ハイパフォーマンス
コンピューティング 88－4
（２００１．１０．２６）

データ分散の図式表現による多重ループの計算分散及び評価

佐藤　真琴
日立製作所システム開発研究所

要旨

従来，HPFが規定する一般の規則的データ分散をあるh次元配列に適用し，この配列の要素で，その添字がループ制御変数の線型式となるものに従って一般のd重ループを分散する場合，非効率なコードしか生成できなかった．本論文では，データ分散の図式表現の上で添字写像等の逆写像を表現することにより上記に対する計算分散アルゴリズムを与えた．また生成コードにおける添字参照の擬周期を示し，これからInspector-executor法によるテーブル検索法コードを与えた．このコードは日立SR2201上で実行時解決法の0.4倍から7.8倍のスピードで，提案アルゴリズムを直接使った方法は実行時解決法よりも1.1倍から18倍高速であることがわかった．

Computation Partitioning of Multiple Loops using Diagram Representation of Data Distribution

Makoto Satoh
Systems Development Laboratory, Hitachi, Ltd.

Abstract

For any h-dimensional array of a general regular distribution of HPF and any d-tuple nested loop including an ON HOME directive with a general affine subscript of the array element in its HOME clause, the conventional computation-partitioning methods can not generate effective code. In this paper, we represent any inverse mapping of subscript mapping on a diagram representation of regular data distribution of HPF and provide a computation-partitioning algorithm for the above loop. Moreover, we show the pseudo-period of array references in the generated code and provide a table-lookup code using the Inspector-executor technique. This optimized code outperformed the run-time resolution code by 0.4 to 7.8 times and the code directly obtained from our algorithm outperformed by 1.1 to 18 times on a Hitachi SR2201 supercomputer.

記号

p：プロセッサ数
b：ブロックサイズ
q：プロセッサ番号
L_T ：テンプレートの，ある次元の下限値
e：テンプレートの，ある次元の寸法
c：cyclic(b)分散におけるサイクルの数 $=\lceil e/(p*b)\rceil$
n：テンプレートのAXIS_TYPEがNORMALとなる次元数
v：アライン指示文でアラインソースが"*"となる次元数
r：テンプレートのAXIS_TYPEがREPLICATEDとなる次元数
s：テンプレートのAXIS_TYPEがSINGLEとなる次元数
f_a*i+f_b，F_a*i+F_b：アライン添字とその行列形
i_a*i+i_b，X_a*i+X_b：ON HOME指示文のHOME節中の配列添字及びその行列形
l, u, m：分散対象ループの下限値，上限値，ストライド
$LT(x) := f_n*i_n*x+f_a*i_b+f_b-L_T$
$TL_0(j, q, x) := (p*b*j+b*q+x+L_T-f_a*i_b-f_b-l*f_n*i_n)/(f_n*i_n*m)$

1. はじめに

分散メモリ型計算機や分散共有メモリ型計算機では，各プロセッサのデータアクセスを局所化させてプログラムの実行を高速化するデータ分散が重要である．本論文で考察の対象とするプログラムパターンは，HPF [1]による規則的なデータ分散が指示されたh次元配列，並びに，ループ制御変数の線型式を添字とする配列要素をHOME節[1]に指定したON HOME指示文を含むd重ループである．このパターンを，本論文ではRDLS(h, d)パターン(Regular data Distribution and Linear Subscript)と呼ぶ．このパターンは密行列を使った流体計算など多くの科学技術計算プログラムに現れる重要なものである．

一般のRDLS(h, d)パターンに適用可能な計算分散法として実行時解決法（RR）がある．しかし，その生成コードの実行性能は良くない[2]．

また，ループストライドが定数となるRDLS(h, d)パターンに適用可能な計算分散法としてD system [3]がある．こ

れはプロセッサ数やブロックサイズが定数の場合，オメガテストを用いるため，コンパイルに多くの時間がかかる可能性がある．他方，これらの値が定数でない場合はvirtual processor[7]によるコードを生成する．このコードはblock-cyclic分散ではテーブル検索法コード[5]より遅い[6].

また，テーブル検索法コードは高速であるが，これを用いた従来の研究ではALIGN指示文を含まない特殊なパターン[6]やループストライドが正で，配列次元とループ制御変数が1対1に対応する場合[5]しか適用できていなかった．一方，我々は一般のRDLS(1, 1)パターンに対してこのような高速なテーブル検索コードを得ている[9].

本論文の目的は，一般のRDLS(h, d)パターン，特にhやdが2以上の場合に対して，高速なコードを得ることである．

このために，我々は，一般のRDLS(h, d)パターンに対する図式表現[9]の上で，添字写像及の逆写像を典型的なプログラムパターンに対して与え，この結果と図式表現とを用いて計算分散を与えた．また，部分プロセッサ[1]やMulti-partitioning[12]も図式表現に加え，同じフレームワークで計算分散も可能であることを示した．これらを統合化して一般のRDLS(h, d)パターンに対する計算分散アルゴリズムを与えた．また，上記各パターン及び一般のパターンに対する擬周期も示し，最初の周期における参照添字取得コードをInspectorとし，取得した参照添字から一般の参照添字を得て計算を行なうコードをExecutorとするテーブル検索法コードを導いた．

尚，本論文では，対象配列は共有メモリ上にあるものとし，その配列添字はデータ分散前の添字と同じ（グローバルアドレス表現）とする．

本論文の残りは以下となる．第2章ではデータ分散の図式表現を示し，第3章ではこれを用いて典型的なパターンに対する逆写像（逆添字写像等）と計算分散を与え，一般のパターンに対するアルゴリズムを与える．第4章では上記の典型的なパターン及び一般のパターンに対する参照添字の擬周期を示してテーブル検索法コードを与え，第5章では提案コードの評価を示し，第6章で関連研究を述べ，第7章で本研究のまとめを行なう．

2．データ分散の図式表現

図1は多次元配列に対するHPFの規則的なデータ分散を適用する時の図式表現に2つの拡張を加えたものである．図式中の，FaはZ^nに対応する次元はf_a倍，Z^nに対応する次元は0，Z^vやZ^cに対応する次元は恒等写像となるように定めた行列を表わす．

1つ目の拡張は，ループイタレーション空間Lである．この空間Lはループネスト数をdとすると，Z^dの中へ1対1に埋めこまれる．さらに，LとZ^dから各々，AとZ^{n+v}へ，HOME節中のAのある要素に対する線型添字が定めるアフィン写像X_a*I+X_b（但し，X_a：(h, d)型行列，I：d次元ベクトル，X_b：h次元ベクトル）がある．このアフィン写像を図式に含めることで，計算分散を考えることができる．

2つ目の拡張は，プロセッサ空間$\sum I_p$からプロセッサ空間$\sum I_g$へのマップφである．これは$\sum I_p$を仮想プロセッサ空間（[7]とは異なるが本論文ではこれもVP:virtual processorと呼ぶ），$\sum I_g$をHPFのPROCESSORS指示文で規定された論理プロセッサ空間とみなすことにより，HPFの公認拡張仕様である部分プロセッサ及びHPF公認拡張でさえないMulti-partitioning等のデータ分散の表現を可能とする．

例えば，部分プロセッサは，$\sum I_p$を部分プロセッサから成る空間，マップφを部分プロセッサから論理プロセッサの中への平行移動とみなす（図2(a)）．ここで，図中の矩形上の番号は，矩形に対応する部分配列がマッピングされる（ロジカル）プロセッサ番号を表わす．

また，2次元配列に対するMulti-partitioning[12]は，論理プロセッサ数をpとすると，

$$\sum I_p = I_p \oplus I_p,\quad \sum I_g = I_p,\quad \varphi = (vp_1 - vp_2) \bmod p$$

と表わされる（図2(b)）．ここで，$(vp_1, vp_2) \in \sum I_p$である．

3．逆写像と計算分散

与えられたデータ分散を元に計算分散を行なうことは図式上では$\sum I_p$（または$\sum I_g$）からループイタレーション空間Lへの逆写像を与えることに相当する．

これを求めるには，

（1）VPからPへの写像に対する逆写像，

（2）align指示文によるF_aの逆写像，

（3）逆添字写像

の3つを求めることが必要である．

このうち，F_aはHPFの仕様によりAXIS_TYPEがNORMALな次元では1対1写像になり，SINGLEな次元ではその次元がマップされるプロセッサ番号は唯一に決まり，その他のAXIS_TYPEは計算分散に関係しない[9]ので，計算分散は考え易い．よって，（1）と（3）の考察が重要である．以下，両者の逆写像ととそれを用いた計算分散を与える．

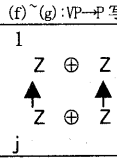
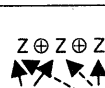
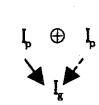
3．1　逆添字写像

多重ループに対しても，基本的に各ループとその制御変数が現れる配列次元に対して定理1[9]と同じ方法で，ループ毎に計算分散を行なう．但し，以下の2つの例外がある．1つ目は1つのループ制御変数が配列の複数の次元に現れる場合であり，他の1つは複数のループ制御変数が1つの配列次元に現れる場合である．以下，これらの場合とその組合せについて，表1の(a)から(e)を用いてRDLS(h, d)の典型的なパターンに対する逆添字写像の求め方を説明する．

(a)　単純ケース

この例ではループ制御変数と配列添字が1対1に対応するので逆添字写像も配列の各次元から各ループイタレーション空間へ個別に求まる．このケースの逆写像は各次元毎の独立な逆写像の和となるので逆写像パターンを∑（直和）と表現する．

表1：プログラムパターンに対する図式・計算分散・擬周期

	プログラム パターン	図式表現 (a)~(e)：添字写像 (f)~(g)：VP→P 写像	逆写像 パターン	計算分散	擬周期
(a)	do i=1, M do j=1, N A(j, i)=		Σ	do i'=LTi(1), LTi(M) do i=TLi(1), TLi(bi) do j'=LTj(1), LTj(N) do j=TLj(1), TLj(bj) A(j, i)=	$\sum_i m_i * \alpha_i$
(b)	do i=1, M A(i, i+1)=		∩	do i1=LT1(1), LT1(M) do i2=LT2(1), LT2(M) do i=max(TL1(1), TL2(1), min(TL1(b1), TL2(b2)) A(i, i+1)=	$m * LCM(\alpha_1, \ldots, \alpha_n)$
(c)	do i=1, M do j=1, N A(i+j)=		∪	do i=1, M do j'=LTj(1), LTj(N) do j=TLj(1), TLj(bj) A(j, i)=	$m*\alpha$
(d)	do i=1, M do j=1, N A(a*i+b*j, c*i+d*j)=		∪∩	do i=1, M do j1=LT1(1), LT1(N) do j2=LT2(1), LT2(N) do j=max(TL1(1), TL2(1), min(TL1(b1), TL2(b2)) A(a*i+b*j, c*i+d*j)=	$m_i * LCM(\alpha_1, \ldots, \alpha_{ni})$
(e)	do i=1, M do j=1, N A(a*i+b*j, c*i+d*j, i)=		Σ∪∩	do i'=LTi(1), LTi(M) do i=TLi(1), TLi(bi) do j1=LT1(1), LT1(N) do j2=LT2(1), LT2(N) do j=max(TL1(1), TL2(1), min(TL1(b1), TL2(b2)) A(a*i+b*j, c*i+d*j, i)=	$\sum_i m_i * LCM(\alpha_1, \ldots, \alpha_{ni})$
(f)	processors P(p) distribute onto VP(p, p) vq1-vq2≡q mod(p)		∪	do vp2=0, p-1 vq1=vq2+q mod(p) ...	対象ループネストの擬周期
(g)	distribute onto P(3:K)		∩	if(3≤vq≤K) then ...	対象ループネストの擬周期

bi, bj, b1, b2：ブロックサイズ，LTi, LTj, LT1, LT2：写像 LT（記号の項参照），TLi, TLj, TL1, TL2：写像 TL_0

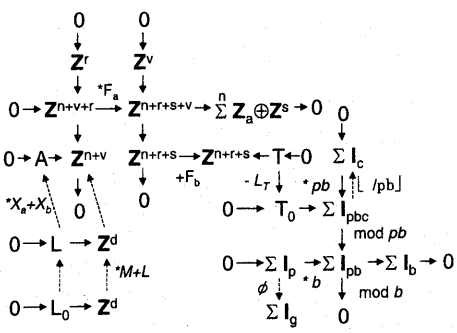


図1：一般の規則的データ分散の図式表現

		(1,2)	(1,3)
		(2,2)	(2,3)
		(3,2)	(3,3)

(a) 部分プロセッサ

0	3	2	1
1	0	3	2
2	1	0	3
3	2	1	0

(b) Multi-partitioning

図2：HPF以外のデータ分散例.

(b)　1制御変数－多次元ケース

この例では，1つのループ制御変数が現れる各配列次元によって分散された各分散後のループ範囲の交わりを計算分散後のループ範囲とする．これは，上記ループ制御変数が上記各次元の制約を同時に満たす必要があるからである．

こうして良いことは以下のようにも説明できる．表1(b)の例では，添字写像の像は

$$\Pi = \{(i,\ i+1) \mid i \in Z\} \qquad (1)$$

なので，ある添字の集合 S が与えられた時，逆添字写像は両者の交わり $S \cap \Pi$ に対してのみ定義でき，逆写像は

$$(i,\ j) \rightarrow i \quad (\text{or}\quad j-1) \qquad (2)$$

と一意に決めることが出来る．このことは S に含まれる i と j の逆像を個別にとって，その交わりを計算すること

$$\{i \mid (i,\ j) \in S\} \cap \{j-1 \mid (i,\ j) \in S\} \qquad (3)$$

と同値である（付録参照）．後者の考え方を用いると，逆写像のパターンは個別に得られた結果の交わりなので∩と表現し，計算分散の結果は表1(b)となる．

(c)　複数制御変数－1配列次元

この例では，逆添字写像は一意に定まらず，その像はあるパラメータを用いたループ制御変数の集合として表現される．例えば，表1(c)のループに対する逆添字写像は

$$k \rightarrow (j,\ k-j) \in (j,\ i),\quad 1 \le j \le k-1$$

となる．したがって，ループ毎に逆添字写像を適用できない．そこで，内側ループでは外側ループ制御変数がループ不変であることを利用し，配列添字を内側ループの制御変数の1次式とみなし内側ループに対して計算分散を行ない，外側ループは計算分散しない．

この考え方は以下と同じである．表1(c)の例では，1つの k(=i+j)に対応する(i, j)の組は i+j=k となり，i, j が各ループのイタレーション範囲内に含まれる全ての制御変数の組として与えられる．各組は一方の値 i が決まれば他方の値 j は j=k−i として一意に決まる．ループ交換が不可能な一般の場合を想定してパラメータとして外側ループのループ制御変数を採用すると k に対応する逆像は

$$\bigcup_j \{(j,\ k-j)\}$$

と表わされる．

これにより逆像のパターンは∪と表わされる．ここで，表1(c)の図式表現における点線の矢印はそれが外側ループのイタレーション空間から配列次元への写像であるため，逆添字写像として計算分散に直接寄与してないことを表わす．

(d)　一般ケース (I)

表1(d)は(b)と(c)の複合ケースである．

まず，(c)と同じ考え方により，配列添字を内側ループの制御変数の1次式とみなし内側ループに対して計算分散を行なう．また，ループ制御変数 j が複数の次元に出現しているので，(b)と同じ考え方により各次元毎のjに関する逆写像の交わりを取ることによって j ループに対する計算分散を得る．

入力: データ分散配列A，テンプレートT，1文のみを含むループネストN，およびその文直前に挿入された配列 A の要素 E を HOME 節に指定した ON HOME 指示文
出力: ループネストNに対する計算分散コード
アルゴリズム:
配列要素Eの全次元を未処理とする．
for (N 中の各ループLに対して，内側から順に外側へ)
　Rを Lのループ範囲とする
　for (配列要素Eの各次元dに対して)
　　if (次元dは未処理 ∧ 次元dは分散される ∧
　　d次元目の添字SはLのループ制御変数iを含む ∧
　　次元dの AXIS_TYPE は NORMAL)
　　次元dを処理済とする
　　添字Sを i の1次式とみなして定理1[9]を適用し，得られたループの計算範囲を R_d とし、$R = R \cap R_d$ を計算する
　　ループL直前に S1，周期ループ，および S2 を生成する
endif; endfor; endfor
for (配列要素Eの各次元dに対して)
　if (次元dの添字が定数 f_b ∧ 次元dは分散される)
　　生成コード全体を以下の IF 文で囲む：
　　"if (q=⌊(f_a*i_b + f_b −L_T) mod (p*b) / b⌋)"
　　但し，q は要素Eの次元dがマップされるプロセッサ次元のプロセッサ番号を表わす．
endif; endfor
for (配列Aのテンプレート Tの各次元dに対して)
　if (次元dの AXIS_TYPE が SINGLE でその添字値が f_b)
　　生成コード全体を以下の IF 文で囲む：
　　"if (q=⌊(f_b −L_T) mod (p*b) / b⌋)"
　　但し，q はテンプレートTの次元dがマップされるプロセッサ次元のプロセッサ番号を表わす．
endif; endfor
for (部分プロセッサの各プロセッサ番号 vq に対して)
(*)　vq の全ての値を渡るループを生成
for (部分プロセッサの各プロセッサ番号 vq に対して)
　vqと同じ軸のプロセッサ番号qとの関係式を条件式とするIF文を(*)のループ中に生成し，生成コード全体をこの IF 文で囲む．

図3：　**計算分散アルゴリズム**.

以上の方法によるパターンは∪∩と表現でき，その計算分散結果は表1(d)で与えられる．

(e)　一般ケース (II)

表1(e)では1，2次元目は(d)と同じパターンだが，3次元目はループ制御変数 i のみで計算分散される．よって，そのパターンは j ループを1，2次元目に従って計算分散し，i ループを3次元目に従って計算分散するので，∑∪∩と表現でき，その計算分散結果は表1(e)で与えられる．

3. 2 VP から P への写像の逆写像

表1の(f)と(g)を用いて，Multi-partitioning と部分プロセッサに対する VP から P への写像の逆写像の求め方を説明する．

(f) Multi-partitioning

この例では，p 個の要素からなるロジカルプロセッサ P が PROCESSORS で宣言され，virtual processor VP がテンプレートのターゲットとなり，vq1, vq2∈ VP と q∈P との間に

vq1 − vq2 ≡ q mod(p)

という関係があることを示している．

表1(f)は，1つ元 q に対して，多くの vq1（及び vq2）が対応するので，その図式は表1(c)の図式と同じパターンである．よって，同様な考え方により逆写像は以下となる．

$$q \rightarrow \bigcup_{vq2} \{ (vq1 = (q + vq2) \bmod (p),\ vq2) \}$$

(g) 部分プロセッサ

P から VP への写像 ϕ は I_g の中への1対1写像なので，逆写像は ϕ の像から I_p への1対1写像になる．よって，計算分散は表1(g)で与えられる．

3. 3 一般のアルゴリズム

図3は一般の RDLS(h, d)パターンに対する計算分散のアルゴリズムである．

最初の2重の for 文によって，内側ループから順に計算分散され，かつ，一つのループ制御変数が複数の配列次元に出現する場合は各々による計算分散の交わりを取る．

2番目と3番目の for ループでは，配列添字が特定の1つの次元プロセッサにマッピングされる場合を扱う．

最後の2つの for ループは VP と P とが異なる場合に対する計算分散を与える．

尚，配列 A の AXIS_TYPE が VANISHED[9]となる次元に制御変数を持つループは分散されないので，アルゴリズムの前半では AXIS_TYPE が NORMAL となる次元のみ考慮する．

4. 最適コード生成

本章では表1の各パターンに対する擬周期[9]を説明し，その結果から各パターン及び一般の RDLS(h, d)パターンに対するテーブル検索法コード[9]を導く．

(a) 複数の独立な次元の擬周期を別々に求めれば良いので，擬周期はそれらの和となる．計算分散コードは，Inspector を独立な次元の数だけ実行し，Executor は1つとなる．

(b) 各次元毎に得られる擬周期を同時に満たす必要があるので，i ループの擬周期はそれらの LCM で与えられる．計算分散コードは1次元－1重ループと同じである．

(c) 1つの配列次元から内側ループを計算分散するので，擬周期や計算分散は1次元－1重ループと同じである．但し，Inspector は外側ループの内側で実行するので，オーバーヘッドが大きくなる可能性が有る．

```
!HPF$ PROCESSORS P(8)
!HPF$ DISTRIBUTE A(CYCLIC(b)) ONTO P
      do i = 0, 99
        do j = 1, 800
!HPF$ ON HOME(A(aj*j+ai*i+bj))
          A(aj*j+ai*i+bj) = 100
      enddo; enddo
```

図4： 評価プログラム．

表2： SR2201 上での実測結果 [sec].

b, aj	IE	ALG	RR	RR /IE	RR/ALG
256, 25	0.413	0.060	0.194	0.47	3.23
256, 3	0.363	0.008	0.144	0.40	18.00
32, 25	0.142	0.111	0.231	1.63	2.08
32, 3	0.057	0.014	0.149	2.61	10.64
4, 25	0.105	0.203	0.232	2.21	1.14
4, 3	0.019	0.026	0.149	7.84	5.73

b: block size, aj: coefficient of j

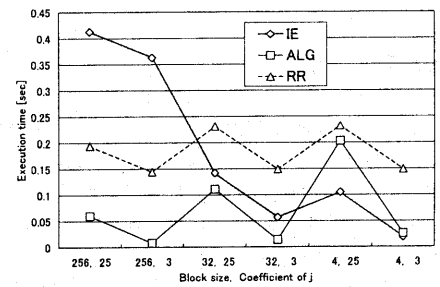


図5： SR2201 上での実測結果 [sec].

(d)と(e)については以上の組合せで得られる．(f)と(g)については，VP と P との関係は計算分散されるループとは関連がないので，(f)や(g)は擬周期に影響は及ぼさない．

5. 評価

提案した種々の計算分散方法の効果を調べるために，図4で示した複数制御変数－1配列次元型のプログラムを，図3のアルゴリズム，4章の最適化コードおよび実行時解決法コードで書きなおし，ブロックサイズとループ j の係数 aj 6つの組に対して実測した．ここで，図4中の ai は参照添字が全て異なるように決め，bj はプロセッサ毎の負荷がほぼ均一になるように決めた．(block, aj) = (256, 3)の場合は各プロセッサが参照した配列要素数は各々，10000±48 の範囲内の数だったが，それ以外では，10000±6 の範囲内の数だった．

測定マシンは分散メモリ型並列計算機である日立

SR2201，プロセッサ数は 8 である．OS は HI-UX/MPP 02-03, Fortran90 コンパイラは OFORT90 V02-06-/E であり，最適化オプションは O(SS)である

表２及び図５は SR2201 上での実測結果である．IE はテーブル検索法コード，ALG は図３のアルゴリズムによる変換結果コード，RR は実行時解決法コードを表わす．表中の値は８プロセッサでの実行時間の平均である．

テーブル検索法コードはブロックサイズが大きい時は実行時解決法 (RR)よりも遅く，スピードは 0.4 倍程度である．これは擬周期がこのプログラムの例では 8*b（b はブロックサイズ）となるので，外側ループの中にある Inspector のオーバーヘッドが大きいためと考えられる．この値が小さい時は最高で RR の 7.8 倍高速である．

また，提案アルゴリズムを直接使った方法は RR よりも 1.1 倍から 18 倍高速である．1.1 倍とスピードが近づいたのは，ブロックサイズに比べて係数 aj の値が大きい時である．これは，提案アルゴリズムがブロックサイズをループイタレーション空間にマップした区間をループ計算後のループ区間とするのに対し，RR は大きなストライドを持つので，提案アルゴリズムの方がループの実行回数が多くなるためである．

6．関連研究

我々はすでに多重ループに対して，内側ループから外側ループに向かって計算分散を行なうアルゴリズムを提案した[10]．本論文ではこれを，一般の cyclic-block 分散，ALIGN 指示文，Multi-partitioning を含む場合に拡張した．

太田ら[11] は計算分散方式について述べているが，cyclic(b)のような複雑なデータ分散については言及していない。また，a(i, i+1)等の，ループ制御変数が複数次元に現われるパターンでは，計算分散後のループ内の文に IF 文が付くので，我々のコードより遅いと思われる．

7. おわりに

HPF が規定する一般の規則的データ分散とループ制御変数の線型式を添字とする配列要素をループ中の ON HOME 指示文の HOME 節に指定した多重ループを RDLS パターンと呼ぶ時，以下を示した．

(1) RDLS パターンに対するデータ分散の図式表現上で部分プロセッサ，Multi-partitioning も表現した．
(2) h 次元配列と d 重ループから成る RDLS パターンに対し，添字写像等の逆写像を図式で与え，計算分散アルゴリズムを与えた．
(3) 計算分散後コードにおける参照添字の擬周期を与え，テーブル検索法コードを与えた．
(4) 複雑なサイクリック・ブロック分散を持つプログラムに提案方法を適用して日立 SR2201 上で実測し，テーブル検索法コードは実行時解決法よりも 0.4 から 7.8 倍高速で，提案アルゴリズムを直接使った方法は実行時解決法よりも 1.1 倍から 18 倍高速であることがわかった．

参考文献

[1] High Performance Fortran Language Specification Version 2.0, Rice Univ., 1997.

[2] S. Hiranandani et al. Compiling Fortran D for MIMD Distributed-Memory Machines, *Communications of the ACM*, Vol. 35, No. 8, pp. 66-80, Aug. 1992.

[3] V. Adve et al. Using Integer Sets for Data-Parallel Program Analysis and Optimization, PLDI'98, pp. 186-198, 1998.

[4] 河田敬義，ホモロジー代数学，岩波数学選書．

[5] S. Chatterjee et al. Generating local addresses and communication sets for data-parallel programs. PPoPP'93, pp. 149-158, San Diego, CA, May, 1993.

[6] K. Kennedy et al. Efficient Address Generation for Block-Cyclic Distributions, ICS '95, pp. 180-184, 1995.

[7] S. Gupta et al. Compiling Array Expressions for Efficient Execution on Distributed-Memory Machines, *J. of Parallel and Distributed Computing*, Vol. 32, pp. 155-172, 1996.

[8] N. H. Naik et al. Parallelization of a class of implicit finite-difference schemes in computational fluid dynamics, *International Journal of High Speed Computing*, 5(1), pp. 1-50, 1993.

[9] 佐藤真琴．データ分散の図式表現と計算分散公式の提案及び評価，情報処理学会研究報告 2001-HPF-87 (SWoPP 2001)，pp. 81-86，2001.

[10] M. Satoh et al. Program Partitioning Optimizations in an HPF Prototype Compiler, in Proceedings of Twentieth Annual International Computer Software and Applications Conference (Compsac '96), pp. 124-131, Seoul, Aug. 1996.

[11] 太田寛，西谷康仁：データ並列言語における多重ループの統一的計算分散方式，情報処理学会論文誌，Vol. 41，No. 5，pp. 1439-1447.

付録

3.1 節における添字写像 i→i，i→i+1 を各々，f，g とし，２次元添字から１次元目，２次元目の成分への射影を p1，p2 とすると，S∩II を定義域とする逆添字写像（式(2)）の像は以下のように表現できる：

$$f^{-1}(p1(S\cap II)) \quad or \quad g^{-1}(p2(S\cap II)) \tag{4}$$

また，式(1)の II は以下となる：

$$II = \{(f(j), g(j)) \mid j \in Z\} \tag{5}$$

ここで，式(4)の第１式は以下となる（第２式も同様）．

$$i \in f^{-1}(p1(S\cap II)) \leftrightarrow f(i) \in p1(S\cap II) \subseteq p1(II)$$
$$\therefore \exists j \text{ s.t. } f(i)=f(j) \leftrightarrow i = j \quad (f : 1 \text{ to } 1 \text{ mapping})$$
$$\therefore (f(i), g(i)) \in S\cap II \leftrightarrow (f(i), g(i)) \in S$$
$$\therefore i \in f^{-1}(p1(S)) \cap g^{-1}(p2(S))$$

右辺は式(3)と同値．式(3)から式(2)を得るのも同様．□