A Direct Approach for Finding Loop Transformation Matrices

LIN Hua (林 华) and LŪ Mi (吕 媚)*
Electrical Engineering Department
Texas A&M University, College Station, TX 77843, U.S.A.
email: {lin1246,mliu}@ee.tamu.edu
Jesse Z. FANG (方之熊)
Hewlett-Packard Lab., P.O. Box 10490, Palo Alto, CA 94303, U.S.A.
e-mail: jfang@plg.bpl.hp.com
Received July, 1995.

Abstract

Loop transformations, such as loop interchange, reversal and skewing, have been unified under linear matrix transformations. A legal transformation matrix is usually generated based upon distance vectors or direction vectors. Unfortunately, for some nested loops, distance vectors may not be computable and direction vectors, on the other hand, may not contain useful information. We propose the use of linear equations or inequalities of distance vectors to approximate data dependence. This approach is advantageous since (1) many loops having no constant distance vectors have very simple equations of distance vectors; (2) these equations contain more information than direction vectors do, thus the chance of exploiting potential parallelism is improved.

In general, the equations or inequalities that approximate the data dependence of a given nested loop is not unique, hence classification is discussed for the purpose of loop transformation. Efficient algorithms are developed to generate all kinds of linear equations of distance vectors for a given nested loop. The issue of how to obtain a desired transformation matrix from those equations is also addressed.

1 Introduction

Loop transformations, such as loop interchange, reversal and skewing, have been proposed for automatically or manually parallelizing sequential nested loops as well as enhancing data locality in memory hierarchies[1–3]. The success of conducting these transformations relies on the collection of information on data dependence in nested loops. Data dependence is usually represented by either distance vectors or direction vectors. Distance vectors—which are defined by the difference of loop indices of dependent iteration—contain sufficient information for optimizing loop transformations; however, for many nested loops computing distance vectors is expensive if it is not impossible, especially when data dependence cannot be represented by finite number of distance vectors. On the other hand, as a compromise, direction vectors—which are defined by the direction of distance vectors—are usually used to approximate distance vectors; however, for many nested loops direction vectors may not contain useful information needed for loop transformation. We use the nested loop in Example 1.1 to illustrate this.

* This research was supported by the Texas Advanced Technology Program under Grant No.99903-165.
Example 1.1:

\[
\begin{align*}
do & I_1 = 0, n \\
do & I_2 = 0, n \\
do & I_3 = 0, n \\
S[I_1 + I_2 + I_3 + 1, I_1 + 2I_2 + 3I_3 + 1] = \cdots \\
\cdots = h(S[I_1 + I_2 + I_3, I_1 + 2I_2 + 3I_3]) \\
\end{align*}
\]

enddo
endo
endo
endo

The size of the iteration space of this triple loop is controlled by variable \( n \). We notice that loop-carried data dependence exists when \( n \geq 1 \). Suppose \( n = 4 \) for instance. We can find that iteration \((0, 2, -1)\) is \textit{flow dependent} on iteration \((0, 0, 0)\) since the data written into \( S[1, 1] \) in iteration \((0, 0, 0)\) is to be read in iteration \((0, 2, -1)\). By subtracting \((0, 0, 0)\) form \((0, 2, -1)\), we can get the corresponding distance vector \((0, 2, -1)\). There is another data dependence between iteration \((0, 4, 0)\) and iteration \((1, 0, 2)\). This is an anti-dependence because \( S[4, 8] \) read in iteration \((0, 4, 0)\) is to be updated in iteration \((1, 0, 2)\). The distance vector is \((1, -4, 2)\). Other data dependence can be found without any difficulty. It is further noticed that the number of distinct distance vector increase with \( n \); in fact, it is proportional to the size of the iteration space. Clearly, distance vectors are not capable to describe the dependence information in this situation. Now let’s look at the direction vectors. Direction vectors, which use symbols “<”, “>”, “=” or their combination, represent the direction of each component of distance vectors. For the above triple loop, \((=, <, >)\) and \((<, >, <)\) are two direction vectors among others — the former is due to distance \((0, 2, -1)\) and the latter is due to distance \((1, -4, 2)\). Obviously, loop \( I_2 \) cannot be parallelized because of the existence of direction vector \((<, >, <)\) which shows the data dependence carried by loop \( I_2 \). We like to know if transformations like loop skewing and loop interchange can be applied to such triple loop that the middle loop of the transformed code can be parallelized. Direction vectors in this case will provide a negative answer because of \((=, <, >)\) and \((<, >, <)\). Nevertheless, our desired transformation does exist. A further study on this nested loop can show that for any distance vector \( d \equiv (d_1, d_2, d_3) \) slope \( d_3 \) is greater than or equal to \(-4\) regardless of the value of \( n \). The theory in [2] tells us that the nested loop can be parallelized through transformation

\[
\left(\begin{array}{c} I_1' \\ I_2' \\ I_3' \end{array}\right) = \left(\begin{array}{ccc} 5 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{array}\right) \left(\begin{array}{c} I_1 \\ I_2 \\ I_3 \end{array}\right)
\]

The parallelized nested loop is as follows:

\[
\begin{align*}
do & I' = 0, 6n \\
doall & I' = \cdots \\
do & I_3 = \cdots \\
S[I_1 + 4I_2 + 3I_3 + 1, 2I_1 - 9I_2 + 3I_3 + 1] = \cdots \\
\cdots = h(S[I_1 + 4I_2 + 3I_3, 2I_1 - 9I_2 + 3I_3]) \\
\end{align*}
\]

enddo
enddoall
endo
endo
endo

Even more coarse parallelism can be achieved through such transformation. We will show, later in this paper, that the following transformation

\[
\left(\begin{array}{c} I_1 \\ I_2 \\ I_3 \end{array}\right) = \left(\begin{array}{ccc} 0 & 2 & 3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{array}\right) \left(\begin{array}{c} I_1' \\ I_2' \\ I_3' \end{array}\right)
\]

can transform the triple loop into the following form which has no dependence carried by the outermost loop.

\[
\begin{align*}
doall & I_1 = \cdots \\
do & I_2 = \cdots \\
do & \cdots \\
S[(I_1' + 4I_2' + 3I_3')/3 + 1, I_1' + I_2' - I_3' + 1] = \cdots \\
\cdots = h(S[I_1' + 4I_2' + 3I_3', I_1' + I_2' - I_3']) \\
\end{align*}
\]

enddo
enddoall
endo
endo

It is worth pointing out that here the failure of recognizing parallelism based on direction vectors is not due to the lack of algorithms, but rather due to that the direction vectors do not contain necessary information.

To deal with this situation, we propose a new approach to approximate data dependence, using linear equations or inequalities of distance vectors. This approach is advantageous in the following sense. First, for many loops whose distance vectors are difficult to obtain, computing equations or inequalities of distance vectors is easy. Second, the information that is important to transformation design is contained in those equations, and moreover, they are easy to obtain. To show this, let us continue the discussion of Example 1.1. It can be shown that any distance vector of this loop satisfies either \( 2d_1 + d_2 - 2 = 0 \) or \( 2d_1 + d_2 + 2 = 0 \), despite that there is no way of enumerating each individual. From these equations, one can easily see that the slope \( \frac{d_1}{d_2} \), where \( d_1 \neq 0 \), is greater or equal to \(-4\), which immediately results in the first transformation shown previously. It can be seen that any distance vector also satisfies equation \( 2d_1 - 2d_2 = 0 \). This property of distance vectors of the loop results in the second transformation shown previously. It is quite obvious that there are loops whose distance vectors do not satisfy linear equations or inequalities, and our approach may not work for those loops.

In complying with this approach, we discuss two important issues in this paper. First, how to determine if the distance vectors of a loop can be approximated by linear equations or inequalities, and how to generate those linear systems. Second, how to design a desired transformation based on linear system of distance vectors.

The rest of this paper is organized as follows. Section 2 discusses program model and some background. The idea of approximating data dependence with linear systems of distance vectors is introduced in Section 3. Section 4 focuses on how to generate all kinds of linear systems of dependence from index functions, ignoring iteration space. Section 5 discusses the loop transformation based on the information provided by linear systems. In Section 6 we continue the discussion of linear systems, considering both index functions and iteration space. The conclusion is given in Section 7.
2 Preliminaries

2.1 Model of Nested Loops

Although it is not a necessary condition for applying our approach, we restrict our attention to perfect nested sequential loops for simplicity. An r-deep nested loop with m-dimensional array references has the following general structure:

```plaintext
do I1 = L1(I), U1(I)
do I2 = L2(I1, I), U2(I1, I)
  ... 
  do I = LI(I1, ... , Ir-1, I), UI(I1, ... , Ir-1, I)
  A(f1(I), ... , fm(I)) = ...
  ... = A(g1(I), ... , gm(I))
  enddo
endo
doendo
```

Here, \( \bar{I} \) is the loop variable and \( \bar{n} \) the loop-independent variable. As functions of \( \bar{I} \) and \( \bar{n} \), \( L_k \)'s and \( U_k \)'s are the lower and upper limits. The iteration space, denoted as \( R \), is defined by \( L_k \) and \( U_k \), and its size is controlled by \( I \) and \( n \). \( f_k(\bar{I}) \)'s and \( g_k(\bar{I}) \)'s are referred to as index functions. We assume that index functions are linear and that their coefficients are loop-invariant constant. Certain types of nested loops which do not meet this requirement may be converted into this class. For example, when the indices of a nested loop with multi-dimensional array are linearized, the coefficients could be a function of variables \( \bar{n} \). This type of nests can be delinearized to meet our assumption\(^4\).

2.2 Data Dependence

When a memory location is shared by two data references, data dependence occurs which imposes a partial order on the execution of iterations. Let iterations \( i \) and \( i' \) be two iterations in \( R \). They are considered being dependent one on the other if \( f_k(i) = g_k(i') \) holds for every \( i \), \( 1 \leq k \leq r \); otherwise, they are independent. Further, data dependence can be classified into several types, and we only concern two types of data dependence in this paper: flow dependence and anti-dependence. Iteration \( i \) is flow dependent on iteration \( i' \) if iteration \( i \) precedes iteration \( i' \), that is \( i' \succ i \); anti-dependence occurs if iteration \( i \) succeeds iteration \( i' \), that is \( i \succ i' \).

Traditionally, information about data dependence is approximated by the set of distance vectors, denoted as \( D \). For iterations, say \( i \) and \( i' \) with \( i' \succ i \), \( d \) is flow dependent on \( i \) and the distance vector \( d \) is defined as \( d = i' - i \). It is important to note that distance vectors are always lexicographically positive. As mentioned, for loops, the number of distinct distance vectors is finite; however, for other loops this number is a function of the size of iteration space. In this paper, we use \( W_k \)'s, \( 1 \leq k \leq r \) to represent a partition of \( D \), the set of distance vectors such that any \( d \in D \) with \( d_j \neq 0 \) and \( d_j = 0 \) for \( 1 \leq j \leq k - 1 \) belongs

---

\(^4\) A vector \( \bar{x} \) is lexicographically positive, denoted as \( \bar{x} \succ 0 \), if there exists \( k \) such that \( x_k > 0 \) and \( x_j \geq 0 \) for any \( 1 \leq j \leq k - 1 \). Vector \( \bar{y} \) is lexicographically greater than vector \( \bar{x} \), denoted as \( \bar{y} \succ \bar{x} \), if \( \bar{y} - \bar{x} \) is lexicographically positive.

---

No. 5 A Direct Approach for Finding Loop Transformation Matrices 241
to \( W_k \). It is possible that some \( W_k \)'s might be empty. Since a distance vector in \( W_k \) must be lexicographically positive, the following fact is trivial.

**Observation 2.1.** For any \( d \) in \( W_k \), \( d_k \succ 0 \).

As mentioned before, the distance vectors of a nested loop could be incomputable in which case direction vectors are usually used to further approximate data dependence. Suppose \( i \) and \( i' \) are two iterations with \( i' \succ i \), the direction vector, \( d_k \), corresponding to \( i \) and \( i' \) is defined as:

\[
d_k = \begin{cases} 
  \langle \langle, \rangle \rangle & \text{if } i_k < i'_k \\
  \langle =, \rangle \rangle & \text{if } i_k = i'_k \\
  \langle >, \rangle \rangle & \text{if } i_k > i'_k 
\end{cases}
\]

Obviously, the number of direction vectors is always finite, and each vector approximates a subset of distance vectors. The drawback of direction vectors is that they may lose useful information for parallelization as seen in Example 1.1.

2.3 Linear Loop Transformations

A nested loop can be converted into a new nested loop by conducting linear transformations over iteration space, without changing the original data dependency. Let \( T \) be the transformation matrix for a given nested loop. Each iteration \( i \) in the new iteration space \( R' \) is a mapping from an iteration \( \bar{i} \) in \( R \), that is \( i = \bar{i} T \). The index functions of the new nested loop are therefore obtained from the original index functions by substituting \( \bar{I} \) with \( \bar{I} T^{-1} \), where \( T^{-1} \) is the inverse of \( T \) and \( \bar{I} \) the loop index variables of the new nested loop.

Readers may refer to \([2, 3]\) and \([1]\) for more details, including how to form new iteration space \( R' \) given \( T \). Several powerful loop transformation approaches, like loop interchange, reversal, shewing and any combination of them, are special cases of such linear transformations. In fact, each of these three transformations can be viewed as an elementary matrix transformation, and the combination of them is viewed as a series products of elementary matrices. For an \( r \)-deep nested loop, the transformation matrix must be a matrix of size \( r \times r \).

To be a legal transformation, i.e., the new nested loop reserves original data dependency, transformation matrix must satisfy the following condition:

**Lemma 2.2**. Let \( T \) be the matrix corresponding to a legal loop transformation on a nested loop with dependence vector set \( D \). Then for any \( d \) in \( D \), \( dT \) is lexicographically positive.

2.4 Loop Parallelization

If the iterations of a loop can be executed simultaneously without violating the partial order imposed by data dependence, the loop of a nested loop is parallelizable. It is a well-known fact that a loop is parallelizable if and only if there is no dependence carried by that loop. It is not hard to see that data dependence carried by loop \( I_k \) exists only if there is a distance vector with form \( (0, 0, \ldots, d_k, d_{k+1}, \ldots, d_r) \) where \( d_k \) is the \( k \)-th component of the vector and \( d_k \neq 0 \).

**Lemma 2.2.** For an \( r \)-deep nested loop, its \( k \)-th loop \( I_k \) is parallelizable if \( W_k \) is empty.

3 Approximating Data Dependence with Linear Systems

A linear system consists of equations and inequalities. Let \( D_T \) and \( D_k \) be the sets of distance vectors of flow-dependence and distance vectors of anti-dependence, respectively,
of a nested loop, L. We say that the dependence of nested loop L can be approximated by linear systems $S_1$ and linear system $S_2$, if any $d_f \in D_f$ satisfies $S_1$ and any $d_s \in D_s$ satisfies $S_2$. Consider the nested loop in Example 3.1. Both flow-dependence and anti-dependence exist when $n > 1$. Suppose iteration $(i_1', i_2')$ and iteration $(i_1, i_2)$ are dependent one on the other. The indices must satisfy equation $i_1 + i_2 + 1 = i_1' + i_2'$. If $(i_1', i_2')$ is flow dependent on $(i_1, i_2)$, then distance $d = (d_1, d_2)$, where $d_1 = i_1' - i_1$ and $d_2 = i_2' - i_2$, satisfies $d_1 + d_2 - 1 = 0$. That is, the flow-dependence of this double loop can be approximated by system $S_1: d_1 + d_2 - 1 = 0$. Likewise, if $(i_1, i_2)$ is anti-dependent on $(i_1', i_2')$, then the distance $d = i_1 - i_1'$ and $d_2 = i_2 - i_2'$ satisfy $d_1 + d_2 + 1 = 0$. Thus, the anti-dependence can be approximated by system $S_2: d_1 + d_2 + 1 = 0$.

**Example 3.1:**

```plaintext
do i_1 = 1, n
   do i_2 = 1, n
      A(l_1 + 2l_2 + 1) = A(l_1 + 2l_2) + ...
   enddo
endo
```

It is important to notice that for a given nested loop, linear systems that can approximate the data dependence are usually not unique. Take Example 3.2 and consider the following equations

$$d_1 + 2d_2 + 5d_3 - 1 = 0$$

or

$$d_1 + d_2 + 3d_3 - 1 = 0$$
$$2d_1 - d_2 - 2 = 0$$
$$d_2 + 3d_3 = 0$$

where the first equation is obtained from the index functions in the first subscript position, the second one is due to the index functions in the second subscript position, the third and the fourth ones are linear combinations of the previous two. It is quite clear that any of these equations approximate the flow-dependence since any distance vectors of flow dependence must satisfy them. Identifying property of linear systems has a direct impact on loop transformations. Further discussion on this matter is included in Sections 4 and 5.

**Example 3.2:**

```plaintext
do i_1 = 1, n
   do i_2 = 1, n
      do i_3 = 1, n
         A(l_1 + 2l_2 + 5l_3 + 1, l_1 + l_2 + 3l_3 + 1) = ...
         ...
         A(l_1 + 2l_2 + 5l_3, l_1 + l_2 + 3l_3 + 1) = ...
      enddo
   enddo
endo
```

A Property of Linear System of Dependence. Suppose both flow-dependence and anti-dependence exist in a nested loop. If the flow-dependence can be approximated by linear system $\sum_{k=1}^{r} c_k d_k + c_0 = 0$, then the anti-dependence can be approximated by linear system $\sum_{k=1}^{r} c_k d_k - c_0 = 0$.

### 4 Linear Systems of Dependence from Index Functions

To find linear system of dependence in general, both index functions and iteration space should be considered. We focus on index functions in this section; in the next section, index functions as well as iteration space will be considered.

#### 4.1 A Sufficient Condition

In this subsection, we give the condition which can be used to determine if the distance vectors of a nested loop can be approximated by linear systems. We consider an r-depth perfectly nested loop with m-dimensional array reference:

$$S : S[f_1, \ldots, f_m] = \ldots$$
$$T : \ldots = h(S[g_1, \ldots, g_m])$$

where

$$f_k(\vec{i}) = a_k,0 + \sum_{j=1}^{r} a_{k,j} i_j$$
$$g_k(\vec{i}) = b_k,0 + \sum_{j=1}^{r} b_{k,j} i_j$$

for $1 \leq k \leq m$. We assume that the coefficients $a_{k,j}$'s and $b_{k,j}$'s are loop-invariant variables, and we also assume that $f_k(\vec{i})$ and $g_k(\vec{i})$ are not constant. An r-deep nested loop of this type is denoted by $L = (f_k, g_k, r)$. By $F$ we denote the matrix of coefficients as follows:

$$F =
\begin{pmatrix}
a_{1,1} - b_{1,1} & a_{1,2} - b_{1,2} & \cdots & a_{1,r} - b_{1,r} \\
a_{2,1} - b_{2,1} & a_{2,2} - b_{2,2} & \cdots & a_{2,r} - b_{2,r} \\
\vdots & \vdots & \ddots & \vdots \\
a_{m,1} - b_{m,1} & a_{m,2} - b_{m,2} & \cdots & a_{m,r} - b_{m,r}
\end{pmatrix}$$

**Theorem 4.1.** If $\text{Rank}(F) \leq m - 1$, where $\text{Rank}(F)$ is the rank of matrix $F$, then distance vectors of a nested loop described above can be approximated by linear system

$$\sum_{k=1}^{r} c_k d_k + c_0 = 0$$

or linear system

$$\sum_{k=1}^{r} c_k d_k - c_0 = 0$$

(Note that this condition is not necessary.)

**Proof.** When $\text{Rank}(F) \leq m - 1$, there exists a nonzero vector $\vec{n} = (n_1, n_2, \ldots, n_m)$ satisfying $\vec{n}^T F = \vec{0}$. Therefore, we have

$$\sum_{k=1}^{m} n_k a_{k,j} = \sum_{k=1}^{m} n_k b_{k,j}, \quad \text{for } 1 \leq j \leq r.$$  

Let $d$ be an arbitrary distance vector in distance set $D$, and iterations $\vec{i}$ and $\vec{j}$ be a pair of dependent iterations represented by $d$. They must satisfy $f_k(\vec{i}) = g_k(\vec{j})$, for $1 \leq k \leq m$ by definition. That is,

$$a_{k,0} + \sum_{j=1}^{r} a_{k,j} i_j = b_{k,0} + \sum_{j=1}^{r} b_{k,j} j_j.$$
for 1 ≤ k ≤ m. By multiplying $n_k$ on both sides of
\[ a_{k,0} + \sum_{j=1}^{r} a_{k,j}i_j = b_{k,0} + \sum_{j=1}^{r} b_{k,j}i_j, \]

for 1 ≤ k ≤ m, and then adding them together, we obtain
\[ \sum_{k=1}^{m} n_k a_{k,1}i_1 + \cdots + \sum_{k=1}^{m} n_k a_{k,i_r}i^r + \sum_{k=1}^{m} n_k a_{k,0} = \sum_{k=1}^{m} n_k b_{k,1}i_1 + \cdots + \sum_{k=1}^{m} n_k b_{k,i_r}i^r + \sum_{k=1}^{m} n_k b_{k,0}. \]

Since
\[ \sum_{k=1}^{m} n_k a_{k,0} = \sum_{k=1}^{m} n_k b_{k,0}, \]
the above equation is equivalent to
\[ \sum_{k=1}^{m} n_k a_{k,1}(i_1 - i_1') + \cdots + \sum_{k=1}^{m} n_k a_{k,i_r}(i_r - i_r') + \sum_{k=1}^{m} n_k a_{k,0} - \sum_{k=1}^{m} n_k b_{k,0} = 0. \]

Remember that $\vec{d}$ could be a flow dependence or an anti-flow dependence. If the former is the case, that is, $\vec{d} = \vec{i} - \vec{i}'$, then we have
\[ \sum_{k=1}^{m} n_k a_{k,1}d_1 + \cdots + \sum_{k=1}^{m} n_k a_{k,i_r}d_r - \sum_{k=1}^{m} n_k a_{k,0} = 0. \]

If the latter is the case, that is, $\vec{d} = \vec{i}' - \vec{i}$, then we have
\[ \sum_{k=1}^{m} n_k a_{k,1}d_1 + \cdots + \sum_{k=1}^{m} n_k a_{k,i_r}d_r + \sum_{k=1}^{m} n_k a_{k,0} - \sum_{k=1}^{m} n_k b_{k,0} = 0. \]

4.2 Generating Linear Systems Using Index Functions

As seen before, in general linear systems of dependence is not unique for a given nested loop. We define two classes of linear systems below. Linear systems of Type I are those whose offset $c_0$'s are nonzero, and linear systems of Type II are those whose offset $c_0$'s equal to zero. It is possible that for a given nested loop, there are several linear systems of the same type that approximate the data dependence. In the loop of Example 3.1, for instance, we have already seen three linear systems of Type I. They are $d_1 + 2d_2 + 3d_3 = 0$, $d_1 + d_2 + 2d_3 = 0$, and $2d_1 + d_2 - 2 = 0$. From transformation point of view, all of them are different, because different linear systems produce different transformations. However, in some applications the difference between the third one and the first two is more significant than that between the first two. For example, the transformation that makes the innermost loop parallelizable can be obtained from either of the first two systems, while the transformation that makes the middle loop parallelizable can be obtained from the third system (more details can be seen later). The other way around does not hold. We refer a linear equation of Type I (or Type II) to as of form $I_4$ (or of form $I_2$ if $c_2 \neq 0$ and $c_1 = 0$), for $q+1 \leq j \leq r$. In this section, we are interested in the following problem: Given an $r$-deep nested loop with $\text{rank}(F) \leq m - 1$, whether the data dependence can be approximated by a linear system of certain form? If the answer is yes, then what is that system?

No. 3

We shall develop a technique to deal with this problem based on manipulating the coefficients of index function. Let $L = (\vec{i}, \vec{g}(\vec{i}), r)$ be an $r$-deep nested loop, where $f_k(\vec{i}) = a_{k,0} + \sum_{j=1}^{r} a_{k,j}i_j$ and $g_k(\vec{i}) = b_{k,0} + \sum_{j=1}^{r} b_{k,j}i_j$. Matrix $F_k$ is defined as follows:
\[ F_k = (F, A_r, A_{r-1}, \ldots, A_k) \]

where
\[ A_j = (a_{1,j}, a_{2,j}, \ldots, a_{m,j})^t. \]

As an example, the coefficient matrix $F_1$ of the loop in Example 4.1 is given as follows:
\[ F_1 = \begin{pmatrix} 1 & 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 & 2 \\ 3 & 1 & 1 & 1 & 2 \end{pmatrix}. \]

Example 4.1.

1. $I_1 = 0, n - 1$
2. $I_2 = 0, n - 1$
3. $I_3 = 0, n - 1$
4. $I_4 = 0, n - 1$

\[ A[2I_1 + I_2 + 2I_3 + 2, 2I_1 + 2I_2 + 2I_3 + 1, 3I_1 + 2I_2 + I_3] = \cdots = h(A[I_1 + I_3, I_1 + I_3 + I_1, 2I_1 + I_3]) \]

endo

We now give the conditions of the existence of linear systems of Type I and Type II.

Theorem 4.2. For an $r$-deep nested loop with $m$-dimensional array reference, the data dependence can be approximated by a linear system of Type I if \( \text{rank}(F) < \text{rank}(F, W) \), where $W = (a_{1,0} - b_{1,0}, a_{2,0} - b_{2,0}, \ldots, a_{m,0} - b_{m,0})^t$; the data dependence can be approximated by a linear system of Type II if \( \text{rank}(F, W) < m \).

To prove this we need a theorem of linear algebra.

Lemma 4.1. Let $P$ be an $m \times m$ matrix with rank($P$) < $m$, and let $Q = (P, A)$ be another matrix where $A$ is an $m \times 1$ matrix. There exists a 1 x m row vector $\vec{x}$ such that $\vec{x}P = 0$ and $\vec{x}Q \neq 0$ if and only if rank($P$) < rank($Q$).

Proof of Theorem 4.2. According to the proof of Theorem 4.1, it is not hard to see that if there exists $n$ such that $nP = 0$ and $nQ \neq 0$, then linear systems of Type I exist. This condition is equivalent to $n^tP = 0$ and $n^t(F, W) \neq 0$. According to Lemma 4.1, this is equivalent to $\text{rank}(F) < \text{rank}(F, W)$.

Consider the loop in Example 4.1 for instance. Since $F = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ and $(F, W) = \begin{pmatrix} 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$, rank($F$) = 1 and rank($F, W$) = 2. Applying Theorem 4.2, we know that linear systems of both types can be used to approximate the data dependence of the triple loop.

We now give the condition for checking if a specific form of linear system exists.

Theorem 4.3. For a 4-deep nested loop with $m$-dimensional array reference, the data dependence can be approximated by a linear system of form $I_4$, $1 \leq q \leq r$, if rank($F_{q+1}$) <
rank($F_{q}$) and rank($F_{q+1}$) < rank($(F_{q+1}, W)$); the data dependence can be approximated by a linear system of form $I_{q}$, 1 ≤ q ≤ r, if rank($F_{q+1}$) < rank($F_{q}$) and rank($F_{q+1}$) = rank($(F_{q+1}, W)$).

Proof. According to the proof of Theorem 4.1, to prove the existence of a linear system of certain form, say form $I_{q}$, we need to show the existence of a vector $\vec{n}$ such that

\[ n_{i}^2 F = \vec{0}, \]
\[ n_{i}^2 V \neq \vec{0}, \]  \hspace{1cm} (2)
\[ n_{i}^2 A_{j} = \vec{0}, \text{ for } q + 1 \leq j \leq r \] \hspace{1cm} (3)
\[ n_{i}^2 A_{q} \neq \vec{0}, \] \hspace{1cm} (4)

where

\[ A_{i} = (a_{1,i}, a_{2,i}, \ldots, a_{m,i})^{T}. \]

Eq.(1) guarantees the existence of linear systems; Eq.(2) guarantees the linear systems are of Type I; Eqs (3) and (4) further assure the linear systems are of Type $I_{q}$. Eqs.(1), (3) and (4) are equivalent to

\[ n_{i}^2(F_{q}, A_{r}, A_{r-1}, \ldots, A_{q+1}) = \vec{0}, \]
\[ n_{i}^2(F_{q}, A_{r}, A_{r-1}, \ldots, A_{q+1}, A_{q}) \neq \vec{0}. \]

Notice that $F_{q+1} = (F_{q}, A_{r}, A_{r-1}, \ldots, A_{q+1})$ and $F_{q} = (F_{q}, A_{r}, A_{r-1}, \ldots, A_{q+1}, A_{q})$. According to Lemma 4.1, the above equations hold if and only if rank($F_{q+1}$) < rank($F_{q}$). Similarly, Eqs.(1), (4) are equivalent to $n_{i}^2 F_{q+1} = 0$ and $n_{i}^2 F_{q+1} = W \neq \vec{0}$. These two equations will be held if and only if rank($F_{q+1}$) < rank($(F_{q+1}, W)$). Thus, we have proved the first half of the theorem; the second half can be handled similarly.

The Gauss elimination algorithm can be used to compute the rank of matrices and to compute vector $\vec{n}$ as well. We shall not give the Gauss elimination algorithm here. Readers can find it in any textbook of numerical algorithms. Let us use Example 4.1 to show the process. Suppose we want to know if the data dependence can be approximated by linear system of form $I_{r}$. According to Theorem 4.2, we need to check where rank($F$) < rank($F_{q}$) and rank($F$) < rank($(F, W)$); if these conditions hold we also need to compute $\vec{n}$ such that $n_{i}^2 F = 0$, $n_{i}^2 F_{q} \neq 0$, and $n_{i}^2 V \neq 0$. One can notice that in the following process these two jobs are to be done in one pass. We start with a unit matrix $U$, coefficient matrix $F_{3}$ and $V$.

\[ U = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad F_{3} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix}, \quad V = \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}. \]

Now, use the first row to cancel the rest of entries in the first column of $F_{3}$.

\[ U \leftarrow \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 1 \\ -1 & 0 & 1 \end{pmatrix}, \quad F_{3} \leftarrow \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \quad V \leftarrow \begin{pmatrix} 2 \\ -1 \\ -2 \end{pmatrix}. \]

Interchange row 2 and row 3,

\[ U \leftarrow \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 1 \\ -1 & 0 & 1 \end{pmatrix}, \quad F_{3} \leftarrow \begin{pmatrix} 1 & 1 & 2 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \quad V \leftarrow \begin{pmatrix} 2 \\ -2 \\ -1 \end{pmatrix}. \]

Notice that $F_{3}$ is diagonalized through so called elementary row operations. From these matrices we see that rank($F$) = 1, rank($F_{3}$) = 2 and rank($(F, W)$) = 2; thus linear system of form $I_{3}$ exists. Row 2 of $U$ forms $n_{1}$, that is, $n_{1} = (-1, 0, 1)$. Since

\[ A_{1} = \begin{pmatrix} 2 \\ 2 \\ 3 \end{pmatrix}, \quad A_{2} = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix}, \quad A_{2} = \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}, \]

we have

\[ c_{1} = n_{1}^2 A_{1} = 1, \quad c_{2} = n_{1}^2 A_{2} = 1, \quad c_{3} = n_{1}^2 A_{3} = -1, \quad c_{0} = n_{1}^2 V = -2. \]

Hence, the desired linear system is $d_{1} + d_{2} - d_{3} + 2 = 0$ for flow dependence and $d_{1} + d_{2} - d_{3} - 2 = 0$ for anti-dependence.

The time complexity for the above process on a matrix of size $m \times r$ is $O(m^2 r)$. The largest matrix among those coefficient matrices is $(F_{1}, V)$, which has a size of $m \times (3r + 1)$. Therefore, checking the existence of a specific form of linear system and computing the corresponding coefficients take time $O(m^2 r^2)$.

Until now we are dealing with generating only one specific form of linear system. In applications, we may want to know all possible form of linear systems. That is, we want to know, given an $r$-deep nested loop, if linear systems with forms of $I_{1}, I_{2}, \ldots, I_{r}, I_{1}, I_{2}, \ldots, I_{r}, I_{1}, I_{2}, \ldots, I_{r}$, exist. One way to do this is to repeat the above process $2r$ times. It will take $O(m^2 r^2)$ time. Here, we present a more elegant way, which takes one pass to identify all forms of $I$ or all forms of $H$.

Let $F_{1}^{r}$ be the diagonalized form of $F_{1}$ obtained by executing elementary row operations; let $U^{r}$ and $V^{r}$ be the matrices obtained after executing those elementary operations which diagonalize $F_{1}$. Any nonzero entry of $F_{1}^{r}$ that is to the right of zero entries is called a breaking entry. In the previous example, $F_{1}^{r}(2, 4)$ and $F_{1}^{r}(3, 5)$ are breaking entries. Notice that no two breaking entries are in the same column. By the $k$-th breaking entry, we refer to the $k$-th breaking entry counted from left to right. We define two functions $Pos(F_{1}^{r})$ and $Pos(W^{r})$. $Pos(F_{1}^{r})$ returns a vector whose $k$-th component is $Pos(F_{1}^{r})(k) = (j_{1}, j_{2})$, where $j_{1}$ and $j_{2}$ are the indices of the $k$-th breaking entry. Function $Pos(W^{r})$ returns a scalar which is $Pos(W^{r}) = j$ such that $W^{r}(j) \neq 0$ and $W^{r}(j) = 0$ for $j_{1} < p \leq m$.

Suppose $L$ is an $r$-deep nested loop with $m$-dimensional array reference. Let $Pos(F_{1}^{r})(k)$ be any component of $Pos(F_{1}^{r})$ and let $Pos(F_{1}^{r})(k) = (p, q)$.

1. If $q > r$ and $p \leq Pos(W^{r})$ then the data dependence of $L$ can be approximated by linear systems of Form $I_{2r-q-1}$; the coefficients of $(c_{1}, \ldots, c_{2r-q-1})$ of the system are the product of $u_{p}^{r}(A_{1}, \ldots, A_{2r-q-1})$, where $u_{p}^{r}$ is the $p$-th row vector of $U^{r}$; the coefficient $c_{0}$ is the product of $u_{p}^{r}(A_{0} + B_{0})$.

2. If $q > r$ and $p > Pos(W^{r})$ then the data dependence of $L$ can be approximated by linear systems of Form $I_{2r-q-1}$; the coefficients of $(c_{1}, \ldots, c_{2r-q-1})$ of the system are the product of $u_{p}^{r}(A_{1}, \ldots, A_{2r-q-1})$, where $u_{p}^{r}$ is the $p$-th row vector of $U^{r}$.

As a summary, we present below the algorithm that finds all forms of linear systems for a given nested loop.
Algorithm: Finding All Forms of Linear Systems of Dependence

Input: coefficient matrices $A$ and $B$ of index functions of an $r$-deep nested loop with $m$-dimensional array reference.

Output: coefficients of all forms of linear systems that approximate the nested loop.

1) Constructing $F_1$ and $V$ and $U$;
2) Using elementary row operation to diagonalize $F_1$, and obtain $F'_1$, $V'$ and $U'$;
3) Computing functions $\text{Pos}(F'_1)$ and $\text{Pos}(W')$;
4) For every component in $\text{Pos}(F'_1)$ do the following.
5) *Let $(p, q)$ be the value of the component $*$

4.1) If $q > r$ and $p \leq \text{Pos}(W')$ then compute $(c_1, \ldots, c_{2r-q-1}) = u_0'(A_1, \ldots, A_{2r-q-1})$; compute $c_0 = u_0'(B_0)$;

4.2) If $q > r$ and $p > \text{Pos}(W')$ then compute $(c_1, \ldots, c_{2r-q-1}) = u_0'(A_1, \ldots, A_{2r-q-1})$.

As an application of this algorithm, we consider the loop in Example 4.1 again. We start with the unit matrix $U$, coefficient matrix $F_1$ and $V$:

$$U = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad F_1 = \begin{pmatrix} 1 & 1 & 1 & 2 & 1 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 3 \end{pmatrix}, \quad V = \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}.$$

After diagonalizing $F_1$ using row elementary operations, we have

$$U' \leftarrow \begin{pmatrix} 1 & 0 & 0 \\ -1 & 0 & 1 \\ -1 & 1 & 0 \end{pmatrix}, \quad F'_1 \leftarrow \begin{pmatrix} 1 & 1 & 1 & 2 & 1 & 2 \\ 0 & 0 & 0 & -1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad V' \leftarrow \begin{pmatrix} 2 \\ -2 \\ -1 \end{pmatrix}.$$

By scanning matrix $F'_1$, we obtain

$$\text{Pos}(F'_1) = ((2, 4), (3, 5))$$

and

$$\text{Pos}(W') = 3.$$

We first consider the component $(p, q) = (2, 4)$. Since $q > r$ and $p \leq \text{Pos}(W')$ hold, linear system of form $I_{2r-q-1}$ (i.e., $I_2$) exists. Notice that $u_0' = (-1, 0, 1)$. Hence,

$$c_1 = u_0' A_1 = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} = 1,$$

$$c_2 = u_0' B_1 = \begin{pmatrix} 0 \\ -1 \\ 2 \end{pmatrix} = 1,$$

$$c_3 = u_0' = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = 1,$$

$$c_0 = u_0' = \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} = 2.$$

The corresponding linear system are $S_f : d_1 + d_2 - d_3 + 2 = 0$ and $S_a : d_1 + d_2 - d_3 - 2 = 0$.

5 Finding Parallelism Through Linear Systems of Dependence

In this section we show how linear systems of dependence can be used to find parallelism in nested loop. Parallelism could be explicit, that is, some loops are parallelizable; it could be non-explicit, that is, parallelizable loop may be obtained through loop transformation. We shall show how linear systems of dependence can help identifying both. Though our attention is on exploring parallelism in nested loops, as mentioned before, loop transforms also have other important applications.

1) Innermost Loop Level Parallelism

For those machines which favor low level parallelism, like superscalar or VLIW, it is desirable that the innermost loop of a nested loop is parallelizable. The following theorem helps to recognize this type of loops.

**Lemma 5.1.** Let $L$ be an $r$-deep nested loop with $m$-dimensional array reference. The innermost loop of $L$ is parallelizable if the data dependence can be approximated by linear system of either form $I_{2r}$, where $q < r$, or form $I_{2r}$. Indeed, if the former is the case, one of the components of any distance vector $(d_1, d_2, \ldots, d_r)$ must be nonzero; thus, $W_{2r}$ is empty since $q > r$. If the latter is the case, any distance vector should not be in the form of $(0, 0, \ldots, 0, d_q)$, where $d_q \neq 0$; thus, $W_{2r}$ is empty. Combining this with the results described in the previous section, we have the follows.

**Corollary 5.1.** Let $L$ be an $r$-deep nested loop with $m$-dimensional array reference. The innermost loop of $L$ is parallelizable if $\text{rank}(F_2) < \text{rank}(F_r V)$ or $\text{rank}(F_r V) < \text{rank}(F_r)$. 
When the condition in Corollary 5.1 does not meet, parallelizable innermost loop may still be available through loop transformations.

Lemma 5.2. Let L be an r-depth nested loop with m-dimensional array reference. There exists a loop transformation T that produces a parallelizable innermost loop, if the data dependence can be approximated by linear systems of form \( I_r \).

Proof. Let \( S_T: \sum_{i=1}^{\alpha} c_i d_i + c_0 = 0 \) and \( S_v: \sum_{i=1}^{\alpha} c_i d_i - c_0 = 0 \) be the loop, where \( c_0 \neq 0 \). The following matrix \( T \) performs the desired transform:

\[
T = \begin{pmatrix}
U_{r-2} & 0 \\
0 & T_0
\end{pmatrix}
\]

where \( U_{r-2} \) is an \((r-2) \times (r-2)\) identity matrix, and \( T_0 = \begin{pmatrix} \mu & 1 \\ 1 & 0 \end{pmatrix} \) with \( \mu = \min(-c \frac{c_{\alpha-1}}{c_0}, n \frac{c_\rho}{c_0}, n \frac{c_{\rho-1}}{c_0}, m \frac{c_{\rho-2}}{c_0}) \). Indeed, due to \( U_{r-2} \), any distance vector \( d \in W_r \), for \( 1 \leq i \leq r-2 \), satisfies \( Td \succ 0 \). It is also clear that any \( d \in W_r \), say \((0, \ldots, 0, d_r)\), will be transformed through \( T \) into \((0, 0, \ldots, 0, d_r, 0)\) which is lexicographically positive. The rest is to show that for any \( d \in W_{r-1} \), \( Td \succ 0 \). We leave this to readers. Note that \( d = (0, \ldots, d_{r-1}, d_r) \) can be assumed. To show that \( Td \succ 0 \), one just need to show \( d_{r-1} \mu + d_r > 0 \).

Consider Example 4.1 again. We have already known from the preceding section the following linear systems \( S_T: d_1 - d_2 + 1 = 0 \) and \( S_v: d_1 - d_2 - 1 = 0 \). Hence, the transformation matrix is \( T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 0 \end{pmatrix} \). The transformed program is as follows.

Example 4.1 (continued):

\[
\begin{align*}
d_1 &= 0, n - 1 \\
d_2 &= 0, n - 1 \\
\text{do } d_3 &= 0, n - 1 \\
A[2d_1 + d_2 + 2d_3 + 2, 2d_1 + 2d_2 + 2d_3 + 1, 3d_1 + 2d_2 + 3] &= \ldots \\
\text{enddo} \\
\text{enddo}
\end{align*}
\]

Corollary 5.3. Let L be an r-depth nested loop with m-dimensional array reference. There exists a loop transformation T that produces a parallelizable innermost loop, if rank(F) < rank(F_\alpha) and rank(F) < rank((F_\alpha, V)).

The conclusion below follows Corollary 5.1 and Corollary 5.2 immediately.

Theorem 5.1. Let L be an r-depth nested loop with m-dimensional array reference. Parallelizable innermost loop is either available through loop transformation or not available if rank(F) < rank((F_\alpha, V)).

2) Parallelizing Outermost Loops

For MIMD machines, higher level parallelism is desirable since the synchronization overhead can be reduced. Exploring the coarse-grain parallelism — making the outermost loop parallelizable — is particularly interesting. It is known that a parallelizable outermost loop is readily available if there is no data dependence carried by loop \( F_0 \), or a parallelizable outermost loop can be obtained through transformation if the distance vectors of a nested loop do not span the entire iteration space[2,3].

No. 3 A Direct Approach for Finding Loop Transformation Matrices 251

Lemma 5.3. Let L be an r-depth nested loop with m-dimensional array reference. The outermost loop is parallelizable if rank(F) < rank(F_\alpha).

Indeed, when this condition holds, according to the results described in the previous section data dependence can be approximated by linear system of form \( I_r \), that is, \( d_1 = 0 \). This implies that there is no data dependence carried by loop 1.

When this condition does not hold, we check the spanning space of distance vectors. It is quite clear that if data dependence can be approximated by linear systems of Type II, i.e., \( c_1 d_1 + c_2 d_2 + \ldots + c_r d_r = 0 \), distance vectors span in a space with dimension less than \( r \). Applying Theorem 4.2, we have the following result.

Lemma 5.4. Let L be an r-depth nested loop with m-dimensional array reference. There is a transformation T that produces a parallelizable outermost loop, if rank(F, V) < m.

Both non-unimodular transformation matrix and unimodular transformation matrix should be available. The way to obtain a unimodular matrix has been introduced in [2,3]. Here we just give the construction of non-unimodular matrix. It is an important advantage of linear system presentation that a non-unimodular transformation matrix can be directly obtained based on the coefficients of a system.

Theorem 4.5. Let \( \sum_{j=1}^{q} c_j d_j = 0 \), where \( 1 \leq q \leq r \) and \( c_q \neq 0 \), be a system of Form 1 of nested loop. The \( r \times r \) transformation matrix

\[
\begin{pmatrix}
c_1 & c_2 & \cdots & c_{q-1} & c_q & 0 & 0 & \cdots & 0 & 0 \\
1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & 1 & 1 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 1 & 1
\end{pmatrix}
\]

produces a parallelizable outermost loop.

To prove the legitimacy of T, we need to verify that \( \vec{d} = Td \succ 0 \) for any \( \vec{d} \in D \). Indeed, for any distance vector \((d_1, d_2, \ldots, d_r)\) in \( W_r \), \( \vec{d} \in \text{the form of } (0, d_1, d_2, \ldots, d_r) \), which is lexicographically positive, since \( d_1 > 1 \). For any distance vector \((d_1, d_2, \ldots, d_r)\) in \( W_r \), \( 2 \leq j \leq q - 1 \), \( \vec{d} \in \text{the form of } (0, d_1, d_2, \ldots, d_r, d_{q-1}, \ldots, d_j) \), which is lexicographically positive since \( d_j > 0 \). Note that set \( W_q \) is empty. For any distance vector \((d_1, d_2, \ldots, d_r)\) in \( W_r \), \( q+1 \leq j \leq r \), \( \vec{d} \in \text{the form of } (0, d_1, d_2, \ldots, d_j, d_{j+1}, \ldots, d_r) \), which is also lexicographically positive since \( d_j > 0 \). Finally, we have already seen that the first component of transformed distance vector is zero. Hence, \( T \) produces a parallelizable outermost loop.

As an application, let us continue with Example 1.1. The coefficient matrix \((F, W)\) is

\[
\begin{pmatrix}
0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 2 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0
\end{pmatrix}
\]

and its rank is 1. Using the technique described in the preceding section, we have the linear system of dependence \( d_2 + 2d_4 = 0 \). Therefore, the desired transformation matrix is

\[
\begin{pmatrix}
0 & 1 & 2 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
0 & 0 & 1
\end{pmatrix}
\]

and the transformed program is as follows.

Example 1.1 (continued):

\[
\begin{align*}
d_1 &= 0, n - 1 \\
d_2 &= 0, n - 1 \\
\text{do } d_3 &= 0, n - 1 \\
\text{enddo} \\
\text{enddo}
\end{align*}
\]
6 Linear Systems of Dependence from Iteration Space and Index Functions as Well

To illustrate our idea about how to consider the effect of index functions and iteration space together, let us first study Example 6.1.

Example 6.1:

\[
\begin{align*}
\text{do } & I_1 = 0, n - 1 \\
\text{do } & I_2 = I_1 + 1, n - 1 \\
S[I_1, I_2] & = h(S[I_2, I_1])
\end{align*}
\]

endo
endo
endo

Since matrix \( F = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \) has a rank of 1, we can apply the technique developed in the previous section as follows. It is not hard to see that distance vectors, if there is any, can be approximated by linear system \( d_1 + d_2 = 0 \). Therefore, innermost loop level and outermost loop level parallelism can be explored. However, the parallelism is not limited to these two levels. In fact, each level of loops can be executed in a parallel fashion since there is no data dependence at all. To see this, information provided by iteration space must be considered.

Suppose \((i_1, i_2)\) and \((i'_1, i'_2)\) are two dependent iterations. Then they must satisfy \( i_1 = i'_2 \) and \( i_2 = i'_1 \). On the other hand, by looking at the functions of loop limits, we realize that the indices must also satisfy \( i_2 > i_1 \) and \( i'_2 > i'_1 \). Combining equation \( i_2 = i'_1 \) with inequality \( i_2 > i_1 \), we have \( i'_2 > i_1 > 0 \); combining the other two equations we have \( i'_2 > i'_1 > 0 \). These imply \( d_1 > 0 \) and \( d_2 > 0 \). Recall that we obtain \( d_1 + d_2 = 0 \) from index functions. We conclude from the inconsistency among these equations that there is no dependence in this double loop.

In the above example, beside the linear system obtained from index functions, we obtained two inequalities, \( d_1 > 0 \) and \( d_2 > 0 \), from the combination of index functions and iteration space. The inequalities of dependence provide additional information about data dependence. In order to obtain inequalities of dependence, a geometrical approach is adopted below. We view the iteration space as polytope, and view the index functions as hyperplanes. The inequalities of dependence is obtained by analyzing the geometry relation between the polytope and hyperplanes. For the sake of simplicity we discuss here one dimensional array reference only; the approach can be easily extended to multi-dimensional array.

Let \( f(\vec{I}) = \sum_{k=1}^{r} a_k l_k + a_0 \) and \( g(\vec{I}) = \sum_{k=1}^{r} b_k l_k + b_0 \) be two index functions of r-depth nested loop with iteration space \( R \). Since the loop limits are formed with linear functions of induction variables, the iteration space \( R \) defined by these functions is a polytope from a geometry point of view. A hyperplane is defined from index functions: \( f(\vec{I}) - g(\vec{I}) = 0 \), i.e., \( \sum_{k=1}^{r} (a_k - b_k) l_k + a_0 - b_0 = 0 \). We say that hyperplane \( f(\vec{I}) - g(\vec{I}) = 0 \) has no intersection with iteration space \( R \), if any instance of iterations in \( R \), say \( (i_1, \ldots, i_r) \), satisfies either

For the proof, the proof is quite straightforward. We only show that for Inequality (5) and (6); others can be handled similarly. Suppose flow dependence exists. Any pair of dependent iterations, say \( i \) and \( i' \), should satisfy

\[
\sum_{k=1}^{r} a_k i_k + a_0 = \sum_{k=1}^{r} b_k i'_k + b_0.
\]
On the other hand, since the hyperplane \( f(\bar{I}) - g(\bar{I}) = 0 \) is above \( R, \bar{i} \) and \( \bar{j} \) should further satisfy
\[
\sum_{k=1}^{r} (a_k - b_k)i_k + a_0 - b_0 > 0
\]
(14)
\[
\sum_{k=1}^{r} (a_k - b_k)j_k + a_0 - b_0 > 0
\]
(15)

Inequality (5) is obtained by combining Equation (13) with (14); inequality (6) is obtained by combining Equation (13) with (15).

6.1 Applications

Theorem 6.2. Suppose \( L \) is an \( r \)-depth nested loop with \( m \)-dimensional array reference and \( R \) its iteration space, and \( f_k(\bar{I}) = \sum a_k,i_kj_k + a_0 \) and \( g_k(\bar{I}) = \sum b_k,i_kj_k + b_0 \), for \( 1 \leq k \leq m \), are index functions. If there is a hyperplane \( f_k(\bar{I}) - g_k(\bar{I}) = 0 \) which has no intersection with \( R \), then
1. the innermost loop \( I_k \) is parallelizable if \( a_k,i_k = b_k,i_k 
eq 0 \);
2. loop \( I_p,p \), for \( q \leq p \leq r \), are parallelizable if either \( a_k,i_p = 0 \), for \( q \leq p \leq r \), or \( b_k,i_p = 0 \), for \( q \leq p \leq r \).

Proof. We assume that hyperplane \( f_k(\bar{I}) - g_k(\bar{I}) = 0 \) is above \( R \); the other case can be handled similarly. If \( a_k,i_k = b_k,i_k 
eq 0 \), the flow dependence is approximated by \( \sum a_k,i_kj_k > 0 \) and \( \sum b_k,i_kj_k > 0 \), and the anti-dependence is approximated by \( \sum a_k,i_kj_k < 0 \) and \( \sum b_k,i_kj_k < 0 \), according to Theorem 5.1. Obviously, any dependence in the form of \((0,0,\ldots,0,d_e)\) cannot satisfy these inequalities, and \( W_p \) is empty. Hence, \( I_p \) is parallelizable. If, for instance, \( a_k,i_k = 0 \) for \( q \leq p \leq r \), we then have \( \sum a_k,i_kj_k < 0 \) for flow dependence and \( \sum b_k,i_kj_k < 0 \) for anti-dependence, which implies that sets \( W_p \)'s for \( q - 1 \leq p \leq r \) are empty. Hence, loop \( I_p \)'s for \( q - 1 \leq p \leq r \) are parallelizable.

7 Conclusion

We present in this paper a new technique for generating loop transformation matrices. In our approach, it is not necessary to compute distance vectors or direction vectors. Instead, linear equations or inequalities of distance vectors are used to approximate data dependence. This method is effective in dealing with nested loops of any depth and is capable of exploiting parallelism of nested loops in different levels. Our future research will be focused on finding an efficient algorithm to determine whether the hyperplane is above or below or intersecting the iteration space \( R \), and finding a general way to fully utilize the information provided by the inequalities in loop transformation.

References


LIN Hua received his B.S. and M.S. degrees in electrical engineering from Fudan University, People’s Republic of China., in 1983 and 1986, respectively. Beginning in 1986, he taught for three years in the Department of Electronics Engineering at Fudan University as a Lecturer, and he is currently a Ph.D. candidate in the Department of Electrical Engineering at Texas A&M University, College Station, TX, USA. His research interests include the design and analysis of parallel algorithms for combinatorial optimization problems, and the parallelizing compiler.

LÜ MI received her M.S. and Ph.D. degrees in electrical engineering from Rice University, Houston, TX, USA, in 1984 and 1987, respectively.

She joined the Department of Electrical Engineering, Texas A&M University in 1987 where she is currently an Associate Professor. Her research interests include parallel computing, distributed processing, parallel computer architectures and applications, computational geometry and VLSI
Parallel Solutions for Large-Scale General Sparse Nonlinear Systems of Equations

HU Chengyi (胡承毅)
Department of Computer and Mathematical Sciences
University of Houston-Downtown
Houston, Texas 77002, U.S.A

Received July, 1995.

Abstract

In solving application problems, many large-scale nonlinear systems of equations result in sparse Jacobian matrices. Such nonlinear systems are called sparse nonlinear systems. The irregularity of the locations of nonzero elements of a general sparse matrix makes it very difficult to generally map sparse matrix computations to multiprocessors for parallel processing in a well balanced manner. To overcome this difficulty, we define a new storage scheme for general sparse matrices in this paper. With the new storage scheme, we develop parallel algorithms to solve large-scale general sparse systems of equations by interval Newton/Generalized bisection methods which reliably find all numerical solutions within a given domain.

In Section 1, we provide an introduction to the addressed problem and the interval Newton's methods. In Section 2, some currently used storage schemes for sparse systems are reviewed. In Section 3, new index schemes to store general sparse matrices are reported. In Section 4, we present a parallel algorithm to evaluate a general sparse Jacobian matrix. In Section 5, we present a parallel algorithm to solve the corresponding interval linear system by the all-row preconditioned scheme. Conclusions and future work are discussed in Section 6.

Keywords: Nonlinear systems of equations, sparse matrix, index storage schemes, interval Newton/generalized bisection algorithm, parallel algorithm.

1 Introduction

A. Nonlinear System of Equations

The general problem to be addressed here is to reliably find all solutions for the nonlinear system of equations

\[
    F(X) = \begin{pmatrix}
        f_0(x_0, x_1, \ldots, x_{N-1}) \\
        f_1(x_0, x_1, \ldots, x_{N-1}) \\
        \vdots \\
        f_{N-1}(x_0, x_1, \ldots, x_{N-1})
    \end{pmatrix} = 0
\]  

(1)

in a given box

This research was partially supported by NSF grants MIP-9208041, CDA-9522157, and ARO grant DAAH-0495-1-0255.

Throughout the paper we will use boldface letters and capital letters to denote interval quantities and vectors, respectively.