

# Further Results for Improving Loop Interchange in Non-adjacent and Imperfectly Nested Loops

Tsung-Chuan Huang and Cheng-Ming Yang

Department of Electrical Engineering

National Sun Yat-Sen University, Kaohsiung 80424, Taiwan, R.O.C.

## Abstract

*Loop interchange is a powerful restructuring technique for supporting vectorization and parallelization. In this paper, we propose a technique which is better to determine whether loops can be interchanged between two non-adjacent loops. We also present a method for determining whether we can directly make loop interchange on an imperfectly nested loop. Some experimental results are also presented to show the effectiveness.*

*Keywords:* Direction vector, Imperfectly nested loop, Loop interchange, Non-adjacent loop.

## 1. Introduction

In converting serial programs into their parallel equivalents, loops are especially important because they possess the greatest amount of potential parallelism in the program structure. Many loops, which reveal no explicit parallelism on the surface, can be finally parallelized or vectorized after making use of appropriate loop transformations. Loop interchange is a kind of loop transformation first proposed by Wolfe[15], which permutes two adjacent levels of a nested loop so that the outer loop becomes the inner loop and vice versa. It is a powerful restructuring technique for supporting vectorization and parallelization. Loop interchange can also be used to reduce memory bank conflicts, enhance cache efficiency, and decrease the number of page faults in a virtual memory system. In addition, many other goals can be satisfied by using this technique. Loop interchange [6,7,9,14,15,16,17] has been a powerful restructuring strategy in supercompiler design. For these reasons, we will propose a technique which is easier to determine whether loops can be interchanged between two non-adjacent loops. Besides, since most studies of loop interchange are restricted in perfectly nested loops, we will also present a method to deal with imperfectly nested loops.

For easier explanation, we begin with some notations. Let  $S$  be a statement enclosed in a  $d$ -nested loop with

indices  $I_1, I_2, \dots, I_d$ , numbered successively from the outermost to the innermost loop, and  $\mathbf{i}=(i_1, i_2, \dots, i_d)$  be an iteration vector, we used  $S(\mathbf{i})=S(i_1, i_2, \dots, i_d)$  to indicate the execution of  $S$  during the particular iteration step while  $I_1=i_1, I_2=i_2, \dots, I_d=i_d$  and call it the instance of  $S$  associated with  $\mathbf{i}$ . The set of all iteration vectors of  $S$  is referred to the execution index set of  $S$ , denoted by  $[S]$ . A  $d$ -nested loop is said to be perfect if and only if the body of loop with index  $I_k$  is exactly that with index  $I_{k+1}$  ( $1 \leq k \leq d-1$ ); otherwise to be imperfect. Whenever we refer to nested loop in this paper, perfect loop is meant unless otherwise stated. Based on these notations, we have following definition:

**Definition 1:** Let  $S$  and  $T$  be two statements in a  $d$ -nested loop and  $(\mathbf{i}, \mathbf{j}) \in [S] \times [T]$ . The distance vector of  $\mathbf{i}$  and  $\mathbf{j}$  is defined by  $\mathbf{u}=\mathbf{j}-\mathbf{i}$ . The direction vector  $\boldsymbol{\theta}$  of  $\mathbf{i}$  and  $\mathbf{j}$  is of length  $d$ , and for all  $k$  ( $1 \leq k \leq d$ ) defined by

$$\theta_k = \begin{cases} "<", & \text{if } u_k > 0 \\ "=", & \text{if } u_k = 0 \\ ">", & \text{if } u_k < 0 \end{cases}$$

■

## 2. Loop interchange for non-adjacent loops

To deal with the interchange between two adjacent loops, there is a well-known result proposed by Wolfe:

**Lemma 1[15]:** If loops  $L_1$  and  $L_2$  are perfectly nested loops,  $L_1$  containing  $L_2$ , the loops can be interchanged if and only if there are no two statements  $S$  and  $T$  (not necessarily distinct) in  $L_2$  such that  $S \delta T$  and their direction vector is  $(<, >)$ .

■

As for the interchange between two non-adjacent loops, Allen[1] classified all possible direction vectors between two arbitrary loops  $L_k$  and  $L_p$  as safe or unsafe for loop interchange:

(i) SAFE (interchangeable)

$$\begin{array}{ccc} L_k & \cdots & L_p \\ < * & \cdots & < \\ < = \cdots < * & \cdots & = \\ = \leq & \cdots & < \\ = \leq & \cdots & = \end{array}$$

(ii) UNSAFE (uninterchangeable)

$$\begin{array}{ccc} L_k & \cdots & L_p \\ < * & \cdots & > \\ = \leq & \cdots & > \\ < = \cdots > * & \cdots & = \end{array}$$

Here \* denotes any choice of  $\{<, =, >, \leq, \geq, \neq\}$  and  $\cdots$  represents an indefinite continuation of the previous entry. Obviously, Allen's method already combined Wolfe's method, because Wolfe only discussed adjacent loop interchange but Allen considered both adjacent and non-adjacent loops.

Although Allen proposed a powerful way to handle non-adjacent loop interchange, he as well as Wolfe, in dealing with adjacent loop interchange, only considers the dependence directions between the loops to be interchanged. This makes many legal loop interchange prevented. Therefore, we propose following lemma to help us determine whether two arbitrary levels in a perfectly nested loop can be interchanged directly. In this lemma, not only the dependence direction of the loops to be interchanged, but also that of prior loop is taken into account.

**Lemma 2:** Let  $\mathbf{L}=(L_1, L_2, \dots, L_d)$ , ( $d \geq 3$ ) be perfectly nested loops such that  $L_t$  contains  $L_{t+1}$  ( $1 \leq t \leq d-1$ ),  $S_1$  and  $S_2$  be two statements enclosed in the perfectly nested loops. Then,  $L_k$  and  $L_p$  ( $k < p$ ) are interchangeable if the direction vector  $\theta$  of  $S_1$  and  $S_2$  is one of the following types:

$$\{(\overset{r}{=}, <, *, \dots, *, *_k, \dots, *, *_p, \dots, *_d)\} \quad (1)$$

$$\{(\overset{r}{=}, <_k, *, \dots, <_p, *, \dots, *_d)\} \quad (2)$$

$$\{(\overset{r}{=}, <_k, =, \dots, <, *, \dots, =_p, *, \dots, *_d)\} \quad (3)$$

$$\{(\overset{k}{=}, <, *, \dots, *, <_p, *, \dots, *_d)\} \quad (4)$$

$$\{(\overset{k}{=}, <, *, \dots, *, =_p, *, \dots, *_d)\} \quad (5)$$

where  $0 \leq r \leq k-2$ , \* may be empty or any choice of  $\{<, =, >, \leq, \geq, \neq\}$ , and  $\cdots$  represents an indefinite continuation of the previous entry. ■

**Proof:** We only prove case (1); the other cases are similar.

Let  $\mathbf{L}_{/(k,p)}$  denote a loop constructed by exchanging the order of the do-control associated with loops  $L_k$  and  $L_p$ , “<” denote the lexicographical order, and “<<” denote the standard execution order. If case (1) is not interchangeable, then we have:

(i)  $S_1(\mathbf{i}) \ll S_2(\mathbf{j}) \Leftrightarrow \mathbf{i} < \mathbf{j}$  and

(ii)  $S_2(\mathbf{j}_{/(k,p)}) \ll_{/(k,p)} S_1(\mathbf{i}_{/(k,p)}) \Leftrightarrow \mathbf{j}_{/(k,p)} < \mathbf{i}_{/(k,p)}$  ( $r+1 < k < p \leq d$ )

But  $\mathbf{j}_{/(k,p)} < \mathbf{i}_{/(k,p)}$  is impossible, because the  $(r+1)$ th entry in the direction vector is “<”. Thus, we must have  $S_1(\mathbf{i}_{/(k,p)}) \ll_{/(k,p)} S_2(\mathbf{j}_{/(k,p)})$ . This implies that we can find an  $x$ ,  $1 \leq x < k$ , such that  $\mathbf{i} <_x \mathbf{j}$ . That is:

$$\begin{array}{l} \mathbf{i}=(i_1, \dots, i_{x-1}, i_x, i_{x+1}, \dots, i_d) \\ \mathbf{j}=(j_1, \dots, j_{x-1}, j_x, j_{x+1}, \dots, j_d) \end{array}$$

with  $i_x < j_x$ . Since in  $\mathbf{i}_{/(k,p)}$  and  $\mathbf{j}_{/(k,p)}$ , the components  $k$  and  $p$  are switched ( $x < k < p \leq d$ ) and  $i_x < j_x$ , their direction vector will be of this form

$$\theta=(\overset{x-1}{=}, <, *, \dots, *_k, \dots, *_p, \dots, *_d), \quad 1 \leq x < k$$

This completes the proof. ■

The first two direction vectors patterns can deal with adjacent loops, while the others treat non-adjacent loops. After applying the maximum and minimum bounds developed in the proof of Banerjee's inequality[4,5,17], we have the following theorem:

**Theorem 1:** Let  $\mathbf{L}=(L_1, L_2, \dots, L_d)$  be perfectly nested loops with loop index variables  $i_1, i_2, \dots, i_d$  respectively, such that  $L_t$  contains  $L_{t+1}$  ( $1 \leq t \leq d-1$ ),  $S_1$  and  $S_2$  be two statements enclosed in the perfectly nested loops  $\mathbf{L}$ . If  $S_1$  and  $S_2$  are of the form

$$S_1 : A(f(i_1, i_2, \dots, i_d)) = \cdots$$

$$S_2 : \cdots = A(g(i_1, i_2, \dots, i_d)) \cdots$$

where all loop lower bounds  $l_j$  and upper bounds  $u_j$  ( $1 \leq j \leq d$ ) in  $\mathbf{L}$  are constants, and  $f$  and  $g$  are linear functions of loop index variables, i.e.,  $f(i_1, i_2, \dots, i_d) = a_0 + \sum_{1 \leq j \leq d} a_j i_j$  and

$g(i_1, i_2, \dots, i_d) = b_0 + \sum_{1 \leq j \leq d} b_j i_j$ , then there exists a legal

loop interchange of loop  $L_k$  and  $L_p$  ( $k < p$ ), if and only if one of the following inequalities, which correspond to the five patterns of direction vectors in Lemma 2 respectively, holds:

$$(1) \sum_{1 \leq j \leq r} [-(a_j - b_j)^-(u_j - l_j) + (a_j - b_j)l_j] + [-(a_{r+1}^- + b_{r+1})^+(u_{r+1} - l_{r+1} - 1) + (a_{r+1} - b_{r+1})l_{r+1} - b_{r+1}]$$

$$+ \sum_{r+2 \leq j \leq d} [-(a_j^- + b_j^+)(u_j - l_j) + (a_j - b_j)l_j]$$

$$\leq b_0 - a_0 \leq$$

$$\sum_{1 \leq j \leq r} [(a_j - b_j)^+(u_j - l_j) + (a_j - b_j)l_j] + [(a_{r+1}^+ - b_{r+1})^+(u_{r+1} - l_{r+1} - 1) + (a_{r+1} - b_{r+1})l_{r+1} - b_{r+1}]$$

$$+ \sum_{r+2 \leq j \leq d} [(a_j^+ + b_j^-)(u_j - l_j) + (a_j - b_j)l_j]$$

$$\begin{aligned}
(2) \quad & \sum_{1 \leq j \leq k-1} [-(a_j-b_j)^-(u_j-l_j)+(a_j-b_j)l_j]+[-(a_k^-+b_k)^+(u_k-l_k-1)+(a_k- \\
& \quad b_k)l_k-b_k] \\
& + \sum_{k+1 \leq j \leq p-1} [-(a_j^-+b_j^+)(u_j-l_j)+(a_j-b_j)l_j]+[-(a_p^-+b_p)^+(u_p-l_p- \\
& \quad 1)+(a_p-b_p)l_p-b_p] \\
& + \sum_{p+1 \leq j \leq d} [-(a_j^-+b_j^+)(u_j-l_j)+(a_j-b_j)l_j] \\
& \leq b_0-a_0 \leq \\
& \sum_{1 \leq j \leq k-1} [(a_j-b_j)^+(u_j-l_j)+(a_j-b_j)l_j]+[(a_k^+-b_k)^+(u_k-l_k-1)+(a_k- \\
& \quad b_k)l_k-b_k] \\
& + \sum_{k+1 \leq j \leq p-1} [(a_j^++b_j^-)(u_j-l_j)+(a_j-b_j)l_j]+[(a_p^+-b_p)^+(u_p-l_p- \\
& \quad 1)+(a_p-b_p)l_p-b_p] \\
& + \sum_{p+1 \leq j \leq d} [(a_j^++b_j^-)(u_j-l_j)+(a_j-b_j)l_j]
\end{aligned}$$

$$\begin{aligned}
(3) \quad & \sum_{1 \leq j \leq k-1} [-(a_j-b_j)^-(u_j-l_j)+(a_j-b_j)l_j]+[-(a_k^-+b_k)^+(u_k-l_k-1)+(a_k- \\
& \quad b_k)l_k-b_k] \\
& + \sum_{k+1 \leq j \leq x-1} [-(a_j-b_j)^-(u_j-l_j)+(a_j-b_j)l_j]+[-(a_x^-+b_x)^+(u_x-l_x- \\
& \quad 1)+(a_x-b_x)l_x-b_x] \\
& + \sum_{x+1 \leq j \leq p-1} [-(a_j^-+b_j^+)(u_j-l_j)+(a_j-b_j)l_j]+[-(a_p-b_p)^-(u_p-l_p)+(a_p- \\
& \quad b_p)l_p] \\
& + \sum_{p+1 \leq j \leq d} [-(a_j^-+b_j^+)(u_j-l_j)+(a_j-b_j)l_j] \\
& \leq b_0-a_0 \leq \\
& \sum_{1 \leq j \leq k-1} [(a_j-b_j)^+(u_j-l_j)+(a_j-b_j)l_j]+[(a_k^+-b_k)^+(u_k-l_k-1)+(a_k- \\
& \quad b_k)l_k-b_k] \\
& + \sum_{k+1 \leq j \leq x-1} [(a_j-b_j)^+(u_j-l_j)+(a_j-b_j)l_j]+[(a_x^+-b_x)^+(u_x-l_x-1)+(a_x- \\
& \quad b_x)l_x-b_x] \\
& + \sum_{x+1 \leq j \leq p-1} [(a_j^++b_j^-)(u_j-l_j)+(a_j-b_j)l_j]+[(a_p-b_p)^+(u_p-l_p)+(a_p- \\
& \quad b_p)l_p] \\
& + \sum_{p+1 \leq j \leq d} [(a_j^++b_j^-)(u_j-l_j)+(a_j-b_j)l_j]
\end{aligned}$$

In this case, if loops  $k$  and  $p$  are directly interchanged, all dependences in level  $k$  of original code become the dependences in level  $x$  of transformed code, where  $k < x < p$  [1].

(4)

$$\begin{aligned}
& \sum_{1 \leq j \leq k} [-(a_j-b_j)^-(u_j-l_j)+(a_j-b_j)l_j]+[-(a_{k+1}^-+b_{k+1})^+(u_{k+1}-l_{k+1}- \\
& \quad 1)+(a_{k+1}-b_{k+1})l_{k+1}-b_{k+1}] \\
& + \sum_{k+2 \leq j \leq p-1} [-(a_j^-+b_j^+)(u_j-l_j)+(a_j-b_j)l_j]+[-(a_p^-+b_p)^+(u_p-l_p- \\
& \quad 1)+(a_p-b_p)l_p-b_p] \\
& + \sum_{p+1 \leq j \leq d} [-(a_j^-+b_j^+)(u_j-l_j)+(a_j-b_j)l_j] \\
& \leq b_0-a_0 \leq \\
& \sum_{1 \leq j \leq k} [(a_j-b_j)^+(u_j-l_j)+(a_j-b_j)l_j]+[(a_{k+1}^-+b_{k+1})^+(u_{k+1}-l_{k+1}- \\
& \quad 1)+(a_{k+1}-b_{k+1})l_{k+1}-b_{k+1}] \\
& + \sum_{k+2 \leq j \leq p-1} [(a_j^++b_j^-)(u_j-l_j)+(a_j-b_j)l_j]+[(a_p^+-b_p)^+(u_p-l_p- \\
& \quad 1)+(a_p-b_p)l_p-b_p] \\
& + \sum_{p+1 \leq j \leq d} [(a_j^++b_j^-)(u_j-l_j)+(a_j-b_j)l_j]
\end{aligned}$$

In this case, if loops  $k$  and  $p$  are directly interchanged, all dependences in level  $p$  of original code become the dependences in level  $x$  of transformed code, where  $k < x < p$  [1].

(5)

$$\begin{aligned}
& \sum_{1 \leq j \leq k} [-(a_j-b_j)^-(u_j-l_j)+(a_j-b_j)l_j]+[-(a_{k+1}^-+b_{k+1})^+(u_{k+1}-l_{k+1}- \\
& \quad 1)+(a_{k+1}-b_{k+1})l_{k+1}-b_{k+1}] \\
& + \sum_{k+2 \leq j \leq p-1} [-(a_j^-+b_j^+)(u_j-l_j)+(a_j-b_j)l_j]+[-(a_p-b_p)^-(u_p-l_p)+(a_p- \\
& \quad b_p)l_p] \\
& + \sum_{p+1 \leq j \leq d} [-(a_j^-+b_j^+)(u_j-l_j)+(a_j-b_j)l_j] \\
& \leq b_0-a_0 \leq \\
& \sum_{1 \leq j \leq k} [(a_j-b_j)^+(u_j-l_j)+(a_j-b_j)l_j]+[(a_{k+1}^-+b_{k+1})^+(u_{k+1}-l_{k+1}- \\
& \quad 1)+(a_{k+1}-b_{k+1})l_{k+1}-b_{k+1}] \\
& + \sum_{k+2 \leq j \leq p-1} [(a_j^++b_j^-)(u_j-l_j)+(a_j-b_j)l_j]+[(a_p-b_p)^+(u_p-l_p)+(a_p- \\
& \quad b_p)l_p] \\
& + \sum_{p+1 \leq j \leq d} [(a_j^++b_j^-)(u_j-l_j)+(a_j-b_j)l_j]
\end{aligned}$$

**Proof:** We only prove case (1); the other cases are similar. ■

After applying the maximum and minimum bounds developed in the proof of Banerjee's inequality [4,5,17], the lower bound of direction vector  $=^t$  is

$$\sum_{1 \leq j \leq r} [-(a_j-b_j)^-(u_j-l_j)+(a_j-b_j)l_j] \quad (1)$$

The lower bound of direction vector  $<$  is

$$-(a_{r+1}^- + b_{r+1}^+) (u_{r+1} - l_{r+1} - 1) + (a_{r+1} - b_{r+1}) l_{r+1} - b_{r+1} \quad (2)$$

The lower bound of direction vector  $*_{r+2}, \dots, *_{r+d}$  is

$$\sum_{r+2 \leq j \leq d} [-(a_j^- + b_j^+) (u_j - l_j) + (a_j - b_j) l_j] \quad (3)$$

Combining (1), (2) and (3), we can obtain the lower bound of case (1). Similarly, the upper bound can be obtained. ■

### 3. Loop interchange for imperfectly nested loops

The conventional method of loop interchange in dealing with imperfectly nested loops is to convert them into perfectly ones by loop distribution, then applies regular loop interchange schemes. In this section, we will proposed a method, by which we can determine whether two loops can be interchanged or not in an imperfectly nested loop as shown in Figure 1:

```

L1: DO I=li, ui
L2: DO J=lj, uj
    S1
L3: DO K=lk, uk
    S2
    ENDDOK
  ENDDOJ
ENDDOI

```

**Figure 1.** A structure of imperfectly nested loop.

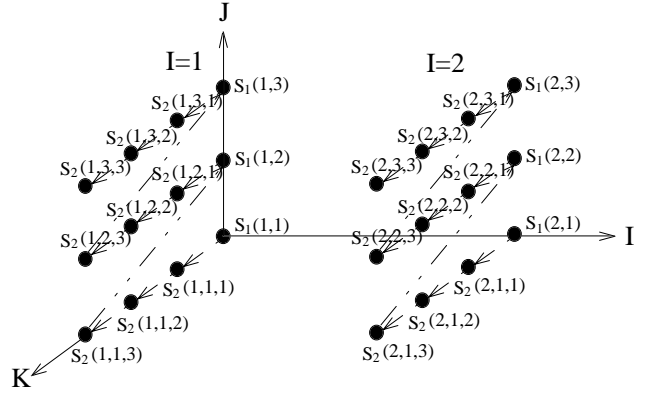
Let's consider the example in Figure 2, whose execution order is shown in Figure 3(a). There is a true dependence from  $S_1$  to  $S_2$  due to array A, since the value of  $A(i,j)$  produced in  $S_1(i,j)$  is used in  $S_2(i+1,j+1,j)$ . This relation is shown in Figure 3(b). The distance vector of  $S_1$  and  $S_2$  is (1,1), and their direction vector is (<,<). In this case,  $L_2$  and  $L_3$  are interchangeable because the original data dependence is preserved after loop interchange. This is demonstrated in Figure 3(c). The resulting loop after loop interchange is shown in Figure 4.

```

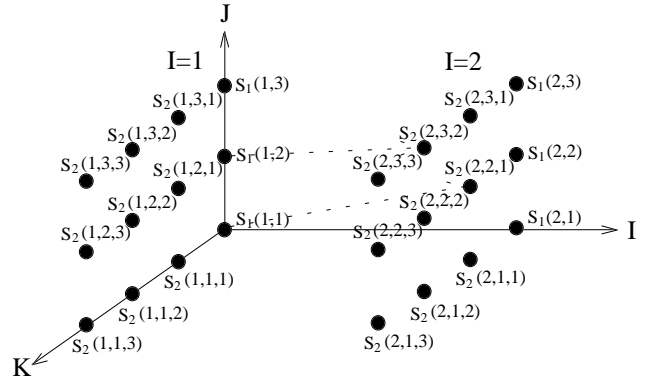
L1: DO I=1,2
L2: DO J=1,3
  S1: A(I,J)=B(I,J)+1
L3: DO K=1,3
  S2: C(I,J,K)=A(I-1,J-1,K)*2
  ENDDOK
ENDDOJ
ENDDOI

```

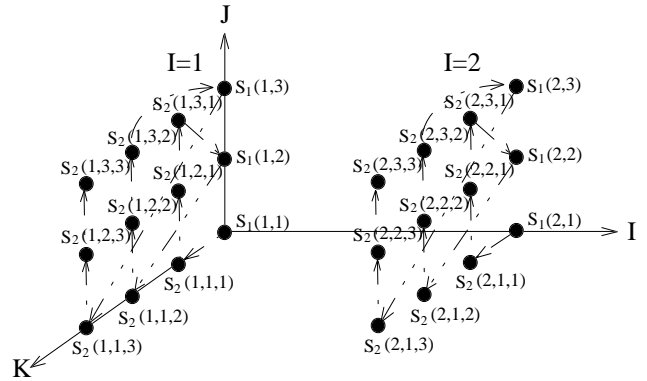
**Figure 2.** An example of true dependence in imperfectly nested loop.



(a)



(b)



(c)

**Figure 3.** (a) The execution order of Figure 2. (b) The iteration space dependence graph of Figure 2. (c) The execution order after loop interchange of Figure 2.

```

L1: DO I=1,2
L3: DO K=1,3
S1: A(I,K,K)=B(I,K,K)+1
L2: DO J=1,3
S2: C(I,J,K)=A(I-1,J-1,K)*2
      ENDDOJ
      ENDDOK
      ENDDOI

```

**Figure 4.** The result after loop  $L_2$  and  $L_3$  are interchanged in Figure 2.

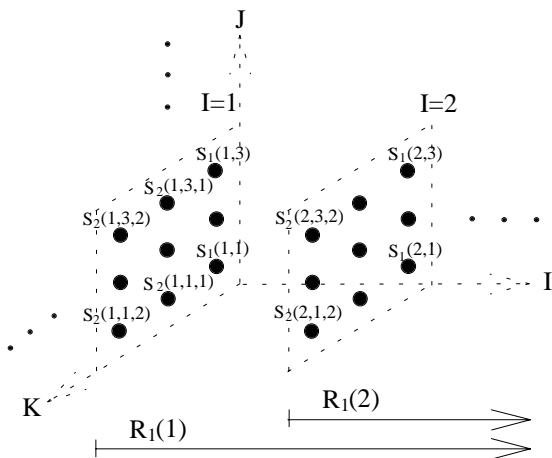
Therefore, from the properties of direction vector, we can see that, for the program segment in Figure 1,  $L_2$  and  $L_3$  are interchangeable if all the data dependence relations are:

- (1)  $S_1(i,j) \delta S_1(i',j')$  where  $i < i'$  or  $i = i', j \leq j'$ ;
- (2)  $S_2(i,j,k) \delta S_2(i',j',k')$  where  $i < i'$  or  $i = i', j \leq j', k \leq k'$ ;
- (3)  $S_1(i,j) \delta S_2(i',j',k')$  where  $i < i'$  or  $i = i', j \leq j', j \leq k'$ ;
- (4)  $S_2(i,j,k) \delta S_1(i',j')$  where  $i < i'$  or  $i = i', j < j', k < j'$ ;

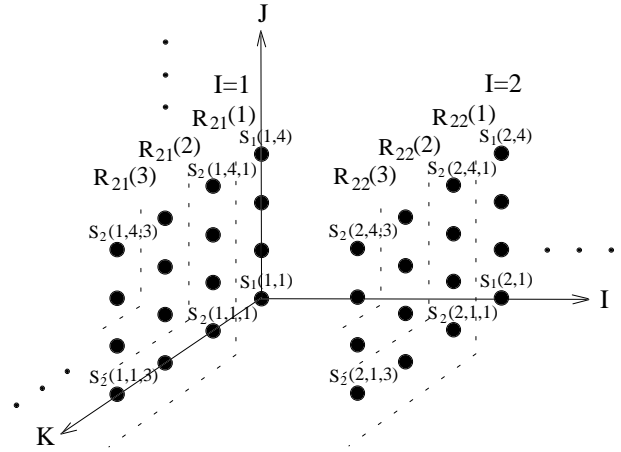
After investigating the iteration spaces of these cases, some properties are revealed. When  $i < i'$ , we get Figure 5, in which  $R_1(1)$  contains  $R_1(2)$ ,  $R_1(2)$  contains  $R_1(3)$ , and so on. When  $i = i', j \leq j'$ , and  $j \leq k'$  we get Figure 6, in which  $R_{2i}(1)$  contains  $R_{2i}(2)$ ,  $R_{2i}(2)$  contains  $R_{2i}(3)$ , and so on. When  $i = i', j < j'$ , and  $k < j'$  we have Figure 7, where  $R_{3i}(3)$  contains  $R_{3i}(2)$ ,  $R_{3i}(2)$  contains  $R_{3i}(1)$ , and so forth. As for the case of  $i = i', j \leq j'$ , and  $k \leq k'$ , the iteration space is similar to that of perfect nested loop.

After loop interchange in imperfectly nested loop, the dependences will preserve when:

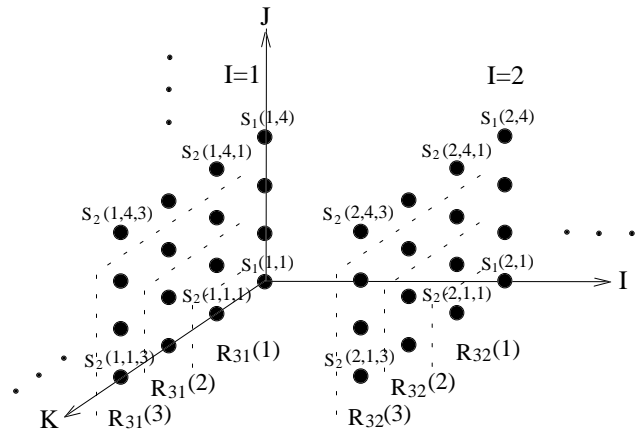
- (1) Statement  $S_1$  or  $S_2$  in region  $R_1(i)$  depends on  $S_1$  or  $S_2$  in region  $R_1(i+1)$ , as shown in Figure 8(a).
- (2) Statement  $S_2$  in region  $R_{2i}(j-1)$  depends on  $S_1(i,j)$ , as shown in Figure 8(b).
- (3)  $S_1(i,j)$  depends on statement  $S_2$  in region  $R_{3i}(j)$ , as shown in Figure 8(c).



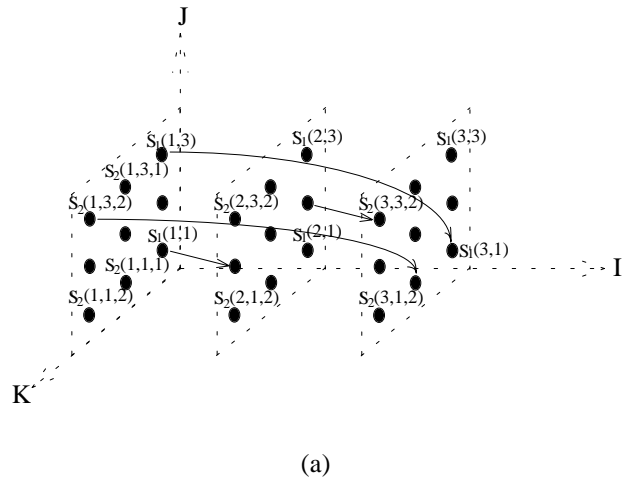
**Figure 5.** The iteration space for  $i < i'$ .



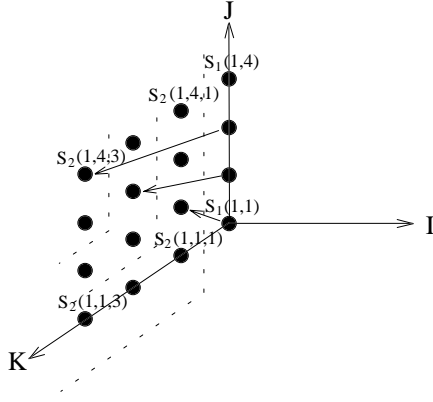
**Figure 6.** The iteration space for  $i = i', j \leq j'$ , and  $j \leq k'$ .



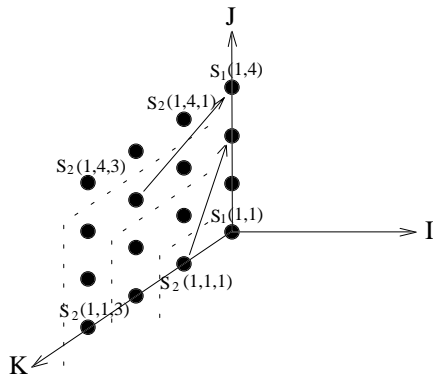
**Figure 7.** The iteration space for  $i = i', j < j'$ , and  $k < j'$ .



(a)



(b)



(c)

**Figure 8.** Example of legal data dependence relation in imperfectly loop interchange. (a) The case for  $i < i'$ . (b) The case for  $i = i', j \leq j'$ , and  $j \leq k$ . (c) The case for  $i = i', j < j'$ , and  $k < k'$ .

The inequality for imperfectly loop interchange is derived in Theorem 3. This inequality is also based on Banerjee inequality, and is more exact than the direction vector in compiler implementation.

**Theorem 3:** For the imperfectly nested loops shown in Figure 1, where  $l_j = l_k, u_j = u_k$ . Let  $S_1$  and  $S_2$  be of this form

$$\begin{aligned} S_1 : & A(f(i,j)) = \dots \\ S_2 : & \dots = A(g(i,j,k)) \dots \end{aligned}$$

where  $f$  and  $g$  are linear functions of loop induction variables. i.e.,

$$\begin{aligned} f(i,j) &= a_i i + a_j j + a_0 & \text{and} \\ g(i,j,k) &= b_i i + b_j j + b_k k + b_0 \end{aligned}$$

Then, loops  $L_2$  and  $L_3$  are interchangeable if the following inequality holds:

$$\begin{aligned} & [-(a_i^- + b_i)^+(u_i - l_i - 1) + (a_i - b_i)l_i - b_i] \\ & + [-(a_j^- + b_j^+)(u_j - l_j) + (a_j - b_j)l_j] \\ & + [-b_k^+(u_k - l_k) - b_k l_k] \\ & \leq b_0 - a_0 \leq \\ & [(a_i^+ - b_i)^+(u_i - l_i - 1) + (a_i - b_i)l_i - b_i] \\ & + [(a_j^+ + b_j^-)(u_j - l_j) + (a_j - b_j)l_j] \\ & + [b_k^-(u_k - l_k) - b_k l_k] \end{aligned}$$

**Proof:** By applying the maximum and minimum bounds developed in the proof of Banerjee's inequality [4,5,17] under the case of direction vector  $(<,*)$ , the inequality can be obtained. ■

## 4. Experimental results

We implement the loop interchange scheme of ours and Allen's respectively using Sage++ and compare their performance. In this experiment, five benchmarks: LAPACK, VECTOR, PARALLEL and DMR are used. Their applications and numbers of perfectly nested loops are listed in Table 1 and Table 2 respectively. There are totally 453 perfectly nested loops in these benchmarks. Two-level nested loop is about 87.4% (396/453), three-level nested loop is about 11.5% (52/453), four-level nested loop is about 0.66% (3/453), and five-level nested loop about 0.44% (2/453). The loop pairs we tested are 590 (396+52\*3+3\*6+2\*10) in total.

The experimental results are shown in Table 3. By our method, there are 502 loop pairs able to be interchanged and 88 loop pairs unable to be interchanged. In other hand, there are 479 loop pairs able to be interchanged and 111 loop pairs unable to be interchanged using Allen's method. Our method finds 23 more loop pairs to be interchangeable. The improvement rate are 3.9 percents.

**Table 1.** Applications of the benchmarks.

Benchmark	Application
LAPACK	Numerical Linear Algebra
VECTOR	Testing the Ability of Analysis on Vector Compiler
PARALLEL	Testing the Ability of Parallelization on Software/Hardware System
DMR	Testing the Time of Linear Algebra Operation on IBM RS/6000-530 System

**Table 2.** Perfectly nested loops in the benchmarks.

Benchmark	Two-level loop	Three-level loop	Four-level loop	Five-level loop	Total perfectly nested loop
LAPACK	258	17	0	0	275
VECTOR	44	8	0	0	52
PARALLEL	87	15	3	2	107
DMR	7	12	0	0	19
Total	396	52	3	2	453

**Table 3.** Experimental results.

Benchmark	Our method		Allen's method	
	Inter-changeable loop pairs	Uninter-changeable loop pairs	Inter-changeable loop pairs	Uninter-changeable loop pairs
LAPACK	274	35	270	39
VECTOR	42	26	38	30
PARALLEL	159	11	148	22
DMR	27	16	23	20
Total	502	88	479	111

## 5. Conclusion

Loop interchange is an important restructuring technique for supporting vectorization and parallelization. In this paper, we categorize the direction vectors between two arbitrary loops into five types for the loops which are interchangeable in a perfectly nested loop. Our classifications covers that of Allen's method and have better performance in determining whether two loops are interchangeable. For more exact in compiler implementation, the inequalities for these five types of direction vectors are derived respectively. We also investigate a kind of imperfectly nested loop and derive the inequality which must hold if and only if two loops can be interchanged.

## REFERENCES

- [1] J. R. Allen, Data Analysis for Subscripted Variable and Its Application to Program Transformations, Ph.D. Dissertation, Department of Mathematical Sciences, Rice University, Houston TX, 1983.
- [2] J. R. Allen and K. Kennedy, Automatic Translation of FORTRAN Programs to Vector Form, *ACM TOPLAS*, Vol.9, (1987) 491-542.
- [3] J. R. Allen, K. Kennedy, C. Porterfield and J. Wareen, Conversion of Control Dependence to Data Dependence, *In Conf. Rec. 10th ACM Sym. Principle of Programming Languages (POPL)*, (1983) 177-189.
- [4] U. Banerjee, Data Dependence in Ordinary Programs, M.S. Thesis Technical Report UIUCDS-R-76-837, Department of Computer Science, University of Illinois at Urbana-Champaign, 1976.
- [5] U. Banerjee, *Dependence Analysis for Supercomputing*, (Boston MA: Kluwer 1988).
- [6] U. Banerjee. Unimodular Transformations of Double Loops. *In Proceedings of the Third Workshop on Languages and Compilers for Parallel Computing*, Irvine, California, August 1-3, 1990.
- [7] U. Banerjee, *Loop Transformations for Restructuring Compilers: The Foundations*. (Boston MA: Kluwer 1993).
- [8] U. Banerjee, *Loop Parallelization*, (Kluwer Academic Pub., 1994).
- [9] P. Feautrier, Some Efficient Solutions to the Affine Scheduling Problem. Part II. Multidimensional Time, *International Journal of Parallel Programming*, 1992(12), pp. 389-420.
- [10] T. C. Huang, C. M. Yang and C. S. Yang, Loop Interchange with Loops Containing Control Dependence, *Proceeding of The First Workshop on Compiler Techniques for High-Performance Computing*, (1995) 30-41.
- [11] K. Kennedy, and K. S. Mckinley, Loop Distribution with Arbitrary Control Flow, *Proceeding of Supercomputing '90, New York*, (November 12-16, 1990) 407-416.
- [12] D. J. Kuck, R. H. Kuhn, D. A. Padua, B. R. Leasure and M. M. Wolfe, Dependence Graphs and Compiler Optimizations, *In Conf. Rec. 8th ACM Sym., Principles of Programming Languages (POPL)*, (1981) 207-218.
- [13] R. A. Towle, Control and Data Dependence for Program Transformation, Ph.D. Thesis, University of Illinois at Urbana-Champaign, March, 1976.
- [14] M. E. Wolf and M. S. Lam, A Loop Transformation Theory and an Algorithm to Maximize Parallelism. *IEEE Transformations on Parallel and Distributed Systems*, vol. 2, no. 4, pp. 452-471. October 1991.
- [15] M. J. Wolfe, Optimizing Supercompiler for Supercomputers, Ph.D. Dissertation, Technical Report 82-1009, Department of Computer Science, University of Illinois at Urbana-Champaign, 1982.
- [16] Y. Q. Yang, C. Ancourt and F. Irigoin, Minimal Data Dependence Abstractions for Loop Transformations: Extended Version, *International Journal of Parallel Programming*, v. 23, n. 4, Aug. 1995, pp. 359-388.
- [17] H. P. Zima and B. Champman, *Supercompiler for Parallel and Vector Computers*, (New York MA: Addison-Wisely, 1991).