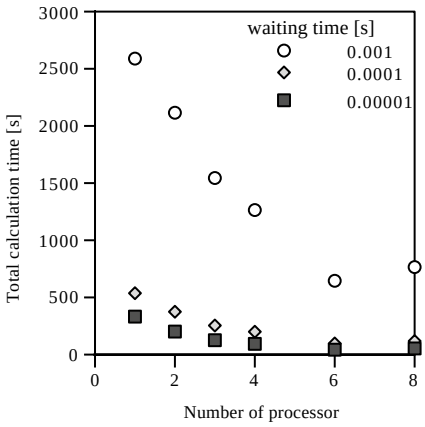

Fig. 16 Total calculation time and number of processors (Ethernet 150 bits)

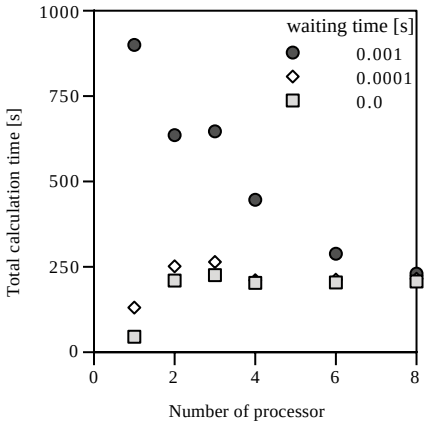

Fig. 17 Total calculation time and number of processors (Ethernet 50 bits)

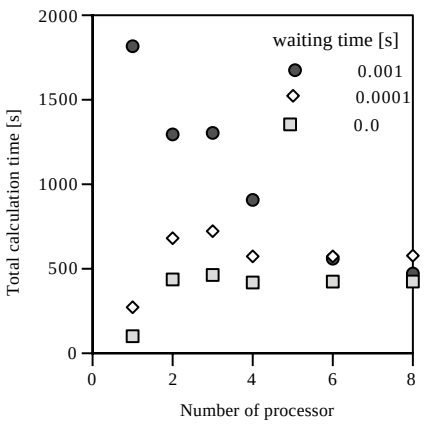

Fig. 18 Total calculation time and number of processors (Ethernet 150 bits)

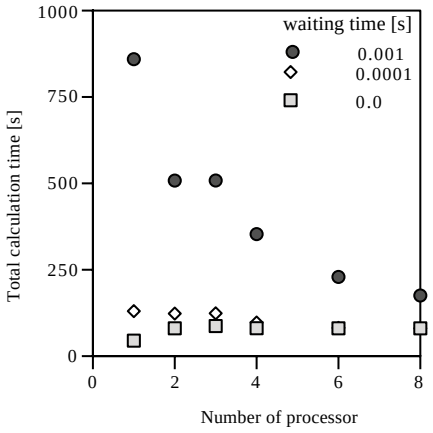

Fig. 19　Total calculation time and number of processors (Ethernet 50 bits)

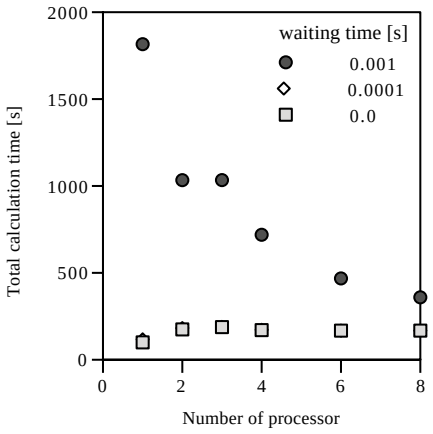

Fig. 20　Total calculation time and number of processors (Ethernet 150 bits)

結果（図 17 から 20）はおおよそ図 6 で示した傾向と一致している．すなわち 2·3 で検討したように一部のプロセッサのみが使用されているものと考えられる．これを確認するために Ethenet において待機時間が短い場合と長い場合において適合度計算がどのプロセッサで何度行われたかを調査する．

それらの回数を表 4 および表 5 に示す．表 4 から待機時間が短い場合には 3 プロセッサのみが使用されていることが明らかである．一方，待機時間が長い場合にはスレーブに使用された数のプロセッサが使用されている．

しかしながら，待機時間が長い場合にでも総計算時間は大きくは減少しない．これは細粒度モデルにおいてはマスタプロセッサが適合度計算を行なわないからである．

１０プロセッサを超えないような PC クラスタシステムではこのモデルは適していないと言え，超並列計算機上でのモデルであると言えよう．

4. 結　　言

本研究では遺伝的アルゴリズムを並列にて処理する際の代表的なモデルとして細粒度モデルおよび粗粒度モデルを説明し，その特徴を検討した．また 8 プロセッサからなる PC クラスタシステム上で数値計算シミュレーションを行うことでこれらのモデルについて検討した．

その結果，細粒度モデルでは本研究で使用したようなプロセッサ数の少ないシステムでは有効でないことが明らかとなった．特に適合度計算に時間がかからないような場合にはどんなに多くのプロセッサを用意しても実際には 3 プロセッサ程度しか使用されない．また，マスタとして 1 プロセッサが占有されるため高い並列化効率が期待できない．

粗粒度モデルはこのような中小規模の PC クラスタモデルに適しているモデルと言える．粗粒度モデルではデータ通信が頻繁に発生しないからである．しかしながら，このモデルにおいても適合度計算にあまり時間を要しないような問題ではデータのロック現象により並列化効率が減少してしまう．そのため，このような場合では通信のスケジュールを最適化する必要がある．

謝　　辞

本研究は文部科学省からの補助を受けた同志社大学の学術フロンティア研究プロジェクトにおける研究の一環として行った．

文　　献

(1) http://swift.lanl.gov/Internal/Computing/Avalon/.

(2) http://www.rwcp.or.jp/lab/pdslab/clusters/home.html.

(3) http://www-c.mcs.anl.gov/home/otc/Guide/faq/nonlinear-programming-faq.html.

(4) http://www.mcs.anl.gov/pgapack/.

(5) http://www-c.mcs.anl.gov/mpi/mpich.

(6) T. Forgaty, C., and R. Huang. Implementing the genetic algorithm on transputer based parallel processing systems. In *Proceedings of Parallel Problem from Nature*, pp. 145–149, 1991.

(7) D. E. Goldberg. *Genetic Algorithms in search, optimization and machine learning.* Addison-Wesly, 1989.

(8) H. Muhlenbein and D. Schilerkamp-Vosen. Predictive models for the breeder genetic algorithm. *Evolutionary Computation*, Vol. 1, No. 1, pp. 25–49, 1993.

Table 4 Number of function call of each slave processor (waiting time = 0.0)

Number of Processors	Master	Slave 1	Slave 2	Slave 3	Slave 4	Slave 5	Slave 6	Slave 7
1	1000	—	—	—	—	—	—	—
2	500	500	—	—	—	—	—	—
3	—	504	496	—	—	—	—	—
4	—	458	355	187	—	—	—	—
6	—	450	341	207	1	1	—	—
8	—	448	337	211	1	1	1	1

Table 5 Number of function call of each slave processor (waiting time = 0.001)

Number of Processors	Master	Slave 1	Slave 2	Slave 3	Slave 4	Slave 5	Slave 6	Slave 7
1	1000	—	—	—	—	—	—	—
2	500	500	—	—	—	—	—	—
3	—	502	498	—	—	—	—	—
4	—	334	333	333	—	—	—	—
6	—	200	200	200	200	200	—	—
8	—	143	143	143	143	143	143	142

(9) L. Nang and K. Matsuo. A survey on the parallel genetic algorithms. *J.SICE*, Vol. 33, No. 6, pp. 500–509, 1994.

(10) T. Starkweather, D. Whitley, and K. Mathias. Optimization using distributed genetic algorithms. In *Proceedings of Parallel Problem from Nature*, pp. 176–184, 1991.

(11) R. Tanese. Distributed genetic algorithms. In *Proceedings of 3rd International Conference on Genetic Algorithms*, pp. 432–439, 1989.