Appeared in the Ninth Workshop on Languages and Compilers for Parallel Computing, San Jose, CA, Aug. 8-10, 1996.

Statement-Level Communication-Free Partitioning Techniques for Parallelizing Compilers

Kuei-Ping Shih[1], Jang-Ping Sheu[1], and Chua-Huang Huang[2]

[1] Department of Computer Science and Information Engineering
National Central University
Chung-Li 32054, Taiwan
E-mail: steven@axp1.csie.ncu.edu.tw
sheujp@csie.ncu.edu.tw

[2] Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210-1277
E-mail: chh@cis.ohio-state.edu

Abstract. This paper addresses the problem of communication-free partitioning of iteration spaces and data spaces along hyperplanes. We consider statement-level partitioning for the iteration spaces. The technique explicitly formulates array references as transformations from statement-iteration spaces to data spaces. Based on these transformations, the necessary and sufficient conditions for the feasibility of communication-free hyperplane partitions are presented.

1 Introduction

It has been widely accepted that local memory access is much faster than memory access involving interprocessor communication on distributed-memory multicomputers. If data and computation are not properly distributed across processors, it may cause heavy interprocessor communication. Excessive interprocessor communication will offset the benefit of parallelization even if the program has a large amount of parallelism. Consequently, parallelizing compilers must pay more attention on the distribution of computation and data across processors to reduce the communication overhead or to completely eliminate the interprocessor communication, if possible. Communication-free partitioning, therefore, becomes an interesting and worth studying issue for distributed-memory multicomputers. In recent years, much research has been focused on the area of partitioning iteration spaces and/or data spaces to reduce interprocessor communication and achieve high-performance computing.

Ramanujam and Sadayappan [5] consider the problem of communication-free partitioning of data spaces along hyperplanes for distributed memory multicomputers. They present a matrix-based formulation of the problem for determining the existence of communication-free partitions of data arrays. Their approach proposes only the array decompositions and does not take the iteration space partitionings into consideration. In addition, they concentrate on fully parallel nested loops and focus on two-dimensional data arrays.

Huang and Sadayappan [3] generalize the approach proposed in [5]. They consider the issue of communication-free hyperplane partitioning by explicitly modeling the iteration and data spaces and provide the conditions for the feasibility of communication-free hyperplane partitioning. However, they do not deal with imperfectly nested loops. Moreover, the approach is restricted to loop-level partitioning, i.e., all statements within a loop body must be scheduled together as an indivisible unit.

Chen and Sheu [1] partition iteration space first according to the data dependence vectors obtained by analyzing all the reference patterns in a nested loop, and then group all data elements accessed by the same iteration partition. Two communication-free partitioning strategies, non-duplicate data and duplicate data strategies, are proposed in this paper. Nevertheless, they require the loop contain only uniformly generated references and the problem domain be restricted to a single perfectly nested loop. They also treat all statements within a loop body as an indivisible unit.

Lim and Lam [4] use affine processor mappings for statements to assign the statement-iterations to processors and maximize the degree of parallelism available in the program. Their approach does not treat the loop body as an indivisible unit and can assign different statement-iterations to different processors. However, they consider only the statement-iteration space partitioning and do not address the issue of data space partitioning. Furthermore, their uniform affine processor mappings can cause a large number of idle processors if the affine mappings are non-unimodular transformations.

In this paper, communication-free partitioning of statement-iteration spaces and data spaces along hyperplanes are considered. We explicitly formulate array references as transformations from statement-iteration spaces to data spaces. Based on these transformations, we then present the necessary and sufficient conditions for the feasibility of communication-free hyperplane partitions. Currently, most of the existing partitioning schemes take an iteration instance as a basic schedulable unit that can be allocated to a processor. But, when the loop body contains multiple statements, it is very difficult to make the loop be communication-freely executed by allocating iteration instances among processors. That is, the chance of communication-free execution found by using these methods is limited. For having more flexible and possible in finding communication-free hyperplane partitions, we treat statements within a loop body as separate schedulable units. Our method does not consider only one of the iteration space and data space but both of them. As in [4], our method can be extended to handle more general loop models and can be applied to programs with imperfectly nested loops and affine array references [6].

The rest of the paper is organized as follows. In Section 2, we introduce notation and terminology used throughout the paper. Section 3 describes the characteristics of statement-level communication-free hyperplane partitioning. The necessary and sufficient conditions for the feasibility of communication-free hyperplane partitioning are presented in Section 4. Finally, the conclusions are given in Section 5.

2 Preliminaries

This section explains the statement-iteration space and the data space. It also defines the statement-iteration hyperplane and the data hyperplane.

2.1 Statement-Iteration Space and Data Space

Let $\mathbf{Q}$, $\mathbf{Z}$ and $\mathbf{Z}^+$ denote the set of rational numbers, the set of integers and the set of positive integer numbers, respectively. The symbol $\mathbf{Z}^d$ represents the set of d-tuple of integers. Traditionally, the iteration space is composed of discrete points where each point represents the execution of *all* statements in one iteration of a loop [8]. Instead of viewing each iteration indivisible, an iteration can be divided into the statements that are enclosed in the iteration, i.e., each statement is a schedulable unit and has its own iteration space. We use another term, *statement-iteration space*, to denote the iteration space of a statement in a nested loop.

The following example illustrates the notion of iteration spaces and statement-iteration spaces.

Example 1: Consider the following nested loop L_1.

```
do $i_1 = 1, N$
   do $i_2 = 1, N$
$s_1$:   $A[i_1, -i_1 - i_2 - 1] = A[i_1 - i_2 - 1, -i_1 + i_2 + 1] +$
                                  $B[i_1 + i_2, i_1 + 2i_2 - 1]$            $(L_1)$
$s_2$:   $B[i_1 - i_2 + 1, i_1 - 2i_2 + 1] = A[i_2 - 1, i_1 - i_2] *$
                                  $B[i_1 + i_2 - 1, i_1 + i_2 - 2]$
enddo  enddo
```

Fig. 1 illustrates the iteration space and statement-iteration spaces of loop L_1 for $N = 5$. In Fig. 1(a), a circle means an iteration and includes two rectangles with black and gray colors. The black rectangle indicates statement s_1 and the gray one indicates statement s_2. In Fig. 1(b) and Fig. 1(c), each statement is an individual unit and the collection of statements forms two statement-iteration spaces. $\square$

The representations of statement-iteration spaces, data spaces and the relations among them are described as follows. Let $\mathcal{S}$ denote the set of statements in the targeted problem domain and $\mathcal{D}$ be the set of array variables that are referenced by $\mathcal{S}$. Consider statement $s \in \mathcal{S}$, which is enclosed in a d-nested loop. The statement-iteration space of s, denoted by $SIS(s)$, is a subspace of $\mathbf{Z}^d$ and

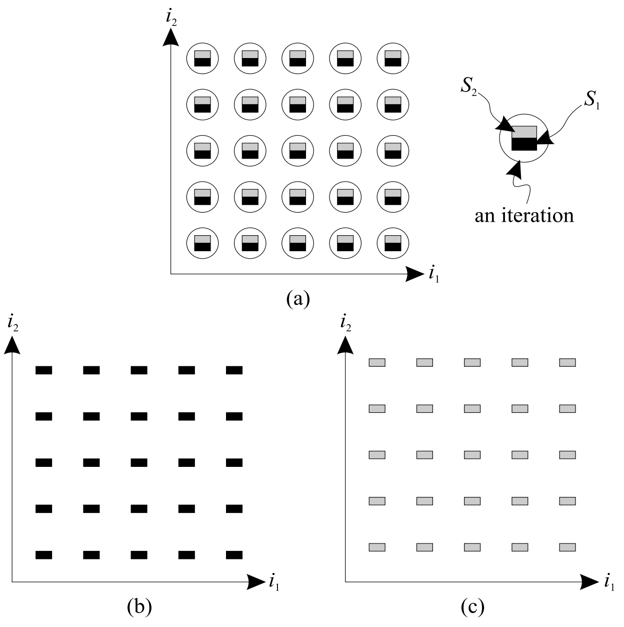

Fig. 1. Loop (L_1)'s iteration space and its corresponding statement-iteration spaces, assuming $N = 5$. (a) $IS(L_1)$, iteration space of loop (L_1). (b) $SIS(s_1)$, statement-iteration space of statement s_1. (c) $SIS(s_2)$, statement-iteration space of statement s_2.

is defined as $SIS(s) = \{[I_1, I_2, \ldots, I_d]^t | LB_i \leq I_i \leq UB_i, \text{ for } 1 \leq i \leq d\}$, where I_i is the loop index variable, LB_i and UB_i are the lower and upper bounds of the loop index variable I_i, respectively. The superscript t is a transpose operator. The column vector $I_s = [I_1, I_2, \ldots, I_d]^t$ is called a *statement-iteration* in statement-iteration space $SIS(s)$, $LB_i \leq I_i \leq UB_i$, for $i = 1, 2, \ldots, d$. On the other hand, from the geometric point of view, an array variable also forms a space and each array element is a point in the space. For exactly describing an array variable, we use *data space* to represent an n-dimensional array v, which is denoted by $DS(v)$, where $v \in \mathcal{D}$. An array element $v[D_1, D_2, \ldots, D_n]$ has a corresponding data index in the data space $DS(v)$. We denote this data index by a column vector $D_v = [D_1, D_2, \ldots, D_n]^t$.

The relations between statement-iteration spaces and data spaces can be built via *array reference functions*. An array reference function is a transformation from statement-iteration space into data space. As most of the existing methods, we require the array references be affine functions of outer loop indices or loop invariant variables. Suppose statement s is enclosed in a d-nested loop and has an array reference pattern $v[a_{1,1}I_1 + a_{1,2}I_2 + \cdots + a_{1,d}I_d + a_{1,0},\ a_{2,1}I_1 + a_{2,2}I_2 + \cdots + a_{2,d}I_d + a_{2,0},\ \ldots,\ a_{n,1}I_1 + a_{n,2}I_2 + \cdots + a_{n,d}I_d + a_{n,0}]$, where $a_{i,j}$ are integer constants, for $1 \leq i \leq n$ and $0 \leq j \leq d$, then the array reference function can be written as:

$$Ref^{s,v}(I_s) = F^{s,v} \cdot I_s + f^{s,v},$$

where

$$F^{s,v} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,d} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,d} \end{bmatrix}, \text{ and } f^{s,v} = \begin{bmatrix} a_{1,0} \\ \vdots \\ a_{n,0} \end{bmatrix}.$$

We term $F^{s,v}$ the *array reference coefficient matrix* and $f^{s,v}$ the *array reference constant vector.* If data index $D_v \in DS(v)$ is referenced in statement-iteration $I_s \in SIS(s)$, then $Ref^{s,v}(I_s) = D_v$.

2.2 Statement-Iteration Hyperplane and Data Hyperplane

A *statement-iteration hyperplane* on statement-iteration space $SIS(s)$, denoted by $\Psi(s)$, is a hyperspace [2] of $SIS(s)$ and is defined as $\Psi_h(s) = \{[I_1, I_2, \ldots, I_d]^t | \delta_1 I_1 + \delta_2 I_2 + \cdots + \delta_d I_d = c_h\}$, where $\delta_1, \ldots,$ and $\delta_d \in \mathbf{Q}$ are the coefficients of the statement-iteration hyperplane and $c_h \in \mathbf{Q}$ is the constant term of the hyperplane. The formula can be abbreviated as $\Psi_h(s) = \{I_s | \Delta \cdot I_s = c_h\}$, where $\Delta = [\delta_1, \ldots, \delta_d]$ is the statement-iteration hyperplane coefficient vector. Similarly, a *data hyperplane* on data space $DS(v)$, denoted by $\Phi(v)$, is a hyperspace of $DS(v)$ and is defined as $\Phi_g(v) = \{[D_1, D_2, \ldots, D_n]^t | \theta_1 D_1 + \theta_2 D_2 + \cdots + \theta_n D_n = c_g\}$, where $\theta_1, \ldots,$ and $\theta_n \in \mathbf{Q}$ are the coefficients of the data hyperplane and $c_g \in \mathbf{Q}$ is the constant term of the hyperplane. In the same way, the formula also can be abbreviated as $\Phi_g(v) = \{D_v | \Theta \cdot D_v = c_g\}$, where $\Theta = [\theta_1, \ldots, \theta_n]$ is the data hyperplane coefficient vector. The hyperplanes that include at least one integer point are considered in this paper.

Statement-iteration hyperplanes and data hyperplanes are used for characterizing communication-free partitioning. We discuss some of these characteristics in the next section.

3 Characteristics of Communication-Free Hyperplane Partitioning

A program execution is communication-free if all operations on each of all processors access only data elements allocated to that processor. A trivial partition strategy allocates all statement-iterations and data elements to a single processor. The program execution of this trivial partitioning is communication-free. However, we are not interested in this single processor program execution because it does not exploit the potential of parallelization and it conflicts with the goal of parallel processing. Hence, in this paper, we consider only nontrivial partitioning, in specific, hyperplane partitioning.

The formal definition of communication-free hyperplane partition is defined as below. Let *partition group*, G,

$$G = \cup_{s \in \mathcal{S}} \Psi_h(s) \bigcup \cup_{v \in \mathcal{D}} \Phi_g(v)$$

be the set of hyperplanes that should be assigned to one processor. The definition of communication-free hyperplane partition can be given as the following.

Definition 1 The hyperplane partitions of statement-iteration spaces and data spaces are said to be communication-free if and only if for any partition group $G = \cup_{s\in\mathcal{S}}\Psi_h(s) \bigcup \cup_{v\in\mathcal{D}}\Phi_g(v)$,

$$\forall I_s \in \Psi_h(s),\ Ref^{s,v}(I_s) \in \Phi_g(v),\ \forall s \in \mathcal{S}, v \in \mathcal{D}.$$

$\square$

As mentioned above, the statement-iterations which access the same array element should be allocated to the same statement-iteration hyperplane. Therefore, it is important to decide statement-iterations that access the same array element. The following lemma states the necessary and sufficient condition that two statement-iterations will access the same array element.

Lemma 1 *For some statement $s \in \mathcal{S}$ and its referenced array $v \in \mathcal{D}$, I_s and I'_s are two statement-iterations on $SIS(s)$ and $Ref^{s,v}$ is the array reference function from $SIS(s)$ into $DS(v)$ as defined above. Then*

$$Ref^{s,v}(I_s) = Ref^{s,v}(I'_s) \Longleftrightarrow (I'_s - I_s) \in Ker(F^{s,v})$$

where $Ker(S)$ denotes the null space of S [2]. $\square$

The proof of this lemma is referred to [6]. We explain the significance of Lemma 1 and show how this lemma can help to find communication-free hyperplane partitions. Communication-free hyperplane partitioning requires those statement-iterations that access the same array element be allocated to the same statement-iteration hyperplane. According to Lemma 1, two statement-iterations access the same array element if and only if the difference of these two statement-iterations belongs to the kernel of $F^{s,v}$. Hence, $Ker(F^{s,v})$ should be a subspace of the statement-iteration hyperplane. Since there may exist many different array references, partitioning a statement-iteration space must consider all array references appeared in the statement. Thus, the space spanned from $Ker(F^{s,v})$ for all array references appearing in the same statement should be a subspace of the statement-iteration hyperplane. The dimension of a statement-iteration hyperplane is one less than the dimension of the statement-iteration space. If there exists a statement s such that the dimension of the spanning space of $Ker(F^{s,v})$ is equal to the dimension of $SIS(s)$, then the spanning space cannot be a subspace of a statement-iteration hyperplane. Therefore, there exists no nontrivial communication-free hyperplane partitioning. From the above observation, we obtain the following theorem.

Theorem 1 *If $\exists s \in \mathcal{S}$ such that*

$$dim(span(\cup_{v\in\mathcal{D}} Ker(F^{s,v}))) = dim(SIS(s)),$$

then there exists no nontrivial communication-free hyperplane partitioning for $\mathcal{S}$ and $\mathcal{D}$. $\square$

Example 2: Consider matrix multiplication.

```
    do $i = 1, N$
        do $j = 1, N$
            do $k = 1, N$
$s$:            $C[i,j] = C[i,j] + A[i,k] * B[k,j]$
    enddo enddo enddo
```

In the above program, there are three array variables, A, B, and C, with three distinct array references involved in statement s. The three array reference coefficient matrices, $F^{s,A}$, $F^{s,B}$, and $F^{s,C}$, are $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$, and $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$, respectively. Thus, $Ker(F^{s,A}) = \{r_1[0,1,0]^t | r_1 \in \mathbf{Z}\}$, $Ker(F^{s,B}) = \{r_2[1,0,0]^t | r_2 \in \mathbf{Z}\}$, and $Ker(F^{s,C}) = \{r_3[0,0,1]^t | r_3 \in \mathbf{Z}\}$. $Ker(F^{s,A})$, $Ker(F^{s,B})$, and $Ker(F^{s,C})$ span $\mathbf{Z}^3$ which has the same dimensionality as the statement-iteration space. By Theorem 1, matrix multiplication has no nontrivial communication-free hyperplane partitioning. □

Theorem 1 can be useful for determining nested loops that have no nontrivial communication-free hyperplane partitioning. Furthermore, when a nontrivial communication-free hyperplane partitioning exists, Theorem 1 can also be useful for finding the hyperplane coefficient vectors. We state this result in the following corollary.

Corollary 1 *For any communication-free statement-iteration hyperplane $\Psi_h(s) = \{I_s | \Delta \cdot I_s = c_h\}$, the following two conditions must hold:*

(1) $span(\cup_{v \in \mathcal{D}} Ker(F^{s,v})) \subseteq \Psi_h(s)$,
(2) $\Delta^t \in (span(\cup_{v \in \mathcal{D}} Ker(F^{s,v})))^{\perp}$,

where $S^{\perp}$ denotes the orthogonal complement space of S. □

Corollary 1 gives the range of communication-free statement-iteration hyperplane coefficient vectors. It can be used for the finding of communication-free statement-iteration hyperplane coefficient vectors. On the other hand, the range of communication-free data hyperplane coefficient vectors is also given as follows.

As mentioned before, the relations between statement-iteration spaces and data spaces can be established via array references. Moreover, the statement-iteration hyperplane coefficient vectors and data hyperplane coefficient vectors are related. The following lemma expresses the relation between these two hyperplane coefficient vectors. A similar result is given in [3].

Lemma 2 *For any statement $s \in \mathcal{S}$ and its referenced array $v \in \mathcal{D}$, $Ref^{s,v}$ is the array reference function from $SIS(s)$ into $DS(v)$. $\Psi_h(s) = \{I_s | \Delta \cdot I_s = c_h\}$ and $\Phi_g(v) = \{D_v | \Theta \cdot D_v = c_g\}$ are communication-free hyperplane partitions if and only if $\Delta = \alpha\Theta \cdot F^{s,v}$, for some α, $\alpha \neq 0$.* □

By Lemma 2, the statement-iteration hyperplane coefficient vector Δ can be decided if the data hyperplane coefficient vector Θ has been determined.

If $F^{s,v}$ is invertible, the statement-iteration hyperplane coefficient vectors can be decided first, then the data hyperplane coefficient vectors can be derived by $\Theta = \alpha' \Delta (F^{s,v})^{-1}$, for some $\alpha', \alpha' \neq 0$. The range of communication-free data hyperplane coefficient vectors can be derived from this lemma. Corollary 1 shows the range of statement-iteration hyperplane coefficient vectors. The next corollary provides the ranges of data hyperplane coefficient vectors.

Corollary 2 *For any communication-free data hyperplane $\Phi_g(v) = \{D_v | \Theta \cdot D_v = c_g\}$, the following condition must hold:*

$$\Theta^t \in (\cup_{s \in \mathcal{S}} Ker((F^{s,v})^t))',$$

where S' denotes the complement set of S. □

The next section describes the communication-free hyperplane partitioning technique. The necessary and sufficient conditions of communication-free hyperplane partitioning for a single perfectly nested loop will be presented.

4 Communication-Free Hyperplane Partitioning for a Perfectly Nested Loop

Each data array has a corresponding data space. However, a nested loop with multiple statements may have multiple statement-iteration spaces. In this section, we will consider additional conditions of multiple statement-iteration spaces for communication-free hyperplane partitioning. These conditions are also used in determining statement-iteration hyperplanes and data hyperplanes.

Suppose $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$ and $\mathcal{D} = \{v_1, v_2, \ldots, v_n\}$, where $m, n \in \mathbf{Z}^+$. The number of occurrences of array variable v_j in statement s_i is $r_{i,j}$, where $r_{i,j} \in \mathbf{Z}^+ \cup \{0\}$, $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. If s_i does not reference v_j, $r_{i,j}$ is set to 0. The previous representation of array reference function can be modified slightly to describe the array reference of statement s_i to variable v_j in the k-th occurrence as $Ref_k^{s_i,v_j}(I_{s_i})$, where $1 \leq k \leq r_{i,j}$. The related representations will be changed accordingly, such as $Ref_k^{s_i,v_j}(I_{s_i}) = F_k^{s_i,v_j} \cdot I_{s_i} + f_k^{s_i,v_j} = D_{v_j}$.

In this section, a partition group that contains a statement-iteration hyperplane for each statement-iteration space and a data hyperplane for each data space is considered. Suppose that the data hyperplane in data space $DS(v_j)$ is $\Phi_g(v_j) = \{D_{v_j} | \Theta_j \cdot D_{v_j} = c_{g_j}\}$, for all $j, 1 \leq j \leq n$. Since $D_{v_j} = Ref_k^{s_i,v_j}(I_{s_i})$, for $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, r_{i,j}$ and $\Theta_j \cdot D_{v_j} = c_{g_j}$, we have

$$\begin{aligned} & \Theta_j \cdot D_{v_j} = c_{g_j} \\ \Leftrightarrow\ & \Theta_j \cdot (F_k^{s_i,v_j} \cdot I_{s_i} + f_k^{s_i,v_j}) = c_{g_j} \\ \Leftrightarrow\ & (\Theta_j \cdot F_k^{s_i,v_j}) \cdot I_{s_i} = c_{g_j} - (\Theta_j \cdot f_k^{s_i,v_j}). \end{aligned}$$

Let

$$\varDelta_i = \varTheta_j \cdot F_k^{s_i,v_j}, \tag{1}$$
$$c_{h_i} = c_{g_j} - (\varTheta_j \cdot f_k^{s_i,v_j}). \tag{2}$$

As a result, those statement-iterations that access the data lay on the data hyperplane $\varPhi_g(v_j) = \{D_{v_j} | \varTheta_j \cdot D_{v_j} = c_{g_j}\}$ will be located on the statement-iteration hyperplane $\varPsi_h(I_{s_i}) = \{I_{s_i} | (\varTheta_j \cdot F_k^{s_i,v_j}) \cdot I_{s_i} = c_{g_j} - (\varTheta_j \cdot f_k^{s_i,v_j})\}$.

To simplify the presentation, we assume all variables v_j appear in every statement s_i. To satisfy that each statement-iteration space contains a unique statement-iteration hyperplane, the following two conditions should be met.

(i) $\forall i$, $\varTheta_j \cdot F_k^{s_i,v_j} = \varTheta_{j'} \cdot F_{k'}^{s_i,v_{j'}}$, $(j \neq j' \vee k \neq k')$,
for $j, j' = 1, 2, \ldots, n$; $k = 1, 2, \ldots, r_{i,j}$ and $k' = 1, 2, \ldots, r_{i,j'}$.

(ii) $\forall i$, $c_{g_j} - (\varTheta_j \cdot f_k^{s_i,v_j}) = c_{g_{j'}} - (\varTheta_{j'} \cdot f_{k'}^{s_i,v_{j'}})$, $(j \neq j' \vee k \neq k')$,
for $j, j' = 1, 2, \ldots, n$; $k = 1, 2, \ldots, r_{i,j}$ and $k' = 1, 2, \ldots, r_{i,j'}$.

Condition **(i)** can infer to the following two equivalent equations.

$$\varTheta_j \cdot F_k^{s_i,v_j} = \varTheta_j \cdot F_1^{s_i,v_j}, \tag{3}$$

for $i = 1, 2, \ldots, m$; $j = 1, 2, \ldots, n$ and $k = 2, 3, \ldots, r_{i,j}$.

$$\varTheta_j \cdot F_1^{s_i,v_j} = \varTheta_1 \cdot F_1^{s_i,v_1}, \tag{4}$$

for $i = 1, 2, \ldots, m$; $j = 2, 3, \ldots, n$. Condition **(ii)** deduces the following two equations, and vice versa.

$$\varTheta_j \cdot f_k^{s_i,v_j} = \varTheta_j \cdot f_1^{s_i,v_j}, \tag{5}$$

for $i = 1, 2, \ldots, m$; $j = 1, 2, \ldots, n$ and $k = 2, 3, \ldots, r_{i,j}$.

$$c_{g_j} = c_{g_1} - \varTheta_1 \cdot f_1^{s_i,v_1} + \varTheta_j \cdot f_1^{s_i,v_j}, \tag{6}$$

for $i = 1, 2, \ldots, m$; $j = 2, 3, \ldots, n$.

Eq. (6) can be used to evaluate the data hyperplane constant terms while some constant term is fixed, say c_{g_1}. Furthermore, we obtain the following results. For some j, c_{g_j} should be the same for all i, $1 \leq i \leq m$. Therefore,

$$c_{g_1} - \varTheta_1 \cdot f_1^{s_i,v_1} + \varTheta_j \cdot f_1^{s_i,v_j} = c_{g_1} - \varTheta_1 \cdot f_1^{s_1,v_1} + \varTheta_j \cdot f_1^{s_1,v_j}, \tag{7}$$

for $i = 2, 3, \ldots, n$ and $j = 2, 3, \ldots, n$. Eq. (7) can be further inferred to obtain the following equation:

$$\varTheta_j \cdot (f_1^{s_i,v_j} - f_1^{s_1,v_j}) = \varTheta_1 \cdot (f_1^{s_i,v_1} - f_1^{s_1,v_1}) \tag{8}$$

for $i = 2, 3, \ldots, m$ and $j = 2, 3, \ldots, n$.

After describing the conditions for satisfying the communication-free hyperplane partitioning constraints, we can conclude the following theorem.

Theorem 2 *Let $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$ and $\mathcal{D} = \{v_1, v_2, \ldots, v_n\}$ be the sets of statements and array variables, respectively. $Ref_k^{s_i,v_j}$ is the array reference function for statement s_i accessing array variables v_j at the k-th occurrence in s_i, where $i = 1, 2, \ldots, m$; $j = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, r_{i,j}$. $\Psi_h(I_{s_i}) = \{I_{s_i} | \Delta_i \cdot I_{s_i} = c_{h_i}\}$ is the statement-iteration hyperplane in $SIS(s_i)$, for $i = 1, 2, \ldots, m$. $\Phi_g(D_{v_j}) = \{D_{v_j} | \Theta_j \cdot D_{v_j} = c_{g_j}\}$ is the data hyperplane in $DS(v_j)$, for $j = 1, 2, \ldots, n$. $\Psi_h(I_{s_i})$ and $\Phi_g(D_{v_j})$ are communication-free hyperplane partitions if and only if the following conditions hold.*

(C1) $\forall i, \Theta_j \cdot F_k^{s_i,v_j} = \Theta_j \cdot F_1^{s_i,v_j}$, for $j = 1, 2, \ldots, n$; $k = 2, 3, \ldots, r_{i,j}$.
(C2) $\forall i, \Theta_j \cdot F_1^{s_i,v_j} = \Theta_1 \cdot F_1^{s_i,v_1}$, for $j = 2, 3, \ldots, n$.
(C3) $\forall i, \Theta_j \cdot f_k^{s_i,v_j} = \Theta_j \cdot f_1^{s_i,v_j}$, for $j = 1, 2, \ldots, n$; $k = 2, 3, \ldots, r_{i,j}$.
(C4) $\Theta_j \cdot (f_1^{s_i,v_j} - f_1^{s_1,v_j}) = \Theta_1 \cdot (f_1^{s_i,v_1} - f_1^{s_1,v_1})$,
for $i = 2, 3, \ldots, m$; $j = 2, 3, \ldots, n$.
(C5) $\forall j, \Theta_j^t \in (\cup_{i=1}^{m} \cup_{k=1}^{r_{i,j}} Ker((F_k^{s_i,v_j})^t))'$.
(C6) $\forall i, \Delta_i = \Theta_j \cdot F_k^{s_i,v_j}$,
for some j, k, $j \in \{1, 2, \ldots, n\}$; $k \in \{1, 2, \ldots, r_{i,j}\}$.
(C7) $\forall i, \Delta_i^t \in (span(\cup_{j=1}^{n} \cup_{k=1}^{r_{i,j}} Ker(F_k^{s_i,v_j})))^{\perp}$.
(C8) $\forall j, j = 2, 3, \ldots, n, \quad c_{g_j} = c_{g_1} - \Theta_1 \cdot f_1^{s_i,v_1} + \Theta_j \cdot f_1^{s_i,v_j}$,
for some $i, i \in \{1, 2, \ldots, m\}$.
(C9) $\forall i, c_{h_i} = c_{g_j} - (\Theta_j \cdot f_k^{s_i,v_j})$,
for some j, k, $j \in \{1, 2, \ldots, n\}$; $k \in \{1, 2, \ldots, r_{i,j}\}$.

□

Theorem 2 can be used to determine whether a nested loop is communication-free. It can also be used as a procedure of finding a communication-free hyperplane partitioning systematically. Conditions **(C1)** to **(C4)** in Theorem 2 are used for finding the data hyperplane coefficient vectors. Condition **(C5)** can check whether the data hyperplane coefficient vectors found in preceding steps are within the legal range. Following the determination of the data hyperplane coefficient vectors, the statement-iteration hyperplane coefficient vectors can be obtained by using Condition **(C6)**. Similarly, Condition **(C7)** can check whether the statement-iteration hyperplane coefficient vectors are within the legal range. The data hyperplane constant terms and statement-iteration hyperplane constant terms can be obtained by using Conditions **(C8)** and **(C9)**, respectively. If one of the conditions is violated, the whole procedure will stop and verify that the nested loop has no communication-free hyperplane partitioning.

On the other hand, combining Equations (3) and (5) together, a sufficient condition of communication-free hyperplane partitioning can be derived as follows.

$$\begin{aligned}\Theta_j(F_1^{s_i,v_j} - F_2^{s_i,v_j}, F_1^{s_i,v_j} - F_3^{s_i,v_j}, \cdots, F_1^{s_i,v_j} - F_{r_{i,j}}^{s_i,v_j},\\ f_1^{s_i,v_j} - f_2^{s_i,v_j}, f_1^{s_i,v_j} - f_3^{s_i,v_j}, \cdots, f_1^{s_i,v_j} - f_{r_{i,j}}^{s_i,v_j}) = 0,\end{aligned}$$

for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. To satisfy the constraint that Θ is a non-zero row vector, the following condition should be true.

$$Rank(F_1^{s_i,v_j} - F_2^{s_i,v_j}, \cdots, F_1^{s_i,v_j} - F_{r_{i,j}}^{s_i,v_j},$$

$$f_1^{s_i,v_j} - f_2^{s_i,v_j}, \cdots, f_1^{s_i,v_j} - f_{r_{i,j}}^{s_i,v_j}) < dim(DS(v_j)), \qquad (9)$$

for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. Note that this condition is similar to the result in [3] for loop-level hyperplane partitioning. We conclude the following corollary.

Corollary 3 *Suppose $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$ and $\mathcal{D} = \{v_1, v_2, \ldots, v_n\}$ are the sets of statements and array variables, respectively. $F_k^{s_i,v_j}$ and $f_k^{s_i,v_j}$ are the array reference coefficient matrix and constant vector, respectively, where $i \in \{1, 2, \ldots, m\}$, $j \in \{1, 2, \ldots, n\}$ and $k \in \{1, 2, \ldots, r_{i,j}\}$. If communication-free hyperplane partitioning exists then Eq. (9) must hold.* □

Theorem 1 and Corollary 3 can be used to check the absence of communication-free hyperplane partitioning for a nested loop, because these conditions are sufficient but not necessary. Theorem 1 is the statement-iteration space dimension test and Corollary 3 is the data space dimension test. To determine the existence of a communication-free hyperplane partitioning, we need to check the conditions in Theorem 2. We show the following example to explain the finding of communication-free hyperplanes of statement-iteration spaces and data spaces.

Example 3: Reconsider loop $L1$. The set of statements $\mathcal{S}$ is $\{s_1, s_2\}$ and the set of array variables $\mathcal{D}$ is $\{v_1, v_2\}$, where $v_1 = A$ and $v_2 = B$. The occurrences of array variables are $r_{1,1} = 2$, $r_{1,2} = 1$, $r_{2,1} = 1$, and $r_{2,2} = 2$.

Since $dim(span(\cup_{j=1}^{2} \cup_{k=1}^{r_{i,j}} Ker(F_k^{s_i,v_j}))) = 1$ is less than $dim(SIS(s_i)) = 2$, for $i = 1, 2$. By Theorem 1, it may exist a communication-free hyperplane partitioning for loop L_1. Again, by Corollary 3, the loop is tested for the possible existence of a nontrivial communication-free hyperplane partitioning. For array variable v_1, the following inequality is satisfied:

$$Rank(F_1^{s_1,v_1} - F_2^{s_1,v_1}, f_1^{s_1,v_1} - f_2^{s_1,v_1}) = 1 < dim(DS(v_1)) = 2.$$

Similarly, with respect to the array variable v_2, the following inequality is obtained:

$$Rank(F_1^{s_2,v_2} - F_2^{s_2,v_2}, f_1^{s_2,v_2} - f_2^{s_2,v_2}) = 1 < dim(DS(v_2)) = 2.$$

Although Eq. (9) holds for all array variables, it still can not ensure that the loop has a nontrivial communication-free hyperplane partitioning.

Using Theorem 2, we further check the existence of a nontrivial communication-free hyperplane partitioning. In the mean time, the statement-iteration and data hyperplanes will be derived if they exist. Recall that the dimensions of data spaces $DS(v_1)$ and $DS(v_2)$ are two, Θ_1 and Θ_2 can be assumed to be $[\theta_{11}, \theta_{12}]$ and $[\theta_{21}, \theta_{22}]$, respectively. The conditions listed in Theorem 2 will be checked to determine the hyperplane coefficient vectors and constants.

By Condition **(C1)** in Theorem 2, the following equations are obtained.

$$\begin{array}{l} \Theta_1 \cdot F_2^{s_1,v_1} = \Theta_1 \cdot F_1^{s_1,v_1} \ (i = 1, j = 1, \text{ and } k = 2,) \\ \Theta_2 \cdot F_2^{s_2,v_2} = \Theta_2 \cdot F_1^{s_2,v_2} \ (i = 2, j = 2, \text{ and } k = 2.) \end{array}$$

By the Condition **(C2)** in Theorem 2,

$$\Theta_2 \cdot F_1^{s_1,v_2} = \Theta_1 \cdot F_1^{s_1,v_1} \ (i = 1 \text{ and } j = 2,)$$
$$\Theta_2 \cdot F_1^{s_2,v_2} = \Theta_1 \cdot F_1^{s_2,v_1} \ (i = 2 \text{ and } j = 2.)$$

By Condition **(C3)** in Theorem 2,

$$\Theta_1 \cdot f_2^{s_1,v_1} = \Theta_1 \cdot f_1^{s_1,v_1} \ (i = 1, j = 1, \text{ and } k = 2,)$$
$$\Theta_2 \cdot f_2^{s_2,v_2} = \Theta_2 \cdot f_1^{s_2,v_2} \ (i = 2, j = 2, \text{ and } k = 2.)$$

By Condition **(C4)** in Theorem 2,

$$\Theta_2 \cdot (f_1^{s_2,v_2} - f_1^{s_1,v_2}) = \Theta_1 \cdot (f_1^{s_2,v_1} - f_1^{s_1,v_1}) \ (i = 2 \text{ and } j = 2,)$$

Substituting $[\theta_{11}, \theta_{12}]$ and $[\theta_{21}, \theta_{22}]$ for Θ_1 and Θ_2, respectively, the above equations form a homogeneous linear system. Solving this homogeneous linear system, we obtain the general solution $(\theta_{11}, \theta_{12}, \theta_{21}, \theta_{22}) = (2r, r, 3r, -2r)$, where $r \in \mathbf{Q} - \{0\}$. Therefore, $\Theta_1 = [2r, r]$ and $\Theta_2 = [3r, -2r]$.

Next, we show Θ_1 and Θ_2 satisfy Condition **(C5)**:

$$(\cup_{i=1}^{m} \cup_{k=1}^{r_{i,1}} Ker((F_k^{s_i,v_1})^t)) = \{c_1[1,1]^t | c_1 \in \mathbf{Q} - \{0\}\},$$
$$\Longrightarrow (\cup_{i=1}^{m} \cup_{k=1}^{r_{i,1}} Ker((F_k^{s_i,v_1})^t))' = \{[r_1, r_2]^t | r_1 \neq r_2, r_1, r_2 \in \mathbf{Q} - \{0\}\},$$
$$\Longrightarrow \Theta_1^t = [2r, r]^t \in (\cup_{i=1}^{m} \cup_{k=1}^{r_{i,1}} Ker((F_k^{s_i,v_1})^t))'.$$

$$(\cup_{i=1}^{m} \cup_{k=1}^{r_{i,2}} Ker((F_k^{s_i,v_2})^t)) = \{c_2[1,-1]^t | c_2 \in \mathbf{Q} - \{0\}\},$$
$$\Longrightarrow (\cup_{i=1}^{m} \cup_{k=1}^{r_{i,2}} Ker((F_k^{s_i,v_2})^t))' = \{[r_1, r_2]^t | r_1 \neq -r_2, r_1, r_2 \in \mathbf{Q} - \{0\}\},$$
$$\Longrightarrow \Theta_2^t = [3r, -2r]^t \in (\cup_{i=1}^{m} \cup_{k=1}^{r_{i,2}} Ker((F_k^{s_i,v_2})^t))'.$$

Now the statement-iteration hyperplane coefficient vectors can be determined using Condition **(C6)** in Theorem 2.

$$\Delta_1 = \Theta_1 \cdot F_1^{s_1,v_1} = \Theta_1 \cdot F_2^{s_1,v_1} = \Theta_2 \cdot F_1^{s_1,v_2} = (r, -r)$$
$$\Delta_2 = \Theta_2 \cdot F_1^{s_2,v_2} = \Theta_1 \cdot F_1^{s_2,v_1} = \Theta_2 \cdot F_2^{s_2,v_2} = (r, r)$$

Note that the statement-iteration hyperplane coefficient vectors may be obtained using many different equations, e.g., Δ_1 can be obtained using $\Theta_1 \cdot F_1^{s_1,v_1}$, $\Theta_1 \cdot F_2^{s_1,v_1}$, or $\Theta_2 \cdot F_1^{s_1,v_2}$. Conditions **(C1)** and **(C2)** in Theorem 2 ensure that all the equations lead to the same result.

For the statement-iteration hyperplane coefficient vectors, Condition **(C7)** is satisfied:

$$span(\cup_{j=1}^{n} \cup_{k=1}^{r_{1,j}} Ker(F_k^{s_1,v_j})) = \{c_3[1,1]^t | c_3 \in \mathbf{Q} - \{0\}\},$$
$$\Longrightarrow (span(\cup_{j=1}^{n} \cup_{k=1}^{r_{1,j}} Ker(F_k^{s_1,v_j})))^{\perp} = \{[r_1, -r_1]^t | r_1 \in \mathbf{Q} - \{0\}\}$$
$$\Longrightarrow \Delta_1^t = [r, -r]^t \in (span(\cup_{j=1}^{n} \cup_{k=1}^{r_{1,j}} Ker(F_k^{s_1,v_j})))^{\perp}.$$

$$span(\cup_{j=1}^{n} \cup_{k=1}^{r_{2,j}} Ker(F_k^{s_2,v_j})) = \{c_4[1,-1]^t | c_4 \in \mathbf{Q} - \{0\}\},$$
$$\Longrightarrow (span(\cup_{j=1}^{n} \cup_{k=1}^{r_{2,j}} Ker(F_k^{s_2,v_j})))^{\perp} = \{[r_2, r_2]^t | r_2 \in \mathbf{Q} - \{0\}\},$$
$$\Longrightarrow \Delta_2^t = [r, r]^t \in (span(\cup_{j=1}^{n} \cup_{k=1}^{r_{2,j}} Ker(F_k^{s_2,v_j})))^{\perp}.$$

Next, we determine the data hyperplane constant terms. Due to the hyperplanes are related to each other, once a hyperplane constant term is determined, the other constant terms will be determined accordingly. Assuming c_{g_1} is known, c_{g_2}, c_{h_1}, and c_{h_2} can be determined using Conditions **(C8)** and **(C9)** as below:

$$\begin{aligned}
c_{g_2} &= c_{g_1} - \Theta_1 \cdot f_1^{s_1,v_1} + \Theta_2 \cdot f_1^{s_1,v_2} = c_{g_1} - \Theta_1 \cdot f_1^{s_2,v_1} + \Theta_2 \cdot f_1^{s_2,v_2} = c_{g_1} + 3r,\\
c_{h_1} &= c_{g_1} - \Theta_1 \cdot f_1^{s_1,v_1} = c_{g_1} - \Theta_1 \cdot f_2^{s_1,v_1} = c_{g_2} - \Theta_2 \cdot f_1^{s_1,v_2} = c_{g_1} + r,\\
c_{h_2} &= c_{g_2} - \Theta_2 \cdot f_1^{s_2,v_2} = c_{g_1} - \Theta_1 \cdot f_1^{s_2,v_1} = c_{g_2} - \Theta_2 \cdot f_2^{s_2,v_2} = c_{g_1} + 2r.
\end{aligned}$$

Similarly, statement-iteration and data hyperplane constant terms can be evaluated using many different equations. However, Conditions **(C3)** and **(C4)** in Theorem 2 ensure that they all lead to the same values.

It is clear that there exists at least one set of nonzero statement-iteration and data hyperplane coefficient vectors such that the conditions listed in Theorem 2 are all satisfied. By Theorem 2, this fact implies that the nested loop has a nontrivial communication-free hyperplane partitioning. The partition group is defined as the set of statement-iteration and data hyperplanes that are allocated to a processor. The partition group for this example follows. $G = \Psi_{h_1}(I_{s_1}) \cup \Psi_{h_2}(I_{s_2}) \cup \Phi_{g_1}(D_{v_1}) \cup \Phi_{g_2}(D_{v_2})$, where

$$\begin{aligned}
\Psi_{h_1}(I_{s_1}) &= \{I_{s_1} \mid [r, -r] \cdot I_{s_1} = c_{g_1} + r\}\\
\Psi_{h_2}(I_{s_2}) &= \{I_{s_2} \mid [r, r] \cdot I_{s_2} = c_{g_1} + 2r\}\\
\Phi_{g_1}(D_{v_1}) &= \{D_{v_1} \mid [2r, r] \cdot D_{v_1} = c_{g_1}\}\\
\Phi_{g_2}(D_{v_2}) &= \{D_{v_2} \mid [3r, -2r] \cdot D_{v_2} = c_{g_1} + 3r\}
\end{aligned}$$

Given loop bounds $1 \le i_1 \le 5$ and $1 \le i_2 \le 5$, for $r = 1$, the constant term c_{g_1} corresponding to statement-iteration hyperplane coefficient vector Δ_1 and Δ_2 are ranged from -5 to 3 and from 0 to 8, respectively. The intersection part of these two ranges means that the two statement-iteration hyperplanes have to be coupled together onto a processor. For the rest, just one statement-iteration hyperplane, either Δ_1 or Δ_2, is allocated to a processor. The constant terms c_{g_2}, c_{h_1}, and c_{h_2} are evaluated to the following values:

$$-2 \le c_{g_2} \le 11, \quad -4 \le c_{h_1} \le 4, \quad \text{and} \quad 2 \le c_{h_2} \le 10.$$

The corresponding parallelized program is as follows.

```
doall $c = -5, 8$
   do $i_1 = \max(c-3, 1), \min(c+1, 5)$
      $i_2 = -i_1 + c + 2$
      $B[i_1 - i_2 + 1, i_1 - 2i_2 + 1] = A[i_2 - 1, i_1 - i_2] * B[i_1 + i_2 - 1, i_1 + i_2 - 2]$
   enddo
   do $i_1 = \max(c+2, 1), \min(c+6, 5)$
      $i_2 = i_1 - c - 1$
      $A[i_1, -i_1 - i_2 - 1] = A[i_1 - i_2 - 1, -i_1 + i_2 + 1] + B[i_1 + i_2, i_1 + 2i_2 - 1]$
   enddo
enddoall
```

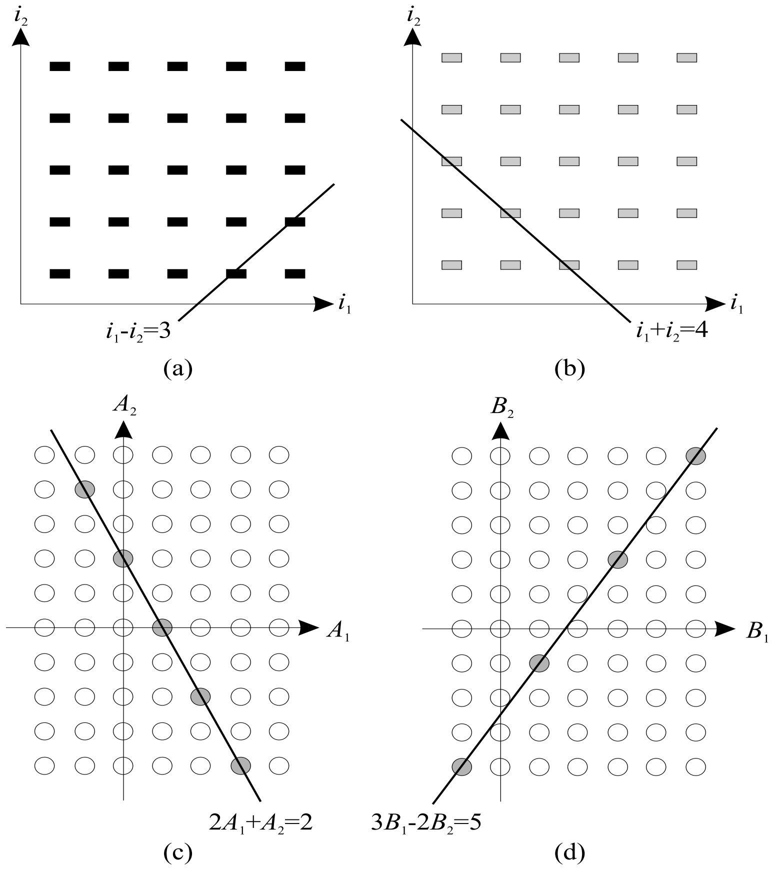

Fig. 2. Communication-free statement-iteration hyperplanes and data hyperplanes for a partition group of loop (L_1), where $r = 1$ and $c_{g_1} = 2$. (a) Statement-iteration hyperplane of $SIS(s_1)$. (b) Statement-iteration hyperplane of $SIS(s_2)$. (c) Data hyperplane of $DS(A)$. (d) Data hyperplane of $DS(B)$.

Fig. 2 illustrates the communication-free hyperplane partitionings for a particular partition group, $r = 1$ and $c_{g_1} = 2$. □

The communication-free hyperplane partitioning technique for a perfectly nested loop has been discussed in this section. Our method treats statements within a loop body as separate schedulable units and considers both iteration and data spaces at the same time. Partitioning groups are determined using affine array reference functions directly, instead of using data dependence vectors.

5 Conclusions

This paper presents the techniques for finding statement-level communication-free hyperplane partitioning for a perfectly nested loop. The technique can also be generalized to deal with sequences of imperfectly nested loops [6].

In reality, most loops are not communication-free. If a program is not communication-free, it is important to identify a subset of iteration and data spaces which are communication-free and then generate communication code for other statement-iterations. The technique presented in this paper can be used for searching subsets of communication-free iteration and data spaces. The future work is to develop heuristics for this searching process and generate efficient code when communication is inevitable.

References

1. T. S. Chen and J. P. Sheu. Communication-free data allocation techniques for parallelizing compilers on multicomputers. *IEEE Transactions on Parallel and Distributed Systems*, 5(9):924–938, September 1994.
2. K. Hoffman and R. Kunze. *Linear Algebra.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, second edition, 1971.
3. C.-H. Huang and P. Sadayappan. Communication-free hyperplane partitioning of nested loops. *Journal of Parallel and Distributed Computing*, 19:90–102, 1993.
4. A. W. Lim and M. S. Lam. Communication-free parallelization via affine transformations. In *Proceedings of the 7th Workshop on Programming Languages and Compilers for Parallel Computing*, August 1994.
5. J. Ramanujam and P. Sadayappan. Compile-time techniques for data distribution in distributed memory machines. *IEEE Transactions on Parallel and Distributed Systems*, 2(4):472–482, October 1991.
6. K.-P. Shih, J.-P. Sheu, and C.-H. Huang. Statement-level communication-free partitioning techniques for parallelizing compilers. Technical Report NCU-PPL-Tr-96-01, Dept. of Computer Science and Information Engineering, National Central University, Taiwan, 1996.
7. M. E. Wolf and M. S. Lam. A data locality optimizing algorithm. In *Proceedings of the ACM SIGPLAN'91 Conference on Programming Language Design and Implementation*, pages 30–44, June 1991.
8. M. J. Wolfe. *High Performance Compilers for Parallel Computing.* Addison-Wesley Publishing Company, 1996.

This article was processed using the LaTeX macro package with LLNCS style.