
COMP 515: Advanced Compilation for Vector and Parallel Processors

Vivek Sarkar
Department of Computer Science
Rice University
vsarkar@rice.edu

<https://wiki.rice.edu/confluence/display/PARPROG/COMP515>

COMP 515

Lecture 21

15 November, 2011



Acknowledgments

- “A Loop Transformation Theory and an Algorithm to Maximize Parallelism” Michael E. Wolf and Monica S. Lam, IEEE TPDS, October 1991
- Slides on Unimodular transformations from “Loop Transformations” lecture
 - www.es.ele.tue.nl/~heco/courses/pam/loop-trafos.ppt

Announcements / Questions

- The next lecture will be on Polyhedral Transformations, and will be given by Dr. Louis-Noel Pouchet from Ohio State University. It will held in DH 3076 during 9:30am - 11:45am on Friday, Nov 18th.
 - There will be no lecture on Thursday, Nov 17th
- We only have 3 lectures remaining (Nov 22, Nov 29, Dec 1)
 - Suggestions welcome on which topics you'd like to focus on
 - Candidates include
 - Instruction Scheduling (Chapter 10)
 - Interprocedural Analysis and Optimization (Chapter 11)
 - Compiling Array Assignments (Chapter 13)
 - Array data flow analysis and Array SSA form
 - How much of Chapters 10 and 11 are already covered in COMP 512?

Transformation Frameworks

- Goal: develop a unified transformation framework in which legality testing and code generation for different transformations can be unified
 - Textbook approach: catalog of transformations
 - Pro: Generality
 - Con: each transformation needs special-case handling
 - Today's lecture: unimodular transformations
 - Pro: unimodular matrix is a simple composable representation
 - Con: limited to transformations of perfect loop nests
 - Friday's lecture: polyhedral transformations
 - Pro: more general than unimodular transformations (includes many cases of loop distribution and fusion)
 - Con: limited to transformation of "static control parts" (SCoP's)
 - PLDI 92: General Framework for Iter-Reordering Transformations
 - Pro: more general than unimodular and some cases of polyhedral
 - Con: limited to transformations of perfect loop nests

Unimodular Matrices

- A unimodular matrix T is a matrix with integer entries and determinant ± 1 .
- This means that such a matrix maps an object onto another object with exactly the same number of integer points in it.
- Its inverse T^{-1} always exist and is unimodular as well.

Unimodular Transformation

$$i' = U \times i$$

New loop index

Unimodular matrix

Old loop index

- A **unimodular matrix** is a square integer matrix that has unit determinant
- A unimodular matrix can be used to specify a unimodular transformation (UT) of a perfect loop nest with affine (trapezoidal) loop bounds
- Any two UTs can be composed to form a new UT by multiplying their unimodular matrices
- The new loop execution order is determined by the transformed index.

Types of Unimodular Transformations

- Loop interchange
- Loop reversal
- Loop skewing with an arbitrary skew factor
- Since unimodular transformations are closed under multiplication, any combination is a unimodular transformation again.

Specifying Loop Interchange as a Unimodular Transformation

$$\begin{bmatrix} i1' \\ i2' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} i1 \\ i2 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} i1 \\ i2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} i1' \\ i2' \end{bmatrix}$$

Original loop nest

```
for (i1 = ...)
  for (i2 = ...) { BODY(i1, i2) }
```

Transformed loop nest

```
for (i1' = ...)
  for (i2' = ...) {
    i1 = i2'; i2 = i1'; BODY(i1, i2) }
```


Specifying Loop Reversal as a Unimodular Transformation

$$\begin{bmatrix} i1' \\ i2' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} i1 \\ i2 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} i1 \\ i2 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} i1' \\ i2' \end{bmatrix}$$

Original loop nest

```
for (i1 = ...)
  for (i2 = ...) { BODY(i1,i2) }
```

Transformed loop nest

```
for (i1' = ...)
  for (i2' = ...) {
    i1 = -i1'; i2 = i2'; BODY(i1,i2) }
```

Specifying Loop Skewing as a Unimodular Transformation

$$\begin{bmatrix} i1' \\ i2' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} i1 \\ i2 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} i1 \\ i2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} i1' \\ i2' \end{bmatrix}$$

Original loop nest

```
for (i1 = ...)
  for (i2 = ...) { BODY(i1, i2) }
```

Transformed loop nest

```
for (i1' = ...)
  for (i2' = ...) {
    i1 = i1'; i2 = -i1' + i2'; BODY(i1, i2) }
```

Loop Skewing (Recap from Lecture 10)

```
DO I = 1, N ! Original DV = { (<,=), (=, <) }
```

```
DO J = 1, N
```

```
S:      A(I,J) = A(I-1,J) + A(I,J-1)
```

```
ENDDO
```

```
ENDDO
```

After skewing ...

```
DO I' = 1, N ! Transformed DV = { (<,<), (=, <) }
```

```
DO J' = I'+1, I'+N
```

```
      I = I' ; J = J' - I' ;
```

```
S:      A(I,J) = A(I-1,J) + A(I,J-1) ! Loop body is unchanged
```

```
ENDDO
```

```
ENDDO
```

Loop interchange to ...

```
DO J' = 2, N+N ! Transformed DV = { (<,<), (<, =) }
```

```
DO I' = max(1,J'-N), min(N,J'-1)
```

```
S:      A(I,J) = A(I-1,J) + A(I,J-1) ! Loop body is unchanged
```

```
ENDDO
```

```
ENDDO
```

Data Dependence Legality Test for Unimodular Transformations

- Given an input distance vector d , the transformed distance vector can be computed as the matrix vector product, $d' = T \times d$, where T is the matrix for the unimodular transformation
- Extend to direction vectors by treating direction entries as ranges of integers
- Check if transformed set of dependence vectors contains a lexicographically negative entry
 - If so, transformation T is illegal

Applying Unimodular Data Dependence Test for Loop Interchange

Reduced case analysis of the direction vectors $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} j \\ i \end{bmatrix}$

(=,=)

The dependence is loop independent, so it is unaffected by interchange

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

(=,<)

The dependence is carried by the j loop.

After interchange the dependence will be (<=), so the dependence will still be carried by the j loop, so the dependence relations do not change.

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

(<,>)

The dependence is carried by the outer loop.

After interchange the dependence will be (>,<), which changes the dependences and results in an illegal direction vector, so interchange is illegal.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} < \\ > \end{bmatrix} = \begin{bmatrix} > \\ < \end{bmatrix}$$

Iteration Space Mapping

- A loop nest is represented as $Bl \leq b$ for iteration vector l
- Example:

$$\begin{array}{l} \text{for}(i=0; i<10;i++) \\ \quad \text{for}(j=i; j<10;j++) \end{array} \begin{pmatrix} -1 & 0 \\ 1 & 0 \\ 1 & -1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix} \leq \begin{pmatrix} 0 \\ 9 \\ 0 \\ 9 \end{pmatrix}$$

- Why does the loop bound matrix have $2N$ rows and N columns for a nest with N loops?

Iteration Space Mapping (contd)

- How to generate new loop bounds after performing a unimodular transformation T ?
- Original loop bounds, $B \times I \leq b$ for iteration vector I
- Transformed iteration vector, $I' = T \times I$
- Transformed loop bounds, $B \times \text{inverse}(T) \times I' \leq b$ for iteration vector I'
- Use inequalities to collect loop bounds
 - In general, insert max functions for lower bounds and min functions for upper bounds

Some fundamental results for Unimodular Transformations (contd)

- “A Loop Transformation Theory and an Algorithm to Maximize Parallelism”, Michael E. Wolf and Monica S. Lam, IEEE TPDS, October 1991

Theorem 3.1: Let $L = \{I_1, \dots, I_n\}$ be a loop nest with lexicographically positive distance vectors $\vec{d} \in D$. The loops in the loop nest can be made fully permutable by skewing.

- Example: distance vectors $\{(0,1), (1,-2), (1,-1)\}$ can be transformed to $\{(0,1), (1,0), (1,1)\}$ by applying the following unimodular transformation:

$$\begin{array}{cc} 1 & 0 \\ 2 & 1 \end{array}$$

- Above result can be extended to direction vectors in some cases

Some fundamental results for Unimodular Transformations (contd)

Theorem 6.1: If loop I_k has dependences such that $\exists \vec{d} \in D : d_k^{\min} = -\infty$ and $\exists \vec{d} \in D : d_k^{\max} = \infty$ then the outermost fully permutable nest consists only of a combination of loops not including I_k .

Theorem 6.2: Let $L = \{I_1, \dots, I_n\}$ be a loop nest with lexicographically positive dependences $\vec{d} \in D$, and $D^i = \{\vec{d} \in D \mid (d_1, \dots, d_{i-1}) \neq \vec{0}\}$. Loop I_j can be made into a fully permutable nest with loop I_i , where $i < j$, via reversal and/or skewing, if

$$\forall \vec{d} \in D^i : (d_j^{\min} \neq -\infty \wedge (d_j^{\min} < 0 \rightarrow d_i^{\min} > 0)) \text{ or}$$
$$\forall \vec{d} \in D^i : (d_j^{\max} \neq \infty \wedge (d_j^{\max} > 0 \rightarrow d_i^{\min} > 0)) .$$

Proof of Theorem 6.2

Proof: All dependence vectors for which $(d_1, \dots, d_{i-1}) \succ \vec{0}$ do not prevent loops I_i and I_j from being fully permutable and can be ignored. If

$$\forall \vec{d} \in D^i : (d_j^{\min} \neq -\infty \wedge (d_j^{\min} < 0 \rightarrow d_i^{\min} > 0))$$

then we can skew loop I_j by a factor of f with respect to loop I_i where

$$f \geq \max_{\{\vec{d} \mid \vec{d} \in D^i \wedge d_i^{\min} \neq 0\}} \lceil -d_j^{\min} / d_i^{\min} \rceil$$

to make loop I_j fully permutable with loop I_i . If instead the condition

$$\forall \vec{d} \in D^i : (d_j^{\max} \neq \infty \wedge (d_j^{\max} > 0 \rightarrow d_i^{\min} > 0)) .$$

holds, then we can reverse loop I_j and proceed as above. \square

Non-perfectly nested loops

- Unimodular transformations have multiple constraints:
 1. They only transform perfectly nested loops
 2. They only reorder iterations, not statement (e.g., they do not support loop distribution and fusion)
 3. They require loop bounds to be trapezoidal
- Polyhedral frameworks address constraints 1 and 2
- Note: a non-perfectly nested loop can be converted to a perfect loop nest using statement guards
 - Pro: enables application of iteration-reordering loop transformations
 - Con: guard evaluation introduces extra overhead


Example of converting a non-perfectly nested loop to a perfectly nested loop

Non-perfectly nested loop:

```
DO I1 = 1,3
  A(I1) = A(I1-1)
  DO I2 = 1,4
    B(I1,I2) = B(I1-1,I2)+B(I1,I2-1)
  ENDDO
ENDDO
```

Perfectly nested loop:

```
DO I1 = 1,3
  DO I2 = 1,4
    DO I3 = 0,1
      IF (I2.EQ.1.AND.I3.EQ.0) A(I1) = A(I1-1)
      ELSE IF (I3.EQ.1) B(I1,I2)=B(I1-1,I2)+B(I1,I2-1)
    ENDDO
  ENDDO
ENDDO
```



REMINDER: Homework #6 (Written Assignment)

Read Section 6 (Memory Cost Analysis) of the following paper discussed in today's lecture, especially the partial derivative analysis on pg 15 (printed page 247):

• [Automatic Selection of High Order Transformations in the IBM XL Fortran Compilers](#). Vivek Sarkar. IBM Journal of Research and Development, 41(3), May 1997

1. Compute the memory cost function and partial derivatives for loops I and J in the following loop nest at the start of Section 9.3.5 of the course textbook. Which loops carry locality? Can all of them be moved to the innermost position?

```
DO I = 1, N
  DO J = 1, M
    A(J+1) = (A(J)+A(J+1))/2
  ENDDO
ENDDO
```

2. Compute the memory cost and partial derivatives for loops I and J in the following transformed loop nest (after skewing) in Section 9.3.5 of the course textbook. Which loops carry locality? Can all of them be moved to the innermost position?

```
DO I = 1, N
  DO j = I, M + I - 1
    A(j-I+2) = (A(j-I+1)+A(j-I+2))/2
  ENDDO
```

```
ENDDO
```

Homework #6 (contd)

- You can make the following simplifying assumptions
 - Only calculate memory cost for a single level of cache, and ignore the TLB
 - Assume a cache line size of $L = 32B$, and an array element size of $8B$ (real*8)
- Homework due by 5pm on Tuesday, November 15th
- Homework should be turned into Amanda Nokleby, Duncan Hall 3137
- Honor Code Policy: All submitted homeworks are expected to be the result of your individual effort. You are free to discuss course material and approaches to problems with your other classmates, the teaching assistants and the professor, but you should never misrepresent someone else's work as your own. If you use any material from external sources, you must provide proper attribution.