ELSEVIER

The Journal of Systems and Software 63 (2002) 99–112

The Journal of Systems and Software

www.elsevier.com/locate/jss

A precise dependence analysis for multi-dimensional arrays under specific dependence direction

Weng-Long Chang [a], Chih-Ping Chu [b,*], Jia-Hwa Wu [b]

[a] Department of Information Management, Southern Taiwan University of Technology, Tainan 701, Taiwan, ROC

[b] Department of Computer Science and Information Engineering, National Cheng Kung University, No. 1, Ta-Hsueh Road, Tainan 701, Taiwan, ROC

Received 28 October 2000; received in revised form 8 January 2001; accepted 26 June 2001

Abstract

In process of automatic parallelizing/vectorizing constant-bound loops with multi-dimensional arrays under specific dependence direction, the Lambda test is claimed to be an efficient and precise data dependence analysis method that can check whether there exist generally inexact '*real-valued*' solutions to the derived dependence equations. In this paper, we propose a precise data dependence analysis method – the multi-dimensional direction vector *I* test. The multi-dimensional direction vector *I* test can be applied towards testing whether there exist generally accurate '*integer-valued*' solutions to the dependence equations derived from multi-dimensional arrays under specific dependence direction in constant-bound loops. Experiments with benchmark showed that the accuracy rate and the improvement rate for the proposed method are approximately 33.3% and 21.6%, respectively.
© 2001 Elsevier Science Inc. All rights reserved.

Keywords: Parallelizing/vectorizing compilers; Data dependence analysis; Loop parallelization; Supercomputing

1. Introduction

Multi-dimensional arrays occur quite frequently in real programs in light of an empirical study reported in Shen et al. (1992). That study also shows that two-dimensional array references account for 36% and three-dimensional array references account for 7% among array references examined. Such a finding highlights a fact, that is, to parallelize/vectorize a loop, an efficient and precise testing method for the data dependence of multi-dimensional arrays is very important.

The question of whether multi-dimensional array references in a loop may be parallelized/vectorized depends upon the resolution of those multi-dimensional array aliases. The resolution of multi-dimensional array aliases is to ascertain whether two references to the same multi-dimensional array within a general loop may refer to the same element of that multi-dimensional array. This problem of m-dimensional arrays, each dimension's expression is the function of n loop index variables, can be reduced to that of checking whether a system of m linear equations with n unknown variables has a simultaneous integer solution, which satisfies the constraints for each variable in the system. It is assumed that m linear equations in a system are written as

$$\begin{aligned} &a_{1,1}X_1 + a_{1,2}X_2 + \cdots + a_{1,n-1}X_{n-1} + a_{1,n}X_n = a_{1,0} \\ &\qquad\vdots \\ &a_{m,1}X_1 + a_{m,2}X_2 + \cdots + a_{m,n-1}X_{n-1} + a_{m,n}X_n = a_{m,0}, \end{aligned} \tag{1.1}$$

where each $a_{i,j}$ is a constant integer for $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant n$. It is postulated that the constraints to each variable in (1.1) are represented as

$$P_{r,0} + \sum_{s=1}^{r-1} P_{r,s}X_s \leqslant X_r \leqslant Q_{r,0} + \sum_{s=1}^{r-1} Q_{r,s}X_s, \tag{1.2}$$

where $P_{r,0}$, $Q_{r,0}$, $P_{r,s}$ and $Q_{r,s}$ are constant integers for $1 \leqslant r \leqslant n$. That is, the bounds for each variable X_r are variable.

If each of $P_{r,s}$ and $Q_{r,s}$ is zero in the limits of (1.2), then (1.2) will be reduced to

$$P_{r,0} \leqslant X_r \leqslant Q_{r,0}, \quad \text{where } 1 \leqslant r \leqslant n. \tag{1.3}$$

* Corresponding author. Tel.: +886-6-27-57-575x62527; fax: +886-6-27-470-76.
E-mail address: chucp@csie.ncku.edu.tw (C.-P. Chu).

That is, the bounds for each variable X_r are constants. Let us use an example to make clear the illustrations stated above. Consider the nested do-loop in Fig. 1.

The lower and upper bounds of the first (outer) and the second (inner) loops are 1 and 100 and 1 and 10, respectively. Therefore, the bounds of the do-loop are constants. This do-loop executes 1000 iterations by consecutively assigning the values $1, 2, \ldots, 100$ to J and $1, 2, \ldots, 10$ to I and by executing the body (the statement ***S***) exactly once in each iteration. The net effect of the do-loop execution is then the ordered execution of the statements:

$$
\begin{aligned}
&A(1,1) = B(1,1) + 1\\
&A(2,2) = B(2,1) + 1\\
&\qquad\quad\vdots\\
&A(999,999) = B(9,100) + 1.\\
&A(1000,1000) = B(10,100) + 1.
\end{aligned}
$$

To ascertain whether two references to the multi-dimensional array A may refer to the same element of A, we have to check if the following two linear equations:

$$10 \times X_1 - 10 \times X_2 + X_3 - X_4 = 0,$$

$$10 \times X_1 - 10 \times X_2 + X_3 - X_4 = 0$$

have a simultaneous integer solution under the constant bounds $1 \leqslant X_1, X_2 \leqslant 100$, and $1 \leqslant X_3, X_4 \leqslant 10$.

It is well known that the problem of finding *integer valued* solutions to a system of linear equations is NP-hard. Therefore, in practice, most well-known data dependence analysis algorithms are used to solve as many particular cases of this problem as possible. The *Banerjee inequalities* handle *real-valued* solutions of one linear equation under the bounds of (1.3) (Banerjee, 1997, 1988). The *Banerjee–Wolfe inequalities* deal with real solutions of one linear equation under the bounds of (1.3) and given direction vectors (Banerjee, 1997, 1988). The *Banerjee algorithm* determines real solutions of one linear equation under the limits of (1.2) and given direction vectors (Banerjee, 1997, 1988). The *Banerjee infinity test* checks real solutions of one linear equation with *symbolic* (*unknown* at compile time) bounds and given direction vectors (Banerjee, 1997; Petersen, 1993). For m linear equation (1.1) with the same constraints, each linear equation has to be tested separately. The *Banerjee test* in this case may generally lose the accuracy

```
DO J = 1, 100
    DO I =1, 10
        S:  A(I+10*(J-1), I+10*(J-1)) =  B(I, J)+1
    ENDDO
 ENDDO
```

Fig. 1. A nested do-loop in Fortran language.

in many practical cases. The I test and the direction vector I test are a combination of the Banerjee inequalities (*Banerjee inequalities* and *Banerjee–Wolfe inequalities*) and the GCD test (Kong et al., 1991, 1993). They figure out *integer* solutions for one linear equation with constant bounds and given direction vectors. The I+ test and the extended version of direction vector I test are a combination of the Banerjee algorithm and the GCD test (Chu and Chang, 1998, 1999). They figure out *integer* solutions for one linear equation with the bounds of (1.2) and given direction vectors. The Lambda test extends the Banerjee inequalities to allow m linear equation (1.1) under the constraints of (1.3) and given direction vectors to be tested simultaneously (Li et al., 1990). The generalized Lambda test extends the Banerjee algorithm to allow m linear equation (1.1) under the constraints of (1.2) and given direction vectors to be tested simultaneously (Chang et al., 1999). The infinity Lambda test extends the Banerjee infinity test to allow m linear equation (1.1) or m nonlinear subscripts with symbolic limits and given direction vectors to be tested simultaneously (Chang and Chu, 2000a). The multi-dimensional I test is extended from the I test and the Lambda test (Chang and Chu, 2000b). It figures out *integer* solutions for linear equations with the bounds of (1.3). The Power test is a combination of Fourier–Motzkin variable elimination with an extension of Euclid's GCD algorithm (Wolfe and Tseng, 1992; Wolfe, 1996). The Omega test combines new methods for eliminating equality constraints with an extension of Fourier–Motzkin variable elimination to integer programming (Pugh, 1992). Though both of the methods gain more accurate outcomes, they have exponential worst-case time complexity. For array references with nonlinear subscripts, the range test can be applied to test data dependency (Blume and Eigenmann, 1998).

In this paper, the Lambda test and the direction vector I test are extended and integrated to treat whether m linear equation (1.1) under the bounds of (1.3) have integer solutions. A theoretical analysis explains that we take advantage of the rectangular shape of the convex sets derived from m linear equations under constant limits in a data dependence testing. An algorithm called the multi-dimensional direction vector I test has been implemented and several measurements have also been performed.

The rest of this paper is proffered as follows. In Section 2, the problem of data dependence subject to direction vectors is reviewed. The summary accounts of the direction vector I test and the Lambda test are presented. In Section 3, the theoretical aspects and the worst-case time complexity of the multi-dimensional direction vector I test are described. Experimental results showing the advantages of the multi-dimensional direction vector I test are given in Section 4. Finally, brief conclusions are drawn in Section 5.

2. Background

The concept of data dependence and the summary accounts of the direction vector I test and the Lambda test are mainly introduced in this section.

2.1. Data dependence

Consider a general loop shown in Fig. 2. It is assumed that S_1 and S_2 are two statements within a general loop. Statements S_1 and S_2 are postulated to be embedded in $d+p$ loops and $d+q$ loops, respectively. Therefore, d loops simultaneously contain statements S_1 and S_2. The d loops are called common loops for statements S_1 and S_2. An array A appears simultaneously within statements S_1 and S_2. In execution of this general loop, if statement S_2 uses the element of array A defined first by statement S_1, then S_2 is true-dependent on S_1. If statement S_2 defines the element of array A used first by statement S_1, then S_2 is anti-dependent on S_1. If statement S_2 redefines the element of array A defined first by statement S_1, then S_2 is output-dependent on S_1.

Each iteration of a general loop is identified by an iteration vector whose elements are the values of the iteration variables for that iteration. For example, the instance of the statement S_1 during iteration $\vec{i} = (i_1, \ldots, i_d, \ldots, i_{d+p})$ is denoted $S_1(\vec{i})$; the instance of the statement S_2 during iteration $\vec{j} = (j_1, \ldots, j_d, \ldots, j_{d+q})$ is denoted $S_2(\vec{j})$. If $(i_1, \ldots, i_d, \ldots, i_{d+p})$ is identical to $(j_1, \ldots, j_d, \ldots, j_{d+q})$ or $(i_1, \ldots, i_d, \ldots, i_{d+p})$ precedes $(j_1, \ldots, j_d, \ldots, j_{d+q})$ lexicographically, then $S_1(\vec{i})$ is said to precede $S_2(\vec{j})$, denoted $S_1(\vec{i}) < S_2(\vec{j})$. Otherwise, $S_2(\vec{j})$ is said to precede $S_1(\vec{i})$, denoted $S_1(\vec{i}) > S_2(\vec{j})$. In the following, Definition 2.1 defines direction vectors.

```
$L_1$ :  DO $I_1 = l_1, u_1$
           ⋮
$L_d$ :    DO $I_d = l_d, u_d$
             ⋮
$L_{d+p}$ :    DO $I_{d+p} = l_{d+p}, u_{d+p}$
$S_1$ :          $A(\cdots)\cdots$
               END DO $I_{d+p}$
             ⋮
$L_{d+q}$ :    DO $I_{d+q} = l_{d+q}, u_{d+q}$
$S_2$ :          $A(\cdots)\cdots$
               END DO $I_{d+q}$
             ⋮
           END DO $I_d$
         END DO $I_1$
```

Fig. 2. A general loop with two statements S_1 and S_2.

Definition 2.1. A vector of the form $\vec{\theta} = (\theta_1, \ldots, \theta_d)$ is termed as a direction vector. The direction vector $(\theta_1, \ldots, \theta_d)$ is said to be the direction vector from $S_1(\vec{i})$ to $S_2(\vec{j})$ if for $1 \leqslant k \leqslant d$, $i_k \theta_k j_k$, i.e., the relation θ_k is defined by

$$\theta_k = \begin{cases} < & \text{if } i_k < j_k, \\ = & \text{if } i_k = j_k, \\ > & \text{if } i_k > j_k, \\ * & \text{the relation of } i_k \text{ and } j_k \text{ can be ignored,} \\ & \text{i.e., can be any one of } \{<, =, >\}. \end{cases}$$

2.2. The direction vector I test

A linear equation with the bounds of (1.3) and any given direction vectors will be said to be integer solvable if the linear equation has an integer solution to satisfy the bounds of (1.3) and any given direction vectors for each variable in the linear equation. The direction vector I test deals with a linear equation by first transforming it to an *interval* equation. Definitions 2.2 and 2.3 cited from (Kong et al., 1991, 1993) define integer intervals and an interval equation.

Definition 2.2. Let $[\alpha_1, \alpha_2]$ represent the integer intervals from α_1 to α_2, i.e., the set of all integers between α_1 and α_2.

To avoid redundancy, throughout this paper we will use the term interval to refer to the integer interval.

Definition 2.3. Let $a_1, \ldots, a_{n-1}, a_n$, L and U be integers. A linear equation

$$a_1X_1 + a_2X_2 + \cdots + a_{n-1}X_{n-1} + a_nX_n = [L, U], \qquad (2.1)$$

which is referred to as an interval equation, will be used to denote the set of ordinary equations consisting of:

$$\begin{aligned} &a_1X_1 + a_2X_2 + \cdots + a_{n-1}X_{n-1} + a_nX_n = L, \\ &a_1X_1 + a_2X_2 + \cdots + a_{n-1}X_{n-1} + a_nX_n = L + 1, \\ &\qquad\vdots \\ &a_1X_1 + a_2X_2 + \cdots + a_{n-1}X_{n-1} + a_nX_n = U. \end{aligned}$$

Similarly, we will use the term interval equation to refer to the integer interval equation throughout this paper. An interval equation (2.1) will be said to be integer solvable if one of the equations in the set, which it defines, is integer solvable. The immediate way to determine this is to test if an integer in between L and U is divisible by the GCD of the coefficients of the left-hand side terms. If $L > U$ in an interval equation (2.1), then there are no integer solutions for the interval equation (2.1). If the expression on the left-hand side of an interval equation (2.1) is reduced to zero items, in the processing of testing, then the interval equation (2.1) will be said to be integer solvable if and only if $U \geqslant 0 \geqslant L$. The following definition and theorems, cited from (Kong et al., 1991, 1993), state in detail how the direction vector I test determines integer solutions of an interval equation under constant bounds and any given direction vectors.

Definition 2.4. Let a variable a_i be an integer $1 \leqslant i \leqslant n$. The positive part a_i^+ and the negative part a_i^- of an integer a_i are defined by $a_i^+ = \mathrm{MAX}\{a_i, 0\}$, and $a_i^- = \mathrm{MAX}\{-a_i, 0\}$.

Theorem 2.1. *Given the interval* equation (2.1) *subject to the constraints of* (1.3) *and a specific direction vector* $\vec{\theta} = (\theta_1, \ldots, \theta_d)$, *where d is the number of common loops and for all* k, $1 \leqslant k \leqslant d$, $\theta_k =<$. *Let*

$$t = \begin{cases} \max(|a_{2k-1}|, |a_{2k}|) & \text{if } a_{2k-1} * a_{2k} > 0 \\ \max(\min(|a_{2k-1}|, |a_{2k}|), |a_{2k-1} + a_{2k}|) & \text{if } a_{2k-1} * a_{2k} < 0. \end{cases}$$

If $t \leqslant U - L + 1$, *then the interval equation*

$$a_1X_1 + \cdots + a_{2d-1}X_{2d-1} + a_{2d}X_{2d} + \cdots + a_nX_n = [L, U]$$

is ($P_{2k,0} \leqslant X_{2k-1} < X_{2k} \leqslant Q_{2k,0}$ *for* $1 \leqslant k \leqslant d$, *and* $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ *for* $2d + 1 \leqslant r \leqslant n$)-*integer solvable if and only if the interval equation*

$$\begin{aligned} &a_1X_1 + \cdots + a_{2k-2}X_{2k-2} + a_{2k+1}X_{2k+1} + \cdots + a_nX_n \\ &\quad = [L - (a_{2k-1}^+ + a_{2k})^+(Q_{2k,0} - P_{2k,0} - 1) - (a_{2k-1} + a_{2k}) \\ &\qquad * P_{2k,0} - a_{2k}, U + (a_{2k-1}^- - a_{2k})^+(Q_{2k,0} - P_{2k,0} - 1) \\ &\qquad - (a_{2k-1} + a_{2k}) * P_{2k,0} - a_{2k}] \end{aligned}$$

is ($P_{2p,0} \leqslant X_{2p-1} < X_{2p} \leqslant Q_{2p,0}$ *for* $1 \leqslant p \leqslant d$, $p \neq k$, *and* $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$, $2d + 1 \leqslant r \leqslant n$)-*integer solvable.*

Proof. Refer to Kong et al. (1993). □

It is very obvious from Theorem 2.1 that the direction vector I test considers a pair of same index variables to justify the movement of the two variables to the right. It is indicated from Theorem 2.1 that a pair of same index variables in Eq. (2.1) can be moved to the right if the coefficients of the two variables have small enough values to justify the movement of the two variables to the right. If all coefficients for variables in Eq. (2.1) have no sufficiently small values to justify the movements of variables to the right, then Theorem 2.1 cannot be applied to result in the immediate movement. While every variable in Eq. (2.1) can not be moved to the right, Theorem 2.2 describes a transformation using the GCD test which enables additional variables to be moved.

Theorem 2.2. *Given the interval Eq.* (2.1) subject to the constraints of (1.3) *and a specific direction vector* $\vec{\theta} = (\theta_1, \ldots, \theta_d)$, *where d is the number of common loops and for all* k, $1 \leqslant k \leqslant d$, $\theta_k =<$. *Let* $g = gcd(a_1, \ldots, a_{n-1}, a_n)$. *The interval equation*

$$a_1X_1 + \cdots + a_{2d}X_{2d} + \cdots + a_nX_n = [L, U]$$

is ($P_{2k,0} \leqslant X_{2k-1} < X_{2k} \leqslant Q_{2k,0}$ *for* $1 \leqslant k \leqslant d$, *and* $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ *for* $2d + 1 \leqslant r \leqslant n$)-*integer solvable if and only if the interval equation*

$$\begin{aligned} &(a_1/g)X_1 + \cdots + (a_{2d}/g)X_{2d} + \cdots + a_nX_n \\ &\quad = [\lceil L/g \rceil, \lfloor U/g \rfloor] \end{aligned}$$

is ($P_{2k,0} \leqslant X_{2k-1} < X_{2k} \leqslant Q_{2k,0}$ *for* $1 \leqslant k \leqslant d$, *and* $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$, $2d + 1 \leqslant r \leqslant n$)-*integer solvable.*

Proof. Refer to Kong et al. (1993). □

2.3. The Lambda test

Geometrically, each linear equation in (1.1) defines a hyperplane π in R^n spaces. The intersection S of m hyperplanes corresponds to the common solutions to all linear equations in (1.1). Obviously, if S is empty then there is no data dependence. Inspecting whether S is empty is trivial in linear algebra (Vaughan, 1986). The bounds of (1.3) and any given direction vectors define a bounded convex set V in R^n. If any of the hyperplanes in (1.1) does not intersect V, then obviously S cannot intersect V. However, even if every hyperplane in (1.1) intersects V, it is still possible that S and V are disjoint. In Fig. 3 from Li et al. (1990), π_1 and π_2 are two such hyperplanes representing two linear equations in (1.1), each of which intersects V. But the intersection of π_1 and π_2 is outside of V. If a new hyperplane which contains the intersection of π_1 and π_2 is found but is disjoint from V, then S and V are immediately gathered not to intersect. In Fig. 3, π_3 is such a new hyperplane. If S and V are disjoint, then there exists a hyperplane which contains S and is disjoint from V. Furthermore, this hyperplane is a linear combination of hyperplanes in (1.1). On the other hand, if S and V intersect, then no such linear combination exists (Li et al., 1990).

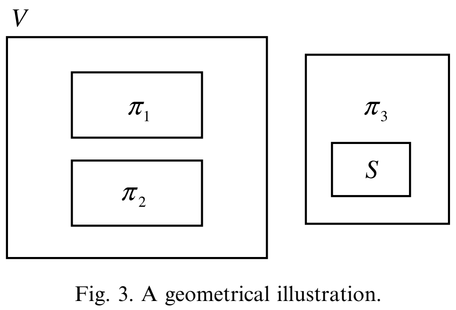

Fig. 3. A geometrical illustration.

In general, the Banerjee inequalities are first applied to test each hyperplane in (1.1). If every hyperplane intersects V, then the Lambda test is employed to simultaneously check every hyperplane.The Lambda test is an efficient and precise data dependence method to deal with (1.1) beneath V. The Lambda test is actually equivalent to the multi-dimensional version of the Banerjee inequality because it can determine simultaneous constrained real-valued solutions. The test forms linear combinations of references that eliminate one or more instances of index variables when direction vectors are not considered. While direction vectors are considered, the Lambda test generates new linear combinations that use a pair of relative index variables. Simultaneous constrained real-valued solutions exist if and only if the Banerjee inequalities find solutions in all the linear combinations generated (Li et al., 1990).

2.4. The multi-dimensional I test

The I test is an efficient and precise data dependence method to ascertain whether there exist *integer* solutions for one-dimensional arrays with constant bounds without direction vectors. The Lambda test is an efficient and precise data dependence method to check whether there exist *real* solutions for multi-dimensional arrays with constant bounds. In our previous work (Chang and Chu, 2000b), we extended the I test and the Lambda test and developed the multi-dimensional I test. The multi-dimensional I test can be applied towards testing whether there are *integer* solutions for multi-dimensional arrays with constant limits without direction vectors. Experiments with benchmark showed that the accuracy rate and the improvement rate for the multi-dimensional I test are approximately 29.6% and 19.2%, respectively.

3. The multi-dimensional direction vector I test

A data dependence problem is considered where subscripts are linear in terms of loop indexes. Bounds for subscripts are presumed to be *constant*. Given the data dependence problem as specified, the multi-dimensional direction vector I test examines a system of equalities and deduces whether the system has integer-valued solutions. In this section, the theoretical aspects and the worst-case time complexity of the multi-dimensional direction vector I test are provided.

Our method is best explained with the aid of a geometrical illustration. Suppose that F_i is the ith interval equation in a system of m interval equations with n unknown variables under the constraints of (1.3), where $1 \leqslant i \leqslant m$. It is assumed that m interval equations are written as:

$$\begin{aligned} &a_{1,1}X_1 + a_{1,2}X_2 + \cdots + a_{1,n-1}X_{n-1} + a_{1,n}X_n = [L_1, U_1],\\ &\qquad\vdots \\ &a_{m,1}X_1 + a_{m,2}X_2 + \cdots + a_{m,n-1}X_{n-1} + a_{m,n}X_n = [L_m, U_m], \end{aligned} \tag{3.1}$$

where each $a_{i,j}$ is a constant integer for $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant n$. It is postulated that the constraints to each variable in (3.1) are represented as

$$P_{r,0} \leqslant X_r \leqslant Q_{r,0}, \tag{3.2}$$

where $P_{r,0}$ and $Q_{r,0}$ are constant integers for $1 \leqslant r \leqslant n$. It is easy to see that the linear equations (1.1) are integer solvable if and only if the interval equations,

$$\begin{aligned} &a_{1,1}X_1 + a_{1,2}X_2 + \cdots + a_{1,n-1}X_{n-1} + a_{1,n}X_n = [a_{1,0}, a_{1,0}],\\ &\qquad\vdots \\ &a_{m,1}X_1 + a_{m,2}X_2 + \cdots + a_{m,n-1}X_{n-1} + a_{m,n}X_n = [a_{m,0}, a_{m,0}], \end{aligned}$$

are integer solvable.

Geometrically, the ith interval equation in (3.1) consists of $U_i - L_i + 1$ linear equations in which each linear equation is parallel each other. Hence, the ith interval equation in (3.1) contains $U_i - L_i + 1$ hyperplanes in which each hyperplane is parallel each other. The intersection S of m interval equations corresponds to the common solutions to all interval equations in (3.1). Obviously, if S is empty then there is no data dependence. The bounds of (3.2) and any given direction vectors define a bounded convex set V in R^n. If any of interval equations in (3.1) does not intersect V, then obviously S cannot intersect V. However, even if every interval equation in (3.1) intersects V, it is still possible that S and V are disjoint. It is assumed that two interval equations in (3.1), respectively, intersect V. But the intersection of them is outside of V. If one can find a new interval equation which contains S but is disjoint from V, then it immediately follows that S and V do not intersect. The following theorem is an extension of Theorem 1 in (Li et al., 1990) and guarantees that if S and V are disjoint, then there must be an interval equation which consists of S and is disjoint from V. Furthermore, this interval equation is a linear combination of equations in (3.1). On the other hand, if S and V intersect, then no such linear combination exists.

Theorem 1. *$S \cap V = \Phi$ if and only if there exists an interval equation, β, only consisting of a linear equations, which corresponds to a linear combination of equations in* (3.1):

$$\left\langle \sum_{i=1}^{m} \lambda_i * \vec{a}_i, \vec{X} \right\rangle = \left[\sum_{i=1}^{m} \lambda_i * a_{i,0}, \sum_{i=1}^{m} \lambda_i * a_{i,0} \right],$$

where $L_i \leqslant a_{i,0} \leqslant U_i$ for $1 \leqslant i \leqslant m$, such that $\beta \cap V = \Phi$. $\langle \vec{a}_i, \vec{X} \rangle$ denotes the inner product of $\vec{a}_i = (a_{i,1}, \ldots, a_{i,n})$ and $\vec{X} = (X_1, \ldots, X_n)$.

Proof. See Appendix A. □

An array $(\lambda_1, \ldots, \lambda_m)$ in Theorem 1 determines an interval equation that contains S. There are infinite number of such interval equations. The tricky part in the multi-dimensional direction vector I test is to examine as few interval equations as necessary to determine whether S and V intersect. We start from the case of $m = 2$, both for convenience of presentation and for practical importance of this case, as described above.

3.1. The case of two dimensional array references

In the case of two dimensional array references, two interval equations in (3.1) are $F_1 = [L_1, U_1]$ and $F_2 = [L_2, U_2]$, where $F_i = a_{i,1}X_1 + \cdots + a_{i,n}X_n$ for $1 \leqslant i \leqslant 2$. An arbitrary linear combination of the two interval equations can be written as $\lambda_1 F_1 + \lambda_2 F_2 = [\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}, \lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}]$, where $L_1 \leqslant a_{1,0} \leqslant U_1$ and $L_2 \leqslant a_{2,0} \leqslant U_2$. The domain of (λ_1, λ_2) is the whole R^2 space. Let $F_{\lambda_1,\lambda_2} = \lambda_1 F_1 + \lambda_2 F_2 = [\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}, \lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}]$, that is, $F_{\lambda_1,\lambda_2} = -(\lambda_1 a_{1,0} + \lambda_2 a_{2,0}) + (\lambda_1 a_{1,1} + \lambda_2 a_{2,1})X_1 + \cdots + (\lambda_1 a_{1,n} + \lambda_2 a_{2,n})X_n = 0$. By (Li et al., 1990), F_{λ_1,λ_2} is viewed in two ways. With (λ_1, λ_2) fixed, F_{λ_1,λ_2} is a linear function of $(X_1, \ldots, X_n)$ in R^n. With $(X_1, \ldots, X_n)$ fixed, it is a linear function of (λ_1, λ_2) in R^2. Furthermore, the coefficient of each variable in F_{λ_1,λ_2} is a linear function of (λ_1, λ_2) in R^2, i.e., $\Psi^{(i)} = \lambda_1 a_{1,i} + \lambda_2 a_{2,i}$ for $1 \leqslant i \leqslant n$. The equation $\Psi^{(i)} = 0$, $1 \leqslant i \leqslant n$, is called a Ψ equation. Each Ψ equation corresponds to a line in R^2, which is called a Ψ line. Each Ψ line separates the whole space into two closed halfspaces $\Psi_i^+ = \{(\lambda_1, \lambda_2) | \Psi^{(i)} \geqslant 0\}$ and $\Psi_i^- = \{(\lambda_1, \lambda_2) | \Psi^{(i)} \leqslant 0\}$ that intersect at the Ψ line. Let $\Phi_{(2k-1,2k)}$ be the sum of the coefficients of X_{2k-1} and X_{2k} in F_{λ_1,λ_2}, where X_{2k-1} and X_{2k} are related by a dependence direction, i.e., $\Phi_{(2k-1,2k)} = \lambda_1(a_{1,2k-1} + a_{1,2k}) + \lambda_2(a_{2,2k-1} + a_{2,2k})$ (Li et al., 1990). From Li et al. (1990), the equation $\Phi_{(2k-1,2k)} = 0$ is called a Φ equation. Each Φ equation corresponds to a Φ line in R^2. Each Φ line separates the whole space into two closed halfspaces $\Phi_i^+ = \{(\lambda_1, \lambda_2) | \Phi^{(i)} \geqslant 0\}$ and $\Phi_i^- = \{(\lambda_1, \lambda_2) | \Phi^{(i)} \leqslant 0\}$ that intersect at the Φ line.

A nonempty set $C \subset R^m$ is a cone if $\varepsilon\vec{\lambda} \in C$ for each $\vec{\lambda} \in C$ and $\varepsilon \geqslant 0$ (Vaughan, 1986). It is obvious that each cone contains the zero vector. Moreover, a cone that includes at least one nonzero vector $\vec{\lambda}$ must consists of the "ray" of $\vec{\lambda}$, namely, $\{\varepsilon\vec{\lambda} | \varepsilon \geqslant 0\}$. Such cones can be clearly viewed as the union of rays. There are at most $n\Psi$ lines and $n/2$ Φ lines, respectively. All Ψ lines and Φ lines divide R^2 space into at most $3n$ regions. Each region contains the zero vector. Any one nonzero element $\vec{\lambda}$ and the zero vector in the region forms the ray of $\vec{\lambda}$, namely, $\{\varepsilon\vec{\lambda} | \varepsilon \geqslant 0\}$. Therefore, each region can be viewed as the union of the rays. It is very obvious from the definition of a cone that each region is a cone (Vaughan, 1986).

In the following, Lemmas 3.1–3.3 are extended from Theorems and Lemmas in Kong et al. (1991, 1993) and Li et al. (1990). Definitions 3.1 and 3.2 are cited from Banerjee (1997, 1988) and Li et al. (1990) directly.

Lemma 3.1. *Suppose that a bounded convex set V is defined simply by the limits of* (3.2) *and a specific direction vector $\vec{\theta} = (\theta_1, \ldots, \theta_d)$, where d is the number of common loops and for all k, $1 \leqslant k \leqslant d$, $\theta_k \in \{<, =, >\}$. If $F_{\lambda_1,\lambda_2} = [\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}, \lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}]$ is $(P_{2k,0} \leqslant X_{2k-1}\theta_k X_{2k} \leqslant Q_{2k,0}$, for $1 \leqslant k \leqslant d$, and $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ for $2d+1 \leqslant r \leqslant n)$-integer solvable for every (λ_1, λ_2) in every Φ line, then $F_{\lambda_1,\lambda_2} = [\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}, \lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}]$ is also $(P_{2k,0} \leqslant X_{2k-1}\theta_k X_{2k} \leqslant Q_{2k,0}$, for $1 \leqslant k \leqslant d$, and $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ for $2d+1 \leqslant r \leqslant n)$-integer solvable for every (λ_1, λ_2) in R^2.*

Proof. See Appendix A. □

It is very obvious from Lemma 3.1 that the multi-dimensional direction vector I test considers a pair of same index variables to justify the movement of the two variables in one new interval equation to the right of the new interval equation. It is indicated from Lemma 3.1 and Theorem 2.1 that a pair of same index variables in one new interval equation can be moved to the right if the coefficients of the two variables in the new interval equation have small enough values to justify the movement of the two variables to the right. If all coefficients for variables in one new interval equation have no sufficiently small values to justify the movements of variables to the right of the new interval equation, then Lemma 3.1 and Theorem 2.1 cannot be applied to result in the immediate movement. While every variable in one new interval equation cannot be moved to the right, Lemma 3.2 describes a transformation using the GCD test which enables additional variables to be moved.

Lemma 3.2. *Suppose that a bounded convex set V is defined simply by the limits of* (3.2) *and a specific direction vector $\vec{\theta} = (\theta_1, \ldots, \theta_d)$, where d is the number of common loops and for all k, $1 \leqslant k \leqslant d$, $\theta_k \in \{<, =, >\}$. Let $g = \gcd(\lambda_1 a_{1,1} + \lambda_2 a_{2,1}, \ldots, \lambda_1 a_{1,n} + \lambda_2 a_{2,n})$. If $(1/g) * F_{\lambda_1,\lambda_2} = [\lceil(\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0})/g\rceil, \lfloor(\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0})/g\rfloor]$ is $(P_{2k,0} \leqslant X_{2k-1}\theta_k X_{2k} \leqslant Q_{2k,0}$, for $1 \leqslant k \leqslant d$, and $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ for $2d+1 \leqslant r \leqslant n)$-integer solvable for every (λ_1, λ_2) in every Φ line, then $(1/g) * F_{\lambda_1,\lambda_2} = [\lceil(\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0})/g\rceil, \lfloor(\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0})/g\rfloor]$ is also $(P_{2k,0} \leqslant X_{2k-1}\theta_k X_{2k} \leqslant Q_{2k,0}$ for $1 \leqslant k \leqslant d$, and $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ for $2d+1 \leqslant r \leqslant n)$-integer solvable for every (λ_1, λ_2)* in R^2.

Proof. Similar to Lemma 3.1. □

As a matter of fact, it suffices to test a single point in each Φ line for determining whether $F_{\lambda_1,\lambda_2} = [\lambda_1 * a_{1,0} +$

$\lambda_2 * a_{2,0}, \lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}]$ or $(1/g) * F_{\lambda_1,\lambda_2} = [\lceil(\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0})/g\rceil, \lfloor(\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0})/g\rfloor]$ is $(P_{2k,0} \leqslant X_{2k-1}\theta_k X_{2k} \leqslant Q_{2k,0}$, for $1 \leqslant k \leqslant d$, and $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ for $2d+1 \leqslant r \leqslant n)$-integer solvable for every (λ_1, λ_2) in those lines.

Lemma 3.3. *Suppose that a bounded convex set V is denoted by the limit of* (3.2) *and a specific direction vector* $\vec{\theta} = (\theta_1, \ldots, \theta_d)$, *where d is the number of common loops and for all k,* $1 \leqslant k \leqslant d$, $\theta_k \in \{<, =, >\}$. *Let* $g = \gcd(\lambda_1 a_{1,1} + \lambda_2 a_{2,1}, \ldots, \lambda_1 a_{1,n} + \lambda_2 a_{2,n})$. *Given a line in* R^2 *corresponding to an equation* $a\lambda_1 + b\lambda_2 = 0$, *if* $F_{\lambda_1,\lambda_2} = [\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}, \lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}]$ *or* $(1/g) * F_{\lambda_1,\lambda_2} = [\lceil(\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0})/g\rceil, \lfloor(\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0})/g\rfloor]$ *is* $(P_{2k,0} \leqslant X_{2k-1}\theta_k X_{2k} \leqslant Q_{2k,0}$ *for* $1 \leqslant k \leqslant d$, *and* $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ *for* $2d+1 \leqslant r \leqslant n)$*-integer solvable in* R^n *space for any fixed point* $(\lambda_1^0, \lambda_2^0) \neq (0,0)$ *in the line, then for every* (λ_1, λ_2) *in the line,* $F_{\lambda_1,\lambda_2} = [\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}, \lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}]$ or $(1/g) * F_{\lambda_1,\lambda_2} = [\lceil(\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0})/g\rceil, \lfloor(\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0})/g\rfloor]$ *is also* $(P_{2k,0} \leqslant X_{2k-1}\theta_k X_{2k} \leqslant Q_{2k,0}$ *for* $1 \leqslant k \leqslant d$, *and* $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ *for* $2d+1 \leqslant r \leqslant n)$*-integer solvable in* R^n *space.*

Proof. Similar to Lemma 3.1. □

Definition 3.1. Given an equation of the form $a\lambda_1 + b\lambda_2 = 0$, a canonical solution of the equation is defined as follows:

$(\lambda_1, \lambda_2) = (1, 0)$, if $a = 0$,

$(\lambda_1, \lambda_2) = (0, 1)$, if $b = 0$,

$(\lambda_1, \lambda_2) = (b, -a)$, if neither of a, b is zero,

$(\lambda_1, \lambda_2) = (1, 1)$, if both of a and b are zero.

Definition 3.2. The Λ set is denoted to be the set of all canonical solutions to Φ equations. Each element, (λ_1, λ_2), in the Λ set corresponds to one interval equation $F_{\lambda_1,\lambda_2} = [\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}, \lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}]$.

There are at most $n/2$ Φ equations if V is denoted by the bounds of (3.2) and any given direction vectors. Each of Φ equations generates a canonical solution according to Definition 3.1. Each canonical solution forms a new interval equation, only containing the only linear equation in light of Definition 3.2. Obviously, new interval equations tested are at most $n/2$ if V is defined by the constraints of (3.2) and any given direction vectors.

The multi-dimensional direction vector I test is employed to simultaneously check every interval equation. The multi-dimensional direction vector I test examines the subscripts from two dimensions, and then figures out the Λ set from Φ equations. Each element in the Λ set determines a new interval equation. The new interval equation is tested to see if it intersects V, by moving variables on left-hand side of one interval equation to right-hand side of the interval equation as done in the direction vector I test for testing each *single* dimension.

We now use the following example to explain the power of the multi-dimensional direction vector I test.

Consider the do-loop in Fig. 1. If we want to determine whether there exists output data dependence of the array A with direction vector $(<, <)$ in Fig. 1, then the linear expressions of the array A with direction vector $(<, <)$ can be transformed into the following linear equations:

$10 \times X_1 - 10 \times X_2 + X_3 - X_4 = 0,$
$10 \times X_1 - 10 \times X_2 + X_3 - X_4 = 0,$

subject to the constant bounds $1 \leqslant X_1$ and $X_2 \leqslant 100$, and $1 \leqslant X_3$ and $X_4 \leqslant 10$, and the limits of a direction vector $X_1 < X_2$ and $X_3 < X_4$.

If the multi-dimensional direction vector I test is applied to resolve the problem, the Φ equations are generated. The Φ equation is $0 * \lambda_1 + 0 * \lambda_2 = 0$. The Φ equation has one canonical solutions $(1, 1)$ in light of Definition 3.1. According to Definition 3.2, the canonical solution $(1, 1)$ yields the following interval equations

$$20 \times X_1 - 20 \times X_2 + 2X_3 - 2X_4 = [0, 0]. \quad \text{(Ex1)}$$

Now the multi-dimensional direction vector I test applies the direction vector I test (Kong et al., 1993) to resolve the interval equation (Ex1). Because there are no sufficiently small values to justify the movement of variables to the right in light of Lemma 3.1 and Theorem 2.1, the multi-dimensional direction vector I test applies Lemma 3.2 to resolve the interval equation (Ex1), and then gains the new interval equation

$$10 \times X_1 - 10 \times X_2 + X_3 - X_4 = [0, 0]. \quad \text{(Ex2)}$$

Next, the multi-dimensional direction vector I test applies Lemma 3.1 and Theorem 2.1 to resolve the interval equation (Ex2), and then the pair of terms X_3 and X_4 in the interval equation (Ex2) is moved to the right-hand side of (Ex2) to gain the new interval equation

$$10 \times X_1 - 10 \times X_2 = [1, 9]. \quad \text{(Ex2.1)}$$

Now the length of the right-hand side interval has been increased to 9. But there is no suitable value for t to justify the movement of the two terms X_1 and X_2 to the right according to Lemma 3.1 and Theorem 2.1. So Lemma 3.2 is again employed to resolve the interval equation (Ex2.1), and then acquires the new interval equation

$$X_1 - X_2 = [1, 0]. \quad \text{(Ex2.2)}$$

Because $1 \leqslant 0$ is false, it is right away inferred that the interval equation (Ex2) is not integer solvable. Therefore, the multi-dimensional direction vector I test in light of Lemmas 3.1–3.3 infers that there is no *integer*-valued solution.

3.2. The case of multi-dimensional array references

We take account of m interval equations in (3.1) with $m > 2$ for generalizing the multi-dimensional direction vector I test. All m interval equations are assumed to be connected; otherwise they can be partitioned into smaller systems. As stated before, we can hypothesize that there are no redundant equations. An arbitrary linear combination of m interval equations in (3.1) can be written as

$$F_{\lambda_1,\dots,\lambda_m} = \left\langle \sum_{i=1}^{m} \lambda_i * \vec{a}_i, \vec{X} \right\rangle = \left[\sum_{i=1}^{m} \lambda_i * a_{i,0}, \sum_{i=1}^{m} \lambda_i * a_{i,0} \right],$$

where $L_i \leqslant a_{i,0} \leqslant U_i$ for $1 \leqslant i \leqslant m$ and $\langle \vec{a}_{\underrightarrow{i}}, \vec{X} \rangle$ denotes the inner product of $\vec{a}_i = (a_{i,1}, \dots, a_{i,n})$ and $X = (X_1, \dots, X_n)$.

Assume that $g = \gcd(\sum_{i=1}^{m} \lambda_i * a_{i,1}, \dots, \sum_{i=1}^{m} \lambda_i * a_{i,n})$. It is to be determined whether

$$F_{\lambda_1,\dots,\lambda_m} = \left[\sum_{i=1}^{m} \lambda_i * a_{i,0}, \sum_{i=1}^{m} \lambda_i * a_{i,0} \right]$$

or

$$(1/g) * F_{\lambda_1,\dots,\lambda_m} = \left[\left\lceil \left(\sum_{i=1}^{m} \lambda_i * a_{i,0} \right) \Big/ g \right\rceil, \left\lfloor \left(\sum_{i=1}^{m} \lambda_i * a_{i,0} \right) \Big/ g \right\rfloor \right]$$

is ($P_{2k,0} \leqslant X_{2k-1}\theta_k X_{2k} \leqslant Q_{2k,0}$ for $1 \leqslant k \leqslant d$, and $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ for $2d+1 \leqslant r \leqslant n$)-integer solvable in R^n space for arbitrary $(\lambda_1, \dots, \lambda_m)$, where θ_k is one of three different direction vectors and d is the number of common loops. By (Li et al., 1990), the coefficient of each variable in $F_{\lambda_1,\dots,\lambda_m}$ is a linear function of $(\lambda_1, \dots, \lambda_m)$ in R^m, which is $\Phi^{(i)} = \sum_{j=1}^{m} \lambda_j (a_{j,2k-1} + a_{j,2k})$ for $1 \leqslant i \leqslant d$. The equation $\Phi^{(i)} = 0$, $1 \leqslant i \leqslant d$, is called a Φ equation. A Φ equation corresponds to a hyperplane in R^m, called a Φ plane. Each Φ plane divides the whole space into two closed halfspaces $\Omega_i^+ = \{(\lambda_1, \dots, \lambda_m) | \Phi^{(i)} \geqslant 0\}$ and $\Omega_i^- = \{(\lambda_1, \dots, \lambda_m) | \Phi^{(i)} \leqslant 0\}$. If V is defined by the constraints of (3.2) and any given direction vectors, then a nonempty set $\bigcap_{i=1}^{n} \Omega_i$, where $\Omega_i \in \{\Omega_i^+, \Omega_i^-\}$, is called a λ region. Every λ region is a cone in R^m space (Li et al., 1990; Vaughan, 1986). The λ regions in R^m space have several lines as the frame of their boundaries. Each line (called a λ line) is the intersection of some Φ equations (Li et al., 1990; Vaughan, 1986).

In the following, Lemmas 3.4 and 3.5 are an extension of Theorems and Lemmas in (Kong et al., 1993; Li et al., 1990), respectively.

Lemma 3.4. *Suppose that a bounded convex set V is defined simply by the limits of* (3.2) *and a specific direction vector* $\vec{\theta} = (\theta_1, \dots, \theta_d)$, *where d is the number of common loops and for all k,* $1 \leqslant k \leqslant d$, $\theta_k \in \{<, =, >\}$. *Let* $g = \gcd(\sum_{i=1}^{m} \lambda_i * a_{i,1}, \dots, \sum_{i=1}^{m} \lambda_i * a_{i,n})$. *If*

$$F_{\lambda_1,\dots,\lambda_m} = \left[\sum_{i=1}^{m} \lambda_i * a_{i,0}, \sum_{i=1}^{m} \lambda_i * a_{i,0} \right]$$

or

$$(1/g) * F_{\lambda_1,\dots,\lambda_m} = \left[\left\lceil \left(\sum_{i=1}^{m} \lambda_i * a_{i,0} \right) \Big/ g \right\rceil, \left\lfloor \left(\sum_{i=1}^{m} \lambda_i * a_{i,0} \right) \Big/ g \right\rfloor \right]$$

is ($P_{2k,0} \leqslant X_{2k-1}\theta_k X_{2k} \leqslant Q_{2k,0}$, *for* $1 \leqslant k \leqslant d$, *and* $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ for $2d+1 \leqslant r \leqslant n$)*-integer solvable for every* $(\lambda_1, \dots, \lambda_m)$ *in every λ line, then*

$$F_{\lambda_1,\dots,\lambda_m} = \left[\sum_{i=1}^{m} \lambda_i * a_{i,0}, \sum_{i=1}^{m} \lambda_i * a_{i,0} \right]$$

or

$$(1/g) * F_{\lambda_1,\dots,\lambda_m} = \left[\left\lceil \left(\sum_{i=1}^{m} \lambda_i * a_{i,0} \right) \Big/ g \right\rceil, \left\lfloor \left(\sum_{i=1}^{m} \lambda_i * a_{i,0} \right) \Big/ g \right\rfloor \right]$$

is also ($P_{2k,0} \leqslant X_{2k-1}\theta_k X_{2k} \leqslant Q_{2k,0}$ *for* $1 \leqslant k \leqslant d$, *and* $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ for $2d+1 \leqslant r \leqslant n$)*-integer solvable for every* $(\lambda_1, \dots, \lambda_m)$ *in* R^m *space.*

Proof. Similar to Lemma 3.1. □

Lemma 3.5. *Given a line in* R^m *which crosses the origin of the coordinates and let* $g = \gcd(\sum_{i=1}^{m} \lambda_i * a_{i,1}, \dots, \sum_{i=1}^{m} \lambda_i * a_{i,n})$. *If*

$$F_{\lambda_1,\dots,\lambda_m} = \left[\sum_{i=1}^{m} \lambda_i * a_{i,0}, \sum_{i=1}^{m} \lambda_i * a_{i,0} \right]$$

or

$$(1/g) * F_{\lambda_1,\dots,\lambda_m} = \left[\left\lceil \left(\sum_{i=1}^{m} \lambda_i * a_{i,0} \right) \Big/ g \right\rceil, \left\lfloor \left(\sum_{i=1}^{m} \lambda_i * a_{i,0} \right) \Big/ g \right\rfloor \right]$$

is ($P_{2k,0} \leqslant X_{2k-1}\theta_k X_{2k} \leqslant Q_{2k,0}$ *for* $1 \leqslant k \leqslant d$, *and* $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ *for* $2d+1 \leqslant r \leqslant n$)*-integer solvable in* R^n *space for any fixed point* $(\lambda_1^0, \dots, \lambda_m^0) \neq (0, \dots, 0)$ *in the line, then for every* $(\lambda_1, \dots, \lambda_m)$ *in the line,*

$$F_{\lambda_1,\dots,\lambda_m} = \left[\sum_{i=1}^{m} \lambda_i * a_{i,0}, \sum_{i=1}^{m} \lambda_i * a_{i,0} \right]$$

or

$$(1/g) * F_{\lambda_1,\dots,\lambda_m} = \left[\left\lceil \left(\sum_{i=1}^{m} \lambda_i * a_{i,0} \right) \Big/ g \right\rceil, \left\lfloor \left(\sum_{i=1}^{m} \lambda_i * a_{i,0} \right) \Big/ g \right\rfloor \right]$$

is also ($P_{2k,0} \leqslant X_{2k-1}\theta_k X_{2k} \leqslant Q_{2k,0}$, *for* $1 \leqslant k \leqslant d$, *and* $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ *for* $2d+1 \leqslant r \leqslant n$)*-integer solvable in* R^n *space.*

Proof. Similar to Lemma 3.3. □

The detail of the multi-dimensional direction vector I test in the general case is not considered since the discussion is similar to the case of $m = 2$.

3.3. The algorithm

We now summarize the illustration into an algorithm. The algorithm is described below.

Input: M interval equation (3.1), the constraints (3.2) to each variable in (3.1) and a set of λ values $(\lambda_1, \ldots, \lambda_m)$.

Output:

no: Eq. (3.1) under the constraints (3.2) have no *integer valued* solutions.

yes: Eq. (3.1) under the constraints (3.2) have *integer valued* solutions.

maybe: The proposed method cannot conclude whether Eq. (3.1) under the constraints (3.2) have *integer valued* solutions.

Method:

Step 1. According to $(\lambda_1, \ldots, \lambda_m)$, we can obtain a new interval equation,

$$F_{\lambda_1,\ldots,\lambda_m} = \left\langle \sum_{i=1}^{m} \lambda_i * \vec{a}_i, \vec{X} \right\rangle = \left[\sum_{i=1}^{m} \lambda_i * a_{i,0}, \sum_{i=1}^{m} \lambda_i * a_{i,0} \right],$$

where $\langle \vec{a}_i, \vec{X} \rangle$ denotes the inner product of $\vec{a}_i = (a_{i,1}, \ldots, a_{i,n})$ and $\vec{X} = (X_1, \ldots, X_n)$.

Step 2. The direction vector I test is applied to deal with the new interval equation.

Step 3. If the direction vector I test finds there exists an integer solution, then a result of **yes** is returned and the processing is terminated. Otherwise, go to Step 4.

Step 4. If the direction vector I test determines there exist no integer solutions, then a result of **no** is returned and the processing is terminated. Otherwise, go to Step 5.

Step 5. The direction vector I test cannot determine if there exists an integer solution. A result of **maybe** is returned and the processing is terminated.

If the multi-dimensional direction vector I test returns a result of **yes** or **no**, then that result is accurate; i.e., a returned value of **yes** means that the equations have integer-valued solutions and a returned value of **no** means that the equations have no integer-valued solutions. A returned value of **maybe**, on the other hand, means that the multi-dimensional direction vector I test does not derive whether the equations have *integer valued* solutions.

3.4. Time complexity

The main phases for the multi-dimensional direction vector I test include: (1) calculating λ values and (2) examining each interval equation. λ values are easily determined according to Φ equations and Definition 3.1. It is clear that the time complexity to computing a λ value is $O(y)$ from Definition 3.1, where y is a constant. Each λ value corresponds to an interval equation. Each interval equation is tested to see if it intersects V, by moving the pairs of variables in left-hand side of one interval equation to right-hand side of the interval equation as done in the direction vector I test for one *single* dimension. The worst-case time complexity of the direction vector I test is $O(n^{2*}y + n^{*}y)$ (Kong et al., 1991, 1993), where n is the number of variables in interval equations. Hence, the time complexity of for the multi-dimensional direction vector I test examining an interval equation is at once derived to be $O(n^{2*}y + n^{*}y + y)$. The number of interval equations checked in the multi-dimensional direction vector I test is at most

$$\prod_{i=1}^{m}(U_i - L_i + 1) * \binom{\frac{n}{2}}{m-1},$$

where m is the number of *original* references and L_i and U_i are lower and upper bounds in right-hand side of *original* interval equations for $1 \leqslant i \leqslant m$, in light of statements in Sections 3.1 and 3.2 and (Li et al., 1990). Therefore, the worst-case time complexity for the multi-dimensional direction vector I test is immediately inferred to be

$$O\left(\begin{bmatrix} \frac{n}{2} \\ m-1 \end{bmatrix} * (n^2 * y + n * y + y) * \left(\prod_{i=1}^{m}(U_i - L_i + 1) \right) \right).$$

Two-dimensional arrays with linear subscripts appear quite frequently in real programs, as clearly indicated from statements in Section 1. Lower and upper bounds are the same in right-hand side of *original* interval equations in original references in real programs. Therefore, the number of interval equations examined in each two-dimensional array tested is at most $n/2$ according to statements in Section 3.1 and (Li et al., 1990). If the multi-dimensional direction vector I test is applied to deal with the array, then their worst-case time complexity is $O((n/2) * (n^2 * y + n * y + y))$. The worst-case time complexity for the Lambda test dealing with the same array is $O((3n/2) * (n + y))$. However, in general, the efficiency of the multi-dimensional direction vector I test is only slightly poorer than that of the Lambda test and the direction vector I test because the number of variables, n, in the interval equation tested is generally very small.

4. Experimental Results

We tested the multi-dimensional direction vector I test and performed experiments on Personal Computer Intel Pentium III through the benchmark codes cited from five numerical packages EISPACK, LINPACK, Parallel Loops, Livermore Loops and Vector Loops (Smith, 1976; Dongarra et al., 1991; Levine et al., 1991). 17,433 pairs of array references with the same pair of array references but with different direction vectors were found to have coupled subscripts (coupled references are groups of reference positions sharing one or more index variables (Li et al., 1990; Wolfe and Tseng, 1992)). Meanwhile, it is also found that all of the lower bounds are constants and all of the upper bounds are *unknown* variables at *compile* time. Therefore, those *symbolic* upper bounds are assumed to be constants 100. The choice of 100 as the upper bound is arbitrary. In (Petersen, 1993) it is reported that for the Perfect Benchmarks data dependence testing results (number of dependencies, independencies and unanalyzable subscripts) from the original unknown loop bounds are quite close to that from the assumed constant loop bounds. This implies that our assumption to the constant bounds does not change the dependence features existed in the original benchmarks. The multi-dimensional I test is only applied to test those arrays with linear subscripts and under constant bounds.

The results obtained (Table 1) reveals the multi-dimensional direction vector I test determined that there were *definite* (*yes* or *no*) results for 3765 pairs of coupled arrays under constant bounds. The multi-dimensional direction vector I test in this experiment is only applied to test those arrays with coupled subscripts and under constant bounds, and it found in total 3765 cases that had *definite* (*yes* or *no*) results. The "accuracy rate" in Table 1 refers to, when given a set of coupled subscripts with constant bounds, how often the multi-dimensional direction vector I test detects a case where there is a *definite* (*yes* or *no*) result. Let b be the number of the coupled subscripts with constant bounds found in our experiments, and let c be the number that is detected to have *definite* (*yes* or *no*) results. Thus the accuracy rate is denoted to be equal to c/b. In our experiments, 11,312 pairs of array references were found to have coupled subscripts and constant bounds, and 3765 of them were found to have *definite* (*yes* or *no*) results. So the accuracy rate for the multi-dimensional direction vector I test was about 33.3%. Similarly, the "improvement rate" refers to how often the multi-dimensional direction vector I test gives a definitive (yes/no) result for a set of coupled subscripts with constant, variable, and unknown bounds. Let d be the number of the coupled subscripts with constant, variable, and symbolic bounds found in our experiments. Thus the improvement rate is denoted to be equal to c/d. In our experiments, 17,433 pairs of array references were found to have coupled subscripts with constant, variable and symbolic bounds, and 3765 of them were found to have *definite* (*yes* or *no*) results. So the improvement rate for the multi-dimensional direction vector I test was equal to 21.6%.

In our experiments, it is found that there are different frequencies of coupled subscripts in different benchmark codes. Improvement rate for each benchmark is shown in Table 2 in which each row shows how many cases for each benchmark were checked to have definitive results. For instance, the first row reveals that 7722 pairs of array references with the same pairs of array references but with different direction vectors from EISPACK were found to have coupled subscripts, and 975 of them were found to have definitive results. Therefore, the improvement rate of the multi-dimensional direction vector I test for EISPACK is 12.6%. For all of benchmark codes in our experiments, the improvement rate to each benchmark was from 12.6% to 48.4%. This indicates that for multi-dimensional arrays with coupled sub-

Table 1
Testing capability of the multi-dimensional direction vector I test for 17,433 pairs of benchmark array references with the same pairs of array references but with different direction vectors

Pairs of arrays with *definitive* results			Accuracy rate	Improvement rate
Constant bounds (11,312)[a]	Variable bounds (6121)[a]	Overall (17,433)[a]		
3765	Not applicable	3765	33.3%	21.6%

[a] Number of array reference pairs.

Table 2
The improvement rate of the multi-dimensional direction vector I test for array references in different benchmark

Benchmark	Pairs of arrays checked	Pairs of examined arrays with definitive results	Improvement rate (%)
EISPACK	7722	975	12.6
LINPACK	1458	331	22.7
Parallel loops	7867	2273	28.9
Livermore loops	215	104	48.4
Vector loops	171	82	47.9

scripts the improvement rate of the multi-dimensional direction vector I test varies with the benchmark tested.

The Omega test analyzer and the Power test analyzer, we implemented them according to the proposed algorithm in (Pugh, 1992; Wolfe and Tseng, 1992), were also employed to resolve the same benchmark. The results obtained (Table 3) reveals the Omega test determined that there were definitive results (*yes* or *no*) for 7232 pairs of coupled arrays under constant bounds and 3871 pairs of coupled arrays under variable bounds. The results obtained (Table 3) also shows the Power test determined that there were definitive results (*no*) for 6722 pairs of coupled arrays under constant bounds and 3826 pairs of coupled arrays under variable bounds. The results reflect that the Omega test and the Power test are more precise than the multi-dimensional direction vector I test. This is because that the coefficients of a number of loop index variables in coupled arrays checked in the benchmark are equal to *zero*, causing the multi-dimensional direction vector I test fail to be used under such a condition. Besides, in our experiments, it is also found that using the multi-dimensional direction vector I test to analyze the dependence for those array elements with subscripts transformed from induction variable substitution is quite precise and efficient.

Suppose that K_{MDVI}, K_{P} and k_{O} are the execution time to treat data dependence problem of a coupled-subscript array for the multi-dimensional direction vector I test, the Power test and the Omega test, subsequently. Table 4 shows the speed-up of the multi-dimensional direction vector I test against the Power test and the Omega test. Each row in Table 4 shows how many times the execution time of the Power test and the Omega test took longer than the execution time of the multi-dimensional direction vector I test. For example, the first row shows that there were 48 subroutines in which the execution time of the Power test took from 4.1 to 9.7 times longer than that of the multi-dimensional direction vector I test. The third row shows that there were 57 subroutines in which the execution time of the Omega test took from 5.2 to 10.8 times longer than that of the multi-dimensional direction vector I test. In our experiments, the execution time per analysis of the Power test and the Omega test was indicated to take from 4.1 to 16.3 times and from 5.2 to 18.7 times longer than that of the multi-dimensional direction vector I test, respectively.

Table 4
The speed-up of the multi-dimensional direction vector I test when compared with the Power test and the Omega test for 3765 pairs of coupled references with constant bounds

	Speed-up	Total number of subroutines involved
$K_{\mathrm{P}}/K_{\mathrm{MDVI}}$	4.1–9.7	48
$K_{\mathrm{P}}/K_{\mathrm{MDVI}}$	10.5–16.3	72
$K_{\mathrm{O}}/K_{\mathrm{MDVI}}$	5.2–10.8	57
$K_{\mathrm{O}}/K_{\mathrm{MDVI}}$	12.3–18.7	63

The superiority of testing efficiency of the multi-dimensional direction vector I test over that of the Omega test for the stated dependence problem can also be deduced from time complexity analysis. The Omega test based on the least remainder algorithm, a variation of Euclid's algorithm, and Fourier's elimination method (Banerjee, 1997; Pugh, 1992) consists of three major computations: eliminating equality constraints, eliminating variables in inequality constraints, and finding integer solutions (that is an integer programming problem). The time complexity for these steps are $O(mn\log|c| + mnp + mn)$, $O(n^2s^2)$ and $O(k^n)$ (Banerjee, 1997; Pugh, 1992), respectively, where m, n, c, p, s, k denote the number of equality constraints, the number of variables, the coefficient with the largest absolute value in equality constraints, the number of passes to eliminate all the variables that become unbound, the number of inequality constraints, and the absolute value of coefficient of variable in inequality constraints, subsequently. So the overall time complexity of the Omega test is $O(mn\log|c| + mnp + mn + n^2s^2 + k^n)$. Obviously, the multi-dimensional direction vector I test is significantly superior to that of the Omega test in terms of testing efficiency. In (Pugh, 1992) it is reported that the Omega test has *exponential* worst-case time complexity. Wolfe (Wolfe and Tseng, 1992) and Triolet (Triolet et al., 1986) also found that Fourier–Motzkin variable elimination for dependence testing takes from 22 to 28 times longer than Banerjee method, a part of the multi-dimensional direction vector I test.

The study in Pugh (1992) stated that (1) the cost of scanning array subscripts and loop bounds to build a

Table 3
Testing capability of the Omega test and the Power test for the same benchmark

Testing methods	Pairs of arrays with *definitive* results			Accuracy rate for constant bounds (%)	Improvement rate (%)
	Constant bounds (11,312)[a]	Variable bounds (6121)[a]	Overall (17,433)[a]		
Omega test	7232	3871	11,103	65.1	63.7
Power test	6722	3826	10,548	59.4	60.5

[a] Number of array pairs.

dependence problem was typically 2–4 times of the copying cost (the cost of building a system of dependence equations) for the problem, and (2) the dependence analysis cost for more than half of *simple* arrays tested was typically 2–4 times of the copying cost, but the dependence analysis cost to other simple arrays and all of the *regular*, *convex* and *complex* arrays tested was more than 4 times of the copying cost. Based on such results, we can estimate and conclude that the analysis cost of data dependence for parallelizing/vectorizing compilation occupies generally about 30–60% of total compiling time. In other words, improvements on dependence testing performance could result in significant compiling performance of a parallelizing/vectorizing compiler.

5. Conclusions

The multi-dimensional direction vector I test can ascertain whether integer-valued solutions exist when testing multi-dimensional array references with linear subscripts and constant bounds. Like the direction vector I test, the multi-dimensional direction vector I test is based on moving the pairs of related variables on left-hand side of an interval equation to right-hand side of the interval equation. The multi-dimensional direction vector I test is exactly equivalent to a version of the Lambda test with generating integer solution under constant bounds and any given direction vectors because it can determine simultaneous constrained integer-valued solutions.

The Power test is a combination of Fourier–Motzkin variable elimination with an extension of Euclid's GCD algorithm (Wolfe and Tseng, 1992, 1996). The Omega test combines new methods for eliminating equality constraints with an extension of Fourier–Motzkin variable elimination (Pugh, 1992). The two tests currently have the highest precision and the widest applicable range in the field of data dependence analysis for testing arrays with linear subscripts. However, the cost of the two tests is very expensive because the worst-case of Fourier–Motzkin variable elimination is exponential in the number of free variables (Banerjee, 1997; Pugh, 1992; Wolfe and Tseng, 1992). It is found in our experiment that the Omega test and the Power test take 5.2–18.7 times and 4.1–16.3 times longer, respectively in execution than the multi-dimensional direction vector I test when testing the dependence of multi-dimensional arrays.

The multi-dimensional direction vector I test integrates the Lambda test and the direction vector I test and, according to the time complexity analysis, only has slightly poorer efficiency than that of the Lambda test and the direction vector I test. Therefore, depending on the application domains, the multi-dimensional direction vector I test can be applied independently or together with the Power test and the Omega test to analyze data dependence for linear-subscript multi-dimensional array references.

Appendix A

Proof of Thoerem 1. ($\Leftarrow$) The interval equation, β, contains S and is disjoint from V. So we can immediately derive that S is disjoint from V.

($\Rightarrow$) For the convenience of the proof, (3.1) are rewritten as $A * \vec{Y} = O$, where

$$A = \begin{pmatrix} -a_{1,0} & a_{1,1} & \cdots & a_{1,n} \\ \vdots & \vdots & \vdots & \vdots \\ -a_{m,0} & a_{m,1} & \cdots & a_{m,n} \end{pmatrix}_{(m)*(n+1)},$$

$$\vec{Y} = \begin{pmatrix} 1 \\ X_1 \\ \vdots \\ X_n \end{pmatrix}_{(n+1)*1},$$

O is a $m*1$ zero matrix, and $L_i \leqslant a_{i,0} \leqslant U_i$ for $1 \leqslant i \leqslant m$. We can let $S = \{(X_1, \ldots, X_n) : A\vec{Y} = \mathrm{O}\}$, $V = \{(X_1, \ldots, X_n) : P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ for $1 \leqslant r \leqslant n$ and X_{2k-1} and X_{2k} satisfy constraints of direction vectors for for $1 \leqslant k \leqslant d$, where d is the number of common loops.$\}$, $S' = \{(1, X_1, \ldots, X_n) : \forall (X_1, \ldots, X_n) \in S\}$, and $V' = \{(1, X_1, \ldots, X_n) : \forall (X_1, \ldots, X_n) \in \mathrm{V}\}$. Because $S \cap V = \Phi$, we can infer $S' \cap V' = \Phi$.

We let $\alpha = Span(\vec{b}_1, \ldots, \vec{b}_m)$, where $\vec{b}_i = (-a_{i,0}, a_{i,1}, \ldots, a_{i,n})$. $\forall \vec{C} \in \alpha$ and $\vec{D} \in S'$, we can obtain the inner product of $\vec{C}$ and $\vec{D}$ as follows

$$\begin{aligned} \left\langle \vec{C}, \vec{D} \right\rangle &= \left\langle \sum_{i=1}^{m} \lambda_i * \vec{b}_i, \vec{D} \right\rangle \\ &= \lambda_1(-a_{1,0} + a_{1,1}X_1 + \cdots + a_{1,n}X_n) \\ &\quad + \cdots + \lambda_m(-a_{m,0} + a_{m,1}X_1 + \cdots + a_{m,n}X_n) \\ &= \lambda_1(0) + \cdots + \lambda_m(0) = 0. \end{aligned}$$

Therefore, we can at once derive that α is the orthogonal complementary space of S'. For any $\vec{Z}$ in V', consider $\vec{P}_Z$, the projection of $\vec{Z}$ on S'. Since $\|\vec{P}_Z - \vec{Z}\|$ is a continuous function on V' and V' is bounded, there must be exist $\vec{Z}_0$ in V' such that $\|\vec{P}_{Z_0} - \vec{Z}_0\| = \min_{\vec{Z} \in V'} \|\vec{P}_Z - \vec{Z}\|$. This is the minimum distance between S' and V'. Since $\vec{Z}_0 - \vec{P}_{Z_0}$ is orthogonal to S', it must be in α. Hence, the equation $\langle \vec{Z}_0 - \vec{P}_{Z_0}, \vec{D} \rangle = 0$ is a linear combination of equations in (3.1), i.e., $\vec{Z}_0 - \vec{P}_{Z_0} = \lambda_1 * \vec{b}_1 + \cdots + \lambda_m * \vec{b}_m$. The equation $\langle \vec{Z}_0 - \vec{P}_{Z_0}, \vec{D} \rangle = 0$ is actually equal to $\langle \sum_{i=1}^{m} \lambda_i * \vec{a}_i, \vec{X} \rangle = \sum_{i=1}^{m} \lambda_i * a_{i,0}$. Therefore, the equation, $\langle \sum_{i=1}^{m} \lambda_i * \vec{a}_i, \vec{X} \rangle = \sum_{i=1}^{m} \lambda_i * a_{i,0}$, is transformed to one new interval equation,

$$\left\langle \sum_{i=1}^{m} \lambda_i * \vec{a}_i, \vec{X} \right\rangle = \left[\sum_{i=1}^{m} \lambda_i * a_{i,0}, \sum_{i=1}^{m} \lambda_i * a_{i,0} \right].$$

Let β be the new interval equation. Hence, we can immediately conclude that the new interval equation, β, which contains S. According to Claim 1 cited from (Li et al., 1990) in the following, $\langle \vec{Z}_0 - \vec{P}_{Z_0}, \vec{Z} \rangle > 0$ for any $\vec{Z}$ in V'. This is to say that each element $\vec{X}$ in V satisfies $\langle \sum_{i=1}^{m} \lambda_i * \vec{a}_i, \vec{X} \rangle > \sum_{i=1}^{m} \lambda_i * a_{i,0}$. Therefore, we can at once derive $\beta \cap V = \Phi$. □

Claim 1. $\langle \vec{Z}_0 - \vec{P}_{Z_0}, \vec{Z} \rangle > 0$ for any $\vec{Z}$ in V'.

Proof. Refer to Li et al. (1990). □

Proof of Lemma 3.1. (1) The direction vector of the kth common loop, θ_k, has three different cases, i.e., θ_k is one of three elements in the set $\{<, =, >\}$. For convenience of proof, we first prove that θ_k is the direction vector '<'. From the direction vector I test in (Kong et al., 1993), because $F_{\lambda_1,\lambda_2} = [\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}, \lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}]$ is ($P_{2k,0} \leqslant X_{2k-1} < X_{2k} \leqslant Q_{2k,0}$ for $1 \leqslant k \leqslant d$, and $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ for $2d+1 \leqslant r \leqslant n$)-integer solvable for every (λ_1, λ_2) in every Φ line, there must be at least one element in V such that $F_{\lambda_1,\lambda_2} - (\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}) = 0$.

(2) We have that $F_{\lambda_1,\lambda_2} - (\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}) = 0$ for any point (λ_1, λ_2) on every Φ line according to the assumption of the lemma. It is immediately concluded that $F_{\lambda_1,\lambda_2} = [\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}, \lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}]$ is ($P_{2k,0} \leqslant X_{2k-1} < X_{2k} \leqslant Q_{2k,0}$, for $1 \leqslant k \leqslant d$, and $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ for $2d+1 \leqslant r \leqslant n$)-integer solvable for every point (λ_1, λ_2) on the boundaries of each cone.

(3) Every point in each cone can be expressed as a linear combination of some points on the boundary of the same cone, as being a well-known fact in the convex theory. Any point (λ_5, λ_6) in a cone is assumed to be capable of being represented as $(\varepsilon\lambda_1 + \tau\lambda_3, \varepsilon\lambda_2 + \tau\lambda_4)$, where (λ_1, λ_2) and (λ_3, λ_4) are points in the boundary of the cone and $\varepsilon \geqslant 0$ and $\tau \geqslant 0$. Because

$$\begin{aligned}
&F_{\lambda_5,\lambda_6}(X_1, \ldots, X_n) - (\lambda_5 a_{1,0} + \lambda_6 a_{2,0}) \\
&\quad = F_{\varepsilon\lambda_1+\tau\lambda_3,\varepsilon\lambda_2+\tau\lambda_4}(X_1, \ldots, X_n) - (\varepsilon\lambda_1 + \tau\lambda_3)a_{1,0} \\
&\qquad - (\varepsilon\lambda_2 + \tau\lambda_4)a_{2,0} \\
&\quad = \varepsilon * (F_{\lambda_1,\lambda_2}(X_1, \ldots, X_n) - (\lambda_1 a_{1,0} + \lambda_2 a_{2,0})) \\
&\qquad + \tau * (F_{\lambda_3,\lambda_4}(X_1, \ldots, X_n) - (\lambda_3 a_{1,0} + \lambda_4 a_{2,0})) \\
&\quad = \varepsilon * 0 + \tau * 0 = 0,
\end{aligned}$$

we thus secure $F_{\lambda_5,\lambda_6} = [\lambda_5 * a_{1,0} + \lambda_6 * a_{2,0}, \lambda_5 * a_{1,0} + \lambda_6 * a_{2,0}]$ is ($P_{2k,0} \leqslant X_{2k-1} < X_{2k} \leqslant Q_{2k,0}$ for $1 \leqslant k \leqslant d$, and $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ for $2d+1 \leqslant r \leqslant n$)-integer solvable for any point (λ_5, λ_6) in each cone. Of course it is also true in the whole R^2 space. Therefore, for any point (λ_1, λ_2) in R^2 space, $F_{\lambda_1,\lambda_2} = [\lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}, \lambda_1 * a_{1,0} + \lambda_2 * a_{2,0}]$ is ($P_{2k,0} \leqslant X_{2k-1} < X_{2k} \leqslant Q_{2k,0}$ for $1 \leqslant k \leqslant d$, and $P_{r,0} \leqslant X_r \leqslant Q_{r,0}$ for $2d+1 \leqslant r \leqslant n$)-integer solvable in R^n space. The proof of other direction vectors is similar to that of the direction vector '<'. □

References

Banerjee, U., 1997. Dependence Analysis. Kluwer Academic Publishers, Norwell, MA.

Banerjee, U., 1988. Dependence Analysis for Supercomputing. Kluwer Academic Publishers, Norwell, MA.

Blume, W., Eigenmann, R., 1998. Nonlinear and symbolic data dependence testing. IEEE Transaction on Parallel and Distributed Systems 9 (12), 1180–1194.

Chu, C.-P., Chang, W.-L., 1998. The extension of direction vector *I* test. In: Proceedings of the 10th IASTED International Conference Parallel and Distributed Computing and Systems, Nevada, USA, pp. 484–489.

Chang, W.-L., Chu, C.-P., 1999. The *I*+ test. In: Lecture Notes in Computer Science, vol. 1656.

Chang, W.-L., Chu, C.-P., Wu, J., 1999. The generalized lambda test: a multi-dimensional version of Banerjee's algorithm. International Journal of Parallel and Distributed Systems and Networks 2 (2), 69–78.

Chang, W.-L., Chu, C.-P., 2000a. The infinity lambda test: a multi-dimensional version of Banerjee's infinity test. Parallel Computing 26, 1275–1295.

Chang, W.-L., Chu, C.-P. 2000b. The multi-dimensional *I* test. Technical Report NCKUCSIE-R-2000-01. Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, ROC.

Dongarra, J., Furtney, M., Reinhardt, S., Russell, J., 1991. Parallel loops - a test suite for parallelizing compilers: description and example results. Parallel Computing 17, 1247–1255.

Kong, X., Klappholz, D., Psarris, K., 1991. The *I* test. IEEE Transaction on Parallel and Distributed Systems 2 (3), 342–359.

Kong, X., Klappholz, D., Psarris, K., 1993. The direction vector *I* test. IEEE Transaction on Parallel and Distributed Systems 4 (11), 1280–1290.

Levine, D., Callahan, D., Dongarra, J., 1991. A comparative study of automatic vectorizing compilers. Parallel Computing 17, 1223–1244.

Li, Z., Yew, P.-C., Zhu, C.-Q., 1990. An efficient data dependence analysis for parallelizing compilers. IEEE Transaction on Parallel and Distributed Systems 1 (1), 26–34.

Petersen, P.M., 1993. Evaluation of programs and parallelizing compilers using dynamic Analysis techniques. Ph.D. Thesis. University of Illinois at Urbana-Champaign, January 1993.

Pugh, W., 1992. A practical algorithm for exact array dependence analysis. Communication of the ACM 35 (8), 102–114.

Shen, Z., Li, Z., Yew, P.-C., 1992. An empirical study of Fortran programs for parallelizing compilers. IEEE Transaction on Parallel and Distributed Systems 1 (3), 356–364.

Smith, B.J., 1976. Matrix Eigensystem Routines-Eispack Guide. Springer, Heidelberg.

Triolet, R., Irigoin, F., Feautrier, P., 1986. Direct parallelization of call statements. In: Proceedings of SIGPLAN Symposium on Compiler Construction, Palo Alto, CA, pp. 176–185.

Vaughan, W.J., 1986. A Residuals Management Model of the Iron and Steel Industry: A Linear Programming Approach. Universal Microfilms International, Ann Arbor, MI.

Wolfe, M., Tseng, C.W., 1992. The power test for data dependence. IEEE Transaction on Parallel and Distributed Systems 3 (5), 591–601.

Wolfe, M., 1996. High Performance Compilers for Parallel Computing. Addison-Wesley, Redwood City, CA.

Weng-Long Chang received his BS degree in computer science and information engineering from Feng Chia University, Taiwan, in 1988 and MS and Ph.D. degrees in computer science and information engineering from the National Cheng Kung University, Taiwan, in 1994 and 1998, respectively. He is currently an Assistant Professor in the Department of Information Management of Southern Taiwan University of Technology, Taiwan. His research interests include languages, tools and compilers for parallel computing.

Chih-Ping Chu received a BS degree in agricultural chemistry from National Chung Hsing University, Taiwan, and MS degree in computer science from the University of California, Riverside, and a Ph.D. degree in computer science from Louisiana State University. He is currently a professor in the Department of Computer Science and Information Engineering of National Cheng Kung University, Taiwan. His research interests include parallel computing, parallel processing, component-based software development, and internet computing.

Jia-Hwa Wu received his BS degree in mechanical engineering from Feng Chia University, Taiwan, in 1981 and MBA degree in industrial management from National Cheng Kung University, Taiwan, in 1986. He is currently a doctoral candidate in computer science and information engineering at the National Cheng Kung University, Taiwan. His research interests include parallelizing compilers, data mining, and internet computing.