数理解析研究所講究録
1068巻 1998年 26-38

交互方向乗数法のベクトル並列計算機VPP500における実行

高松大・経営　山川栄樹　(Eiki Yamakawa)

1. はじめに

数理計画問題に対する並列アルゴリズムには，行列や関数を分割し，もとの問題を二つの取扱いやすい問題に分解して得られるものが少なくない．交互方向乗数法もそのようなアルゴリズムの一つであり，ネットワーク構造をもつ問題などに適用すると，高度な並列アルゴリズムを得ることができる．

まず，一般的な凸計画問題

$$\text{minimize } F(x)+G(y) \qquad \text{subject to } Mx-y=0 \qquad (1)$$

と，その Fenchel の双対問題

$$\text{minimize } G^*(v)+F^*(w) \quad \text{subject to } -M^{\top}v-w=0 \quad (2)$$

について考える．ただし，M は $m\times n$ 行列，F,G は $\Re^n,\Re^m$ 上の閉真凸関数，F^*,G^* はそれらの共役関数である．問題 (1) に対する交互方向乗数法の反復は，つぎのように記述される．

$$x^{(k+1)} := \arg\min_x\{F(x)+(\lambda^{(k)})^{\top}Mx+\tfrac{t}{2}\|Mx-y^{(k)}\|^2\},$$
$$y^{(k+1)} := \arg\min_y\{G(y)-(\lambda^{(k)})^{\top}y+\tfrac{t}{2}\|Mx^{(k+1)}-y\|^2\},$$
$$\lambda^{(k+1)} := \lambda^{(k)}+t(Mx^{(k+1)}-y^{(k+1)}).$$

明らかに，F, G が分離可能で $M^\top M$ が(ブロック)対角行列ならば，各部分問題を並列的に解くことができる．以下では，このアルゴリズムを主交互方向乗数法と呼ぶ．一方，問題 (2) に対する交互方向乗数法の反復は，つぎのように記述される．

$$v^{(k+1)} := \arg\min_v \{G^*(v) - (\mu^{(k)})^\top M^\top v + \tfrac{s}{2}\|M^\top v + w^{(k)}\|^2\},$$
$$w^{(k+1)} := \arg\min_w \{F^*(w) - (\mu^{(k)})^\top w + \tfrac{s}{2}\|M^\top v^{(k+1)} + w\|^2\},$$
$$\mu^{(k+1)} := \mu^{(k)} - s(M^\top v^{(k+1)} + w^{(k+1)}).$$

今度は，G^*, F^* が分離可能で $MM^\top$ が(ブロック)対角行列ならば，各部分問題を並列的に解くことができる．なお，各部分問題に対する Fenchel の双対問題を考えると，この反復は

$$z^{(k+1)} := \arg\min_z \{G(Mz) + (w^{(k)})^\top z + \tfrac{t}{2}\|z - x^{(k)}\|^2\},$$
$$x^{(k+1)} := \arg\min_x \{F(x) - (w^{(k)})^\top x + \tfrac{t}{2}\|z^{(k+1)} - x\|^2\},$$
$$w^{(k+1)} := w^{(k)} + t(z^{(k+1)} - x^{(k+1)}),$$

と等価である．ただし，$t = 1/s$ である．そこで以下では，このアルゴリズムを双対交互方向乗数法と呼ぶことにする．

Eckstein [2] は並列計算機 CM-2 を用いて主交互方向乗数法の計算実験を行い， Eckstein and Fukushima [3] は並列計算機 CM-5 の上で双対交互方向乗数法の計算実験を行っている．これらの文献では，主交互方向乗数法は反復が進んでも制約条件の相対誤差があまり縮小しないのに対して，やや粒度の大きい双対交互方向乗数法は効率的に並列実行できると主張してい

る．しかし，使用した計算機や収束判定の方法に違いがあり，並列化効率も定量的に示されていない．また，これらの計算実験では，CM-2 や CM-5 に特有な Segmented Scan Operation [1] を用いて線形代数演算を行っているなどの問題がある．

そこで本報告では，二つのアルゴリズムを 2 次輸送問題に適用し，現在最もよく利用されているベクトル並列計算機の上で実行する方法を提案して，性能の比較を行うことにする．

2. 主交互方向乗数法

ここでは，2 部グラフ: $\mathcal{G} = (\mathcal{N}_1, \mathcal{N}_2, \mathcal{A})$ 上の 2 次輸送問題

$$\begin{array}{lll} \text{minimize} & \Sigma_{(i,j)\in\mathcal{A}} \left\{ \frac{d_{ij}}{2}(x_{ij})^2 + c_{ij}x_{ij} \right\} & \\ \text{subject to} & \Sigma_{j:(i,j)\in\mathcal{A}}\, x_{ij} = \alpha_i\,, & i \in \mathcal{N}_1\,, \\ & \Sigma_{i:(i,j)\in\mathcal{A}}\, x_{ij} = \beta_j\,, & j \in \mathcal{N}_2\,, \\ & x_{ij} \geq 0, & (i,j) \in \mathcal{A}, \end{array} \tag{3}$$

について考える．$\mathcal{G}$ の接続行列の行を $m_1^\top, \cdots, m_{|\mathcal{N}_1|+|\mathcal{N}_2|}^\top \in \Re^{|\mathcal{A}|}$ として $M = [\mathrm{diag}\{m_1\}, \cdots, \mathrm{diag}\{m_{|\mathcal{N}_1|+|\mathcal{N}_2|}\}]^\top$ とおくならば，$M^\top M$ は対角行列になる．このとき，F と G を枝および節点について分離可能な関数に選んで，問題 (3) を問題 (1) に帰着させることができる．なお，主交互方向乗数法を実行する際には，$y^{(k+1)}$ と $\lambda^{(k+1)}$ の計算式に現れる $Mx^{(k+1)}$ を，緩和パラメータ $\rho \in (0,2)$ を用いて $(1-\rho)y^{(k)} + \rho M x^{(k+1)}$ と修正し，

収束の加速を図ることが多い．結局，問題 (3) に対する主交互方向乗数法の反復は，つぎのように書き下すことができる [2]．

$$
\begin{aligned}
x_{ij}^{(k+1)} &:= \max\Big\{0,\, t\Big(2z_{ij}^{(k)} + \frac{p_i^{(k)}}{\zeta_i} + \frac{q_j^{(k)}}{\xi_j}\Big) + \nu_i^{(k)} + \upsilon_j^{(k)} - c_{ij}\Big\} \\
&\qquad\qquad /(d_{ij} + 2t), \qquad (i,j) \in \mathcal{A}, \\
r_i^{(k+1)} &:= \alpha_i - \Sigma_{j:(i,j)\in\mathcal{A}}\, x_{ij}^{(k+1)}, \\
\nu_i^{(k+1)} &:= \nu_i^{(k)} + \rho\, t\, r_i^{(k+1)}/\zeta_i, \qquad i \in \mathcal{N}_1, \\
p_i^{(k+1)} &:= (1-\rho)p_i^{(k)} + \rho\, r_i^{(k+1)}, \\
s_j^{(k+1)} &:= \beta_j - \Sigma_{i:(i,j)\in\mathcal{A}}\, x_{ij}^{(k+1)}, \\
\upsilon_j^{(k+1)} &:= \upsilon_j^{(k)} + \rho\, t\, s_j^{(k+1)}/\xi_j, \qquad j \in \mathcal{N}_2, \\
q_j^{(k+1)} &:= (1-\rho)q_j^{(k)} + \rho\, s_j^{(k+1)}, \\
z_{ij}^{(k+1)} &:= (1-\rho)z_{ij}^{(k)} + \rho\, x_{ij}^{(k+1)}, \qquad (i,j) \in \mathcal{A}.
\end{aligned}
$$

ただし，ζ_i, ξ_j は節点 $i \in \mathcal{N}_1$, $j \in \mathcal{N}_2$ の次数である．

現実の大規模な問題では，節点の数は膨大であるが各節点の次数は非常に少ないと考えられる．そこで，データの分割，ベクトル化，並列化はいずれも節点に関して行うことにした．すなわち，供給節点 $\mathcal{N}_1$ および需要節点 $\mathcal{N}_2$ に関する情報は，それぞれ使用可能なプロセッサに重複なく分配する．一方，枝に関する情報は，対応する供給節点と需要節点を担当するプロセッサに重複して持たせる．そして，各プロセッサは，担当する節点に対する演算をベクトル処理により実行する．ただし，その節点に接続する複数の枝に関する情報を集約しなければなら

ない場合には，逐次処理により実行する．なお，$x_{ij}^{(k+1)}$ の値をその供給節点 $i \in \mathcal{N}_1$ に対応するプロセッサで計算する際には，需要節点ごとに各プロセッサが分割して保持する $q_j^{(k)}/\xi_j + v_j^{(k)}$ の値を，また，$x_{ij}^{(k+1)}$ の値をその需要節点 $j \in \mathcal{N}_2$ に対応するプロセッサで計算する際には，供給節点ごとに各プロセッサが分割して保持する $p_i^{(k)}/\zeta_i + \nu_i^{(k)}$ の値を，他のすべてのプロセッサにあらかじめ伝達しておくことにする．

3. 双対交互方向乗数法

2 次輸送問題 (3) に対して，双対交互方向乗数法を適用することを考える．行列 $MM^\top$ を対角行列にするために，節点 $\mathcal{N}_2$ と枝 $\mathcal{A}$ の接続行列を M とする．このとき，F と G を節点 $\mathcal{N}_1$ および $\mathcal{N}_2$ について分離可能な関数に選んで，問題 (3) を問題 (1) に帰着させることができる．なお，双対交互方向乗数法を実行する場合も，収束を加速する目的で，$x^{(k+1)}$ と $w^{(k+1)}$ の計算式に現れる $z^{(k+1)}$ を $(1-\rho)x^{(k)} + \rho\, z^{(k+1)}$ と修正することが多い．結局，問題 (3) に対する双対交互方向乗数法の反復は，つぎのように書き下すことができる [3]．

$$
\begin{aligned}
v_j^{(k+1)} &:= [t\, s_j^{(k)} + \Sigma_{i:(i,j)\in\mathcal{A}}\, w_{ij}^{(k)}]/\xi_j\,, && j \in \mathcal{N}_2\,,\\
\overline{w}_{ij}^{(k+1)} &:= (1-\rho) w_{ij}^{(k)} + \rho\, v_j^{(k+1)}, && (i,j) \in \mathcal{A},\\
\overline{c}_{ij}^{(k+1)} &:= c_{ij} - \overline{w}_{ij}^{(k+1)} - t x_{ij}^{(k)}, && (i,j) \in \mathcal{A},
\end{aligned}
$$

$$x_{i\cdot}^{(k+1)} := \arg\min\Big\{\Sigma_{j:(i,j)\in\mathcal{A}}\big[\tfrac{\bar{d}_{ij}}{2}(x_{ij})^2 + \bar{c}_{ij}^{(k+1)}x_{ij}\big] \;\Big|\; \Sigma_{j:(i,j)\in\mathcal{A}}\, x_{ij} = \alpha_i,\ x_{i\cdot} \geq 0\Big\}, \qquad i \in \mathcal{N}_1,$$

$$s_j^{(k+1)} := \beta_j - \Sigma_{i:(i,j)\in\mathcal{A}}\, x_{ij}^{(k+1)}, \qquad j \in \mathcal{N}_2,$$

$$w_{ij}^{(k+1)} := \overline{w}_{ij}^{(k+1)} - t(x_{ij}^{(k+1)} - x_{ij}^{(k)}), \qquad (i,j) \in \mathcal{A}.$$

ただし, $\bar{d}_{ij} = d_{ij} + t$ である． $x_{ij}^{(k+1)}$ を求める部分問題は連続型の資源配分問題で，組合せ的な解法が知られている [7]．すなわち，適当に添字をつけかえて $\bar{c}_{i1}^{(k+1)} \leq \cdots \leq \bar{c}_{i\zeta_i}^{(k+1)}$ として，

$$\nu_i^{(k+1)} := [\,\alpha_i + \Sigma_{h=1}^{\hat{h}_i}(\bar{c}_{ih}^{(k+1)}/\bar{d}_{ih})\,]\,/\,[\,\Sigma_{h=1}^{\hat{h}_i}(1/\bar{d}_{ih})\,] > \bar{c}_{i\hat{h}_i}^{(k+1)}$$

となる最大の $\hat{h}_i$ を見つけて，つぎのようにおけばよい．

$$x_{ih}^{(k+1)} := \max\{0,\, (\nu_i^{(k+1)} - \bar{c}_{ih}^{(k+1)})/\bar{d}_{ih}\}, \quad h = 1, \ldots, \zeta_i\,.$$

双対交互方向乗数法においても，データの分割，ベクトル化，並列化は節点に関して行う．したがって，節点および枝に関する情報のプロセッサへの分割方法や各プロセッサにおける処理の形態は，主交互方向乗数法の場合と同じである．なお，$\overline{w}_{ij}^{(k+1)}$ の値をその供給節点 $i \in \mathcal{N}_1$ に対応するプロセッサで計算する際には，需要節点ごとに各プロセッサが分割して保持する $v_j^{(k+1)}$ の値を他のすべてのプロセッサに伝達する．一方，$x_{ih}^{(k+1)}$ の値をその需要節点 $h \in \mathcal{N}_2$ に対応するプロセッサで計算する際には，供給節点ごとに各プロセッサが分割して保持する $\nu_i^{(k+1)}$ の値を，他のすべてのプロセッサに伝達しておく．

4. 計算実験

計算実験は，京都大学大型計算機センターのベクトル並列計算機 VPP500 上で行った．VPP500 の各プロセッサには，200 MFLOPS のスカラーユニット，1.6 GFLOPS のベクトル演算装置，および 256 Mbytes のローカルメモリが装備されている [6]．プロセッサ間のデータ転送と同期制御はクロスバーネットワークにより実現され，その転送速度は 400 Mbps である．

アルゴリズムのコーディングには，プログラミング言語 VPP FORTRAN 77 を用いた．VPP FORTRAN 77 では，総和演算，条件文，あるいは飛出しがある DO ループも，一定の条件のもとで自動的にベクトル化される．また，多重 DO ループでは，原則として最も内側のループが自動的にベクトル化されるが，ベクトル化対象ループの実行文を二重に展開し，すぐ外側の DO ループの回転数を半分にするベクトルアンローリングとよばれる最適化が自動的に行われる [4]．

VPP500 は SPMD (Single Program Multiple Data) 型の並列計算機で，プログラム中にコンパイラ指示行を挿入して並列処理の制御を行う [5]．まず，プログラムの先頭に PROCESSOR 文を置いて使用するプロセッサの数を宣言し，LOCAL 文などにより大規模配列の要素を各プロセッサへ分割する．一方，PARALLEL REGION 文と END PARALLEL 文で囲まれたプログラムは各プロ

セッサで並列処理されるが，DO ループに SPREAD DO 文を付加すると，繰返し処理を各プロセッサに分割実行させることができる．なお，配列要素に対する演算を分割実行した場合には，UNIFY 文を用いて実行結果を他のプロセッサに転送したり，グローバル関数によりデータを集約する必要が生じる．

計算実験は，ランダムに生成した 2 次輸送問題 (3) に対して行った．生成した 2 部グラフにおいて，供給節点 $\mathcal{N}_1$ の数と需要節点 $\mathcal{N}_2$ の数は同じで，枝 $\mathcal{A}$ の総数は供給節点数の 8 倍または 16 倍である．各節点は少なくとも 2 本の規則的な枝をもつが，他の枝はすべてランダムに生成されたものである．一方，目的関数の係数 c_{ij} と d_{ij} は，それぞれ区間 $[0, 100]$, $[0.1, 1.0]$ から一様に選んだ．さらに，変数 x_{ij} の値を区間 $[0, 100]$ からランダムに選び，これらの値に対してすべての制約条件がなりたつように，右辺定数 α_i および β_j の値を決定した．

実験においては，枝 $\mathcal{A}$ の総数を 16384 から 1048576 まで変化させた．そして，それぞれのサイズについて乱数の種類を変えて 5 題の問題例を生成し，これらの平均をもって実験結果とした．2 次輸送問題 (3) の変数の数は，枝の総数と一致する．一方，等式制約条件の数は節点の総数と等しいが，節点数と枝数の関係から，変数の数の 1/4 または 1/8 となる．

主交互方向乗数法の収束判定条件は，

$$\max_{i\in\mathcal{N}_1}|r_i^{(k+1)}| \le 10^{-6}\max_{i\in\mathcal{N}_1}\alpha_i,$$
$$\max_{j\in\mathcal{N}_2}|s_j^{(k+1)}| \le 10^{-6}\max_{j\in\mathcal{N}_2}\beta_j,$$
$$\max_{(i,j)\in\mathcal{A}}|x_{ij}^{(k+1)} - z_{ij}^{(k)}| \le 10^{-6},$$

とした．一方，双対交互方向乗数法では，

$$\max_{j\in\mathcal{N}_2}|s_j^{(k+1)}| \le 10^{-6}\max_{j\in\mathcal{N}_2}\beta_j,$$
$$\max_{(i,j)\in\mathcal{A}}|x_{ij}^{(k+1)} - x_{ij}^{(k)}| \le 10^{-6},$$

が成り立った場合に反復を終了させた．なお，収束判定による計算負荷を削減するため，これらの条件の一つが満たされた場合にのみ他の条件の判定に進むことにしている．

まず，緩和パラメータを $\rho = 1.6$, ペナルティパラメータを

$$t = \bar{t} \equiv [\,(|\mathcal{N}_1| + |\mathcal{N}_2|)\max_{(i,j)\in\mathcal{A}}\{c_{ij}\}\,]\,/\,(50|\mathcal{A}|)$$

に選んで実験を行った結果を表 1,2 に示す．なお，係数 c_{ij} の選び方により，節点の平均次数が 8 の問題で $\bar{t} \cong 0.5$，次数が 16 の問題で $\bar{t} \cong 0.25$ 程度である．なお，各表において，p は使用したプロセッサの数である．また，結果が "-" の欄は，メモリ不足で実行できなかったことを示す．表 1,2 より，主交互方向乗数法と双対交互方向乗数法のいずれも，ベクトル化により性能が大幅に向上するとともに，並列化によって処理効率がさらに改善していることがわかる．しかし，双対交互方向乗数法は，主交互方向乗数法に比べて反復回数は少ないものの多くの計算時間を要している．これは，各反復で連続型の資源配分

表1: 主交互方向乗数法の実験結果 ($\rho = 1.6, t = \bar{t}$)

問題サイズ			反復回数	計算時間 [秒]				
				逐次処理	ベクトル処理	ベクトル並列処理		
$\|\mathcal{N}_1\|$	$\|\mathcal{N}_2\|$	$\|\mathcal{A}\|$				$p=2$	$p=4$	$p=8$
2048	2048	16384	86	6.861	0.203	0.130	0.083	0.062
4096	4096	32768	91	26.09	0.436	0.258	0.153	0.103
8192	8192	65536	113	88.32	1.105	0.638	0.353	0.221
16384	16384	131072	137	218.0	2.721	1.548	0.843	0.492
32768	32768	262144	211	792.2	8.616	4.820	2.648	1.511
65536	65536	524288	161	1246.	13.15	7.343	3.948	2.225
131072	131072	1048576	947	-	-	91.83	48.59	27.01
1024	1024	16384	63	3.847	0.136	0.090	0.059	0.046
2048	2048	32768	68	9.350	0.285	0.173	0.103	0.072
4096	4096	65536	69	37.40	0.601	0.341	0.192	0.120
8192	8192	131072	81	113.8	1.427	0.795	0.424	0.249
16384	16384	262144	159	455.2	5.655	3.119	1.644	0.905
32768	32768	524288	363	2425.	25.70	14.19	7.429	4.071
65536	65536	1048576	339	-	-	27.68	14.39	7.766

表2: 双対交互方向乗数法の実験結果 ($\rho = 1.6, t = \bar{t}$)

問題サイズ			反復回数	計算時間 [秒]				
				逐次処理	ベクトル処理	ベクトル並列処理		
$\|\mathcal{N}_1\|$	$\|\mathcal{N}_2\|$	$\|\mathcal{A}\|$				$p=2$	$p=4$	$p=8$
2048	2048	16384	53	18.68	0.480	0.267	0.153	0.098
4096	4096	32768	59	59.77	1.196	0.622	0.336	0.196
8192	8192	65536	66	191.8	2.767	1.502	0.734	0.403
16384	16384	131072	67	449.1	5.738	3.113	1.582	0.787
32768	32768	262144	74	1390.	13.91	7.542	3.831	1.964
65536	65536	524288	110	4182.	41.57	22.59	11.45	5.873
131072	131072	1048576	532	-	-	148.5	75.71	39.34
1024	1024	16384	52	19.89	0.563	0.319	0.190	0.128
2048	2048	32768	55	54.53	1.271	0.666	0.370	0.224
4096	4096	65536	56	148.5	2.751	1.363	0.721	0.405
8192	8192	131072	61	459.1	6.476	3.408	1.611	0.857
16384	16384	262144	71	1227.	15.93	8.442	4.214	2.003
32768	32768	524288	85	3810.	37.37	19.88	9.976	5.001
65536	65536	1048576	161	-	-	68.88	34.59	17.46

問題を解く際に $\bar{c}_{ij}^{(k+1)}$ を並べ換える必要があるためと考えられる．なお，変数の非負制約に対する相補条件の絶対誤差は，主交互方向乗数法が $10^{-4} \sim 10^{-3}$ 程度であったのに対して，双対交互方向乗数法では $10^{-5} \sim 10^{-4}$ 前後であった．

つぎに，プロセッサ数を $p = 8$ に固定して，パラメータ ρ と t の値を変化させた場合の実験結果を表3,4に示す．表より，いずれの方法でも ρ の値は $1.6 \sim 1.8$ 程度が望ましいことがわかる．一方，パラメータ t の値も $\bar{t} \sim 2\bar{t}$ 程度が適切で，値が過大でも過小でも収束が遅くなることが確かめられる．

5. おわりに

一般的な凸計画問題とその双対問題に対する交互方向乗数法を2次輸送問題に適用し，ベクトル並列計算機 VPP500 で実行する方法を示した．そして，計算実験により，大規模な問題を効率的に処理できることを確認した．2次輸送問題以外の凸計画問題において，主交互方向乗数法と双対交互方向乗数法のどちらが適するかを見極めることは，今後の課題である．

謝辞

日頃よりご指導いただいている京都大学大学院情報学研究科の福島雅夫教授に感謝致します．なお，本研究の一部は，京

表3: 主交互方向乗数法の実験結果 (ρ, t の値を変化させた場合)

問題サイズ			計算時間 [秒]					
			$t = \bar{t}$			$\rho = 1.6$		
$\|\mathcal{N}_1\|$	$\|\mathcal{N}_2\|$	$\|\mathcal{A}\|$	$\rho = 1.0$	$\rho = 1.4$	$\rho = 1.8$	$t = \bar{t}/2$	$t = \bar{t}$	$t = 2\bar{t}$
2048	2048	16384	0.099	0.071	0.060	0.104	0.062	0.119
4096	4096	32768	0.161	0.116	0.099	0.168	0.103	0.190
8192	8192	65536	0.340	0.244	0.207	0.388	0.221	0.329
16384	16384	131072	0.773	0.555	0.449	0.966	0.492	0.621
32768	32768	262144	2.379	1.694	1.343	2.899	1.511	1.388
65536	65536	524288	3.599	2.553	1.991	4.628	2.225	2.463
131072	131072	1048576	43.10	30.87	24.01	50.18	27.01	14.12
1024	1024	16384	0.076	0.053	0.058	0.097	0.046	0.075
2048	2048	32768	0.115	0.081	0.085	0.151	0.072	0.104
4096	4096	65536	0.201	0.139	0.140	0.264	0.120	0.173
8192	8192	131072	0.405	0.285	0.250	0.496	0.249	0.311
16384	16384	262144	1.468	1.038	0.799	1.794	0.905	0.584
32768	32768	524288	6.536	4.653	3.613	7.777	4.071	2.124
65536	65536	1048576	12.43	8.891	6.895	14.79	7.766	3.877

表4: 双対交互方向乗数法の実験結果 (ρ, t の値を変化させた場合)

問題サイズ			計算時間 [秒]					
			$t = \bar{t}$			$\rho = 1.6$		
$\|\mathcal{N}_1\|$	$\|\mathcal{N}_2\|$	$\|\mathcal{A}\|$	$\rho = 1.0$	$\rho = 1.4$	$\rho = 1.8$	$t = \bar{t}/2$	$t = \bar{t}$	$t = 2\bar{t}$
2048	2048	16384	0.150	0.109	0.099	0.188	0.098	0.168
4096	4096	32768	0.269	0.210	0.190	0.345	0.196	0.308
8192	8192	65536	0.592	0.445	0.385	0.752	0.403	0.584
16384	16384	131072	1.231	0.900	0.717	1.655	0.787	1.135
32768	32768	262144	2.927	2.174	1.822	3.846	1.964	2.682
65536	65536	524288	9.173	6.702	5.216	11.38	5.878	5.456
131072	131072	1048576	59.12	44.35	35.36	73.61	39.34	20.44
1024	1024	16384	0.209	0.148	0.147	0.266	0.128	0.144
2048	2048	32768	0.357	0.258	0.245	0.453	0.224	0.226
4096	4096	65536	0.662	0.473	0.434	0.846	0.405	0.422
8192	8192	131072	1.362	0.980	0.844	1.708	0.857	0.854
16384	16384	262144	3.152	2.283	1.896	4.018	2.003	1.678
32768	32768	524288	7.783	5.701	4.475	9.675	5.001	3.963
65536	65536	1048576	26.79	19.79	15.66	33.14	17.46	8.518

都大学大型計算機センター開発計画によるものである.

参考文献

[1] Y. Censor and S.A. Zenios: *Parallel Optimization; Theory, Algorithm and Applications*, Oxford University Press, New York, 1997.

[2] J. Eckstein: "The Alternating Step Method for Monotropic Programming on the Connection Machine CM-2", *ORSA Journal on Computing*, Vol. 5 (1993), pp. 84–96.

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

[4] 富士通株式会社: UXP/V VP プログラミングハンドブック, V10 用, 1997.

[5] 富士通株式会社: UXP/V VPP プログラミングハンドブック, V10 用, 1997.

[6] 平野彰雄: "MSP ユーザのための VPP 入門," 京都大学大型計算機センター広報, Vol. 28 (1995), pp. 63-75.

[7] 一森哲男: 数理計画法 –最適化の手法–, 共立出版, 1994.