計算機アーキテクチャ 125－3
（1997. 8. 20）

データ分割配置を考慮するループディストリビューション

中西 恒夫[1]，城 和貴[2]，Constantine D. Polychronopoulos[3]，
荒木 啓二郎[4]，福田 晃[1]

[1] 奈良先端科学技術大学院大学情報科学研究科
630–01 奈良県生駒市高山町 8916 番地の 5
[2] 和歌山大学システム工学部情報通信システム学科
[3] Center for Supercomputing Research and Development,
University of Illinois at Urbana-Champaign
[4] 九州大学大学院システム情報科学研究科
E-mail: nakasu-para@is.aist-nara.ac.jp
URL: http://fukuda.aist-nara.ac.jp/

概要

本稿では，自動並列化コンパイラの中間表現であるデータ分割グラフ上で，データ分割配置とループディストリビューションを同時に行うアルゴリズムを提案する．従来の一般的なループディストリビューションアルゴリズムでは，ループネストは可能な限り細かく分割され，また並列化なループは全て並列化される．しかしながら，通信オーバーヘッドが比較的大きな分散メモリ型並列計算機上では，過剰なループネストの分割と並列化は，プロセッサ間通信のために当該ループネストの実行時間が引き延ばされる結果に陥る．提案アルゴリズムでは，データ転送コストを考慮の上で適正なループネストの分割と並列化を行う．

A Loop Distribution Algorithm with Data Partitioning

Tsuneo NAKANISHI[1], Kazuki JOE[2], Constantine D. POLYCHRONOPOULOS[3],
Keijiro ARAKI[4], Akira FUKUDA[1]

[1] Graduate School of Information Science,
Nara Instituite of Science and Technology
8916–5, Takayamacho, Ikoma, Nara 630–01, JAPAN
[2] Department of Computer and Communication Sciences,
Faculty of Systems Engineering, Wakayama University
[3] Center for Supercomputing Research and Development,
University of Illinois at Urbana-Champaign
[4] Graduate School of Information Science and Electrical Engineering,
Kyushu University

ABSTRACT

In this paper we propose an algorithm which performs data partitioning and loop distribution on the data partitioning graph, which is an intermediate representation for parallelizing compilers. Loop distribution algorithms used in common partitions and parallelizes loop nests as much as possible. However, excessive partitioning and parallelizing of loop nests cause poor results in terms of execution time due to communication overheads on distributed memory parallel computers whose inter-processor communication is high. The proposed algorithm partitions and parallelizes loop nests reasonably with communication overheads consideration.

1 はじめに

ループディストリビューション [1] は，自動並列化／ベクトル化コンパイラにおいて，逐次ループネストを複数のループネストに分割し，そのうち並列実行可能なループネストを並列化することにより，ループネストの実行時間の短縮を図るループ変形技法である．しかしながら，その結果として得られるループネスト群を分散メモリ型並列計算機上で実行する場合，近年問題になっているプロセッサとメモリあるいはバスのスピード格差を考えれば，ループネストを分割した結果生じるプロセッサ間通信により当該ループネストの実行時間が延長されることが考えられる．過剰にループネストを分割した場合は，たとえ分割後に得られたループネストが並列化できたとしても，実行時間の短縮にはつながらない．分散メモリ型並列計算機上でプロセッサ間通信を削減するためには，データの適切な分割配置が極めて重要である．ループネストがどのように分割されるか，分割の結果得られる個々のループネストが逐次実行されるか並列実行されるかにより，プロセッサ間通信の量や配列の最適の分割配置が異なってくる．したがって，ループディストリビューションとデータの分割配置の問題は，決して個別に解決される問題ではなく，同時に解決されるべき問題である．そこで本稿では，本研究ですでに提案しているデータ分割グラフ（DPG: Data Partitioning Graph）上でデータ転送コストを考慮の上で，ループディストリビューションならびにデータ分割配置を同時に行うアルゴリズムを提案する．

2 ループディストリビューション

ループディストリビューションは，与えられたループネストをそれと等価な複数のループネスト群に分割するし，そのうち並列実行可能なループネストを並列化するループ変形技法のひとつである．図1上のループネストにループディストリビューションを適用した例を，図1下に示す．逐次ループのみからなるループネストの一部が，DOALL ループに変換されていることに着目されたい．

ループディストリビューションは対象ループネストのデータ依存グラフ（DDG: Data Dependence Graph）を中間表現として用いる．今，対象ループネストの DDG が $G=(V,E)$ として表されるものとする．但し，Vはタスクを表す節点の集合，Eはタスク間のデータ依存を表す枝の集合である．以下に一般的なループディストリビューションのアルゴリズムを示す．

1. G の強連結部分を検出する．各々の強連結部分に属する節点の集合を S_i $(1 \leq i \leq p)$ とする．
2. 新しいグラフ $\tilde{G}=(\tilde{V},\tilde{E})$ を生成する．$\tilde{V}$はGの各強連結部分を意味する節点の集合，すなわち $\tilde{V}=\{S_i | 1 \leq i \leq p\}$ である．また，G において異なる強連結部分 S_u，S_vにそれぞれ始点，終点が属する枝が存在する時に限り，$\tilde{G}$の節点 S_uから S_vへ向かう枝 (S_u,S_v) を設ける．すなわち，$\tilde{E}=\{(S_u,S_v) | (v_u,v_v) \in E, v_u \in S_u, v_v \in S_v\}$ である．

```
// Original
DO I=1, N
  DO J=1, M
    C(I,J) = A(I,J)*B(I,J)
    A(I+1,J+1) = C(I,J-2)/2 + C(I-1,J)*3
    D(I,J) = D(I-1,J-1) + 1
    B(I,J+4) = D(I,J) - 1
  EndDO
EndDO

// after Loop Distribution
DO I=1, N
  DO J=1, M
    D(I,J) = D(I-1,J-1)+1
  EndDO
EndDO
DOALL I=1, N
  DOALL J=1, M
    B(I,J+4) = D(I,J) - 1
  EndDOALL
EndDOALL
DO I=1, N
  DO J=1, M
    C(I,J) = A(I,J)*B(I,J)
    A(I+1,J+1) = C(I,J-2)/2 + C(I-1,J)*3
  EndDO
EndDO
```

図 1: プログラム例

3. 2で得られた$\tilde{G}$をトポロジカルソートする．ソート結果の節点列を $\langle \tilde{v}_1, \tilde{v}_2, ..., \tilde{v}_p \rangle$ とする．
4. $i=1$ から p まで昇順で，$\tilde{v}_i$に属する G の節点のタスクからなるループネストを生成する．$\tilde{v}_i$ の強連結部分が節点単独（枝を含まない）ならば，当該ループネストを並列化する．

上述のアルゴリズムでは，データ転送のオーバーヘッドは全く考慮されず，ループネストは可能な限り細かく分割され，並列化されていることに着目されたい．

3 提案アルゴリズム

本節では，データ転送コストを考慮の上でデータ分割配置ならびにループディストリビューションを行うアルゴリズムを提案する．提案アルゴリズムは，本研究ですでに提案しているデータ・プログラム同時分割アルゴリズム — CDP^2アルゴリズム：Combined Data Program Partitioning Algorithm[2] — を拡張したものとなっており，中間表現としては本研究で提案している DPG を用いる．

前節に示した一般的なループディストリビューションアルゴリズムは並列実行可能なループネストは可能な限り並列化し，またループネストは可能な限り細かく分割している．しかしながら，通信オーバーヘッドの大きな計算機においては，並列実行可能な場合でも逐次実行した方が実行時間が短いケースが生じ得る上，過剰なループネストの分割は実行時間の延長につながり得る．本節で提案するアルゴリズムは，データ転送コストを考慮の上でループネストを適正に分割すると同時に，データ分割配置の最適化を行う．

3.1 前提条件

対象アーキテクチャ：提案アルゴリズムが対象とするアーキテクチャは，i) 個々のプロセッサは遅延0でアクセスできるローカルメモリを有し，ii) 個々のプロセッサは他のプロセッサのローカルメモリ，すなわちリモートメモリに一定の遅延でアクセスでき，iii) リモートメモリアクセスに伴う遅延は，アクセスされるリモートメモリの位置に依存しないものと仮定する．

対象ループネスト：対象ループネストはパーフェクトループネストとし，ループボディの繰り返し数はコンパイル時に既知であるものとする．また，ループボディ内には条件分岐を持たないものとする．

生成されるループネスト群：提案アルゴリズムにより得られるループネスト群のループネストは，並列実行される場合はネスト内の全てのループはDOALLループとして実行されるものとする．したがって並列実行されるループネスト内に，ループによって運ばれるデータ依存はない．

本稿では紙面の制約により，データの分割配置について，スカラ変数の取り扱いについては割愛する．

3.2 DPGの概要

提案アルゴリズムは中間表現としてDPGを用いている．DPGの一例として，図1上のプログラムを表すDPGを図2に示す．

DPGには2種類の節点，C-nodeとD-nodeが存在し，D-nodeにはスカラ変数の集合を表すスカラD-nodeと配列を表す配列D-node[1]とがある．本稿ではスカラ変数の取り扱いについては議論しないので，ここでは配列D-nodeのみを検討する．C-nodeはタスクを表し，図2では円形の節点として表されている．配列D-nodeはタスクの全インスタンスによりアクセスされる配列の部分配列を表し，図2では配列D-nodeが方形の節点として表されている．枝には，C-nodeとC-nodeの間に張られる制御フロー，制御依存，データ依存を表す枝のほか，データアクセスを表す枝が存在する．データアクセス枝には，D-nodeからC-nodeに張られるReadアクセス枝と，C-nodeからD-nodeに張られるWriteアクセス枝が存在し，それぞれC-nodeのタスクがD-nodeの変数・配列にReadアクセス，Writeアクセスすることを意味する．

本稿では，DPGを形式的に$G_{DPG} = (\{CV, DV\}, \{CE_f, CE_c, CE_d, DE\})$と表記する．$CV$，$DV$，$CE_f$，$CE_c$，$CE_d$，$DE$はそれぞれ，C-node，D-node，制御フロー枝，制御依存枝，データ依存枝，データアクセス枝の集合である．また，与えられたDPGのC-nodeとデータ依存枝のみからなるDDGを形式的に$G_{DDG} = (CV, CE_d)$と表記する．

3.3 問題の定義

ループディストリビューションおよびデータ分割配置の問題は，G_{DPG}のC-nodeおよびD-nodeをいかにグルーピングするかの問題としてとらえることができる．同じグループに属するC-nodeのタスクは，ループディストリビューション後に同じループネストに属する．同じ逐次ループネストに対応するグループに属するC-nodeとD-nodeについては，そのC-nodeのタスクが実行されるプロセッサのローカルメモリにそのD-nodeの変数・配列が配置される．同じ並列ループネストを表すグループに属するC-nodeとD-nodeについては，そのC-nodeのタスクのインスタンスが実行されるプロセッサのローカルメモリにそのD-nodeの配列の要素が配置される．ループディストリビューションの際は，G_{DDG}の任意の強連結部分について，その強連結部分に属する全てのC-nodeのタスクは，同じループネストに属するようにしなければならない．したがって，グルーピングの時はそれらのC-nodeは全て同じグループに属するようにしなければならない．

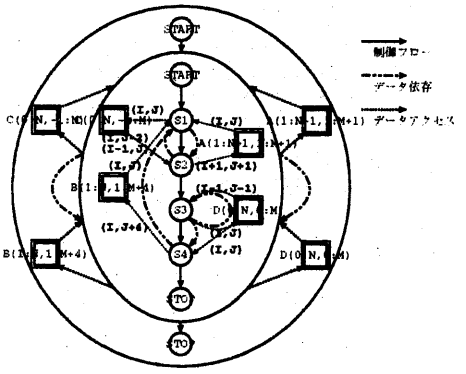

図2: DPGの一例

C-nodeおよびD-nodeのグルーピングを一意に定めるには，G_{DPG}のデータ依存枝を，両端のC-nodeが同じグループに属するデータ依存枝の集合*IntraPartition*と，両端のC-nodeが異なるグループに属するデータ依存枝の集合*InterPartition*とにクラス分けする．但し，G_{DDG}の強連結部分に属する全てのC-nodeのタスクは，同じループネストに属していなければならないので，強連結部分に含まれるデータ依存枝は全て*IntraPartition*にクラス分けしなければならない．さらにそのクラス分けに矛盾せぬように，G_{DPG}のデータアクセス枝を，両端のC-nodeとD-nodeが同じグループに属するデータアクセス枝の集合*IntraPartitionDE*と，両端のC-nodeとD-nodeが異なるグループに属するデータアクセス枝の集合*InterPartitionDE*とにクラス分けする．以上の処理により，最終的に図3に示すように，C-nodeとD-nodeのグルーピングが一意に定まる．

得られた個々のグループについてループネストが構成される．通過C-nodeが全て同じグループに属する，*IntraPartition*のデータ依存枝のみからなる閉路が存在する場合，そのグループについて構成されるループネストは逐次実行のみ許される．そうでない場合は，そのグループについて構成されるループネストは逐次実行，並列実行いずれも可能であり，与えられたループネストの実行時間が最小化

[1]D-nodeのこの分類はこれまで提案してきたDPGから拡張された部分である．

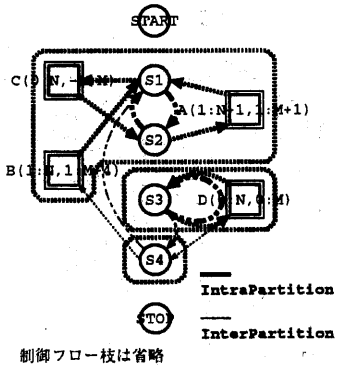

図 3: C-node と D-node のグルーピング

されるようにいずれかの実行形態を選択しなければならない．

以上をまとめると，データ分割配置を考慮するループディストリビューションの問題は，実行時間が最小化される，データ依存枝の $Intra/InterPartition$ へのクラス分け，データアクセス枝の $Intra/InterPartitionDE$ へのクラス分け，その結果得られる各 C-node および D-node のグループに対応するループネストの実行形態（逐次か並列か）の決定を探索する問題と定義することができる．

3.4 アルゴリズム

C-node の属性：G_{DPG} の各 C-node $cv \in CV$ に，当該タスクの 1 インスタンス実行に要する時間 $\omega_{CV}(cv)$ と，当該タスクが並列ループネストに属するか逐次ループネストに属するかを示す $mode(cv)$ を，属性として設ける．$mode(cv)$ は，cv のタスクが逐次ループネストに属する時，並列ループネストに属する時，いずれのループネストに属するか未定の時，それぞれ値 S，P，X を採るものとする．

$mode$ はアルゴリズムの最初で初期化する．G_{DDG} の任意の強連結部分について，その強連結部分がひとつの C-node 単独でならないならば，その強連結部分内の任意の C-node cv は必ず逐次ループネストに属するので，$mode(cv)$ を値 S で初期化する．強連結部分がひとつの C-node cv 単独でなる場合は，その C-node は逐次／並列いずれの実行形態も採り得るので，$mode(cv)$ を値 X で初期化する．

データアクセス枝の属性：C-node cv と配列 D-node dv の間のデータアクセス枝 de に，3 つの属性 $var(de)$，$\omega_{DE}(de)$，$\Omega_{DE}(de)$ を設ける．$var(de)$ は，cv のタスクのインスタンスが dv の配列へアクセスする際の（シンボリックな）添字である．また，$\omega_{DE}(de)$ は cv のタスクのインスタンスが $var(de)$ のデータをアクセスする時の通信コスト，$\Omega_{DE}(de)$ は cv のタスクの全インスタンスが dv の配列をアクセスする時の通信コストである．

枝のクラス分け：データアクセス枝を $Intra/InterPartitionDE$ へクラス分けする際は，データ依存枝の $Intra/InterPartition$ へのクラス分けの決定に矛盾しないようにしなければならない．これを保証するためには DPG の以下の性質が利用できる．

性質 1 DPG の任意のデータ依存枝 $ce = (cv_u, cv_v)$ について，

- ce がフロー依存枝ならば，
  $\exists dv \in DV : (cv_u, dv) \in DE, (dv, cv_v) \in DE$
- ce が出力依存枝ならば，
  $\exists dv \in DV : (cv_u, dv) \in DE, (cv_v, dv) \in DE$
- ce が逆依存枝ならば，
  $\exists dv \in DV : (dv, cv_u) \in DE, (cv_v, dv) \in DE$

□

すなわち，任意のデータ依存枝 ce の両端の C-node とデータアクセス枝で接続される D-node が存在する．このような D-node の集合を $DV(ce)$ と表記することにする．

データ依存枝 ce を $IntraPartition$ にクラス分けした場合と $InterPartition$ にクラス分けした場合の，ce の両端の C-node と $DV(ce)$ に属する D-node の間のデータアクセス枝の許されるクラス分けは，図 4に示すそれぞれ 2 通りと 3 通りである．

属性 $mode$ の選択：C-node の属性 $mode$ はデータ依存枝の $IntraPartition$，$InterPartition$ へのクラス分け決定に矛盾しないように，提案アルゴリズム適用後には S または P の値が与えられなければならない．データ依存枝 ce を $IntraPartition$ にクラス分けした場合と $InterPartition$ にクラス分けした場合の，ce の両端の C-node の属性 $mode$ の値割当は，図 5に示すそれぞれ 2 通りと 4 通りである．並列ループネスト内の全てのループは DOALL ループとするものと仮定しているので，$IntraPartition$ に分類されたループによって運ばれるデータ依存の枝の両端の C-node の $mode$ の値は常に S となる．

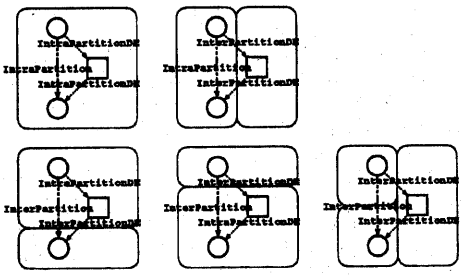

図 4: 矛盾しない枝のクラス分け

並列ループネスト内のデータ分割配置：並列ループネストのイタレーションは複数のプロセッサに配置され並列実行されるため，並列ループネストに対応するグループ内の D-node の配列はこれらのプロセッサのローカルメモリに分散配置されなければならない．本稿では，並列ループネストに対応するグループ内の D-node の配列の分割配置については議論せず，既成の適当な分割配置手法を使用するものとする．

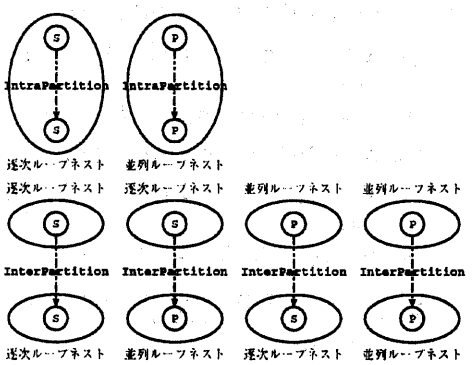

図 5: 矛盾しない属性 $mode$ の値割当

ループネスト群の実行時間の評価：提案アルゴリズムでは，ループディストリビューションの結果得られたループネスト群の実行時間の評価を行う．

実行時間の評価をするにあたり，新たに依存グラフ$\tilde{G}$を生成する．$\tilde{G}$の節点は，データ依存枝，データアクセス枝のクラス分けにより定まるC-nodeおよびD-nodeのグループを意味する．$\tilde{G}$の節点$\tilde{v}_u$から$\tilde{v}_v$へは，G_{DPG}において$\tilde{v}_u$のグループに属する節点から$\tilde{v}_v$のグループに属する節点へのデータ依存枝がある時に限り，枝が設けられる．

$\tilde{G}$の節点$\tilde{v}$にはコストが属性として与えられる．最初に$\tilde{v}$のグループに属する任意のC-nodecvの実行時間$\omega_{CV}(cv)$の総和sを求める．$\tilde{v}$が逐次ループネストに対応する場合は，sとそのループネストの繰り返し数の積を$\tilde{v}$のコストとする．$\tilde{v}$が並列ループネストに対応する場合は，$\tilde{v}$のグループ内のD-nodeの配列に対して適用した分割配置手法に沿った，当該ループネストのイタレーションの実行時間を$\tilde{v}$のコストとする．

$\tilde{G}$の枝$\tilde{e}=(\tilde{v}_u,\tilde{v}_v)$にもコストが属性として与えられる．$\tilde{v}_u$，$\tilde{v}_v$のいずれのグループも逐次ループネストに対応する場合，$\tilde{v}_u$に属するC-node（D-node）と$\tilde{v}_v$に属するD-node（C-node）の間の任意のデータアクセス枝deの通信コスト$\Omega_{DE}(de)$，および$\tilde{v}_u$に属するC-node，$\tilde{v}_v$に属するC-nodeと他の逐次ループネストに対応するグループ$\tilde{v}_x$に属するD-nodedv_xとの間の任意のデータアクセス枝de_xの通信コスト$\Omega_{DE}(de_x)$，および$\tilde{v}_u$に属するC-node，$\tilde{v}_v$に属するC-nodeと他の並列ループネストに対応するグループ$\tilde{v}_y$に属するD-nodedv_yとの間の任意のデータアクセス枝de_yの通信コスト$\omega_{DE}(de_y)$の総和を$\tilde{e}$のコストとする．$\tilde{v}_u$，$\tilde{v}_v$のいずれかのグループが並列ループネストに対応する場合は，Ω_{DE}の代わりにω_{DE}を用いて同様の計算をし，$\tilde{e}$のコストとする．

こうして得られたコスト付依存グラフ$\tilde{G}$のクリティカルパス長は，ループディストリビューションにより得られるループネスト群の最小並列実行時間を与える．このクリティカルパス長をループネスト群実行時間の評価値として用いる．

分枝限定法：得られるループネスト群の実行時間を最小化するような，データ依存枝の$Intra/Inter$-$Partition$へのクラス分け決定，ならびにそれに矛盾しないデータアクセス枝の$Intra/Inter$-$PartitionDE$へのクラス分け決定，属性$mode$への値割当を決定を探索するにあたり，提案アルゴリズムでは分枝限定法を用いる．分枝限定法の部分解は六つ組(I,U,DI,DU,m,a)で表され，Iは$IntraPartition$にクラス分けされたデータ依存枝の集合，Uはクラス分け未定のデータ依存枝の集合，DIは$IntraPartitionDE$にクラス分けされたデータアクセス枝の集合，DUはクラス分け未定のデータアクセス枝の集合，mは全C-nodeの属性$mode$の値割当である．また，aはI，U，DI，DU，mが定めるループディストリビューションを適用して得られる，ループネスト群の実行時間評価であり，評価の際はクラス分け未定のデータ依存枝／データアクセス枝はないものとして扱う．

アルゴリズム：提案アルゴリズムを以下に示す．

1. 初期解$(\phi,CE_d,\phi,DE,m_0,a_0)$をリスト$lis$に挿入する．但し，$m_0$は「C-nodeの属性」の項で述べた通りに初期化された属性$mode$の値割当，a_0はデータ依存枝，データアクセス枝のクラス分けが全く定まっていない状態で定められるループディストリビューションを適用して得られる，ループネスト群の実行時間評価である．
2. lisより，a最小の暫定解(I,U,DI,DU,m,a)を取り出す．
3. $U=\phi$ならばアルゴリズム終了．I，DI，mが定めるループディストリビューションが最適のループネストの分割方法を与える．
4. クラス分け未定のデータ依存枝$ce=(cv_u,cv_v)\in U$を選択する．
5. ceがG_{DDG}の強連結部分に含まれる時8へ飛ぶ．
6. mにおいて$mode(cv_u)$の値がSまたはXならば，
   (a) mにおいて$mode(cv_v)$の値がSまたはXならば，$m':=m$とした後，m'において$mode(cv_u):=S$，$mode(cv_v):=S$，手続き$Inter$呼出．
   (b) mにおいて$mode(cv_v)$の値がPまたはXならば，$m':=m$とした後，m'において$mode(cv_u):=S$，$mode(cv_v):=P$，手続き$Inter$呼出．
7. mにおいて$mode(cv_u)$の値がPまたはXならば，
   (a) mにおいて$mode(cv_v)$の値がSまたはXならば，$m':=m$とした後，m'において$mode(cv_u):=P$，$mode(cv_v):=S$，手続き$Inter$呼出．
   (b) mにおいて$mode(cv_v)$の値がPまたはXならば，$m':=m$とした後，m'において$mode(cv_u):=P$，$mode(cv_v):=P$，手続き$Inter$呼出．
8. cv_uからcv_vへのデータ依存枝のみからなる全てのパス上の，データ依存枝の集合をBとする．

9. $ce \in B$かつ $ce \in CE_d - U - I$なるデータ依存枝が存在するならば，2へ戻る．

10. Bに属する各々のデータ依存枝 $ce' = (cv'_u, cv'_v) \in B$について以下の処理を施す．

    (a) mにおいて $mode(cv'_u)$ の値が Sまたは X，かつ $mode(cv'_v)$ の値が Sまたは X ならば，$m' := m$ とした後，m'において $mode(cv'_u) := S$，$mode(cv'_v) := S$，手続き $Intra$ 呼出．

    (b) mにおいて $mode(cv'_u)$ の値が Pまたは X，かつ $mode(cv'_v)$ の値が Pまたは X，かつ ce'がループによって運ばれるデータ依存の枝でないならば，$m' := m$ とした後，m'において $mode(cv'_u) := P$，$mode(cv'_v) := P$，手続き $Intra$ 呼出．

11. 2へ戻る．

手続き $Inter$では任意の $dv \in DV(ce)$ について以下の処理を施す．但し，C-nodecv_uと D-nodedvの間のデータアクセス枝を de_u，C-nodecv_vと D-nodedvの間のデータアクセス枝を de_vとする．

1. $DU' := DU - \{de_u, de_v\}$，$DI'_1 := \phi$，$DI'_2 := \phi$，$DI'_3 := \phi$．

2. もし $de_u \in DU$かつ $de_v \in DU$ならば，$DI'_1 := DI \cup \{de_a\}$，$DI'_2 := DI \cup \{de_v\}$，$DI'_3 := DI$．

3. もし $de_u \notin DU$かつ $de_v \in DU$ならば，

   (a) もし $de_u \in DI$ならば，$DI'_1 := DI$．

   (b) もし $de_u \notin DI$ならば，$DI'_2 := DI \cup \{de_v\}$，$DI'_3 := DI$．

4. もし $de_u \in DI$かつ $de_v \notin DU$ならば，

   (a) もし $de_v \in DI$ならば，$DI'_1 := DI \cup \{de_u\}$，$DI'_3 := DI$．

   (b) もし $de_v \notin DI$ならば，$DI'_2 := DI$．

5. もし DU'に dvと cv_uの間，または dvと cv_vの間のデータアクセス枝がなければ，$U' := U - \{ce\}$，$I' := I$．

6. $DI'_i \neq \phi$ならば，新しい暫定解 $(I', U', DI'_i, DU', m', a')$ を lis に挿入する $(i = 1, 2, 3)$．a'は I'，U'，DI'_i，DU'が定めるループディストリビューションにより得られるループネスト群の実行時間評価である．

手続き $Intra$ では，任意の $dv' \in DV(ce')$ について以下の処理を施す．但し，C-nodecv'_uと D-nodedv'の間のデータアクセス枝を de'_u，C-nodecv'_vと D-nodedv'の間のデータアクセス枝を de'_vとする．

1. $DU' := DU - \{de'_u, de'_v\}$，$DI'_4 := \phi$，$DI'_5 := \phi$．

2. もし $de'_u \in DU$かつ $de'_v \in DU$ならば，$DI'_4 := DI \cup \{de'_u, de'_v\}$，$DI'_5 := DI$．

3. もし $de'_u \notin DU$かつ $de'_v \in DU$ならば，

   (a) もし $de'_u \in DI$ならば，$DI'_4 := DI \cup \{de'_v\}$．

   (b) もし $de'_u \notin DI$ならば，$DI'_5 := DI$．

4. もし $de'_u \in DU$かつ $de'_v \notin DU$ならば，

   (a) もし $de'_v \in DI$ならば，$DI'_4 := DI \cup \{de_u\}$．

   (b) もし $de'_v \notin DI$ならば，$DI'_5 := DI$．

5. もし DU'に dv'と cv'_uの間，または dv'と cv'_vの間のデータアクセス枝がなければ，$U' := U - \{ce'\}$，$I' := I$．

6. $DI'_i \neq \phi$ならば，新しい暫定解 $(I', U', DI'_i, DU', m', a')$ を lis に挿入する $(i = 4, 5)$．a'は I'，U'，DI'_i，DU'が定めるループディストリビューションにより得られるループネスト群の実行時間評価である．

4 おわりに

本稿では，データ転送コストを考慮の上で，ループディストリビューションとデータ分割配置を同時に行うアルゴリズムを提案した．従来の一般的なループディストリビューションアルゴリズムは，ループネストは可能な限り細かく分割し，得られたループネストのうち並列化可能なものは全て並列化していた．これに対し提案アルゴリズムは，ループディストリビューションにより得られるループネスト群の実行時間が最小になるように，ループネストを分割・並列化する．

本稿では並列ループネストのイタレーション間のデータ分割配置は既成のものを用いるものとした．今後の研究課題は，並列ループネスト内のデータ分割配置もDPG上で行い，提案アルゴリズムの一部として組み込むことが挙げられる．

参考文献

[1] Hans Zima and Barbara Chapman, *Supercompilers for Parallel and Vector Computers*, Addison-Wesley, 1991.

[2] Tsuneo Nakanishi, Kazuki Joe, Hideki Saito, Akira Fukuda, and Keijiro Araki, "The CDP^2 Algorithm: A Combined Data and Program Partitioning Algorithm on the Data Partitioning Graph," *Proc. of the 1995 International Conference on Parallel Processing (ISBN 0-8493-2616-8)*, Vol.II, pp.177-181, CRC Press, August 1995.

[3] Tsuneo Nakanishi, Kazuki Joe, Akira Fukuda, Keijiro Araki, Hideki Saito, and Constantine D.Polychronopoulos, "The Data Partitioning Graph: Extending Data and Control Dependencies for Data Partitioning," *Proc. of the 7th International Workshop on Languages and Compilers for Parallel Computing*, Lecture Notes in Computer Science 892 (ISBN 3-540-58868-X), pp.170-185, Springer-Verlag, 1995.