On Tiling as a Loop Transformation

Jingling Xue
Department of Mathematics, Statistics and Computing Science
University of New England, Armidale 2351, Australia

ABSTRACT

This paper is a follow-up to Irigoin and Triolet's earlier work and our recent work on tiling. In this paper, tiling is discussed in terms of its effects on the dependences between tiles, the dependences within a tile and the required dependence test for legality. A necessary and sufficient condition is given for enforcing the data dependences of the program, while Irigoin and Triolet's atomic tile constraint is only sufficient. A condition is identified under which both Irigoin and Triolet's and our constraints are equivalent. The results of this paper are discussed in terms of their impact on dependence abstractions suitable for legality test and on tiling to optimise a certain given goal.

Keywords: Tiling, loop transformation, dependence analysis, code generation.

1. Introduction

Blocked algorithms are widely known to achieve high performance on parallel computers with a memory hierarchy [8,9]. Tiling is a loop transformation that a parallelising compiler can use to automatically create blocked versions of programs (e.g., linear algebra code) [12,22,24]. The compiler can choose tiles so that cache locality of the program is improved [4,6,10,13,19,21,23] and communication between processors reduced [3,15,17,18].

As tiling emerges as a viable loop transformation, it is becoming increasingly important to understand various properties of tiling when it is used to restructure a loop nest. This paper is a follow-up to Irigoin and Triolet's earlier work [12] and our recent work [25]. We provide answers to several important but previously unanswered questions and discuss some problems that we intend to pursue in the future work.

In this paper, we consider perfectly nested loops with dependences represented as distance or direction vectors. We restrict ourselves to the *parallelepiped tiling* — a special case of monohedral tilings [20], where all tiles are parallelepipeds of the same size and shape. We use $D \in \mathbb{Z}^{n\times m}$ to denote the *dependence matrix* whose columns are all m dependence vectors of a loop nest. Sometimes we abuse the notation by using D as if it was a set consisting of all m dependence vectors of the loop nest. Therefore, when writing $d \in D$, we mean d is a column of D. As is customary, 0 and 1 represent all-zero and all-one matrices or vectors of appropriate sizes, respectively. The notation $\succ$ denotes the usual lexicographic order "larger than" on vectors, and other lexicographic order operators like $\succcurlyeq$, $\prec$ and $\preccurlyeq$ are also used in the usual sense. All lexicographic order operators on two matrices are column-wise. All relational operators on two matrices (or vectors) are element-wise. $\lfloor\ \rfloor$ and $\lceil\ \rceil$ are the usual floor and ceiling operators, respectively; their applications on a matrix (or vector) are element-wise. For example, if A is an $n \times m$ matrix, $\lfloor A \rfloor \succcurlyeq 0$ means that for every column $a_k = \begin{bmatrix} a_{1,k} \\ \vdots \\ a_{n,k} \end{bmatrix}$ of A, $\begin{bmatrix} \lfloor a_{1,k} \rfloor \\ \vdots \\ \lfloor a_{n,k} \rfloor \end{bmatrix} \succcurlyeq 0$.

In their seminal work [12], Irigoin and Triolet modeled a parallelepiped tiling by a matrix transformation $H \in \mathbb{Q}^{n \times n}$ that divides the iteration space into parallelepiped tiles by systems of hyperplanes. Geometrically, the rows of H are the normals to the faces of a parallelepiped tile. Irigoin and Triolet imposed three constraints on H:

- H is non-singular, ensuring that the tile size is bounded, i.e., that all dimensions of the iteration space are tiled;
- $H^{-1} \in \mathbb{Z}^{n \times n}$ is integral, ensuring that all tiles contain the same number of iterations;
- $HD \geqslant 0$, ensuring that no cyclic dependences exist between tiles. This is known as the *atomic tile constraint* and was introduced to facilitate the scheduling of tiles on parallel computers. Each tile is an atomic unit of work to be scheduled on a processor. Once a tile is scheduled, it runs to completion without preemption. As discussed in Section 5, this constraint is sufficient but not necessary to preserve the data dependences of the program.

In this paper, we define tiling as a one-to-one matrix transformation from $\mathbb{Z}^n$ to $\mathbb{Z}^{2n}$, where n is the depth of a loop nest. We discuss tiling in terms of its effects on the dependences between tiles, the dependences within a tile and the required dependence test for legality. We characterise how the set of dependence vectors for a loop nest is transformed by a given tiling transformation. In the special case of rectangular tilings, the closed-form transformed dependences are provided. While Irigoin and Triolet used $HD \geqslant 0$ as a sufficient condition, this paper gives $\lfloor HD \rfloor \succcurlyeq 0$ as a necessary and sufficient condition for legality of a tiling. We show that both $HD \geqslant 0$ and $\lfloor HD \rfloor \succcurlyeq 0$ are equivalent if $|HD| < 1$. We discuss the implications of our results on dependence abstractions suitable for legality test and on tiling to optimise a certain given objective function.

2. Data Dependence Vectors

The data dependence vectors in D for a loop nest can be computed using standard dependence analysis techniques [2,11,14,24]. A dependence vector for an n-deep perfect loop nest is an n-vector $d = (d_1, \cdots .d_n)$, where the k-th entry d_k corresponds to the k-th loop (counting from outermost to innermost). Each entry d_k can take two kinds of values:

- *Distance*: $d_k \in \mathbb{Z}$ is an integer value.
- *Direction*: $d_k \in \{<, \leqslant, >, \geqslant, *\}$ is a direction value [24]. Note that '=' and '$\neq$' are not used here: '=' can be denoted as 0 and '$\neq$' refined and represented by '$<$' and '$>$'.

A dependence vector is called a *distance vector* if all its entries are integers, and a *direction vector* otherwise. Often direction vectors have to be used when some dependences cannot be represented as a finite number of distance vectors.

In the presence of direction vectors, each entry d_k of a dependence vector d can be represented as a range of integers by $[d_k^{\min}, d_k^{\max}]$ in a unified manner [21]. Therefore, the direction values '$<$', '$\leqslant$', '$>$', '$\geqslant$', and '$*$' become shorthands for $[1, \infty]$, $[0, \infty]$, $[-\infty, -1]$,

$[-\infty, 0]$, and $[-\infty, \infty]$, respectively. If d_k is a distance value x, then $d_k^{\min} = d_k^{\max} = x$. Hence, a single dependence vector d represents a set of distance vectors:

$$\mathcal{E}(d) = \{(e_1, \cdots, e_n) \mid e_k \in \mathbb{Z}, d_k^{\min} \leqslant e_k \leqslant d_k^{\max}\}$$

Now, all lexicographic order operators can be extended to operate on general dependence vectors. Let d be a dependence vector. We define $d \succ 0$ if $\forall e \in \mathcal{E}(d) : e \succ 0$. That is, d is lexicographically positive if all distance vectors in $\mathcal{E}(d)$ are. We do likewise for $\succcurlyeq$, $\prec$ and $\preccurlyeq$.

In a sequential loop nest, all its dependence vectors are lexicographically positive.

Assumption 1 $D \succ 0$.

To test legality for a tiling (Section 5), we need to be able to calculate $\lfloor Hd \rfloor$, where $H \in \mathbb{Q}^{n \times n}$ and d is a direction vector. In [21], the ordinary arithmetic on integers was extended to operate on direction values. Since $H \in \mathbb{Q}^{n \times n}$, we introduce one more extension to handle $\lfloor\ \rfloor$. As in [21], we define component addition to be:

$$[a, b] + [c, d] = [a + c, b + d]$$

where for all $x \in \mathbb{Z} \cup \{\infty\}$, $x + \infty$ is ∞ and for all $x \in \mathbb{Z} \cup \{-\infty\}$, $x + (-\infty)$ is $-\infty$. Similarly, multiplication by a scalar $s \in \mathbb{Q}$ is defined as:

$$s[a, b] = \begin{cases} [sa, sb] & \text{if } s \geqslant 0 \\ [sb, sa] & \text{if } s < 0 \end{cases}$$

where $s\infty$ is ∞ for positive s, 0 if $s = 0$, and $-\infty$ for negative s, and likewise for $s(-\infty)$. We use $\lfloor\ \rfloor$ in the normal manner, except that $\lfloor \frac{\infty}{s} \rfloor$ is ∞ for positive s and $-\infty$ for negative s, and likewise for $\lfloor \frac{-\infty}{s} \rfloor$. Finally, component subtraction $a - b$ is defined to be $a + (-1)b$.

This component arithmetic is consistent (or conservative) in the sense that if $e \in \mathcal{E}(d)$, then $He \in \mathcal{E}(Hd)$, where $H \in \mathbb{Q}^{n \times n}$, but the converse is not necessarily true.

3. Tiling Transformation

Tiling decomposes an n-dimensional loop nest into a $2n$-dimensional loop nest, where the outer n *tile loops* step between tiles and the inner n *element loops* step the points within a tile.

Example 1 Consider a double loop:

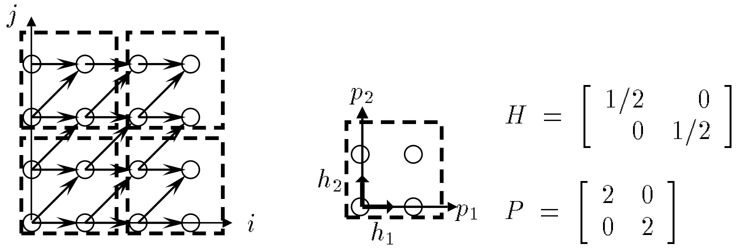
```
do i = 0, N
   do j = 0, N
      A(i, j) = A(i − 1, j − 1) + A(i − 1, j)
```

with the following dependences:

$$D = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

Figure 1(a) shows a 2×2 rectangular tiling of the double loop, and the tiled program is:

```
do ii = 0, N, 2
   do jj = 0, N, 2
      do i = ii, min(ii + 1, N)
         do j = jj, min(jj + 1, N)
            A(i, j) = A(i − 1, j − 1) + A(i − 1, j)
```



(a) Tiling (N=3)   (b) The fundamental tile

Fig. 1: A rectangular tiling of a double loop.

When tiling an n-dimensional iteration space, all tiles are n-dimensional parallelepipeds of the same size and shape. We call the tile at the origin the *fundamental tile*. Since all tiles are translations of the fundamental tile, a tiling transformation can be characterised by means of characterising the fundamental tile. As an n-dimensional parallelepiped, the fundamental tile can be defined either by an integral matrix $P \in \mathbb{Z}^{n\times n}$ whose columns are its n edge vectors emitting from the origin or by a rational matrix $H \in \mathbb{Q}^{n\times n}$ whose rows are the normal vectors to its n faces adjoining at the origin. P and H has the relationship: $P = H^{-1}$. Figure 1(b) shows the H and P for the rectangular tiling shown in Figure 1(a).

A tiling for an n-deep loop nest is a mapping from $\mathbb{Z}^n$ to $\mathbb{Z}^{2n}$:

$$\tau : \mathbb{Z}^n \longrightarrow \mathbb{Z}^{2n}, \quad \tau(I) = \begin{bmatrix} \lfloor HI \rfloor \\ I - H^{-1}\lfloor HI \rfloor \end{bmatrix} \tag{1}$$

where $\lfloor HI \rfloor$ identifies the coordinates of the tile (in the column basis of H^{-1}) that I is mapped to, and $I - H^{-1}\lfloor HI \rfloor$ gives the coordinates of I within that tile relative to the tile origin $\lfloor HI \rfloor$ (in the basis of the original iteration space but translated by $-\lfloor HI \rfloor$). In (1), τ is completely specified by H. We shall speak of H as a tiling transformation.

To ensure that all tiles contain the same number of iterations, it is required that $P = H^{-1} \in \mathbb{Z}^{n\times n}$.

By the definition of (1), we always have:

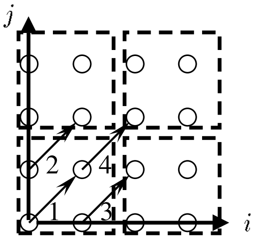
$$0 \leqslant HI - \lfloor HI \rfloor < 1$$

Theorem 1 *The mapping in (1) is well-defined, e.g., injective.*

Proof. If I and J are so mapped that $\begin{bmatrix} \lfloor HI \rfloor \\ I - H^{-1}\lfloor HI \rfloor \end{bmatrix} = \begin{bmatrix} \lfloor HJ \rfloor \\ J - H^{-1}\lfloor HJ \rfloor \end{bmatrix}$, then we have $I = J$ by an algebraic simplification. Thus, the mapping in (1) is well-defined and injective. ■

4. Transformation of Dependence Vectors

The dependence vectors in D enforce a partial order on the execution of the iterations for a loop nest. This partial order summarises all iteration-reordering constraints imposed by data dependences. As an iteration-reordering transformation, a tiling may change the execution order of the iterations for a loop nest, but it must not violate any data dependence constraints.



$$\underset{d}{\begin{bmatrix}1\\1\end{bmatrix}} \longmapsto \left\{ \underset{d'_1}{\begin{bmatrix}0\\0\\1\\1\end{bmatrix}}, \underset{d'_2}{\begin{bmatrix}0\\1\\1\\-1\end{bmatrix}}, \underset{d'_3}{\begin{bmatrix}1\\0\\-1\\1\end{bmatrix}}, \underset{d'_4}{\begin{bmatrix}1\\1\\-1\\-1\end{bmatrix}} \right\}$$

Fig. 2: Transformation of dependence vectors (Fig. 1 cont'd). d =(1,1) is mapped to four dependence vectors: its instances labeled 1, 2, 3, and 4 are mapped to d'_1, d'_2, d'_3 and d'_4, respectively.

Let D' be the set of dependence vectors that result after a tiling is applied. Since a dependence vector in D is an n-vector, every dependence vector in D' is a $2n$-vector.

Definition 1 (**Legality of Tiling**) A tiling is *legal* if $D' \succ 0$.

This section discusses how to construct the transformed set of dependence vectors D'. Unlike a unimodular transformation, a tiling may map a single dependence vector to more than one dependence vector. This is illustrated in Figure 2.

4.1. Distance Vectors

Let Φ be the iteration space of a loop nest. According to (1), a tiling H maps a distance vector $d \in D$ to the following set of distance vectors:

$$\left\{ \begin{bmatrix} \lfloor H(I+d) \rfloor - \lfloor HI \rfloor \\ d - H^{-1}(\lfloor H(I+d) \rfloor - \lfloor HI \rfloor) \end{bmatrix} \mid I, I+d \in \Phi \right\}$$

Using the properties of $\lfloor\ \rfloor$, this can be simplified by restricting I within the fundamental tile:

$$\left\{ \begin{bmatrix} \lfloor H(I+d) \rfloor - \lfloor HI \rfloor \\ d - H^{-1}(\lfloor H(I+d) \rfloor - \lfloor HI \rfloor) \end{bmatrix} \mid 0 \leqslant HI < 1 \right\}$$

which can be further simplified by noting that $\lfloor HI \rfloor = 0$ when $0 \leqslant HI < 1$:

$$d' = \left\{ \begin{bmatrix} \lfloor H(I+d) \rfloor \\ d - H^{-1}\lfloor H(I+d) \rfloor \end{bmatrix} \mid 0 \leqslant HI < 1 \right\}$$

4.2. Direction Vectors

In the case when d is a direction vector, a simple-minded approach would be to compute its transformed dependence vectors from the distance vectors in $\mathcal{E}(d)$:

$$d'_{\text{dis}} = \left\{ \begin{bmatrix} \lfloor H(I+e) \rfloor \\ e - H^{-1}\lfloor H(I+e) \rfloor \end{bmatrix} \mid 0 \leqslant HI < 1, e \in \mathcal{E}(d) \right\}$$

This may be infeasible in the case when $\mathcal{E}(d)$ contains infinitely many distance vectors.

A practical approach is to use our component arithmetic to operate on direction vectors directly:

$$d'_{\rm dir} \;=\; \left\{ \left[\begin{array}{c} \lfloor H(I+d) \rfloor \\ d - H^{-1}\lfloor H(I+d) \rfloor \end{array} \right] \mid 0 \leqslant HI < 1 \right\}$$

This works because our component arithmetic on direction values guarantees $d'_{\rm dir} \supseteq d'_{\rm dis}$. In fact, any component arithmetic will do as long as it ensures that $d'_{\rm dir} \supseteq d'_{\rm dis}$ is satisfied.

4.3. Transformed Dependences

Hence, all dependences in the tiled program can be found using the following theorem.

Theorem 2 *(***Transformed Set of Dependence Vectors** *D') A tiling H maps D to D' given by:*

$$D' = \left\{ \left[\begin{array}{c} \lfloor H(I+d) \rfloor \\ d - H^{-1}\lfloor H(I+d) \rfloor \end{array} \right] \mid 0 \leqslant HI < 1, d \in D \right\} \tag{2}$$

As an example, this theorem allows us to find out the transformed dependence vectors for the example in Figure 1:

$$D' \;=\; \left[\begin{array}{rrrrrr} 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & -1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 0 & 0 \end{array} \right]$$

where the first four columns are mapped from $(1, 1)$ and the last two from $(1, 0)$.

As a second example, consider a double loop with one direction vector:

$$D \;=\; \left[\begin{array}{c} 1 \\ < \end{array} \right]$$

Using again the tiling transformation H in Figure 1, the transformed dependences are:

$$D' \;=\; \left[\begin{array}{cc} 0 & 1 \\ \leqslant & \leqslant \\ 1 & -1 \\ * & * \end{array} \right]$$

In both of these two examples, the tiling transformation H in Figure 1 is legal.

If all dependence vectors in D are distance vectors, the transformed dependences given in Theorem 2 are exact. In this case, Definition 1 can be tightened as follows.

Definition 2 (**Legality of Tiling When D Contains Distance Vectors Only**) A tiling is *legal* if and only if $D' \succ 0$.

However, when direction vectors are necessary, information about the dependences may be lost when a tiling is applied. As was discussed in Section 4.2, we can only guarantee that $d'_{\rm dir} \supseteq d'_{\rm dis}$ but $d'_{\rm dir} = d'_{\rm dis}$ is not necessarily true. Therefore, when computing the dependence vectors for a loop nest, distance vectors should be used whenever possible.

From Theorem 2, the dependences between tiles and those internal to a tile can be derived.

$$
\left\{ \begin{bmatrix} d'_1 \\ \vdots \\ d'_n \\ d'_{n+1} \\ \vdots \\ d'_{2n} \end{bmatrix} \middle| \begin{bmatrix} d'_k \\ d'_{n+k} \end{bmatrix} \in \begin{cases} \left\{ \begin{bmatrix} \lfloor \frac{d_k}{s_k} \rfloor \\ d_k - s_k \lfloor \frac{d_k}{s_k} \rfloor \end{bmatrix}, \begin{bmatrix} \lceil \frac{d_k}{s_k} \rceil \\ d_k - s_k \lceil \frac{d_k}{s_k} \rceil \end{bmatrix} \right\} & \text{if } d_k \in \mathbb{Z} \\ \left\{ \begin{bmatrix} 0 \\ d_k \end{bmatrix}, \begin{bmatrix} < \\ * \end{bmatrix} \right\} & \text{if } d_k \in \{<, \leqslant\} \\ \left\{ \begin{bmatrix} 0 \\ d_k \end{bmatrix}, \begin{bmatrix} > \\ * \end{bmatrix} \right\} & \text{if } d_k \in \{>, \geqslant\} \\ \left\{ \begin{bmatrix} * \\ * \end{bmatrix} \right\} & \text{if } d_k = `*' \end{cases} \right\}
$$

Fig. 3: Transformed dependences for rectangular tilings.

The set of dependence vectors between tiles is given by:

$$
D'_T \quad = \quad \{\lfloor H(I+d) \rfloor \mid 0 \leqslant HI < 1, d \in D\}
$$

To find out the dependence vectors internal to a tile, we note that the first component in (2) must be 0, i.e., $\lfloor H(I+d) \rfloor = 0$. Since $0 \leqslant HI < 1$, this implies $|Hd| < 1$. Hence,

$$
D'_E \quad = \quad \{d \mid |Hd| < 1, d \in D\}
$$

Finally, in the important special case when H is a rectangular tiling, the closed-form transformed dependences can be obtained from Theorem 2. Let $H = \mathrm{diag}(s_1, \cdots, s_n)$, where $s_k > 1$ is the tile length along the k-th dimension. If $d = (d_1, \cdots, d_n)$ is a dependence vector, then its transformed dependences are shown in Figure 3.

5. Legality Test

Irigoin and Triolet's atomic tile constraint $HD \geqslant 0$ is only sufficient for enforcing legality of a tiling H. This section introduces $\lfloor HD \rfloor \succcurlyeq 0$ as a necessary and sufficient condition. In addition, we show that $HD \geqslant 0$ and $\lfloor HD \rfloor \succcurlyeq 0$ are equivalent if $|HD| < 1$. We provide a geometric interpretation of $|HD| < 1$ and discuss its implications on dependence abstractions suitable for legality test. These results allow us to understand why Irigoin and Triolet's atomic tile constraint $HD \geqslant 0$ has often been used in tiling *uniform dependence loops*, i.e., nested loops with all dependences representable solely as distance vectors.

Theorem 3 *A tiling is legal if and only if* $\lfloor HD \rfloor \succcurlyeq 0$.

Proof. By Definition 1, we show that $D' \succ 0 \iff \lfloor HD \rfloor \succcurlyeq 0$.

"$\Longrightarrow$" Assume to the contrary that there exists a $d \in D$ such that $\lfloor Hd \rfloor \prec 0$. By setting $I = 0$ in (2), we find that D' contains $\begin{bmatrix} \lfloor Hd \rfloor \\ d - H^{-1}\lfloor Hd \rfloor \end{bmatrix} \prec 0$. This contradicts the assumption $D' \succ 0$.

"$\Longleftarrow$" Let $d \in D$ be a dependence vector. It suffices to show that all its transformed dependence vectors are lexicographically positive. By (2), all these transformed dependence vectors are given by:

$$
d' \quad = \quad \left\{ \begin{bmatrix} \lfloor H(I+d) \rfloor \\ d - H^{-1}\lfloor H(I+d) \rfloor \end{bmatrix} \mid 0 \leqslant HI < 1 \right\}
$$

We are given that $\lfloor Hd \rfloor \succcurlyeq 0$. By further using $0 \leqslant HI$, we obtain $\lfloor H(I+d) \rfloor \succcurlyeq 0$. For each fixed but arbitrary I that satisfies $0 \leqslant HI < 1$, it suffices to show that $d(I)' = \begin{bmatrix} \lfloor H(I+d) \rfloor \\ d - H^{-1}\lfloor H(I+d) \rfloor \end{bmatrix} \succ 0$. Two cases are distinguished:

Case 1: $\lfloor H(I+d) \rfloor \succ 0$. Trivially, $d(I)' \succ 0$.

Case 2: $\lfloor H(I+d) \rfloor = 0$. We have $d(I)' = \begin{bmatrix} 0 \\ d \end{bmatrix} \succ 0$, since $d \succ 0$ by Assumption 1. ■

This theorem allows a larger class of tilings to be used than Irigoin and Triolet's atomic tile constraint.

Example 2 Consider a double loop:

do $i = 0, N$
  do $j = 0, N$
    $A(i,j) = A(i,j-1) + A(i-2,j+1)$

with the following dependences:

$$D = \begin{bmatrix} 0 & 2 \\ 1 & -1 \end{bmatrix}$$

Using the tiling transformation in Figure 1, Figure 4 shows the rectangular tiling of the iteration space. This tiling is legal according to Theorem 3, since

$$\lfloor HD \rfloor = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix} \succcurlyeq 0$$

but illegal if Irigoin and Triolet's atomic tiling constraint is used:

$$HD = \begin{bmatrix} 0 & 1 \\ 1/2 & -1/2 \end{bmatrix} \not\geqslant 0$$

Using Theorem 2, we can verify that all transformed dependence vectors are lexicographically positive:

$$D' = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & -1 \end{bmatrix}$$

and the tiling is indeed legal by Definition 1.

$HD \geqslant 0$ defines a system of linear inequalities while $\lfloor HD \rfloor \succcurlyeq 0$ is non-linear. Therefore, we prefer to use $HD \geqslant 0$ when tiling to optimise a certain given objective function. We describe below a theorem that, along with one immediate corollary, allows $HD \geqslant 0$ to be used equivalently in place of $\lfloor HD \rfloor \succcurlyeq 0$ for an important class of programs that includes uniform dependence loops as a special case.

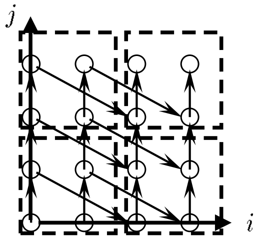


Fig. 4: An illustration of $\lfloor HD \rfloor \succcurlyeq 0$ and $HD \geqslant 0$ for Example 2 ($N = 3$).

Theorem 4 *If* $|HD| < 1$*, then*

$$\lfloor HD \rfloor \succcurlyeq 0 \quad \Longleftrightarrow \quad HD \geqslant 0 \tag{3}$$

Proof. "$\Longleftarrow$" is self evident. To prove "$\Longrightarrow$", we show that for a fixed but arbitrary dependence vector $d \in D$, $Hd \geqslant 0$ holds when $|Hd| < 1$ and $\lfloor Hd \rfloor \succcurlyeq 0$. Let $d' = Hd = (d'_1, \cdots, d'_n)$. Assume to the contrary that $\exists\, k : 0 < k \leqslant n$ such that $\forall\, 0 < i < k : d'_i \geqslant 0$ and $d'_k < 0$. That $|Hd| < 1$ means that $\forall\, 0 < i \leqslant n : -1 < d'_i < 1$, implying that $\lfloor d' \rfloor = \lfloor Hd \rfloor = (\overbrace{0, \cdots, 0}^{k-1}, -1, \cdots) \prec 0$. This contradicts the assumption that $\lfloor Hd \rfloor \succcurlyeq 0$. Thus, if $\lfloor Hd \rfloor \succcurlyeq 0$, we must have $Hd \geqslant 0$. ∎

In this theorem, however, $|HD| < 1$ is not necessary to establish (3). To illustrate this and the theorem itself, let us consider Examples 1 and 2 again. In both examples, the same tiling matrix H in Figure 1 is used. Example 1 satisfies the hypothesis $|HD| < 1$ of Theorem 4. Thus, either $HD \geqslant 0$ or $\lfloor HD \rfloor \succcurlyeq 0$ can be used to check the legality of the tiling. To understand why $|HD| < 1$ is not necessary to establish (3), suppose dependence vector $(4, 4)$ is added to Example 1. In this revised example, $|HD| < 1$ is not satisified because $\left| H \begin{bmatrix} 4 \\ 4 \end{bmatrix} \right| = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \not< \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. We note that (3) will still hold. In Example 2, $|HD| < 1$ is not satisified, either, because $\left| H \begin{bmatrix} 2 \\ -1 \end{bmatrix} \right| = \left| \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right| = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \not< \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. In this case, however, (3) does not hold.

Theorem 4 has an immediate but important corollary.

Corollary 1 *Let* $D_s \in \mathbb{Z}^{n \times p}$ *be a submatrix of* $D \in \mathbb{Z}^{n \times m}$ *such that all dependence vectors in* D *are nonnegative linear combinations of those in* D_s*. If* $|HD_s| < 1$*, then* $\lfloor HD \rfloor \succcurlyeq 0 \iff HD \geqslant 0$.

Geometrically, $|HD| < 1$ means that each dependence vector $d \in D$ is *contained* within the fundamental tile. This is possible only if d is a distance vector. Therefore, Theorem 4 and Corollary 1 have the following implications on dependence abstractions suitable for tiling and the required dependence test for legality.

1. **Distance vectors are preferable over direction vectors.** Two reasons can be identified in favour of using distance vectors as a dependence abstraction for tiling. First, the transformed dependences for distance vectors can be computed exactly. Second, the distance vectors can potentially validate $|HD| < 1$ so that according to Theorem 4, the linear constraints $HD \geqslant 0$ can be used for legality test.

If some dependences for a loop nest are not representable as a finite number of distance vectors, one may choose to approximate the dependences for the entire loop nest by a dependence abstraction called the *dependence cone* [12]. A (pointed) cone is characterised by a finite number of integer vectors called *extremal rays* $R = [r_1, \cdots, r_p]$ such that all data dependences of the program can be expressed as nonnegative linear combinations of the rays. The dependence cone is more precise than direction vectors [26] and may enable $HD \geqslant 0$ to be used for legality test due to Corollary 1.

Consider a double loop from [12]:

$$\begin{array}{l} \textsf{do } i = 1, N \\ \quad \textsf{do } j = 1, M \\ \quad\quad A(i+j) = A(i) + A(i+j) \end{array}$$

with the dependences specified as both direction vectors and a dependence cone:

$$D = \begin{bmatrix} < \\ * \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$$

R is more precise than D. To see this, we note that the tiling transformation

$$H = \begin{bmatrix} 1/2 & -1/2 \\ 0 & 1/2 \end{bmatrix}$$

is legal if R is used but will be rendered as illegal if D is used. In addition, due to Corollary 1, R may enable $HD \geqslant 0$ to be used for legality test.

However, the dependence cone abstraction is not without problems. First, the dependence cone conservatively approximates the dependences of a loop nest, reducing the search space for legal tilings. Second, the dependence cone for the tiled program can be very imprecise, since the lengths of the sides of a tile are not represented in the cone. This can be relaxed somewhat if the dependence polyhedron abstraction is used [7,26]. Further research in this area is needed.

2. $HD \geqslant 0$ **can be used for tiling uniform dependence loops.** Prior work on tiling uniform dependences loops has used $HD \geqslant 0$ exclusively to enforce the data dependences of the programs in the class [3,25]. In practice, the tile size is sufficiently larger than the magnitudes of distance vectors (which have small integers only [11]). This means that all distance vectors are contained within the fundamental tile, implying that $|HD| < 1$ in Theorem 4 is satisfied. Hence, in practice, $HD \geqslant 0$ can be used to enforce the dependences of uniform dependence loops sufficiently and necessarily.

6. Tiling Nested Loops

When enforcing the data dependences of a program, $\lfloor HD \rfloor \succcurlyeq 0$ is exact but non-linear, but the converse is true for $HD \geqslant 0$. This section discusses how to use both constraints to advantage when tiling to optimise a certain given goal. We illustrate the basic idea by finding tilings that require a minimal amount of communication between processors.

The problem of minimising communication between processors for uniform dependence loops was studied in [3,17,19,25]. The size (computation volume) of a tile for a tiling transformation H is $\mathcal{V}_{\text{comp}}(H) = \frac{1}{|\det(H)|}$. Given the tile size as an input design parameter, the problem of finding a tiling that induces the minimal amount of communication between tiles was formulated as a combinatorial problem:

$$\begin{array}{rl} \textbf{Minimise} & \mathcal{V}_{\text{comm}}(H) = \frac{1}{|\det(H)|} \sum_{i=1}^{n} \sum_{j=1}^{m} (HD)_{i,j} \\ \textbf{Subject to} & \mathcal{V}_{\text{comp}}(H) = \frac{1}{|\det(H)|} = \nu \\ & HD \geqslant 0 \end{array} \tag{4}$$

and solved analytically in [3,25]. Here, the objective function $\mathcal{V}_{\text{comm}}(H)$ approximates the communication volume of a tile. $HD \geqslant 0$ defines the search space for all legal tilings (Remark 2 at the end of Section 5).

These research efforts are limited to uniform dependence loops only. Sometimes, a loop nest is almost but not quite uniform due to the presence of a few global (spiral or wraparound) dependences. We discuss by examples how to exploit the results in this paper to deal with this kind of global data dependences.

Consider a double loop:

do $i = 0, N$
  do $j = 0, N$
    $A(i,j) = A(i,j-1) + A(i-1,j+1) + A(i-1,j-1) + A(i-N,j+1)$

with the following dependences:

$$D = \begin{bmatrix} 0 & 1 & 1 & N \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

If D is used in (4), the optimal tiling is found to be:

$$H = \sqrt{\tfrac{1}{\nu}} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

and induces the communication volume $\mathcal{V}_{\text{comm}}(H) = 2(N+2)\sqrt{\nu}$ for a tile. This solution is misleading because the objective function $\mathcal{V}_{\text{comm}}(H)$ in (4) grossly over-approximates the communication volume when the global dependence $\begin{bmatrix} N \\ -1 \end{bmatrix}$ is used as if it existed uniformly across the entire iteration space. By ignoring this global dependence vector, the following desirable optimal tiling is found:

$$H = \sqrt{\tfrac{6}{\nu}} \begin{bmatrix} \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 \end{bmatrix}$$

which has the communication volume $\mathcal{V}_{\text{comm}}(H) = 2\sqrt{6\nu}$. Note that $\begin{bmatrix} N \\ -1 \end{bmatrix}$ is contained in the cone generated by the other three dependence vectors of the program: $\begin{bmatrix} N \\ -1 \end{bmatrix} = (N-1)\begin{bmatrix} 0 \\ 1 \end{bmatrix} + N\begin{bmatrix} 1 \\ -1 \end{bmatrix}$. By Corollary 1, this tiling is legal.

Let us modify this program slightly by deleting the second use of A in the right-hand:

do $i = 0, N$
  do $j = 0, N$
    $A(i,j) = A(i,j-1) + A(i-1,j-1) + A(i-N,j+1)$

The dependences of the program are:

$$D = \begin{bmatrix} 0 & 1 & N \\ 1 & 1 & -1 \end{bmatrix}$$

Again by ignoring $\begin{bmatrix} N \\ -1 \end{bmatrix}$, we obtain the following optimal tiling:

$$H = \sqrt{\tfrac{1}{\nu}} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix}$$

which has the communication volume $\mathcal{V}_{\mathrm{comm}}(H) = 2\sqrt{\nu}$. This time, $\begin{bmatrix} N \\ -1 \end{bmatrix}$ cannot be expressed as a nonnegative linear combination of the other dependence vectors of the program. So Corollary 1 does not apply. However, a simple calculation shows that $H \begin{bmatrix} N \\ -1 \end{bmatrix} = \sqrt{\tfrac{1}{\nu}} \begin{bmatrix} -N-1 \\ N \end{bmatrix}$. By permuting the two rows of H, the resulting tiling

$$H = \sqrt{\tfrac{1}{\nu}} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$$

is legal by Theorem 4 and can be used to tile the given program.

So the results of this paper can be used to deal with nested loops that are uniform once a few global dependences are disregarded. One future work is to apply the results in this paper to tile general nested loops with dependences represented as direction vectors.

7. Code Generation

The tiled program consists of $2n$ loops with the outer n tile loops iterating over tiles and and the inner n elements loops stepping through the points within a tile. When generating the tiled program to implement a tiling transformation, care must be taken to deal with the tiles at the border of the iteration space. A border tile may have its origin located outside the iteration space or contain less points than a tile inside the iteration space or both.

Several code generation approaches have been proposed [1,12,21]. Ancourt and Irigoin's approach [1] constructs the tiled program by scanning linear constraints representing the tiled iteration space such that the tile loops never iterate over non-empty tiles. On the other hand, Wolf and Lam's approach [21] is simpler and faster but results in the execution of empty tiles at run time. In the case when the iteration space is irregular, many empty tiles may be created. Wolf and Lam's approach generates the tile loops to scan the points in the smallest rectangle enclosing the iteration space and then constructs the element loops to scan the points confined within a tile and the iteration space.

A tradeoff between the two approaches is possible. Let $BI \leqslant b$ define the iteration space of an n-deep loop nest, where $I_1, \cdots, I_n$ are n loop variables (counted from outermost to innermost). In (1), H can be expressed as:

$$\begin{bmatrix} I_1 \\ \vdots \\ I_n \end{bmatrix} = H^{-1} \begin{bmatrix} T_1 \\ \vdots \\ T_n \end{bmatrix} + \begin{bmatrix} E_1 \\ \vdots \\ E_n \end{bmatrix}, \text{ where } 0 \leqslant H \begin{bmatrix} E_1 \\ \vdots \\ E_n \end{bmatrix} < 1 \tag{5}$$

where $T_1, \cdots, T_n$ are the loop variables for the tiled loops and $E_1, \cdots, E_n$ for the element loops. Substituting for $I_1, \cdots, I_n$ from (5) into $BI \leqslant b$, we get the following inequalities:

$$B(H^{-1} \begin{bmatrix} T_1 \\ \vdots \\ T_n \end{bmatrix} + \begin{bmatrix} E_1 \\ \vdots \\ E_n \end{bmatrix}) \leqslant b, \quad 0 \leqslant H \begin{bmatrix} E_1 \\ \vdots \\ E_n \end{bmatrix} < 1 \tag{6}$$

1. **Creating the tile loops.** There are two approaches to generating the limits for the tile loops. If we disallow the tile loops to create any empty tiles, then we may use Ancount and Irigoin's algorithm [1], Feautrier's PIP [5] or Pugh's Omega system [16]. Otherwise, Fourier-Motzkin elimination or similar algorithms may be used. In the second approach, the empty tiles created are never one tile away from the border of the iteration space, as may be the case when Wolf and Lam's approach is used.
2. **Creating the element loops.** There are two approaches depending on whether the loop body needs to be modified or not. The first approach generates the limits for the element loops from (6) using any of the existing algorithms mentioned above. In this case, we must substitute for $I_1, \cdots, I_n$ from (5) into the array subscripts in the loop body. The second approach avoids having to modify the loop body by generating the limits for the element loops from the inequalities:

$$B \begin{bmatrix} I_1 \\ \vdots \\ I_n \end{bmatrix} \leqslant b, \quad 0 \leqslant H(\begin{bmatrix} I_1 \\ \vdots \\ I_n \end{bmatrix} - H^{-1} \begin{bmatrix} T_1 \\ \vdots \\ T_n \end{bmatrix}) < 1$$

where $I_1, \cdots, I_n$ will be used as the loop variables for the element loops.

8. Conclusion

In this paper, we discussed how to calculate the transformed dependences after a tiling transformation is applied. While Irigoin and Triolet's atomic tiling constraint is sufficient, this paper gave a sufficient and necessary condition to check the legality of a tiling transformation. A condition was also identified under which both Irigoin and Triolet's and our constraints are equivalent. The results of this paper were discussed in terms of their implications on dependence abstractions suitable for legality test.

Prior work has shown that tiling is very useful in improving cache locality and parallelism of a program. The results of this paper can be used to find optimal tilings to optimise a given goal. This paper demonstrated this by finding communication-minimal tilings for nested loops that are uniform if a few global dependences are disregarded.

9. Acknowledgements

I am grateful to the anonymous referees for their valuable comments which have helped in improving the quality of this paper. This work is supported by an Australian Research Council Grant A49600987.

References

1. C. Ancourt and F. Irigoin. Scanning polyhedra with DO loops. In *PROc. of the 3rd ACM SIGPLAN Symp. on Principles & Practice of Parallel Programming (PPoPP)*, pages 39–50. ACM Press, Apr. 1991.
2. U. Banerjee. *Dependence Analysis for Supercomputing*. Engineering and Computer Science: Parallel Processing and Fifth Generation Computing. Kluwer Academic Publishers, 1988.
3. P. Boulet, A. Darte, T. Risset, and Y. Robert. (Pen)-ultimate tiling. *Integration, the VLSI Journal*, 17:33–51, 1994.

4. S. Carr and K. Kennedy. Compiler blockability of numerical algorithms. In *Supercomputing '92*, pages 114–124, Minneapolis, Minn., Nov. 1992.
5. J.-F. Collard, T. Risset, and P. Feautrier. Construction of DO loops from systems of affine constraints. Technical Report 93-15, Ecole Normale Supérieure de Lyon, May. 1993.
6. K. Cooper, K. Kennedy, and N. McIntosh. Cross-loop reuse analysis and its application to cache optimizations. In *Proc. of the 9th Workshop on Languages and Compilers for Parallel Computing*, Aug. 1996.
7. A. Darte and F. Vivien. Optimal fine and medium grain parallelism detection in polyhedral reduced dependence graphs. In *Proc. of the 1996 International Conference on Parallel Architectures and Compilation Techniques*, pages 281–291, Boston, MA,, 1996.
8. J. J. Dongarra, S. J. Hammarline, and D. C. Sorensen. Block reduction of matrices to condensed forms for eigenvalue computations. *J. of Computer Application and Mathematics*, 27:216–227, 1989.
9. K. Gallivan, W. Jalby, U. Meier, and A. H. Sameh. Impact of hierarchical memory systems on linear algebra algorithm design. *Int. J. of Supercomputer Applications*, 2:12–48, 1988.
10. G. R. Gao, V. Sarkar, and S. Han. Locality analysis for distributed shared-memory multiprocessors. In *Proc. of the 9th Workshop on Languages and Compilers for Parallel Computing*, Aug. 1996.
11. Gina Goff, Ken Kennedy, and Chau-Wen Tseng. Practical dependence testing. In *Proc. of the ACM SIGPLAN'91 Conf. on Programming Language Design and Implementation*, pages 15–29, Toronto,, Jun. 1991.
12. F. Irigoin and R. Triolet. Supernode partitioning. In *Proc. of the 15th Annual ACM Symposium on Principles of Programming Languages*, pages 319–329, San Diego, California., Jan. 1988.
13. M. S. Lam, E. E. Rothberg, and M. E. Wolf. The cache performance and optimizations of blocked algorithms. In *Proc. of the 2nd International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 63–74, Santa Clara, California, Apr. 1991.
14. D.E. Maydan, J. L. Hennessy, and M. S. Lam. Efficient and exact data dependence analysis. In *Proc. of the SIGPLAN 1991 PLDI*, pages 1–14, 1991.
15. H. Ohta, Y .Saito, M. Kainaga, and H. Ono. Optimal tile size adjustment in compiling for general DOACROSS loop nests. In *1995 ACM International Conference on Supercomputing*, pages 270–279. ACM Press, 1995.
16. W. Pugh. The Omega test: A fast and practical integer programming algorithm for dependence analysis. *Comm. ACM*, 35(8):102–114, Aug. 1992.
17. J. Ramanujam and P. Sadayappan. Tiling multidimensional iteration spaces for multicomputers. *J. of Parallel and Distributed Computing*, 16(2):108–230, Oct. 1992.
18. A. Rogers and K. Pingali. Compiling for distributed memory architectures. *IEEE Transactions on Parallel and Distributed Systems*, 5(3):281–298, Mar. 1994.
19. R. Schreiber and J. J. Dongarra. Automatic blocking of nested loops. Technical Report 90.38, RIACS, May 1990.
20. G. Shephard. *Tilings and Patterns: An Introduction*. Series in Mathematical Sciences. W. H. Freeman and Co., 1989.
21. Michael E. Wolf, , and Monica S. Lam. A data locality optimizing algorithm. In *Proc. of the ACM SIGPLAN'91 Conf. on Programming Language Design and Implementation*, Jun. 1991.

22. M. J. Wolfe. More iteration space tiling. In *Supercomputing '88*, pages 655–664, Nov. 1989.

23. M. J. Wolfe. *Optimizing Supercompilers for Supercomputers*. Research Monographs in Parallel and Distributed Computing. MIT Press, 1989.

24. M. J. Wolfe. *High Performance Compilers for Parallel Computing*. Addision-Wesley, 1996.

25. J. Xue. Communication-minimal tiling of uniform dependence loops. In *Proc. of the 9th Workshop on Languages and Compilers for Parallel Computing*, Lecture Notes in Computer Science, pages 300–319, San Jose, 1996. Springer-Verlag.

26. Y.Q. Yang, C. Ancourt, and F. Irigoin. Minimal data dependence abstractions for loop transformations. In *Proc. of the 7th Workshop on Languages and Compilers for Parallel Computing*, Ithaca, Aug 1994.