確率学習による適応度評価を導入した遺伝的アルゴリズムに基づく動的負荷均衡

棟朝雅晴† 高井昌彰† 佐藤義治†

分散システムを有効に利用するために，システム内の各計算機の負荷を一様化することが必要である．分散管理型の動的負荷均衡アルゴリズムにおいては，それぞれの計算機で独立して負荷状態の観測およびタスク転送の決定を行う．本論文では，負荷の重い計算機からのタスク転送要求をマルチキャストで実現した分散管理型の動的負荷均衡アルゴリズムを提案する．本手法の特徴は，タスク転送要求の送出先リストを符号化し，適応度評価に確率学習オートマトンを組み合わせた遺伝的アルゴリズムを用いることで転送要求の成功率を向上させることにある．シミュレーション実験により従来の手法との比較検討を行い，提案する手法がシステムの平均応答時間，タスク転送要求の成功率，および動的な負荷変化への適応性の点において優れていることを示した．

A Dynamic Load Balancing Scheme Using a Genetic Algorithm with Stochastic Learning

MASAHARU MUNETOMO,† YOSHIAKI TAKAI † and YOSHIHARU SATO †

It is necessary to balance the load of each processor in a distributed system in order to utilize the system effectively. In a dynamic load balancing algorithm with distributed control, each processor observes load status of the system and dispatches tasks independently. We propose a dynamic load balancing scheme with distributed control which employs stochastic multicast messages. We encode a sending set of the requests for task dispatch into a binary string to which genetic operations with stochastic learning are applied in order to increase the probability for the requests to be accepted. Through simulation studies, we compared our scheme with some conventional load balancing methods. The results show the effectiveness of our scheme concerning mean response time of the tasks, success rate of the requests, and adaptability to environmental changes.

1. はじめに

分散システムにおいて計算機利用率の向上と応答時間の短縮のために，負荷均衡アルゴリズムに関する研究が数多くなされている．負荷均衡アルゴリズムは大きく静的負荷均衡と動的負荷均衡に分類できる[1]．静的負荷均衡アルゴリズムは，発生するタスクや分散システムに関してあらかじめ得られている情報を用いてタスクの割当てを決定することで，システムの利用率を向上させる．これに対して動的負荷均衡アルゴリズムは，処理の進行に伴いタスクを負荷の重い計算機から負荷の軽い計算機へ転送することで計算機間の負荷を均一化し，間接的にシステムの有効利用をはかる．

動的負荷均衡は，1台の計算機で集中して負荷情報の管理とタスク転送の決定を行う集中管理型と，それぞれの計算機で独立して負荷情報の管理とタスク転送の決定を行う分散管理型に分けられる．システムを構成する計算機が数台程度の場合，集中管理型により十分な負荷均衡を行うことができるが，システムが大規模なものとなるのに伴い集中管理が困難となることは容易に推測される．本論文は，大規模分散システムへの適用を目指した分散管理型の動的負荷均衡を対象とする．

分散管理型の動的負荷均衡は，タスク転送の要求をタスクの送り手，すなわち負荷の重い計算機から送出する送り手主導 (sender-initiative) の方式[2),3)]と，タスクの受け手，すなわち負荷の軽い計算機から送出する受け手主導 (receiver-initiative) の方法に分類できる[3)]．

送り手主導の方式において，ブロードキャストによりタスク転送の要求を実現することには，負荷の軽い計算機がシステム中に存在する場合，必ずそれを発見できる利点がある．しかし大規模な分散システムでは

† 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

通信量が膨大となり，明らかに非効率的である．また，ユニキャストを用いた実現として，受け入れ先が見つかるまでタスク転送の要求を一定限度内で繰り返し送出する方法が考えられる．この場合，システム中で多くの計算機が過負荷状態にある時には，高い確率で要求が拒絶され，無駄な要求が繰り返し送出される欠点がある．従ってシステムサイズが比較的大きい場合には，全体の中から選ばれた複数の計算機に対してタスク転送の要求を同時に送出する，マルチキャストによる実現が適当であると考えられる．

マルチキャストを用いて転送要求を行う場合問題となるのは，どのような計算機群に対して要求を送出するかということである．これにより要求の成功率（負荷の軽い計算機が少なくとも1つ発見される確率）が大きく変化する．固定された一定の計算機群に要求を送るという方法では，ある計算機がいつまでも負荷の軽い計算機を発見できない可能性がある．

送り手主導の動的負荷均衡をマルチキャストで実現する場合，タスク転送の要求を送出すべき計算機群を負荷状況の変化に対応して速やかにかつ安定して求められる機構が必要である．そこで，我々は遺伝的アルゴリズム（Genetic Algorithms, 以下GAと略す）[4]の持つ頑健性に着目し，さらに確率学習オートマトン（Stochastic Learning Automaton, 以下SLAと略す）を適応度の評価に用いることで，動的に変化する環境の中で効率良く負荷の軽い計算機を発見する手法であるGeSLA動的負荷均衡アルゴリズムを開発した．

本手法では，タスク転送の要求を送出する計算機群をビット列の形に符号化し，これをGAにおける個体と見なす．ある計算機でタスク転送が必要になると，あらかじめ割り当てられた個体集団の中から適応度に比例した確率で1つの個体が選出され，そこに示されている計算機群に対して要求がマルチキャストされる．

個体の適応度はタスク転送要求の成功・失敗の結果に依存する．適応度の最も簡単な計算方法として，過去の一定回数の成功率を用いる方法がある[5)~7)]．この場合，ある個体を選択してタスク転送要求が失敗してもその個体の適応度のみが減少し，他の個体の適応度には影響しないため，長い間使われていない個体の適応度が不正確になるという問題点がある．

GeSLA動的負荷均衡アルゴリズムにおいては，適応度評価にSLAを用いることで，タスク転送要求の成功・失敗の結果を集団内すべての個体の適応度値に反映させる．この結果，直接使用されていない個体に関しても適応度値の変化が伝搬し，動的に変化する環境に速やかに適応することが可能となる．

2. GeSLA動的負荷均衡アルゴリズム

本アルゴリズムで前提とするシステムは，自立した計算機（以下ではノードと呼ぶ）が通信リンクを介して結合されている分散計算システム[1)]である．ノードの待ち行列モデルを**図1**に示す．Distributorはタスクの転送を決定する．Waiting queue, Task queueはそれぞれタスク転送の前と後における待ち行列である．ノード内でタスクが発生すると，タスクはWaiting queueに入る．以下では，負荷状態の決定およびタスク転送に要する時間はタスクの実行に要する時間に比して少ないものと仮定する．つまり，Waiting queueにおける待ちタスクの数はTask queueに比べて無視しうるほど少ないものとする．

負荷状態の観測については，*Heavy, Normal, Light*の3状態からなる方法[8)]を採用し，その判断基準として，Task queueの長さ$L(L\geq 0)$を用いた．つまり，閾値L_1，$L_2(L_1<L_2)$により負荷状態は次のように決定される．

$$\begin{cases} 0\leq L<L_1 & \rightarrow \quad Light \\ L_1\leq L<L_2 & \rightarrow \quad Normal \\ L_2\leq L & \rightarrow \quad Heavy \end{cases} \qquad (1)$$

本手法では，あるノードでタスクが発生し，かつノードの負荷が*Heavy*であるならば，Distributorは負荷が*Light*のノードを探し，そのTask queueへ向けてタスクを転送する．従って，タスクは実行前に高々一度しか転送されず，実行途中で転送されることはない．

タスク転送は以下の4種類のメッセージを用いて実現される．

Request タスク転送要求
Accept タスク転送要求の受け入れ
Reject タスク転送要求の拒絶
Dispatch タスク転送（転送対象のタスクを含む）

図2にGeSLA動的負荷均衡アルゴリズムの概念図を示す．図の$P0$〜$P7$はそれぞれのノードを表している．タスク転送要求をする*Request*メッセージの送出先は{0, 1}からなる文字列の形に符号化されている．

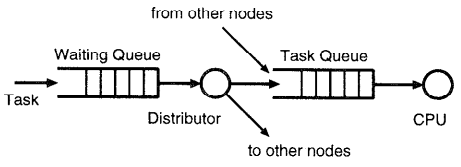

図1 ノードの待ち行列モデル
Fig. 1 A queueing model of a node.

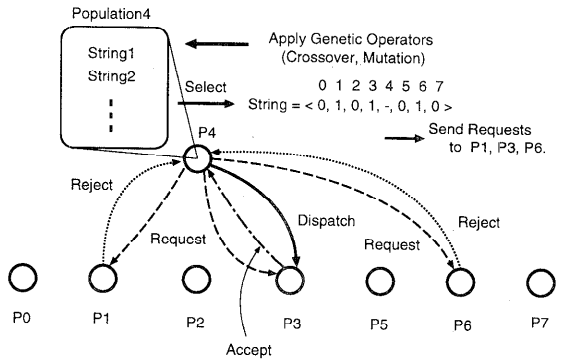

図2 GeSLA 動的負荷均衡アルゴリズムの概念図
Fig. 2 Overview of the GeSLA load balancing algorithm.

文字列の i 番目の文字が1ならばノード Pi にメッセージを送出し，0ならば送出しない．この文字列が遺伝的操作の対象となる個体となり，各ノードごとに比較的少数（10～20個程度）の個体集団を構成する．ここで各個体に対して，それが選択される確率として適応度値が与えられる．従って，1つの個体集団全体における適応度値の和は1である．初期集団はランダムに生成された n 個の個体により構成され，適応度の初期値として $1/n$ が与えられる．

あるノードで新たにタスクが発生し，そのノードの負荷が *Heavy* である場合，個体集団から確率的に選択された個体の文字列の内容に従って *Request* メッセージがマルチキャストで送出される．成功（*Accept* メッセージの受信）または失敗（*Reject* メッセージの受信）の結果に依存して，SLA を用いた適応度値の更新が行われる．これにより成功率の高い個体の適応度値は増加し，逆に成功率の低い個体の適応度値は減少する．さらに集団に対して遺伝的操作である crossover と mutation が適用され，新たな個体が生成される．

本アルゴリズムにおいては，同じ文字列を持つ個体に対して異なった適応度が割り当てられるのを防ぐため，個体の重複は許されない．直接重複の検出を行い，検出された場合には mutation をさらに適用することで重複を除去する．個体の多様性を維持する方法としてはニッチング[9]などが提案されているが，完全に重複を排除することはできない．

GeSLA 動的負荷均衡アルゴリズムの具体的な手順を以下に示す．

1. タスクの発生時に自ノードの負荷状態の観測を行い，それに従って以下の処理を実行する．
   - 負荷が *Heavy* の場合，以下の2～5に示されるタスク転送処理を実行する．
   - 負荷が *Light* または *Normal* の場合にはタスクを自ノードの Task queue に移す．その後，再び1に戻り，新たなタスクの発生を待つ．
2. 個体の適応度値に応じた確率に従って，集団内から1個体を選択する．
3. タスク転送要求のため，選択された個体の記述に従って，*Request* メッセージを送出する．
4. タスク転送要求の成否により，以下の処理を行う．
   - タスク転送要求が成功した場合，すなわち負荷が *Light* であるノードが少なくとも1つ見い出され，そこから *Accept* メッセージが戻ってきた場合には，そのノードの Task queue へ向けてタスクが転送され，SLA による適応度値の更新が行われる．*Accept* メッセージを返したノードが複数存在する場合にはランダムに1つを選択する．
   - タスク転送要求が失敗した場合，すなわち戻ってきたメッセージがすべて *Reject* である場合には，タスクの転送は一切行われず，タスクは自ノードの Task queue に移される．その後，SLA による適応度の更新および遺伝的操作が集団に対して適用され，新たな個体が生成される．集団内で最も適応度の低い個体は，新たに生成された個体と置き換えられる．
5. 再び1に戻り新たなタスクの発生を待つ．

3. SLA による学習

動的に変化する環境において適応度の評価に時間を要することは，正確な適応度の獲得を困難にする．動的負荷均衡アルゴリズムの場合，ある個体の適応度をタスク転送要求の成功率として直接評価するためにはその個体に従ったタスク転送要求を繰り返し送出しなくてはならないが，その間に対象とするシステム自体の状態が変化してしまう．GeSLA 動的負荷均衡アルゴリズムでは，適応度評価に SLA を採用することで，変化する環境に適応して逐次的に適応度を調整することを可能にしている．

ここで m 個の個体から成る集団内における各個体の適応度値を $p_k(k=1, 2, \cdots, m)$ とする．タスク転送要求の結果が戻ってきた際，その成否により p_k に対して linear reward-penalty scheme[10]が適用される．i 番目の個体に基づくタスク転送要求の結果，負荷が *Light* であるノードが少なくとも1つ見出された場合にその要求は成功したと見なされ，式（2）に従って p_i の値を増加させる．

$$p_i(n+1)=p_i(n)+\sum_{j\neq i}^{m} f(p_j(n)),$$

$$p_j(n+1)=p_j(n)-f(p_j(n)),\ \forall j\neq i. \qquad (2)$$

負荷が *Light* であるノードが1つも見つからなかった場合には失敗と見なされ，式（3）に従って p_i の値を減少させる．

$$p_i(n+1)=p_i(n)-\sum_{j\neq i}^{m} g(p_j(n)),$$

$$p_j(n+1)=p_j(n)+g(p_j(n)),\ \forall j\neq i. \qquad (3)$$

ここで $f(\cdot)$, $g(\cdot)$ は以下に示される非負の連続関数である．

$$f(p)=ap/r,\ g(p)=b/(m-1)-bp. \qquad (4)$$

式（4）において，$0<a<1, 0<b<1$ であり，r はタスク転送要求の送出数（選択された個体の文字列中での1の数）である．$f(p)$ に関しては全体を r で割った形となっているが，これは，より少ない転送要求で成功した個体に，より大きな適応度値を割り当て，タスク転送に使われるメッセージ数の少ない個体を有利にするためである．また $g(p)$ は，行動 i が失敗した場合に p_i の値を減少させ，$p_j(j\neq i)$ の値を増加させるとともに一様分布に近づける（$b=1$ のときには p_j がすべて同じ値となる）働きをもつ．

4. 遺伝的操作による適応学習

タスク転送要求が失敗した場合，適応度の更新を行った後に一定確率で遺伝的操作の crossover と mutation を集団に適用し，新たな個体を生成する．

crossover の処理が起動されると，集団から2つの個体がランダムに選択される．この個体ペアに対して uniform crossover[11]が適用され，新たな2つの個体が生成される．生成された個体の適応度は，それらの親となる個体から継承される．すなわち，親の適応度値が f_1, f_2 のとき，生成された個体の持つ適応度の初期値を $(f_1+f_2)/2$ と定める．crossover によって新たに生成された2個体のうち，ランダムに選ばれた1個体が集団内で最も適応度の低い個体と置き換えられる．

mutation の処理が起動されると，集団から1つの個体がランダムに選択され，その個体中の1文字が別の文字に変化することで新たな個体が生成される．生成された個体の適応度の初期値は親となる個体と同一とする．mutation によって新たに生成された個体は集団内で最も適応度の低い個体と置き換えられる．

以上の遺伝的操作により集団内で個体の重複が発生した場合には，新たに生成された個体に対してさらに mutation を適用することで重複を除去する．また，遺伝的操作の直後では，適応度 p_i がそれぞれの親から継承されるため，適応度の制約条件 $\sum_i p_i=1$ が成立しない可能性がある．そのため各個体の適応度値を $p_i/\sum_j p_j$ によって調整し，適応度の制約条件を満足させている．

crossover の手法として uniform crossover が使用されているのは，個体の文字列中における位置が近いノードが必ずしも同じような負荷状態を持つとは限らないので，one-point crossover や two-point crossover のような文字の位置関係を使用した手法を用いる必然性がなく，単に部分列を交換すると良いためである．各個体はタスク転送要求の送出先となるノード群を示しているため，crossover による部分列の交換はそれらノード群に関する情報の交換を行っていると言える．この働きにより，mutation のみの場合に比べて学習速度の向上が期待できる．

本動的負荷均衡アルゴリズムにおいて，building block（積み木）[4] はタスク転送の要求を送出する（もしくはしない）ノード群を示している．例えば building block $H_1=***11*$ はノード P_3 と P_4 に対して要求を送ることに対応し，$H_2=0*0***$ はノード P_0 と P_2 に対して要求を送らないことを示す．H_1 と H_2 を含む個体が高い適応度を持つ，すなわちタスク転送要求に対する成功率が高いものとすると，crossover によりこの両者が組み合わされ，さらに成功率の高い個体が生成される．

5. シミュレーション実験

ワークステーション上に実現したシミュレータを使用し，以下の項目についてシミュレーション実験を行った．

1. タスクの発生割合と平均応答時間との関係
2. タスクの発生割合と転送要求の成功率
3. 負荷の非一様性と平均応答時間との関係
4. 負荷の動的な変化に対する追従性
   (a) 過渡的な負荷変化
   (b) 周期的な負荷変化
5. タスク転送要求先の自己組織化
6. 遺伝的操作の方法に関する比較

実験1から3においては，システム全体として見た場合の負荷均衡の効果を評価する．1ではタスクの生成から終了までの平均応答時間を観測することで，負荷均衡の性能を評価する．2では実際のタスク転送要求がどの程度成功しているかを比較し，平均応答時間の差が生じている原因を調べる．3ではシステム内の負荷の一様度を変化させることで，手法の頑健性を評価する．

実験4においては，負荷状態の過渡的な変化に対して学習がどの程度追従しているかを，Task queueの待ち行列長を観測することで比較する．負荷状態の変化パターンとしては，さまざまな形態が考えられる．ここでは，(1) ユーザがタスクの継続的な投入を急に開始するという状況を想定した過渡的な負荷状態の変化，および，(2) 特定のユーザがあるノードを占有して一定周期で使用するという状況を想定した周期的な負荷状態の変化，の2つの場合について実験を行う．

実験5ではタスク転送要求が実際にどのようなノード群に対してなされているかを観測する．これにより，負荷の軽いノードに対して重点的かつ組織的にタスク転送要求を送出することで，要求の成功確率を向上させていることを明らかにする．

実験6ではcrossoverの方法の違いによる影響と，遺伝的操作により一度に置き換えられる個体の数を変化させた場合の影響について実験を行った．

本論文では，タスクの発生割合を単位時間あたりのタスクの発生確率とタスクの平均実行時間の積として定義する．

実験に当たって，GeSLA動的負荷均衡アルゴリズムの比較対象として以下の2つを取り上げた．

(a) ユニキャストで実現された学習なしの送り手主導の方式

(b) SLAによる学習を行う送り手主導の方式

比較対象(a)は，ユニキャストによりランダムに選ばれたノードに対してタスク転送要求を送出し，それが成功した場合には，そのノードへタスクを転送し，失敗した場合には，さらに別のノードへ転送要求を送出する手法である．転送要求は，それが成功するまで，もしくはある一定回数に達するまで繰り返される．

比較対象(b)は，先に述べた(a)にSLAによる学習を適用した方式である．各ノードiに対してタスク転送要求を送る確率p_i ($\sum_i p_i=1$) を割り当て，この確率ベクトル$<p_i>$について，SLAによる学習を適用することで，転送要求の成功確率を向上させる．(a)と同様に，転送要求はそれが成功するまで，もしくはある一定回数に達するまで繰り返し送出される．

比較対象(a)，(b)のアルゴリズムを以下にまとめて示す．

1. タスクの発生時に自ノードの負荷状態の観測を行い，それに従って以下の処理を行う．
   - ●負荷が*Heavy*の場合，以下の2〜4に示されるタスク転送処理を実行する．
   - ●負荷が*Light*または*Normal*の場合にはタスクを自ノードのTask queueに移す．その後，再び1に戻り，新たなタスクの発生を待つ．
2. (a)では一様確率でランダムに，(b)ではベクトル$<p_i>$に従って，1つのノードiを選択し，そこへ*Request*メッセージを送出する．
3. タスク転送要求の成否により，以下の処理を行う．
   - ●タスク転送要求が成功した場合，すなわち，負荷が*Light*であるノードが見出され，そこから*Accept*メッセージが戻ってきた場合には，そのノードのTask queueに向けてタスクが転送される．(b)ではその後，式(2)に基づいてSLAによる$<p_i>$の更新が行われる．
   - ●タスク転送要求が失敗した場合，すなわち戻ってきたメッセージが*Reject*である場合には，2.に戻りタスク転送要求を繰り返す．転送要求が一定回数以上失敗した場合には，タスクの転送は一切行われず，タスクは自ノードのTask queueに移される．*Reject*メッセージを受け取るたびに，(b)では式(3)に基づいてSLAによる$<p_i>$の更新が行われる．
4. 再び1に戻り新たなタスクの発生を待つ．

以下の実験について共通の実験条件を**表1**に示す．本シミュレーションでは1 msを1単位時間とした．

以下ではそれぞれ10回の実験を行い，その平均値をグラフ上および表に示されるデータとしている．

表1 共通の実験条件
Table 1 Simulation conditions.

分散システムのノード数	40
ネットワークトポロジー	バス結合
ネットワークの帯域幅	10 Mbps
タスクの平均実行時間（指数分布）	100 ms
タスクの平均サイズ（指数分布）	10 KBytes
メッセージのサイズ（Request, Accept, Reject）	128 Bits
メッセージのサイズ（Dispatch）	128Bits+転送タスクサイズ
シミュレーション中のタスク実行数	100,000
送り手主導方式およびSLAにおける最大要求送出数	10
GeSLA動的負荷均衡アルゴリズムの条件	
ノード当たりの個体集団サイズ	10
crossoverの起動確率	0.1
mutationの起動確率	0.2
SLA，GeSLA共通の学習パラメータ	
式(4)におけるaの値	0.15
式(4)におけるbの値	0.25

5.1 システムの平均応答時間

タスクの発生割合と平均応答時間の関係を**図3**に示す．この場合，40個のノードのうち，10個に発生割合4λ (λ：システム全体としての発生割合) で集中してタスクが発生するという非一様負荷により実験を行っ

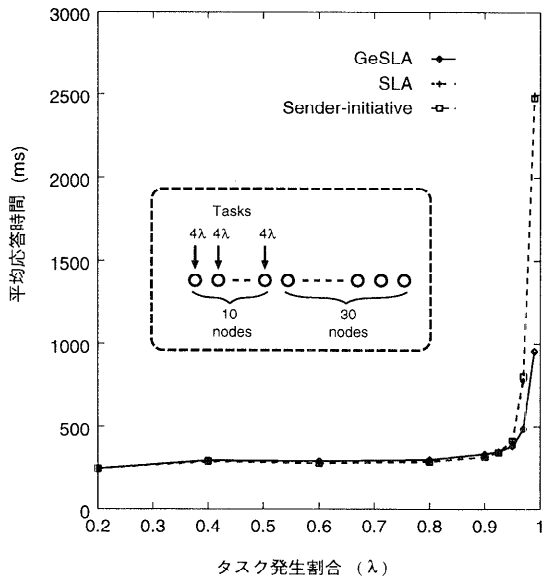

図3 タスクの発生割合と平均応答時間
Fig. 3 Task arrival rate and mean response time.

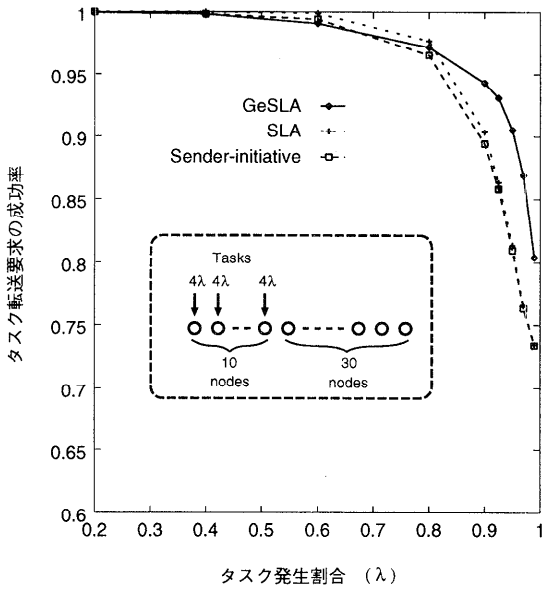

図4 タスクの発生割合とタスク転送要求の成功率
Fig. 4 Task arrival rate and success rate of the requests.

た．

タスクの発生割合が小さい場合，手法間に有意な差は見られないが，発生割合が0.95を超える領域では，GeSLAが他のアルゴリズムに比べて少ない平均応答時間を達成している．つまり，システムの負荷が非常に重い場合にGeSLAにおける学習が，単純な送り手主導の方式やSLAによる学習を付加した方式よりも，効果的に機能していることが分かる．

また，SLAを用いた手法と単純な送り手主導を比較すると，両者の間に有意な差は認められない．ノード数が少ない場合にはSLAを用いることで改善を見ることができるが，本実験のように多くのノードを使用している場合には，適切な確率ベクトルを得るまでの学習に多くの時間が必要となるためである．SLAの学習を速くするためには，式(4)におけるパラメータ a, b の値を大きなものとすることも考えられるが，その場合には一度の成功・失敗で確率ベクトルが急激に変化するため振動現象を起こし，適切な確率ベクトルが生成されにくくなる．一方GeSLAでは遺伝的操作，特にcrossoverにより転送要求先リストの部分列が交換されるため，速やかな学習が実現できているものと考えられる．

5.2 タスク転送要求の成功率

タスク転送要求の成功率に関する実験結果を**図4**に示す．図の横軸はタスクの発生割合を示し，縦軸はタスク転送要求の成功率を示す．この実験においても，40個のノードのうち，10個に発生割合 4λ (λ：システム全体としての発生割合）で集中してタスクが発生するという非一様負荷により実験を行った．

システムの負荷が軽くタスク発生割合が0.8以下の場合には，手法による差は認められない．しかし，発生割合が0.8を超えると，GeSLAのタスク転送要求の成功率は他の2手法よりも5〜10%高くなっている．システムの負荷が重い場合には，待ち行列長が長くなり，タスク転送要求が多数送出されているため，成功率のわずかな差が平均応答時間に強い影響を及ぼす．この結果は前節で示した平均応答時間の結果を裏づけるものと考えられる．また，SLAによる手法で改善が見られていないが，これは5.1節で述べたのと同じ理由による．

5.3 負荷の非一様性の影響

40個のノードのうち n 個 (n=10, 20, 30, 40) のノードに対して発生割合 $\lambda\times 40/n$ (λ=0.99) でタスクを発生させた場合の実験結果を図5に示す．

タスクがシステム全体に一様に発生している場合 (n=40) には手法による差はほとんど見られない．しかし発生割合に大きな偏りが見られる場合 (n=10, 20) では，他の手法に比べてGeSLAにおける平均応答時間が格段に改善されている．一方，SLAによる手法は学習の速度が遅いため，非一様度がそれほど大きくない場合 (n=20, 30) には単純な送り手主導の方法に比べて若干の改善が見られるものの，それ以外の場合にはほとんど改善が見られない．

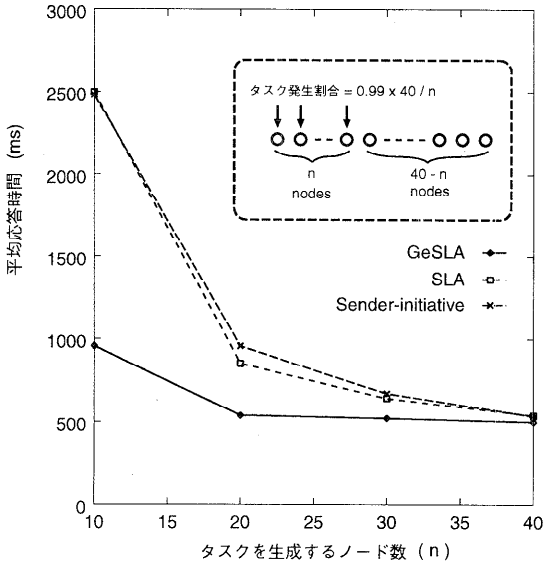

図5 負荷の非一様性と平均応答時間
Fig. 5 Load imbalance and mean response time.

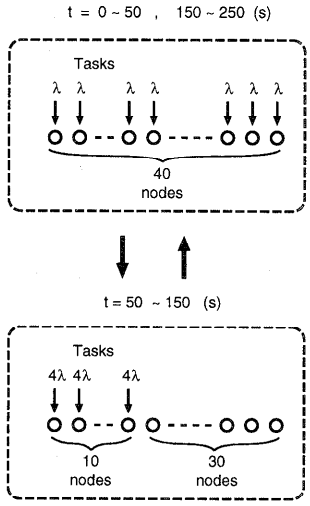

図6 過渡的な負荷分布の変化
Fig. 6 A transitional change of loads.

GeSLA においては，ノードの負荷に関する情報が間接的に集団内の個体に蓄えられ，“負荷の軽いノードにタスク転送要求を送出すること”を表現する building block が遺伝的操作により集団内に数多く含まれる．そのため，システム内で特に負荷の偏りが大きく，その状態が定常的に継続するような状況においては，集団内に蓄積された情報を用いることにより転送要求の成功率が向上し，全体の平均応答時間が減少する．

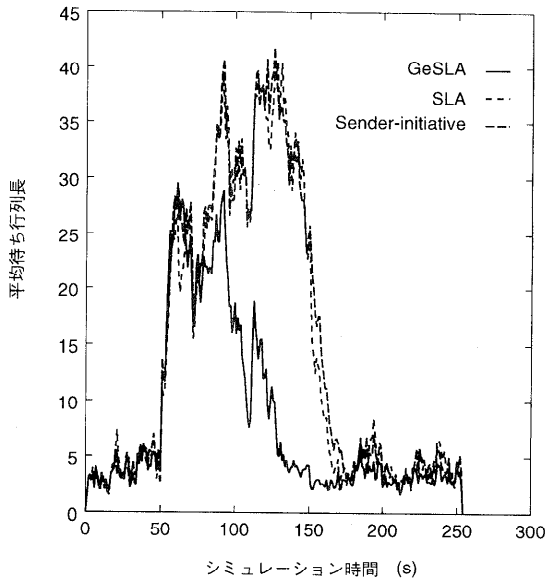

図7 過渡的な負荷変化におけるタスク待ち行列長の推移
Fig. 7 Transition of the task queue length in a transitional change of loads.

5.4 負荷変化に対する追従性

5.4.1 過渡的な負荷変化

ここでは負荷状態の過渡的な変化に対する追従性について述べる．この実験は，通常の使用状態から負荷の偏りの大きな状態へ急激に環境が変化した場合を想定している．

負荷状態の変化パターンを図6に示す．シミュレーション時間 $t=0\sim50$(s) および $t=150\sim250$(s) では一様な発生割合 λ で各ノードにタスクが発生する．$t=50\sim150$(s) では，10個のノードに対してのみ $4\times\lambda$ の発生割合でタスクが発生し，残りのノードでは発生しない．本実験では λ の値として 0.99 を用いた．

図7は，タスクが常に発生している10個のノードにおける Task queue の待ち行列長の平均値を示している．図の横軸はシミュレーション時間，縦軸は待ち行列長の平均値である．

タスクが一様に発生している間はどの手法もほぼ同じ平均待ち行列長を示している．タスクの発生状況が変化した後，単純な送り手主導および SLA による方法では待ち行列長が長くなる一方であるのに対して，GeSLA では $t=100$(s) を超えたあたりから，待ち行列長が急激に短くなっている．

GeSLA では1つの集団に対して，$t=50\sim100$(s) の間に約1,800回のタスク転送要求がなされ，約70回の遺伝的操作が集団に適用されている．1度の遺伝的操作で1個体のみが変更されるので，集団の大きさが10

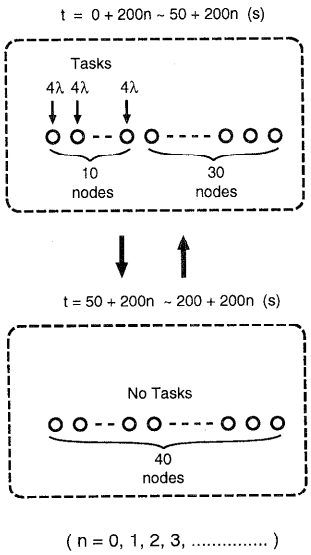

図8 周期的な負荷分布の変化
Fig. 8 Cyclic changes of loads.

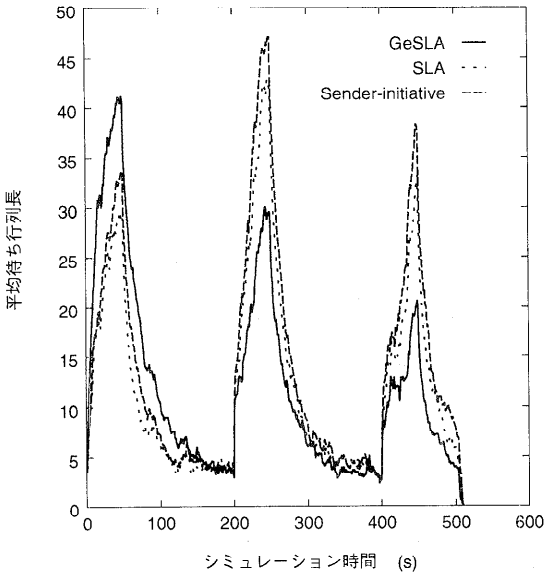

図9 周期的な負荷変化における待ち行列長の推移
Fig. 9 Transition of the task queue length in cyclic changes of loads.

であることから，通常の遺伝的アルゴリズムの約7世代に相当する時間で学習の効果が現れている．

5.4.2 周期的な負荷変化

システムの負荷状態が周期的に変化する場合について述べる．この実験は，特定のノードがある周期で使用されて断続的に負荷が重くなる場合を想定している．

負荷の変化パターンを図8に示す．1周期200(s)のうち，はじめの50(s)では，10個のノードに対してのみ$4\times\lambda$の発生割合でタスクが発生し，他のノードでは発生しない．残りの150 (s) ではシステム内でタスクは発生しない．本実験ではλの値として0.99を用いた．

タスクが発生している10台のノードにおけるTask queueの待ち行列長の平均値の推移を**図9**に示す．単純な送り手主導による手法では，待ち行列長のピークは減少せず，逆に大きくなる場合もある．これは，タスクの実行時間の分布における確率的な揺らぎの影響が表面化しているものと考えられる．SLAによる学習を付加した手法では若干の改善が見られるが，同じタスク生成パターンを繰り返しているにも関わらず，待ち行列長の改善は少ない．SLAでは個々のタスク転送要求の成功・失敗の情報を1つの確率ベクトルのみを用いて学習しており，過去のタスク生成パターンを記憶できないためと考えられる．

一方GeSLAでは，最初のタスク発生においてのみ待ち行列長が他の手法よりも長くなっているが，周期的なタスク発生の進行に伴い，徐々に待ち行列長のピークが小さくなっている．過去のタスク発生パターンを集団内に個体の形で蓄えているため，2回目以降はその情報を用いたタスク転送要求を行うことができ，タスク転送の成功率が向上しているものと考えられる．

以上示した負荷変化への追従性に関して，単純な送り手主導に対してSLAによる改善は少ない．これは5.1節で述べたように，ノード数が比較的多い場合，SLAでは適切な確率ベクトルを得るまでに多くの時間を要することにその原因がある．特に負荷状態が変化した場合には，それに速やかに追従する必要があるため，SLAでは有効な学習を実現できない．

5.5 タスク転送要求先の自己組織化

ここではシステム内の各ノードにおけるタスク転送要求先に着目する．実験条件として，40台のノードからなるシステムのうち，ノード番号1から20までの20台のノードで継続的に発生割合$2\times\lambda$でタスクを発生させる．システム全体としてのタスクの発生割合は$\lambda=0.99$とした．100,000個のタスクを実行するまでシミュレーションを行い，それが終った時点での結果を用いた．

図10はタスク転送要求のノード別の送出回数を図示したものである．図の黒丸はそれぞれのノードが受けとったタスク転送要求の回数を示しており，その半径が回数に比例している．縦軸は転送要求の送出元のノード番号，横軸は送出先のノード番号をそれぞれ表

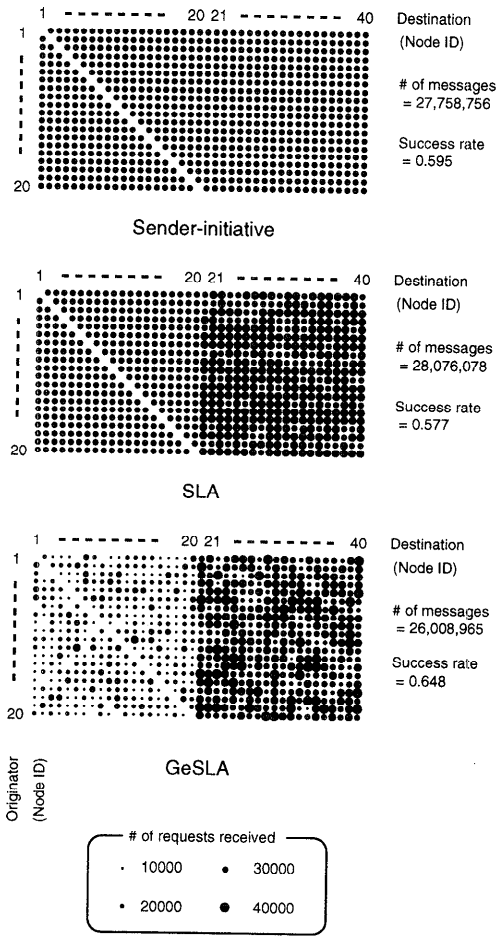

図10 タスク転送要求の送出先ノードの分布
Fig. 10 Distribution of the destination nodes of task transfer requests.

している．ノードでタスクが発生しない限り，そこからの転送要求は生じないので，送出先のノード番号21から40までについては省略をしている．また，全体として送出されたメッセージの数と要求の成功率をそれぞれのグラフの横に示している．

送り手主導による方式では，負荷の状態にかかわらずすべてのノードに対して一様に要求を送出している．一方，SLA, GeSLA ではタスクが発生していて負荷の重い20台のノードを避け，残りの比較的負荷の軽いノード（番号21～40）に重点的に要求を送出している．この傾向はGeSLAによる方式において，きわめて顕著である．その結果，GeSLA では他の手法に比べてより少ないメッセージ数で高いタスク転送要求の成功率を実現している．

GeSLA動的負荷均衡において特筆すべきことは，タスク転送の要求先となる負荷の軽いノード群の中にはっきりとしたばらつきが見られるということである．例えば，1番のノードは要求の送り先として，29,40番のノードを多用し，また，20番のノードでは29,30番のノードを多く利用している．すなわち，タスク転送要求の流れを見た場合，システム内でタスクが継続的に発生することに伴って，転送要求先のノードのグループが自己組織的に形成されている．この結果，少ない要求回数で成功率の高いタスク転送が実現されていると考えられる．

表2 Crossoverの方法についての平均応答時間の比較
Table 2 A comparison of the crossover methods.

crossover の方法	平均応答時間
Uniform	958.7
1-point	1174.3
2-point	1107.7
3-point	1230.0
5-point	1120.8
10-point	1051.1

表3 置き換えられる個体数についての平均応答時間の比較
Table 3 A comparison of the number of strings to be replaced.

置き換えられる個体数	平均応答時間
1	958.7
2	1191.3
3	1206.1
4	1261.5
5	2002.2

5.6 遺伝的操作の方法に関する比較

遺伝的操作に関係して，(1) crossover の方法，(2) 遺伝的操作により一度に置き換えられる個体数，の2点について平均応答時間の比較実験を行った．実験条件は基本的には実験1と同じであり，タスクの発生割合を0.99とした．

表2 に crossover の比較結果を示す．この結果によると，今回採用した uniform crossover は最も短い平均応答時間を示しており，次いで個体長40に対して10ヶ所で交叉を行う 10-point crossover が良い性能を示している．GeSLA 動的負荷均衡では個体の文字列中の位置関係が特別な意味を持たないので，位置関係を利用する 1-point crossover などよりも，uniform crossover のように building block の交換を頻繁に行う手法が効率的であることに起因するものと考えられる．

表3 に遺伝的操作で一度に置き換えられる個体数の比較結果を示す．置き換えられる個体数が1の場合（最劣1個体）が最も短い平均応答時間を実現しており，

個体数が増えるにつれて平均応答時間が増加している．これは，遺伝的操作により生成された個体の適応度の初期値が親から継承され，真の適応度との間に誤差を生じるためと考えられる．特に多数の個体を一度に置き換えた場合には，集団内の個体の持つ適応度値の誤差が無視できないほど大きなものとなる．例えば，置き換えられる個体数が5である場合，crossover, mutationについてそれぞれ適応度の低い順番に5つの個体が置き換えられる．そのため全個体数に相当するのべ10個体が置き換えられてしまい，集団における各個体の持つ適応度値は著しく不正確なものとなる．

以上の実験結果より，本論文で採用したuniform crossoverおよび最劣1個体の置き換えという手法は妥当であると考えられる．

6. おわりに

本論文では，送り手主導に基づく分散管理型の動的負荷均衡アルゴリズムに確率学習オートマトンと遺伝的操作による学習を導入したGeSLA動的負荷均衡アルゴリズムを提案した．

シミュレータを用いた実験により提案手法が，(1) システムの平均応答時間，(2) タスク転送要求の成功率，(3) システムの負荷状態変化への追従性，の3点において，単純な送り手主導の方式やSLAのみによる学習を付加した方式よりも優れていることが確かめられた．その理由としては，(1) タスク転送要求を負荷の軽いノードに集中して送出していること，(2) 転送要求がノードのグループに対して組織的に送出されており，それぞれの要求送り元ノードについて，異なる要求送出先グループが形成されていること，の2点が考えられる．

今後の課題としては，タスクの受け入れ側からタスク転送要求を送出する受け手主導の考え方も導入した手法の開発があげられる．

参考文献

1) Suen, T. T. Y. and Wong, J. S. K.: Efficient Task Migration Algorithm for Distributed Systems, *IEEE Trans. Parallel and Distributed Systems,* Vol. 3, No. 4, pp. 488-499 (1992).
2) Eager, D. L., Lazowska, E. D. and Zahorjan, J.: Adaptive Load Sharing in Homogeneous Distributed Systems, *IEEE Trans. Softw. Eng.,* Vol. 12, No. 5, pp. 662-675 (1986).
3) Shivaratri, N. G., Krueger, P. and Singhal, M.: Load Distributing for Locally Distributed Systems, *IEEE Computer,* Vol. 25, No. 12, pp. 33-44 (1992).
4) Holland, J. H.: *Adaptation in Natural and Artificial Systems,* University of Michigan Press (1975).
5) 棟朝雅晴，高井昌彰，佐藤義治：遺伝的操作を用いた動的負荷分散アルゴリズムの提案，電子情報通信学会1993年秋季大会講演論文集 (6)，p. 88 (1993).
6) 棟朝雅晴，高井昌彰，佐藤義治：分散制御型動的負荷分散における遺伝的操作の導入，北海道大学工学部研究報告，No. 167, pp. 127-135 (1994).
7) Munetomo, M., Takai, Y. and Sato, Y.: A Genetic Approach to Dynamic Load Balancing in a Distributed Computing System, *Proceedings of the First IEEE Conference on Evolutionary Computation,* pp. 418-421 (1994).
8) Reed, D.A. and Fujimoto, R.M.: *Multicomputer Networks: Message-Based Parallel Processing,* MIT Press (1987).
9) Goldberg, D.E.: *Genetic Algorithms in Search, Optimization and Machine Learning,* Addison Wesley (1989).
10) Narendra, K. S. and Thathachar, M. A. L.: Learning Automata—A Survey, *IEEE Trans. System, Man, and Cybernetics,* Vol. 4, No. 4, pp. 323-334 (1974).
11) Syswerda, G.: Uniform Crossover in Genetic Algorithms, Schaffer, J.D. (ed.), *Proceedings of the Third International Conference on Genetic Algorithms,* pp. 2-9, Morgan Kaufmann Publishers (1989).

(平成6年10月31日受付)
(平成7年2月10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．