非均質環境における誤差逆伝播法の矩形分割マッピングによる並列高速化

小澤洋司[†,☆] 菅谷至寛[†] 阿曽弘具[†]

階層型ニューラルネットワークとその学習法の 1 つである誤差逆伝播法は様々な分野で広く応用されているが，その処理量は膨大である．そのため，処理時間短縮のために並列化の研究が数多く行われている．しかし，各プロセッサの能力が均質な環境を対象としたものが多く，そのまま能力が非均質な環境に適用すると能力の低い計算機に全体の性能が制限される．本研究では，誤差逆伝播法の並列化を矩形分割問題と考えることで，非均質な環境において適切なマッピングを行う手法（SRPM 法：Static Rectangular Partitioning Mapping）を提案する．各プロセッサが担当する処理を矩形で表すことで，能力と処理量を適切に対応させ，通信時間が最小となるマッピングを一意に決定できる．さらに，あらかじめ適切な能力比を求めることが困難である場合に有用となる，適切な能力比を推定し，動的にマッピングを更新する手法（DRPM 法：Dynamic Rectangular Partitioning Mapping）を提案する．実験によりそれぞれの有効性を確認する．

Parallelizing of a Back Propagation in Heterogeneous Environment Using a Rectangular Partitioning Mapping Method

Yoji Ozawa,[†,☆] Yoshihiro Sugaya[†] and Hirotomo Aso[†]

Back propagation (BP), which is one of the neural network training algorithms, has been applied to various fields. It requires long processing time, so many researchers have tackled this problem using parallel processing. But most of them aim for homogeneous environment. When those are applied to heterogeneous environment, the lowest processor restricts whole performance. We propose a SRPM method (Static Rectangular Partitioning Mapping), based on an idea that parallelization of BP is modeled as a rectanguler partitioning. The model enables to decide an appropriate mapping pattern with respect to a load-balancing and communication time. We propose a DRPM method (Dynamic Rectangular Partitioning Mapping), which estimates ability and updates mapping pattern dynamically. The method is efficient and useful for environments in which it is difficult to get information of accurate processor's ability. And we show their efficiency experimentally.

1. はじめに

階層型ニューラルネットワークの 1 つである多層パーセプトロンとその学習法である誤差逆伝播法（以下 BP 法：Back Propagation）は，良い識別性能を持つ識別器を，学習サンプルから比較的容易に構成できる．そのため，様々な分野で広く応用されている．しかし，学習に要する処理量は膨大なものとなるため処理時間の短縮が必要とされており，高速化のアプローチとして並列化の研究がさかんに行われている[1)~3)]．BP アルゴリズムには 2 種類の並列性（データ並列性，ノード並列性）が内在しており，これらを適切に組み合わせることが必要である．多くの研究は，各プロセッサの能力が均等な環境を対象としている．しかし，各プロセッサの能力が異なる計算機クラスタのような非均質な環境で，既存の均質環境を対象としたアルゴリズムを用いると，同期待ちのために最低の能力のプロセッサに全体の性能が制限されてしまう．これを避けるために，各プロセッサの能力に応じて処理を割り当てる必要がある．しかし，2 種類の並列性に関して非均質性を考慮することは困難な問題であった．また，並列性の組合せの割合により全体の性能が変化することが知られており，適切な組合せの割合を決定する手法が提案されているが[4),5)]，正確に求めるためには多くのパラメータを必要とし，困難な作業となる．本研究では BP 法の並列化を矩形分割問題としてモデル化

† 東北大学大学院工学研究科
Graduate School of Engineering, Tohoku University
☆ 現在，株式会社日立製作所
Presently with Hitachi, Ltd.

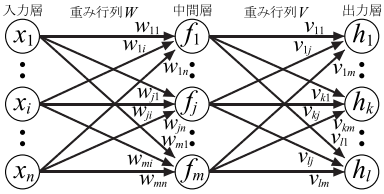

図 1 3 層パーセプトロン
Fig. 1 Multilayer perceptron.

することでこれらの問題を解決する．

また，非均質性に適切に対応するためにあらかじめ各プロセッサの能力の見積りが必要となる．しかし，適切な見積りを求めることは一般に困難であるうえに，能力の見積りを間違うと全体の性能が劣化する．そこで，本研究では BP 法の処理を行っている間に動的に適切な能力を求め，それに応じてマッピングを変化させる手法を提案する．これにより，能力がまったく未知の環境などにも対応できることが期待される．

2. 誤差逆伝播法（BP 法）

BP 法は多層パーセプトロンにおいて，ある入力パターンから所望のパターンが出力されるように，学習サンプルを用いて結合重みを調整するアルゴリズムである．本研究では 3 層パーセプトロン（**図 1**）を対象とする．n，m，l をそれぞれ入力層，中間層，出力層のユニット数とし，入力層と中間層を結合する重み行列を $\boldsymbol{W}$（$m \times n$ 行列），中間層と出力層を結合する重み行列を $\boldsymbol{V}$（$l \times m$ 行列）とする．入力を n 次元ベクトル $\boldsymbol{x}$，学習サンプル集合を X（学習サンプル数 s）とする．BP 法は Forward フェーズ，Backward フェーズ，Modify フェーズで構成される．Forward フェーズでは，入力されたサンプルに対して中間層の出力 $\boldsymbol{f}$（m 次元ベクトル），出力層の出力 $\boldsymbol{h}$（l 次元ベクトル）が順に計算される．

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{\sigma}\boldsymbol{W}\boldsymbol{x} \tag{1}$$

$$\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{\sigma}\boldsymbol{V}\boldsymbol{f}(\boldsymbol{x}) \tag{2}$$

ここで，$\boldsymbol{\sigma}$ はベクトルの各要素にシグモイド関数 $\sigma(u) = 1/(1+e^{-u})$ をかける演算である．

Backward フェーズでは，重み更新量 $\Delta\boldsymbol{W}$，$\Delta\boldsymbol{V}$ を以下の式より計算する．

$$\Delta\boldsymbol{V}(\boldsymbol{x}) = \boldsymbol{\delta}^{(2)}(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x})^T \tag{3}$$

$$\Delta\boldsymbol{W}(\boldsymbol{x}) = \boldsymbol{\delta}^{(1)}(\boldsymbol{x})\boldsymbol{x}^T \tag{4}$$

ここで，$\boldsymbol{\delta}^{(1)}$，$\boldsymbol{\delta}^{(2)}$ は修正誤差と呼ばれ，その各要素は以下のようにして求める．

$$\delta_k^{(2)}(\boldsymbol{x}) = (h_k(\boldsymbol{x}) - d_k(\boldsymbol{x}))h_k(\boldsymbol{x})(1-h_k(\boldsymbol{x})) \quad (1 \le k \le l)$$

$$\delta_j^{(1)}(\boldsymbol{x}) = f_j(\boldsymbol{x})(1-f_j(\boldsymbol{x}))\sum_{k=1}^{l} v_{kj}\delta_k^{(2)}(\boldsymbol{x}) \quad (1 \le j \le m)$$

ここで，$\boldsymbol{d}(\boldsymbol{x})$（$l$ 次元ベクトル，要素 $d_k(\boldsymbol{x})$）は教師信号である．

Modify フェーズでは，重み更新量を用いて，重み行列を更新する．

$$\boldsymbol{V}^{new} = \boldsymbol{V}^{old} - \epsilon\sum_{\boldsymbol{x}\in X}\Delta\boldsymbol{V}(\boldsymbol{x}) \tag{5}$$

$$\boldsymbol{W}^{new} = \boldsymbol{W}^{old} - \epsilon\sum_{\boldsymbol{x}\in X}\Delta\boldsymbol{W}(\boldsymbol{x}) \tag{6}$$

ϵ は学習定数と呼ばれる，微小係数である．

すべての学習サンプルに対して，Forward，Backward フェーズを行った後，一括して Modify フェーズを行う．このようにすべての学習サンプルに対する重み更新量を計算した後，一括して更新を行う手法を**バッチ学習**と呼ぶ．この一連の計算をイタレーションと呼び，誤差が閾値以下になるまで繰り返す．

3. 誤差逆伝播法の並列化

BP 法に内在するデータ並列性とノード並列性による並列化手法[1] について述べる．さらに，2 種類の並列化手法を組み合わせた手法について述べる．

3.1 データ並列

学習サンプル集合 X を分割し，その部分集合を各プロセッサに割り当てる．各プロセッサは割り当てられた学習サンプルについて重み更新量の部分和を求める．各プロセッサが保持する部分和の総和が重み更新量となるため，通信を行い部分和の総和を求める．その結果を各プロセッサに配り直す．重み更新量 $\Delta\boldsymbol{V}$，$\Delta\boldsymbol{W}$ を通信するため，データ並列による通信要素数は $(n+l)m$ となる．

3.2 ノード並列

重み行列を分割し，重み更新の計算を並列化する[4]．重み行列 $\boldsymbol{W}$（$m \times n$ 行列）を**行方向**に，重み行列 $\boldsymbol{V}$（$l \times m$ 行列）を**列方向**に分割する．つまり，ノード並列は中間層のユニット数 m を分割し，処理を割り当てることに相当する．プロセッサ数が 3 の場合を以下に示す．

$$\boldsymbol{W} = \left(\begin{array}{c}\boldsymbol{W}^{(1)}\\\hline\boldsymbol{W}^{(2)}\\\hline\boldsymbol{W}^{(3)}\end{array}\right), \boldsymbol{V} = \left(\boldsymbol{V}^{(1)} \,\middle|\, \boldsymbol{V}^{(2)} \,\middle|\, \boldsymbol{V}^{(3)}\right)$$

表 1 各プロセッサにおける BP 法の計算量
Table 1 The amount of calculation.

	加減算	乗除算	シグモイド演算
Forward フェーズ	$(n+l-1)m_i s_i - ls_i$	$(n+l)m_i s_i$	$m_i s_i + ls_i$
Backward, Modify フェーズ	$(n+l)m_i s_i + 2ls_i$	$(n+2l+1)m_i s_i + 2ls_i$	—

n：入力層のユニット数　l：出力層のユニット数　m_i：プロセッサ P_i に割り当てられた中間層のユニット数
s_i：プロセッサ P_i に割り当てられた学習サンプル数

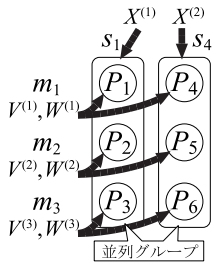

図 2 2 種類の並列化手法の組合せ（$N = 6$, $N_{DP} = 2$）
Fig. 2 Combination of two parallelizations.

このように分割することで，式 (2) は，$\boldsymbol{h}(\boldsymbol{x}) = \sum_i \boldsymbol{\sigma V}^{(i)} \cdot \boldsymbol{\sigma W}^{(i)} \boldsymbol{x}$ と表せる．したがって，プロセッサ間通信は Forward フェーズ計算時，$\boldsymbol{h}(\boldsymbol{x})$ の部分和の転送の 1 度のみとすることができる．通信要素数は ls となる．

3.3 データ並列とノード並列の組合せ

2 種類の並列化手法は組み合わせることができる（**図 2**）．複数のプロセッサからなる並列グループを作成し，分割した学習サンプル集合を割り当てる（並列グループの数を N_{DP} とし，データ並列度と呼ぶ）．そして，各並列グループ内でノード並列を行う．プロセッサ P_i に割り当てる学習サンプル数を s_i，分割された重み行列の中間層のユニット数を m_i とする．プロセッサ P_i の BP 法の計算量は**表 1** のようになる．ls_i は全体の計算量に比べて小さいため，P_i の計算量は $m_i s_i$ にほぼ比例すると見なせる．ノード並列による通信は並列グループ内のプロセッサ間通信となる．データ並列による通信は，各並列グループで求めた重み更新量の部分和を通信するため，異なる並列グループのプロセッサ間の通信となる．つまり，重み行列の同じ部分を保持しているプロセッサ間の通信となる．

4. 非均質性を考慮した過去の研究

プロセッサの能力が非均質な場合，能力比と処理量の対応が不適切であると通信の同期待ちなどが発生するため，プロセッサの能力を有効に利用できず，並列高速化の妨げとなる．本章では非均質性を考慮した過去のマッピング手法について述べ，その問題点を明らかにする．

東ら[4] は以下のようにして非均質性を考慮し，処理のマッピングを行った．まず能力の低いプロセッサから順に N/N_{DP} 個ずつ選択し，並列グループを作成する．各並列グループの中で最低のプロセッサの能力比で学習サンプルを分割し，並列グループに割り当てる．そして，ノード並列に関しての割当ては均等とする．以下では手法 H と呼ぶ．

表 2 手法 H と手法 H_{rev} の割当て
（$N = 6$, $N_{DP} = 2$）
Table 2 Mappings of method H and method H_{rev}.

手法	$s_1 : s_4$	$m_1 : m_2 : m_3$
手法 H	1.0 : 2.5	1.0 : 1.0 : 1.0
手法 H_{rev}	1.0 : 2.5	1.0 : 1.5 : 2.0

手法 H は並列グループ内のプロセッサ間の非均質性，つまりノード並列に関しての非均質性について考慮していない．そこで，ノード並列に関しても最低の能力の並列グループ内のプロセッサの能力比で m を分割し，それに対応する処理を割り当てるように簡単な改良を施す．これを手法 H_{rev} と呼ぶ．

図 2 のプロセッサの能力比が $p_1 : p_2 : p_3 : p_4 : p_5 : p_6 = 1.0 : 1.5 : 2.0 : 2.5 : 3.0 : 3.5$ であった場合（非均質環境）の手法 H，H_{rev} の割当てを**表 2** に示す．手法 H_{rev} では改良を行い，ノード並列に関して非均質性を考慮したが，問題点が 2 つあげられる．まず，プロセッサの能力比と割り当てる処理量が対応していないことである．たとえば，図 2 で P_2 と P_5 にノード並列に関して割り当てる処理量は均等であるため，処理量の割合はデータ並列に関する割合となる．しかし，データ並列の分割の割合は P_1 と P_4 の能力比であるため，P_2 と P_5 の能力比には適切に対応していない．このように正確に各プロセッサの能力に対応していないため，各プロセッサの能力を有効に活用できない．

また，プロセッサ全体に対して可能なデータ並列度は数通り（N の約数の個数分）存在する．これは，データ並列とノード並列の組み合わせ方が複数考えられることを意味している．ニューラルネットワーク（NN）のサイズ，学習サンプル数などの条件により，処理時間が最小となる最適なデータ並列度が異なることが知られており，適切な選択をする必要がある．過去の研究[4),5)] において，並列度選択手法が提案されているが，

正しく選択するためには計算能力や通信性能など，多くのパラメータが必要となり，それらを事前に正しく測定するのは困難な作業となる．

5. 矩形分割を用いたマッピング手法

本章では従来手法の問題点をふまえ，適切に非均質性を考慮し，一意にマッピング（処理の割当て）を決定する手法（SRPM 法：Static Rectangular Partitioning Mapping）を提案する．まず，BP 法の並列化を矩形分割問題としてモデル化する．これにより，計算量と通信時間を同時に考えることができる．計算量を適切に割り当て，通信時間を最小にする分割が最適となるが，最適な分割を選択するのは非常に困難な問題となる．したがって，いくつかの条件を加えた分割方法を用い，その分割の中で最適な分割を選択するアルゴリズムを提案する．

5.1 矩形分割問題へモデル化

Beaumont ら[7] は非均質環境での行列乗算の並列化のため，行列そのものを単位正方形と見なして，1 つのプロセッサに割り当てる部分行列を矩形と見なし，並列化問題を単位正方形を矩形に分割する問題に帰着させた．

BP 法において，重み更新の処理の大部分は行列演算である．それゆえ，Beaumont らによる行列乗算の並列化手法を，この処理の並列化（ノード並列）つまり重み行列の分割法へ利用することが考えられるが，複数の問題が生じるため，適切ではない．まず，3.2 節で述べた重み行列 $\boldsymbol{W}$ を行方向に，重み行列 $\boldsymbol{V}$ を列方向にという分割を行わない場合，通信回数が増加してしまう．そして，データ並列と組み合わせる場合には手法 H，H_{rev} と同様に適切なデータ並列度を選択する必要が生じる．

本研究では単位正方形を行列と考えるのではなく，水平方向の分割はデータ並列，垂直方向の分割はノード並列に関するマッピングを表すと考える．つまり，矩形の幅を $s_i^*(=s_i/s)$，高さを $m_i^*(=m_i/m)$ とする．たとえば，図 2 の並列化を**図 3** のようにモデル化する．このモデル化により，以下のような理由から，データ並列とノード並列両手法を考慮した適切なマッピングを考えることができる．

3.3 節で述べたようにプロセッサ P_i の計算量が $m_i s_i$ にほぼ比例することから，P_i の計算量は $m_i^* \times s_i^*$ の矩形の面積で表現でき，各プロセッサの能力に応じて矩形の面積を定めることで正確な負荷分散が実現できる．

プロセッサ P_i のデータ並列に関する通信の要素数は $(n+l)m_i$，ノード並列に関する通信の要素数は ls_i であり，通信量は s_i と m_i により決まる．同じ部分の学習サンプルの部分集合，あるいは重み部分行列を担当するプロセッサ間で通信が行われるため，通信を行うプロセッサのグループは矩形分割モデルにおいて各々垂直方向，あるいは水平方向に並んだプロセッサとなる．したがって，矩形の幅と高さ（形状と呼ぶ）と配置で通信を考慮することができる．このようにモデル化を行うことで，データ並列とノード並列を両方考慮した適切な割当てを容易に考えることができる．

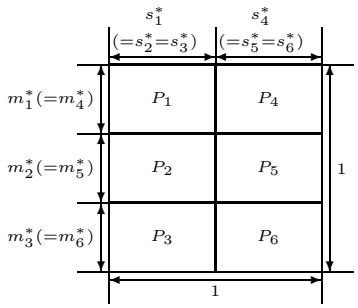

図 3 矩形分割問題にモデル化
Fig. 3 Rectangular partitioning modeling.

5.2 分割方法

本研究では矩形分割モデルを利用し，処理の割当てを次のような考えのもとで決める．

- 面積：面積は各プロセッサの計算量に比例するため，各プロセッサの能力により決める．プロセッサ P_i の能力を p_i （$\sum_{i=1}^{N} p_i = 1$）とし，矩形の面積 $m_i^* s_i^* = p_i$ とする．これにより，計算を適切に分配でき，各プロセッサの能力を十分に利用することができる．
- 矩形の形状と配置：矩形の形状（幅 s_i^*，高さ m_i^*）と配置を決めることで通信時間を概算することができる（概算の方法については後で詳しく述べる）．通信時間が最小のものが最良の分割であると考え，通信時間が最小になる形状と配置にする．

矩形の面積，形状と配置の決定により，分割が決まる．しかし，分割の仕方は非常に多数存在し，その中から上記の 2 つの条件を満たす分割を選択することは困難である．したがって，本研究ではさらに以下の条件を加え，その中で通信時間が最小となる最適なものを選択する．

- 列を固定する．**図 4**（左）で示すように，垂直線は途中で曲がることはなく，直線である．
- 列ごとに能力の低いプロセッサから並べる．

これらの条件を加えると，矩形分割の仕方は**図 5** のようになる．ここで，列数を C，第 c 列に割り当て

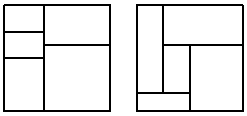

図 4 列が固定されている分割（左），固定されていない分割（右）
Fig. 4 Column-fixed partitioning.

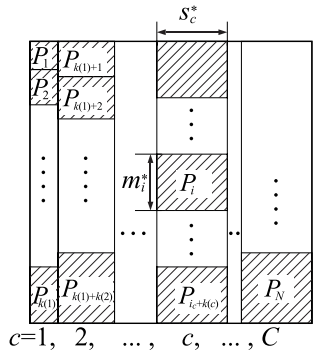

図 5 矩形分割方法.
Fig. 5 Rectangular partitioning pattern.

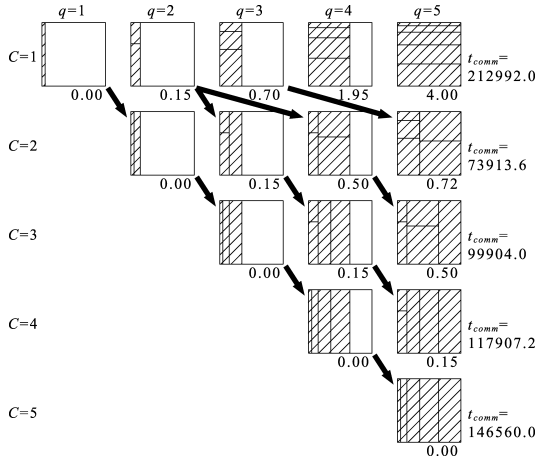

図 6 分割決定アルゴリズムの流れ．プロセッサ数 $N = 5$．プロセッサの能力（0.05, 0.10, 0.20, 0.30, 0.35）．NN サイズ n-m-l：203-80-26．学習サンプル数 $s = 1024$．各単位正方形の右下の値が $t(C, q)$ である．
Fig. 6 Process of the partition deciding algorithm.

られたプロセッサ数（垂直方向の分割数）を $k(c)$ とする．第 c 列の幅 $s_c^*(= 1 \times s_c^*)$ は $k(c)$ 台のプロセッサの総能力とする．すなわち，s_c^* は式 (7) のようになる．

$$s_c^* = \sum_{i=i_c+1}^{i_c+k(c)} p_i \ , \quad i_c = \sum_{j=1}^{c-1} k(j) \tag{7}$$

この分割をしたときの，全体の通信時間 t_{comm} は式 (8) で概算できる．

$$t_{comm} = 2ls \cdot \max_c(s_c^*(k(c)-1)) + 2(l+n)m(C-1) \tag{8}$$

第 1 項は垂直方向の通信時間を表している．垂直方向の通信はノード並列に関する通信であり，同じ列の中（並列グループ内）のプロセッサが出力層の値を求めるための値を共有する通信である．この通信は，1 つのプロセッサに値を集め，足し合わせた後，その結果をブロードキャストする．本研究ではプロセッサ間通信は，1 対 1 通信で行っている．つまり，1 列に $k(c)$ 台のプロセッサがあるので，通信回数は $2(k(c)-1)$ となる．また，列幅は固定されており，各列は同時に通信をすることが可能であるため，通信時間は列の中で最大のものとなる．したがって，通信回数 $2(k(c)-1)$ に通信要素数 $ls \cdot s_c^*$ を乗じたものが各列の通信時間となり，その中の最大値を垂直方向の通信時間とする．

第 2 項は水平方向の通信時間を表している．水平方向の通信は同時に行えない場合があるので，通信要素数は $(l+n)m$ とする．水平方向の通信を行うプロセッサ数は C 台であるので，第 1 項の場合と同様に考え，通信時間は $2(l+n)m(C-1)$ となる．

この通信時間 t_{comm} が最小となる C と $\{k(c)\}$ を求め，最適な分割として採用する．

5.3 分割決定アルゴリズム

通信時間 t_{comm} を最小にする分割を決定するアルゴリズムは，動的計画法を用いる．C を列数，q を割当て済みプロセッサ数とする．

図 6 のように，各 C，q で通信時間が最小となる分割を決めていく．この C，q に依存して変化する t_{comm} の第 1 項の変化する部分を，次のようにおく．

$$t(C, q) = \max_{c \le C}(s_c^*(k(c)-1)) \tag{9}$$

t_{comm} の第 1 項は垂直方向の通信を表しており，各列の最大値である．したがって，$t(C, q)$ は以下のように計算できる．q' 台のプロセッサが列数 $C-1$ に分割されているとして，その分割に新しい列 C を加える．その列には，$k(C) = q - q'$ 台のプロセッサを割り当てる．つまり，垂直方向に $q - q'$ 分割する．その新しい列に関する通信時間は $t'(q', q) = (\sum_{i=q'+1}^{q} p_i)(q - q' - 1)$ である．$t'(q', q)$ と $t(C-1, q')$ の大きい方がその分割での垂直方向の通信時間となる．したがって，各 q' $(q' = C-1, ..., q-1)$ において，それぞれの分割をしたときの通信時間を求め，その中で最小となるものが $t(C, q)$ を与える．すなわち，次のようになる．

$$t(C, q) = \min_{q'}\{\max\{t'(q', q), t(C-1, q')\}\} \tag{10}$$

(1) プロセッサを能力 p_i に関して昇順に並べる．
(2) /* $t(C,q)$ を順に求める．*/
for $C = 1$ **to** N
for $q = C$ **to** N

$$t(C,q) = \min_{q' \in [C-1, q-1]} \left(\max \left((\sum_{i=q'+1}^{q} p_i)(q - q' - 1), t(C-1, q') \right) \right)$$

end for
end for
(3) $t_{comm}(C) = 2ls \cdot t(C,N) + 2(l+n)m(C-1)$ を $C = 1, 2, ..., N$ について計算し，最小となる C を求め，その時の分割を選択する．

図 7 分割決定アルゴリズム
Fig. 7 Partition deciding algorithm.

この計算の例を図6に示す．この図では，$t(C-1, q')$ を与える分割に対して，新しい列を加えたとき，最小となる $t(C,q)$ の分割を矢印で示している．

すべて計算した後，$t(C,N)$ の値を用いて，t_{comm} を計算し，C の値を動かして，最小となる分割を決定する．具体的なアルゴリズムを**図 7** に示す．このようにして，適切に能力に対応した処理を割り当て，通信時間が最小となる矩形分割，すなわちマッピングが一意に決定できる．

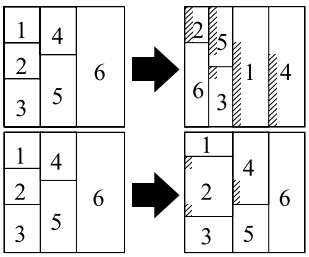

図 8 2種類の再マッピング手法．全体で再マッピング（上）．各列ごとに再マッピング（下）．斜線部分の高さは再マッピングに際して転送が必要となる重み行列の量を示している
Fig. 8 Two re-mapping methods.

6. 動的なマッピング手法

前章ではプロセッサの能力比は既知であるとし，それを用いて非均質性を考慮したマッピングを行った．全体の性能はこの能力比の見積りの正確さに大きく影響される．適切な能力の見積りを行うことは非均質環境を対象とするうえで非常に重要な処理であるが，計算と通信の割合など処理の特性によっても適切な割当ては異なり，適切な見積りは非常に困難である．こうした問題に対応するために，本章では前章で提案した SRPM 法を拡張して，BP 法の処理中に動的にマッピングを変更する手法（DRPM 法：Dynamic Rectangular Partitioning Mapping）を提案する．

BP 法は同様の処理を，誤差が閾値以下になるまで繰り返す処理である．したがって，ある時点までの処理時間を用いて能力比を再推定し，動的にマッピングを変更する．再マッピングには，ある程度の時間（コスト）がかかるため，それ自体のコストの削減も考慮する．

あらかじめ能力比の概算（初期能力比）を与えるがより適切な能力比を求めて高速化することを目的とする場合と，計算機の能力が未知の状態で処理を開始しても高い性能となることを目的とする初期能力比なしの場合の 2 つの場合を考える．

6.1 再マッピング手法

再マッピング自体のコストと処理時間の削減のバランスを考慮し，本研究では次の 2 種類の再マッピング手法を用いる．これらはプロセッサの処理時間の不釣り合いの程度に応じて使い分ける．

- 全体で再マッピング：新しい能力比を用いて，初めからマッピングし直す（**図 8**（上））．
- 各列ごとに再マッピング：列数 C，各列に割り当てられるプロセッサ数 $k(c)$，およびプロセッサの順番は変えず，新しい能力比に応じて分割位置のみを変更する（図 8（下））．

全体で再マッピングを行う場合，各列に割り当てられるプロセッサとその数 $k(c)$ を求め直す必要がある．列ごとに再マッピングをする場合は，C，$k(c)$，プロセッサの順番は不変であるため，再推定した能力比に応じて分割位置を変更するだけでよく，計算コストを削減することができる．

各プロセッサは自分が担当する部分の処理に用いる部分重み行列のみを保持し更新する．再マッピングを行うことで担当する部分が変化するため，再マッピング以前にその部分を担当していたプロセッサから，通信を用いてその部分の重み行列を得る必要がある．図 8 の斜線部分の高さはその重み行列の量を示している．列ごとに再マッピングを行うことにより，明らかに全体で再マッピングを行う場合よりも通信による転送が必要な重み行列は少なくなる．さらに，垂直方向に隣

接したプロセッサどうしが通信すれば十分であり，通信を行うプロセッサの対の決定が容易である．全体で再マッピングを行うと様々なプロセッサ間で通信する必要があり，コストが大きくなる．したがって，基本的にはコストの小さい列ごとの再マッピングを行い，それでは対応できないほどの負荷の不均衡がある場合は全体で再マッピングを行う．

なお，全プロセッサが完全な学習サンプル集合を保持しており，同じ学習サンプル集合を繰り返し使用するため，再マッピングをしても学習サンプルをプロセッサ間でやりとりする必要はない．

6.2 プロセッサ能力比の再推定

過去の実際の処理時間を用いてプロセッサの能力比の再推定を行う．まず，再推定に用いる処理時間について考える．並列化された BP 法の処理は，計算 1（Forward フェーズ）→通信 1（垂直方向）→計算 2（Backward フェーズ）→通信 2（水平方向）→計算 3（Modify フェーズ）という順番で行われる．各列ごとに再マッピングを行うとき用いる能力比は計算 1，通信 1，計算 2 の処理時間 t_1 を用いて求め，全体で再マッピングを行うとき用いる能力比は計算 2 の処理時間 t_2 を用いて求める．再マッピング手法により用いる処理時間を変えて能力比を求めるのは次のような理由による．計算時間に加え，通信時間を考慮することでタスクの性質を反映した適切な能力の見積りが行える．しかし，全体で再マッピングする場合，各列に割り当てられるプロセッサ数 $k(c)$ をはじめ，分割が大きく変化し，それにともない通信時間も変化するため，全体で再マッピングを行う場合は通信時間を含まない処理時間 t_2 を用いる方が能力比の適切な見積りができる．なお，処理時間 t_2 に計算 1 の時間を含めない理由は，各プロセッサ間の計算 1 の計算量の割合と計算 2 の計算量の割合はほぼ等しいため，計算 2 を用いれば十分なためである．

この処理時間を用いて，プロセッサの能力を次のようにして再推定する．あるプロセッサに割り当てた処理量と，そのときの処理時間は比例すると考えられるため，**図 9** のように縦軸に処理量を横軸にそのときの処理時間をとり，各プロセッサに関してプロットすると直線上に分布する．プロセッサの能力は単位時間あたりの処理量としているため，最小 2 乗法を用いて直線を推定し，その傾きを新しい能力とする．また，過去 L 回の処理時間の値を用いて新しい能力を推定する．本研究では予備実験により，$L = 6$ とした．

6.3 再マッピングを行う条件

イタレーション R 回ごとにプロセッサ間の処理時間の不釣り合いの程度を検査し，再マッピングの必要性を判断する（予備実験により $R = 20$ とした）．再マッピングを行う条件を**図 10** に示す．t_{min}，t_{max} は各々プロセッサの中での処理時間 t_1 の最小値，最大値である．T_{whole}，T_{line} は閾値であり，予備実験を基に，$T_{whole} = 0.4$，$T_{line} = 0.8$ とした．

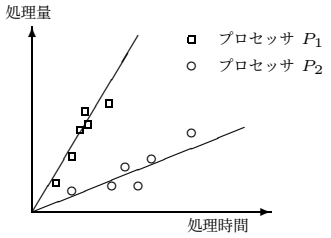

図 9 最小 2 乗法によりプロセッサの能力比を推定
Fig. 9 Estimating performance of processors using least-square method.

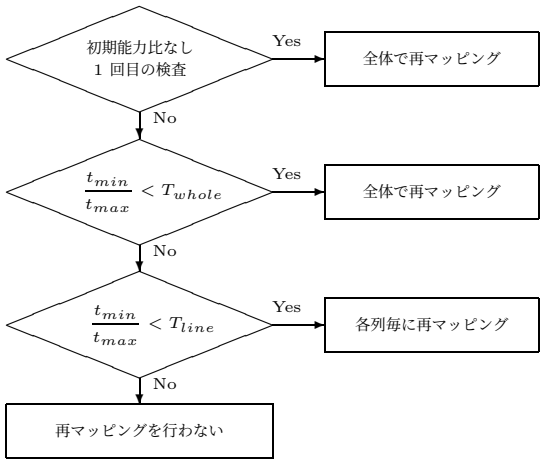

図 10 再マッピングを行う条件
Fig. 10 Condition of the re-mapping.

初期能力比がない場合，処理開始時には均等に処理をマッピングするため，1 回目の検査後に大きくマッピングを変更する必要がある．これが，1 つ目の条件である．そして，後半 2 つの条件はプロセッサ間の処理時間の不釣り合いに関する条件である．各プロセッサでの処理時間が等しいときに最高の性能となると考えられるため，処理時間 t_1 の比を計算することでプロセッサ間の処理時間の不釣り合いを検査し，その程度により用いる再マッピング手法を決定する．

7. 性能評価

SRPM 法と DRPM 法の性能を評価することにより，非均質性を考慮したマッピング法の有効性と処理の実行中に適切な能力比の推定を行うことの有効性を

表 3 実験条件
Table 3 Experimental conditions.

CPU	Intel Pentium III 800 MHz, 500 MHz
OS	Linux 2.4.19
通信ライブラリ	LAM/MPI version7.0
NIC	1000base-T (32bit/33 MHz PCI)
NN サイズ	n-m-l: 203-80-26
学習サンプル数	$s = 1024$

確認する．本研究ではプロセッサの能力の違いや他の実行中ジョブによる非均質環境を想定しており，事前に最適な能力比を求めることは困難であると仮定している．実験においても真の能力比が分からないことを前提に実験条件を決めている．

7.1 実験条件

実験条件は**表 3** のとおりである．BP 法で扱うタスクは NETtalk[8] とした．2 種類のプロセッサ（800, 500 MHz）を用い，さらに，複数のジョブをバックグラウンドで実行して，見かけの能力を落とすことで非均質性を実現した．実験対象の各手法で必要な設計パラメータであるプロセッサの能力は，クロック周波数に比例し，プロセッサあたりのジョブ数に反比例すると単純に仮定して算出した（もちろん，これは BP 法の処理に使うプロセッサの真の能力比には一致しない）．実験には，2 種類の環境を用意した．各環境でのプロセッサの能力比は次のようになっている．条件 A（非均質性：強/ 0.25, 0.31, 0.63, 1.0, 1.0, 0.42, 0.67, 0.63）．条件 B（非均質性：弱/ 0.63, 0.63, 0.63, 1.0, 0.63, 1.0, 1.0, 0.63）．プロセッサは 4〜8 台使用した．8 台未満の場合は，左から順に使用する．つまり，条件 A でプロセッサ数が 5 台のときに使用するプロセッサの能力比は 0.25, 0.31, 0.63, 1.0, 1.0 である．用いた手法は以下の 5 手法であり，手法 1，2，SRPM 法は処理中にマッピングを変更しない静的なマッピング手法である．

- 手法 1：非均質性未考慮（すべてのプロセッサに均等に処理をマッピングする）．
- 手法 2（4 章で述べた手法 H_{rev}）
- SRPM 法（5 章で提案した手法）
- DRPM 法（初期能力比あり）
- DRPM 法（初期能力比なし）

7.2 評価手法

非均質環境は様々な非均質性が考えられるため，非均質性によらない評価を行うために，評価尺度として並列化効率 γ を導入する．並列化効率 γ は，使用した各計算機のスループットの総和に対する，並列処理の時間から求めたスループットの割合とする．スループットは処理時間の逆数とし，N 台での処理時間を $T^{(N)}_{parallel}$，プロセッサ P_i の 1 台での処理時間を $T^{(i)}_{series}$ とすると並列化効率 γ は式 (11) で求められる．

$$\text{並列化効率}\gamma = \frac{\left(T^{(N)}_{parallel}\right)^{-1}}{\sum_{i=1}^{N}\left(T^{(i)}_{series}\right)^{-1}} \qquad (11)$$

通信時間などがまったくなく，各プロセッサが最大限にその能力を使用できたときに並列化効率が 1.0 となる．したがって，並列化効率が 1.0 に近いほどよい並列化手法であるといえる．

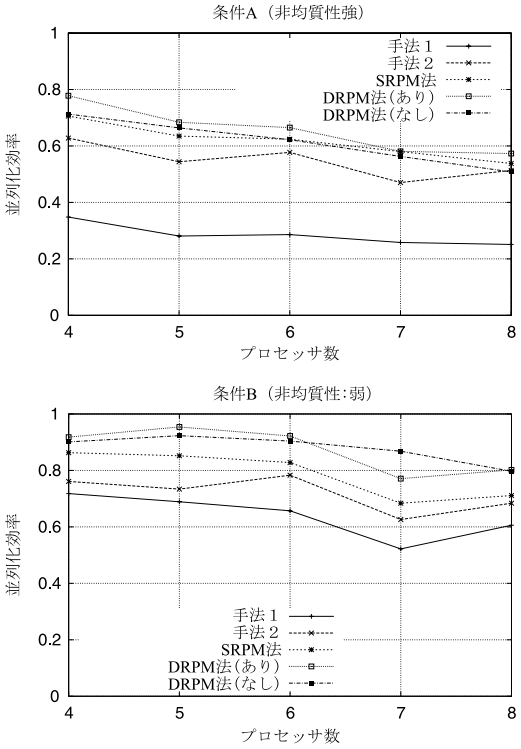

図 11 実験結果
Fig. 11 Experimental results.

7.3 実験結果

実験結果を**図 11** に示す．横軸がプロセッサ数 N，縦軸が並列化効率 γ である．なお，手法 1，2 の値は，複数通りとりうるデータ並列度 N_{DP} の中で最高の性能となる場合の値である．非均質性を考慮していない場合は効率が非常に悪く，非均質性を考慮する必要があることが確認できる．手法 2 と SRPM 法を比較すると，SRPM 法の方が良い性能となった．SRPM 法は手法 2 よりも適切にプロセッサの能力の非均質性を考慮してマッピングを行い，能力を有効に利用することが可能になったことが確認できる．さらに，手法 2

の並列化効率は各プロセッサ数に対して複数通りあるデータ並列度の中で最高の値である．したがって，手法 2 は最高の性能となるデータ並列度の選択，つまりマッピングを選択する必要があるが，SRPM 法ではマッピングを一意に決定できるため，手法 2 よりも優れた手法であるといえる．

さらに，$N = 5$，7 で手法 2 の性能が落ちている．これは手法 2 はデータ並列度が 1 か N しかとれないためである．つまり，手法 2 が並列化手法としてノード並列かデータ並列しか用いることができないのに対して，SRPM 法では両並列化手法を用いることができ，このような場合にも柔軟に対応できる．

次に SRPM 法，DRPM 法（あり），DRPM 法（なし）を比較する．多くの場合で DRPM 法（あり）が最も良い性能となっているが，初期能力比（あり）と（なし）との差はわずかである．SRPM 法は仮定した初期能力比を用いてマッピングしているが，その能力比は厳密に正確なものではなく，BP 法の処理中に推定した能力比を用いる DRPM 法が高性能になっている．実験において推定された能力比の例を示す．条件 B（非均質性：弱），DRPM 法（なし）により推定された能力比は，0.49，0.50，0.49，1.0，0.52，0.98，1.0，0.51 となった．仮定から求めた初期能力比の 0.63 が 0.5 近辺の値をとり，初期能力比と実際の能力比は異なる値となっている．また，初期能力比が同じプロセッサ間でも，マッピングの仕方により通信の様子が異なるため，プロセッサ間に能力比の差が生じ，それを反映したマッピングは通信を考慮した適切なものとなっていると推定される．

DRPM 法の再マッピング発生の様子とその効果を確認する．プロセッサ数 $N = 4$，条件 B（非均質性：弱）の環境において，横軸に再マッピングのための検査の回数，縦軸に検査間のイタレーション R 回の処理時間をとり，実行された再マッピングの種類をプロットしたグラフを**図 12** に示す．初期能力比がある場合はない場合よりも 1 回目の検査までの処理時間が短い．条件 B の環境が変化していないためマッピングは 1 つに収束することが期待されるが，それが 3 回目以降で起きている．1〜2 回の再マッピングを行った後，どちらの場合も同等の処理時間となり，短時間のうちに適切なマッピングとなった．この結果は，処理時間中に他のユーザのジョブの開始や終了のようなステップ関数的な負荷変動が起こり，環境が変化した場合，能力比の変化を検知し，短時間で再マッピングが行われることを示唆している．DRPM 法の（あり），（なし）の性能の差は収束までのステップ数の差を反映したものと推測される．

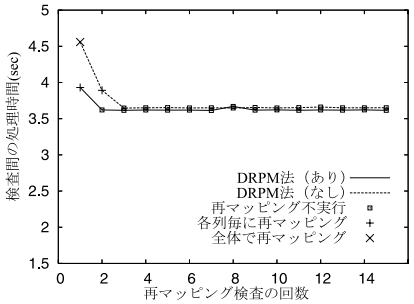

図 12 再マッピング発生の様子とその効果
Fig. 12 Moment of a re-mapping and its efficacy.

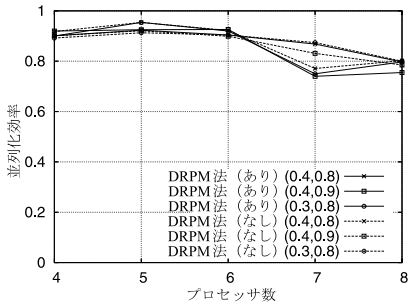

図 13 閾値 T_{whole}，T_{line} の影響
Fig. 13 Parameter's influence.

全体および列ごとの再マッピングを行う条件に関する閾値 T_{whole}，T_{line} の値は，全体での再マッピングが起こり難くなるようにし，逆に列ごとの再マッピングは容易に起こることを許容するように設定した．これらの値が変化した場合の影響について調べた．条件 B（非均質性：弱）の環境において，$(T_{whole}, T_{line}) = (0.4, 0.9)$，$(0.3, 0.8)$，$(0.4, 0.8)$ としたときの実験結果を**図 13** に示す．ほとんどの場合でほぼ同等の性能となり，0.4，0.8 の値のとき最大となっていることはこの値の妥当性を示している．ただし，小さな性能差がある場合もあり，最適な値を得るためのチューニングが必要かもしれないが，今後の課題である．

新しい能力の推定に用いる過去の処理時間のデータ数 L に関して，$L = 3$，6，8 と変化させた予備実験をあらかじめ行った．その結果は割愛せざるをえないが，大きな差は見られず，最大性能が $L = 6$ で実現されていることが多かった．つまり，頻繁な負荷の変動がない場合には，L が極端な値でなければ，その変化の性能への影響は小さいと考えられ，$L = 6$ の妥当性を確認した．このパラメータは最小 2 乗法で傾きを推定するためのデータ数であり，データ数が小さければその傾きの信頼性は低下し，また大きければより過去のデータを利用することになり現在の環境への追随

性が悪くなる．その観点からは負荷変動率などに応じて動的に値を変更する方法が必要となるが，今後の課題である．

再マッピングの検査間のイタレーション回数 R に関しては，その値が小さければ検査のオーバヘッドが増加し，大きければオーバヘッドは小さくなるが緩やかな負荷変動にしか追随できないことになる．R の値を決めるため，$R = 10$，20，30 と変化させて予備実験を行った．$R = 10$（初期能力比なし）のときの性能がやや低くなり，再マッピングの必要性の検査のオーバヘッドの影響が見られたが，それ以外では性能はほぼ同等となり，$R = 20$ が適切であると判断された．ただし，非均質性の違いなどによる全体の能力の差によりイタレーションあたりの処理時間は異なるので，実際の処理時間から R を決定することは有効と考えられ，またそれにより負荷変動にもより適切に対応できることになる．パラメータの動的な決定による本手法の改良は今後の課題である．

8. ま と め

プロセッサの能力の非均質性を考慮した BP 法のマッピング手法である SRPM 法を提案した．本手法では，BP 法の並列化を矩形分割によってモデル化することで，プロセッサの能力に応じた処理の分配と通信時間を同時に考慮することが可能となった．そして，通信時間が最小になる分割を最適な分割と考え，その分割を選択するアルゴリズムを提案した．このアルゴリズムが必要とするパラメータは各プロセッサの能力のみであり，非常にシンプルな手法である．さらに，動的にマッピングを変更する DRPM 法を提案した．計算機クラスタ上に提案手法を実装し，高速に処理が行えることを確認した．また，能力が未知の環境を対象とする場合など制約がある場合でも高い性能となることを確認した．

今回は計算能力の非均質性のみを扱った．しかし，通信に関しても非均質な環境が存在する．動的にマッピングを行うことで，ある程度，通信能力に関する非均質性を考慮することができるが，適切なマッピングを考える時点で通信能力の非均質性も考慮できるように拡張することが今後の課題である．

参 考 文 献

1) Singer, A.: Implementation of Artificial Neural Network on the Connection Machine, *Parallel Computing*, Vol.14, pp.305–315 (1990).
2) Ayoubi, R.A. and Bayoumi, M.A.: Efficient Mapping Algorithm of Multilayer neural Network on Torus Architecture, *IEEE Trans. Parallel and Distributed Systems*, Vol.14, No.9, pp.932–943 (2003).
3) Suresh, S., Omkar, S.N. and Mani, V.: Parallel Implementation of back-Propagation Algorithm in Networks of Workstations, *IEEE Trans. Parallel and Distributed Systems*, Vol.16, No.1, pp.24–34 (2005).
4) 東 大亮，菅谷至寛，阿曽弘具：非均一な並列計算機環境におけるニューラルネットワークの学習，情報処理学会研究報告 HPC-89-1, pp.1–6 (2002).
5) 小澤洋司，菅谷至寛，阿曽弘具：非均質環境を考慮した誤差逆伝播法のノード並列アルゴリズム，情報技術レターズ，Vol.2, LA-002, pp.5–6 (2003).
6) Andonie, R., Chronopoulos, A.T., Grosu, D. and Galmeanu, H.: Distributed Backpropagation Neural Networks on a PVM Heterogeneou System, *Proc. Parallel and Distributed Computing and Systems Conf.*, pp.555–560 (1998).
7) Beaumont, O., Boudet, V., Rastello, F. and Robert, Y.: Matrix Multiplication on Heterogeneous Platforms, *IEEE Trans. Parallel and Distributed Systems*, Vol.12, No.10, pp.1033–1051 (2001).
8) Sejnowski, T.J. and Rosenberg, C.R.: Parallel networks that learn to pronounce English text, *Complex Systems*, Vol.1, pp.145–168 (1987).

(平成 17 年 3 月 25 日受付)
(平成 17 年 10 月 11 日採録)

小澤 洋司（正会員）

平成 14 年東北大学工学部通信工学科卒業．平成 16 年同大学院工学研究科博士前期課程修了．平成 16 年より日立製作所．在学中，並列分散処理の研究に従事．

菅谷 至寛（正会員）

平成 7 年東北大学工学部通信工学科卒業．平成 9 年同大学院工学研究科博士前期課程修了．平成 14 年同大学院工学研究科博士後期課程修了．平成 12 年より同大学院工学研究科助手．博士（工学）．シストリックアルゴリズム設計支援システム，タスク割当て，並列ソフトウェア等の研究に従事．並列分散処理全般に興味を持つ．電子情報通信学会会員．

阿曽　弘具（正会員）

昭和 43 年東北大学工学部電気工学科卒業．昭和 49 年同大学院博士課程修了．昭和 48 年東北大学工学部助手，昭和 54 年名古屋大学工学部講師．昭和 57 年同助教授，昭和 61 年東北大学工学部助教授を経て，平成 3 年同教授．現在，同大学院工学研究科教授．工学博士．その間，学習オートマトン，セル構造オートマトン，並行処理理論，シストリックアルゴリズム設計論，文字認識，音声認識，ニューラルネットワーク等の研究に従事．平成 3 年度電子情報通信学会業績賞受賞．IEEE，ACM，EATCS，電子情報通信学会，人工知能学会，日本認知科学会，LA 各会員．