# 並列計算機を用いた最適化計算の実際

山川　栄樹

## 1．はじめに

並列処理が身近なものとなる中で，数理計画問題に対してもさまざまな並列アルゴリズムが提案されている［4，5］．その多くは，双対理論と関数（行列）分割の考え方にもとづいており，もとの問題の近似モデルを複数の独立な部分問題に分解し，各反復でこれらを並列的に解くことによって原問題の解に至るような点列を生成するものである．特に，主問題または双対問題の変数を単位とするような粒度の細かい分割が行なわれ，生成された部分問題の解が解析的に求められる場合には，データ並列型のアルゴリズムが得られる．一方，多品種流問題などのように，もとの問題の制約条件が特殊なブロック構造をもつ場合には，その双対問題の目的関数もある種のブロック構造を示すことが多い．そこで，双対問題をこのブロック構造に沿って分解すれば，比較的粒度の粗いコントロール並列型のアルゴリズムを得ることができる．

ここ数年の間に，いろいろな処理機構をもつ大規模並列計算機が相次いで商品化され，遺伝子解析や自然言語処理など膨大な計算を要する分野を中心に利用が始まっている．しかし，使用する並列計算機固有の性質を十分意識した上でプログラミングを行なわない限り，アルゴリズムのもつ理論的な並列化効率はおろか，計算機本来の性能すら十分に引き出せないこともある．

並列計算については，本誌1992年8月号に特集が組まれており，計算機のハード・ソフトからアルゴリズムの可能性とその限界に至るまで，幅広い視点から解説されている．本稿では，筆者が実際に並列計算機を用いて行なった数理計画問題に対する並列アルゴリズムの開発と検証の結果をもとに，並列計算機を用いて現在どの程度の最適化計算が実行でき，アルゴリズムの実現にはどのような問題点が存在するかについて述べることにする．

やまかわ えいき　エイ・ティ・アール人間情報通信研究所
〒619-02 京都府相楽郡精華町光台2丁目2番地

## 2．並列計算機CM-5

国内では，北陸先端科学技術大学院大学や筆者の勤務する研究所など10余りの研究機関がCM-5を導入している．数理計画の分野においても，CM-5を用いた数値実験結果を含む論文がすでにいくつか報告されている［1，2，3，6，9］．そこで，CM-5とそのソフトウェアを簡単に紹介しよう．詳細は，World Wide Webサービス http：／／www．think．com／を通じて入手可能な文献［8］を参照されたい．CM-5はSPMD（Single-Program Multiple-Data）型の並列計算機であり，多くのデータに対して一斉に同じ処理を行なうデータ並列型のアルゴリズムと，各プロセッサが独立に異なる処理を行なうコントロール並列型のアルゴリズムの双方を実行できる．本文中に示す数値実験に用いたCM-5は，32台のプロセッサから成る．各プロセッサは，32 MFLOPSの浮動小数点演算用ベクトル・ユニットを4個搭載している．また，ベクトル・ユニットには8 Mbyteのローカル・メモリが装備され，メッセージ伝達（message passing）にもとづく分散メモリ型の疎結合システムを構成している．

データ並列型のアルゴリズムは，CM FortranやC*という固有のプログラミング言語によってコーディングされる．一方，コントロール並列型のアルゴリズムの記述には，FORTRAN 77やCなどの従来型の言語が利用できる．また，コントロール並列型のアルゴリズムにおいて個々のプロセッサが実行する手続きを，データ並列型の言語を用いて記述することもできる．プログラムの実行形態は，コンパイルの際に利用者が指定する．データ並列型で実行する場合には，ベクトル・ユニットが並列処理の単位となり，それらの上で唯一のプログラムが動作する．データは，各ベクトル・ユニットに付属するローカル・メモリに分散し

表1　配列全体に対する加減乗除の実験結果

| $n$ | $m$ | L | 計算時間 (秒) |
|---|---|---|---|
| 1024 | 1 | 100000 | 5.6 |
| 16384 | 1 | 100000 | 14.5 |
| 262144 | 1 | 100000 | 180.4 |
| 1024 | 10 | 10000 | 0.9 |
| 16384 | 10 | 10000 | 8.4 |
| 262144 | 10 | 10000 | 131.1 |
| 1024 | 100 | 1000 | 0.6 |
| 16384 | 100 | 1000 | 8.1 |
| 262144 | 100 | 1000 | 127.6 |

て保持される．ベクトル・ユニットの間で必要となる通信や実行する命令の同期制御は，システムが自動的に行なう．一方，コントロール並列型の実行を指定すると，各プロセッサは，自らが保持するデータの上でシステムから受取ったプログラムのコピーを独立に実行する．プロセッサ間の通信や同期制御は，メッセージ伝達のライブラリーCMMDのルーチンを用いてプログラムに記述しておかなければならない．各プロセッサにデータを分割する方法も，利用者が決定する必要がある．実際には，プログラムが走行しているプロセッサの番号を返すCMMDの関数を利用して，実行すべきモジュールをプロセッサ自身に識別させる方法が一般的である．プログラムがデータ並列型の言語で記述されているならば，各プロセッサは4個のベクトル・ユニットを用いてこれを並列的に実行する．その際，各プロセッサのデータは，ベクトル・ユニットに付属するローカル・メモリに分散して保持される．

CM Fortranは Fortran 90に準拠しており，配列全体を1つの実体として扱う．ベクトル $x$, $z$ の各成分の値を格納する長さ $n$ の1次元配列を $X$, $Z$ とするとき，文

```
W=SUM (X * Z)
```

は内積 $w=x^Tz$ を与える．ここで，X * Zは配列X, Zの対応する要素同士のかけ算，SUMは配列の全要素の和を返す組込み関数である．並列処理の形態は，配列要素の配置を定めるレイアウト文で決まる．たとえば，

```
CMF$ LAYOUT X(:NEWS), Z(:NEWS)
```

とすると，$n$ 個の仮想的なプロセッサにX, Zの要素が1つずつ割当てられ，X * Zは各要素について並列に処理される．一方，2次元配列Aに対して

```
CMF$ LAYOUT A(:NEWS, :SERIAL)
```

と指定すると，第1成分の要素番号が同じデータは同一のプロセッサに配置される．したがって，第1成分の要素番号が異なるデータに対する処理は並列的に行なわれるが，第2成分の要素番号が異なるデータに対する処理は逐次的に実行される．また，第2成分の要素番号についての和は，プロセッサ間通信を行なうことなく計算できる．

配列の部分を取り扱う方法も用意されている．文

```
X(3:5)=1.
```

は配列Xの第3～5要素に1.を代入する．また，配列Xの要素のなかで，同じサイズの論理配列Sにおいて値が真である要素に対応するものの最大値は，

```
W=MAXVAL (X, MASK=S)
```

により求められる．配列要素への代入はFORALL構文により並列的に実行できる．[1:$j$]を第 $i$ 要素の値が $i$ であるような長さ $j$ の1次元整数配列とするとき，文

```
FORALL (j=1 :n) Z(j)=
*     SUM (X(MAXVAL([1:j], MASK=
      S(1:j)):j))
```

は，論理配列Sの真値の要素に対応する位置で区切られた配列Xの部分和を求める処理 [13] である．

CM Fortranのプログラムにおいて，連続して記述された並列的に実行可能な処理の固まりはコード・ブロックと呼ばれ，そのサイズが大きいほど実行時の効率はよくなる．IF文やDOループ，サブルーチンや関数呼出しがあると，コード・ブロックは切断される．長さ $n$ の1次元倍精度実数配列T, V, X, Zに対して，文

```
DO 100  I = 1, L
     X=(X+Z) * T/V-Z
          :                  } m 行繰返し.
     X= (X+Z) * T/V-Z
100 CONTINUE
```

を実行した場合の計算時間を表1に示す．配列全体としての加減乗除の実行回数はいずれも100000回であるが，配列の長さ $n$ とDOループ内の行数 $m$ の値が大きいほど，1要素あたりの計算時間は短いことがわかる．本来の性能にはほど遠い結果であるが，条件判定を含む反復解法では，この程度の性能にとどまることが多い．

## 3．データ並列型アルゴリズム

数理計画法のアルゴリズムにおいては，最適化問題を構成する各関数の勾配ベクトルなどを用いた線形代

数的な演算により，探索点の生成を行なうことが多い．特に既存の逐次アルゴリズムを用いて大規模な問題を解こうとするとき，探索方向の決定や近似モデルの修正に伴う線形代数的な演算は，最も計算時間を要する処理の1つとなる．したがってこのような演算のなかで独立に実行できる部分を並列化するだけで，大規模な問題が実用的な時間で解けるようになる場合がある．

たとえば文献［11］では，主双対実行可能内点法の各反復で解くべきニュートン方程式に共役勾配法を適用し，データ並列型のプログラミング環境のもとで実行する方法を示している．正定値対角行列 $D\in R^{n\times n}$ とベクトル $c\in R^n$，$b\in R^m$ および階数が $m$ の行列 $A\in R^{m\times n}$ により定義される次のような凸2次計画問題を考える．

$$\text{目的関数：}\frac{1}{2}x^T Dx+c^T x \rightarrow \text{最小}$$

$$\text{制約条件：}Ax=b,\ x\geq 0.$$

$y\in R^m$，$z\in R^n$ をそれぞれ $Ax=b$，$x\geq 0$ に対する双対変数，$X$，$Z$ をそれぞれ $x$，$z$ の各要素を対角成分とする対角行列，$e\in R^n$ を各要素が1のベクトル，$\mu$ を正のパラメータとするとき，各反復における計算時間の大半は，次の連立方程式を解くために費やされる．

$$A(D+X^{-1}Z)^{-1}A^T\Delta y = A(D+X^{-1}Z)^{-1}(z-\mu X^{-1}e)$$

したがって，共役勾配法の主要な計算は，$Ax$ および $A^Ty$ という2通りの行列とベクトルのかけ算になる．

行列 $A$ が比較的小規模で密な場合には，仮想的なプロセッサをサイズが $m\times n$ の2次元格子状に配置して，その第 $(i,j)$ 成分にベクトル $x$ の第 $j$ 要素，ベクトル $y$ の第 $i$ 要素，および，行列 $A$ の第 $ij$ 要素の値をもたせる．そして，第2成分の要素番号についての和および第1成分の要素番号についての和をとることにすれば，上の2通りのかけ算を容易に実現することができる．

行列 $A$ が大規模で疎な場合には，非零要素に対応するプロセッサのみを直線状に並べる．このとき，同じ添字 $i$ に関する情報を格納するプロセッサを固めて配置すれば，2章で述べた配列の部分和を求める処理を用いて積 $Ax$ を並列的に計算できる．さらに，同じ添字 $j$ に関する情報が固まるように各プロセッサのデータを並べ換えるポインタを用意すれば，積 $A^Ty$ も同様にして計算できる．ところが，比較的少数のベクトル・プロセッサから成る並列計算機上で，非零要素の構造が一様でない大規模行列に対して配列の部分和を求める処理を適用すると，特定のプロセッサへの負荷の集中やプロセッサ間通信の衝突などにより，処理速度が低下する場合がある．そこで，CM-5の数値計算ライブラリーCMSSLでは，全く異なる考え方にもとづくルーチンを提供している．$Ax$ を計算する場合には，非零要素 $A_{ij}$ と $x_j$ をベクトル・レジスタの同じ要素に収集(gather)して $A_{ij}x_j$ を求め，結果を格納すべきベクトルの第 $i$ 要素にこれを散布（scatter）する．複数の積項が散布された要素では，それらの値の和をとる．積 $A^Ty$ を計算する場合も同様である．収集・散布のパターンを生成するために前処理が必要となるものの，積の計算は高速に実行できるため，非零要素の構造が同じ行列に対して何度も繰り返し積を計算する必要がある反復解法のアルゴリズムにおいては，部分和を求める処理よりも有利になる．実際，ランダムに生成した2次計画問題に対する数値実験結果を**表2**に示す．CMSSLのルーチンを利用すると，FORALL構文により部分和を求める場合の2／3程度の計算時間で済むことがわかる．非零要素の構造が一様となるように $A$ の行と列をあらかじめ並べ換えておけば，処理速度はさらに向上すると考えられる．

**表2**　2次計画問題に対する主双対内点法の実験結果

| 係数行列 $A$ のサイズ | | | 計算時間(秒) | |
|---|---|---|---|---|
| $m$ | $n$ | 非零要素数 | FORALL | CMSSL |
| 512 | 4096 | 16384 | 3.78 | 2.84 |
| 1024 | 4096 | 16384 | 5.61 | 3.97 |
| 2048 | 4096 | 16384 | 11.69 | 8.07 |
| 1024 | 8192 | 65536 | 9.75 | 6.76 |
| 2048 | 8192 | 65536 | 15.15 | 9.87 |
| 4096 | 8192 | 65536 | 32.45 | 20.24 |
| 4096 | 16384 | 262144 | 53.77 | 33.99 |
| 8192 | 32768 | 1048576 | 258.61 | 147.48 |

一般に，各制約関数が1変数関数であり，目的関数も1変数関数の和として記述されるような最適化問題は，変数を単位として分解できる．したがって，探索点を生成するための近似モデルとしてこのような分解可能な部分問題を構成すれば，並列計算に適したアルゴリズムが得られる．実際，凸解析における双対理論を用いると，多変数の制約関数も分解可能な形で双対問題に取り込むことが可能である．その際，もとの問題の目的関数は双対変数について分解不可能な形に変換されるが，その項が微分可能であるならば，1次近

似モデルを用いることによって分解可能な部分問題を構成することができる．文献[6]では，一般の凸計画問題に対してこのような考え方にもとづく並列アルゴリズムを提案している．また，ステップ幅を決定するパラメータを自動的に調整するような機構を組み込むことによって，実際の収束性が加速されることを示している．さきほどの凸2次計画問題に対してアルゴリズムを適用すると，部分問題は $m$ 個の独立な問題

$$\text{目的関数}：\frac{\lambda_k}{2}(y_i-y_i^k)^2-\frac{r_i^k}{||a_i||^2}y_i \rightarrow \text{最小}$$

に分解される．ここで，$a_i$ は $A^T$ の第 $i$ 列ベクトル，$\lambda_k$ はステップ幅を決定する正のパラメータ，$r_i^k$ は

$$r^k=Ax^k-b$$

で定義される残差ベクトルの成分で，$x^k$ は補助問題

$$\text{目的関数}：\frac{1}{2}x^T Dx+(A^T y^k+c)^T x \rightarrow \text{最小}$$

$$\text{制約条件}：x \geq 0$$

の解である．各部分問題と補助問題はいずれも解析的に解けるため，効率的なデータ並列型のアルゴリズムが構成される．特に，$A$ が2部グラフ $G=(V_1, V_2, E)$ の接続行列となる輸送問題では，その構造的特徴を利用して必要な計算を高速に実行できる．実際，各部分問題に現われる $||a_i||^2$ は対応する節点の次数になる．また，補助問題に現われる $A^T y^k$ も，$y^k$ の各要素を枝の始点と終点の位置関係を示すポインタを用いて並べ換えたものに $y^k$ 自身を加算するだけで簡単に求められる．各節点の平均次数が8および16の場合の数値実験結果を**表3**に示す．変数の自由度が小さい前者の問題では計算時間が多少長くなるが，後者では枝数が100万を超える問題も1分半程度で解けており，非常に良好な結果といえる．

## 4．コントロール並列型アルゴリズム

解くべき問題がもともと分解しやすい構造をもつ場合には，本質的に逐次的なアルゴリズムも有効な並列アルゴリズムとなる．次の多品種輸送問題を考えよう．

$$\text{目的関数}：\Sigma_{l=1}^{L}\{\frac{1}{2}x_l^T D_l x_l+c_l^T x_l\} \rightarrow \text{最小}$$

$$\text{制約条件}：A_l x_l=b_l, \qquad l=1, \cdots, L,$$

$$\Sigma_{l=1}^{L} x_l \leq \bar{b},$$

$$0 \leq x_l \leq u_l, \qquad l=1, \cdots, L.$$

ただし，$D_l \in R^{n\times n}$ は正定値対角行列，$A \in R^{m\times n}$ は2部グラフ $G$ の接続行列，$c_l$, $\bar{b}$, $u_l$ はいずれも $n$ 次元ベクトル，$b_l$ は $m$ 次元ベクトルである．$A_l x_l=b_l$ に対する双対変数を $y_l \in R^m$，$\Sigma_{l=1}^{L} x_l \leq \bar{b}$ に対するスラック変数と双対変数をそれぞれ $t \in R^n$ と $v \in R^n$，$t$, $v$ の各要素を対角成分とする対角行列をそれぞれ $T$, $V$ と書くとき，主双対実行可能内点法の各反復で解くべきニュートン方程式は，品種ごとに独立な $L$ 個の方程式

$$A\Psi_l A^T \Delta y_l=A(\Psi_l \Delta v-\lambda_l)$$

と，$v$ の探索方向を求めるためのもう1つの方程式

$$\{\Sigma_{l=1}^{L}(\Psi_l-\Psi_l \hat{A}_l \Psi_l)+V^{-1}T\}\Delta v$$
$$=\Sigma_{l=1}^{L}(\lambda_l-\Psi_l \hat{A}_l \lambda_l)+\mu V^{-1}e-t$$

に分解される[12]．ここで，$\Psi_l \in R^{n\times n}$ は正定値対角行列，$\lambda_l$ は $n$ 次元ベクトルであり，$n\times n$ 行列 $\hat{A}_l$ は

$$\hat{A}_l=A^T(A\Psi_l A^T)^{-1}A$$

と定義される．$n$ 次元ベクトル $w_l$ に対して，方程式

$$A\Psi_l A^T s_l=Aw_l$$

の解を $s_l \in R^m$ とすると，$A^T s_l=\hat{A}_l w_l$ である．したがって，$\Delta v$ に関する方程式も品種ごとに並列的に求めた情報を集約して解けることになり，全体としてコントロール並列型のアルゴリズムを構成できることがわかる．

並列計算機上で実行する際には，分解された部分問題を処理するプロセッサ群と，アルゴリズム全体を制御するもう1つのプロセッサから成る計算モデルを用いる．部下に仕事を分配する管理職の姿にたとえて，前者をワーカー・ノード，後者をマスター・ノードと呼ぶ．ここでは，各品種にワーカー・ノードを1つずつ割当て，品種ごとに独立な方程式を共役勾配法を用いて並列的に解く．マスター・ノードは，$\Delta v$ に関する方程式の処理と内点法の制御を行なう．**表4**に数値実験結果を示す．品種数が増えても計算時間の伸びは比較的鈍く，並列化の効果が認められる．また，枝の総流量上限制約の存在を考慮すると，32個のプロセッサを用いた表3の方法と比べても，十分に評価できる結果である．

一方，ブロック対角行列による行列分割を用いると，一般の狭義凸2次計画問題に対してもブロック並列型

**表3**　輸送問題に対する並列アルゴリズムの実験結果

| 2部グラフのサイズ | | | 計算時間 |
|---|---|---|---|
| $\|V_1\|$ | $\|V_2\|$ | $\|E\|$ | (秒) |
| 4096 | 4096 | 32768 | 5.68 |
| 16384 | 16384 | 131072 | 16.58 |
| 65536 | 65536 | 524288 | 173.82 |
| 4096 | 4096 | 65536 | 2.99 |
| 16384 | 16384 | 262144 | 14.30 |
| 65536 | 65536 | 1048576 | 83.17 |

**表4**　多品種輸送問題に対する主双対内点法の実験結果

| 2部グラフのサイズ $\|\mathcal{V}_1\|$ | $\|\mathcal{V}_2\|$ | $\|\mathcal{E}\|$ | 品種数 $L$ | 計算時間（秒） |
|---|---|---|---|---|
| 1024 | 1024 | 16384 | 8 | 1614.08 |
| | | | 16 | 2286.73 |
| | | | 24 | 2885.42 |
| 4096 | 4096 | 32768 | 8 | 8197.79 |
| | | | 16 | 11552.58 |
| | | | 24 | 14996.18 |

のアルゴリズムを構成することができる［9］．問題

目的関数：$\frac{1}{2}x^T Dx+c^T x \rightarrow$ 最小

制約条件：$Ax \leq b$

に対して $G=AD^{-1}A^T$，$h=AD^{-1}c+b$ とおくと，その双対問題は次のように書ける．

目的関数：$\frac{1}{2}z^T Gz+h^T z \rightarrow$ 最小

制約条件：$z \geq 0$．

$G$ をブロック対角行列 $M=\mathrm{diag}\{M_q\}_{q=1,\cdots,Q}$ により分割する．$M$ の分割に対応して $z=(z_1^T,\cdots,z_Q^T)^T$ と書くと，行列分割法の部分問題は，$Q$ 個の小さな問題

目的関数：$\frac{1}{2}z_q^T M_q z_q+(v_q^k)^T z_q \rightarrow$ 最小

制約条件：$z_q \geq 0$

に分解される．そこで，$Q$ 個のワーカー・ノードにおいて，これらの問題を前処理つき共役勾配法を用いて並列的に解く．マスター・ノードでは，係数ベクトル

$v^k=(G-M)z^k+h$

の計算とアルゴリズムの収束判定を行なう．行列 $A$ が

$$A=\begin{pmatrix} A_1 & & \\ & \ddots & \\ & & A_L \\ \bar{A}_1 & \cdots & \bar{A}_L \end{pmatrix}$$

という特殊なブロック構造をもつ場合には，行列 $G$ もブロック構造を示し，その $L+1$ 行および $L+1$ 列ブロック以外のブロック非対角部分は 0 となる．そこで，$M_q$ として行列 $G$ の対応するブロック対角部分を選んだ場合の数値実験結果を**表5**に示す．問題 QP 11, QP 12 の $L$ の値はそれぞれ 7, 15，$\bar{A}=(\bar{A}_1\cdots\bar{A}_L)$ の非零要素数はそれぞれ 8192, 16384 である．一方，$A_l$ のサイズはいずれも 256×1024，その非零要素数はいずれも 8192 である．プロセッサ数 $p=1$ の欄は，双対問題に対して直接共役勾配法を適用した結果である．他の欄の $p$ の値は $Q+1$ である．表より，アルゴリズムの並列化効率は，ブロックの数が多い場合に特に高いことがわかる．

**表5**　ブロック並列アルゴリズムの計算時間（単位：秒）

| 問題名 | プロセッサ数 $p=1$ | $p=3$ | $p=5$ | $p=9$ |
|---|---|---|---|---|
| QP11 | 229.83 | 95.88 | 51.92 | 33.49 |
| QP12 | 747.25 | 180.33 | 102.02 | 63.90 |

**表6**　非同期並列アルゴリズムの計算時間（単位：秒）

| 問題名 | 非同期性のパラメータ $N=1$ | $N=2$ | $N=4$ | $N=8$ |
|---|---|---|---|---|
| QP21 | 160.50 | 167.68 | 172.31 | 183.00 |
| QP22 | 131.99 | 132.51 | 134.15 | 135.55 |

ブロック並列型のアルゴリズムは，分解された部分問題を各ワーカー・ノードが解き終わるのを待って次の反復に入る同期型の並列アルゴリズムである．したがって，もとの問題のブロック構造が一様でない場合には，計算負荷の軽いワーカー・ノードが長い間遊休状態となり，並列化効率の低下をまねく可能性がある．そこで，ある整数 $N \in [1, Q]$ に対して，新たに $N$ 個の部分問題の解をマスター・ノードが得た段階で $v$ の値を最新の $z_q$ にもとづいて評価し直し，遊休状態となったワーカー・ノードに新しい部分問題を与えて反復を進める非同期型の並列アルゴリズムが考えられる［10］．$N=Q$ のとき，アルゴリズムは同期型の並列アルゴリズムと一致し，$N$ の値が小さくなるほど非同期性が強くなる．さきほどと同じ構造をもつ行列 $A$ に対して，$Q=L+1$ とした場合の数値実験結果を**表6**に示す．問題 QP 21, QP 22 において，行列 $A_l$ のサイズはそれぞれ 256×512, 128×512，すべての変数にまたがる行列 $\bar{A}$ の非零要素数はそれぞれ 8192, 16384 である．一方，$A_l$ の非零要素数はいずれも 8192，また，$L$ の値はいずれも 7 である．

アルゴリズムの非同期化が比較的良い結果をもたらしたこの2題について，各プロセッサの稼働状況を調べてみよう．$Q=L+1$ より，$1 \sim L$ 番目のワーカー・ノードは，$A$ の互いに独立なブロック $A_l$ に対する処理を行なう．一方，$L+1$ 番目のワーカー・ノードは，全変数にまたがる $\bar{A}$ を受けもつ．問題 QP 21 の場合は，$N$ を 8 から 1 へ変化させると，ワーカー・ノード $L+1$ の稼働率が 54.1 %から 97.1 %へ急上昇している．すなわち，非同期性の導入によって，問題全体に対して影響力をもつ $\bar{A}$ を担当するプロセッサの稼働

率がよくなって，大きな効果が得られたものと考えられる．一方，問題QP 22の場合は，$N$の減少とともにワーカー・ノード1～$L$の稼働率が46.5％から82.5％へ上昇し，処理時間の短縮に貢献している．しかし，ワーカー・ノード$L+1$の稼働率は97.5％以上のままでほとんど変化しなかったため，問題QP 21の場合ほど大きな効果は得られなかったものと思われる．

## 5．おわりに

本稿では，データ並列型とコントロール並列型という典型的な2種類の並列アルゴリズムについて述べ，大規模で疎な2次計画問題に対する性能を簡単に紹介した．また，計算機を構成する各プロセッサにかかる負荷の問題を中心に，並列アルゴリズムを実際に並列計算機上で効率よく実行するための方策についても考えた．

CM-5をはじめとする多くの並列計算機は，航空工学などの分野でしばしば用いられる空間のメッシュ分割モデルを適用対象と想定して開発されたようである．そこで現われる行列は，非零要素の位置が対称であり，各行の非零要素数もほぼひとしい場合が多い．実際，CMSSLで提供される線形代数的な演算のルーチンは，このような特徴をもつ行列に対して最も有効に動作するといわれている．したがって，構造的に一様ではない横長行列を取り扱う場合が多い数理計画法にとって，現在利用可能な並列計算機は必ずしも都合がよいものであるとは限らない．

文献[7]には，大規模科学技術計算を効率よく並列化できるような新しい並列計算モデルと，それを実行する並列計算機の開発構想が示されている．最適化のアルゴリズムの分野においても，実用的な並列計算環境を獲得するためには，アルゴリズムの開発に携わる多くの人が実際の並列計算に触れ，そこで要求される計算の枠組みを明らかにしていく必要がある．並列最適化の世界には難しい問題がたくさん存在するが，まだまだ多くの可能性も残されている．本稿が，並列計算に興味をおもちの方々にとって，少しでも参考になれば幸いである．

### 参考文献


[1] Eckstein, J.: Parallel Branch-and-Bound Algorithms for General Mixed Integer Programming on the CM-5. *SIAM Journal on Optimization,* Vol. 4 (1994), 794-814.

[2] Eckstein, J. and Fukushima, M.: Some Reformulations and Applications of the Alternating Direction Method of Multipliers. *Large Scale Optimization : State of the Art* (eds. W. W. Hager, D. W. Hearn and P.M. Pardalos). Kluwer Academic Publishers, Dordrecht, 1994, 115-134.

[3] Ferris, M.C. and Mangasarian, O.L.: Parallel Variable Distribution. *SIAM Journal on Optimization,* Vol. 4 (1994), 815-832.

[4] 福島雅夫： 非線形最適化における並列アルゴリズム．システム／制御／情報, Vol. 34 (1990), 223-231.

[5] 福島雅夫： 数理計画問題に対する並列アルゴリズム．第5回RAMPシンポジウム論文集, 1993, 15-28.

[6] Fukushima, M., Haddou, M., Nguyen, V.H., Strodiot, J.-J., Sugimoto, T. and Yamakawa, E.: A Parallel Descent Algorithm for Convex Programming. *Comp. Opt. Appl.,* to appear.

[7] 野木達夫：並列計算モデルPB対ADEPS．応用数理, Vol. 3 (1993), 207-224.

[8] Thinking Machines Corporation：*Connection Machine CM-5 Technical Summary.* Cambridge, Massachusetts, 1993.

[9] Yamakawa, E. and Fukushima, M.：A Block Parallel Conjugate Gradient Method for Separable Quadratic Programming Problems. Information Science Technical Report 94033, Nara Institute of Science and Technology, 1994.

[10] 山川栄樹，牧英一：ブロック構造を持つ2次計画問題に対する非同期並列型共役勾配法．テクニカルレポートTR-H-135，エイ・ティ・アール人間情報通信研究所, 1995.

[11] 山川栄樹，松原康博： 2次計画問題に対する主双対内点法とその数値実験．テクニカルレポートTR-H-105，エイ・ティ・アール人間情報通信研究所, 1994.

[12] 山川栄樹，松原康博，福島雅夫： 2次コスト多品種流問題に対する並列型主双対内点法，投稿中.

[13] Zenios, S.A.：Data Parallel Computing for Network-Structured Optimization Problems. *Comp. Opt. Appl.,* Vol. 3 (1994), 199-242.