17-2-R01

A Loop Transformation for Nested Loops with Non-uniform and Flow Dependences

정 삼 진
천안대학교 정보통신학부 컴퓨터학전공
sjjeong@cheonan.ac.kr

ABSTRACT

Several methods are proposed in order to parallelize loops with non-uniform and flow dependences, but most of such approaches perform poorly due to irregular and complex dependence constraints. This paper proposes an improved tiling method of nested loops with non-uniform and flow dependences for maximizing parallelism. Our approach is based on the Convex Hull theory which has adequate information to handle non-uniform dependences, and also based on minimum dependence distance tiling method. We will first show how to find the incrementing minimum dependence distance. Next, we will propose how to tile the iteration space efficiently according to the incrementing minimum dependence distance and how to transform it into parallel loops. Comparison with some other methods shows more parallelism than other existing methods.

1. INTRODUCTION

The existing parallelizing compilers can parallelize most of the loops with uniform dependences, but they do not satisfactorily handle loops with non-uniform dependences. Most of the time, the compiler leaves such loops running sequentially. Unfortunately, loops with non-uniform dependences are not so uncommon in the real world.

Several works have been done for loops with non-uniform dependences. All of the existing techniques do a good job for some particular types of loops, but show us a poor performance on some other types of loops.

Some techniques, based on Convex Hull theory [7] that has been proven to have enough information to handle non-uniform dependences, are the minimum dependence distance tiling method [5], [6], the unique set oriented partitioning method [4], the three region partitioning [9], and the improved three region partitioning method [3].

This paper will focus on parallelizing perfectly nested loops with non-uniform and flow dependences.

The rest of this paper is organized as follows. Section two describes our loop model,

and reviews some fundamental concepts. Section three presents an improved tiling method for parallelization with nested loops with non-uniform and flow dependences. Section four shows comparison with related works. Finally, we conclude in section five with the direction to enhance this work.

2. PROGRAM MODEL AND DEPENDENCE ANALYSIS

The loop model considered in this paper is doubly nested loops with linearly coupled subscripts and both lower and upper bounds for loop variables should be known at compile time. The loop model has the form in Figure 1, where $f_1(I, J)$, $f_2(I, J)$, $f_3(I, J)$, *and* $f_4(I, J)$ are linear functions of loop variables.

```
do I = l1, u1
do J = l2, u2
A[f1(I, J), f2(I, J)) = . . .
        . . .  =A[f3(I, J), f4(I, J))
```

Figure 1. A doubly nested loop model

The loop in Figure 1 carries cross iteration dependences if and only if there exist four integers (i_1, j_1, i_2, j_2) satisfying the system of linear diophantine equations given by Eq. (1) and the system of inequalities given by Eq. (2).

The general solution to these equations can be computed by the extended GCD [1] or the power test algorithm [8] and forms a **DCH** (Dependence Convex Hull).

$$f_1(i_1, j_1) = f_3(i_2, j_2) \text{ and } f_2(i_1, j_1) = f_4(i_2, j_2) \quad (1)$$

$$l_1 \le i_1, i_2 \le u_1 \text{ and } i_2 \le j_1, j_2 \le u_2 \quad (2)$$

From Eq. (1), (i_1, j_1, i_2, j_2) can be represented as

$(i_1, j_1, i_2, j_2) = (g_1(i_2, j_2), g_2(i_2, j_2), g_3(i_1, j_1), g_4(i_1, j_1))$

where g_i are linear functions.

From Eq. (2), two sets of inequalities can be written as

$$l_1 \le i_1 \le u_1 \text{ and } i_2 \le j_1 \le u_2 \text{ and}$$
$$l_1 \le g_3(i_1, j_1) \le u_1 \text{ and } i_2 \le g_4(i_1, j_1) \le u_2 \quad (3)$$
$$l_1 \le i_2 \le u_1 \text{ and } i_2 \le j_2 \le u_2 \text{ and}$$

$$l_1 \le g_1(i_2, j_2) \le u_1 \text{ and } i_2 \le g_2(i_2, j_2) \le u_2 \quad (4)$$

And, Eq. (3) and (4) form DCHs denoted by DCH1 and DCH2, respectively [4]. Clearly, if we have a solution (i_1, j_1) in DCH1, we must have a solution (i_2, j_2) in DCH2, because they are derived from the same set of Eq. (1). The union of DCH1 and DCH2 is called Complete DCH (**CDCH**), and all dependences lie within the CDCH.

If iteration (i_2, j_2) is dependent on iteration (i_1, j_1), then we have a dependence vector

$$d(i_1, j_1) = (d_i(i_1, j_1), d_j(i_1, j_1)) = (i_2-i_1, j_2-j_1)$$

So, for DCH1, we have

$$d_i(i_1, j_1) = g_3(i_1, j_1) - i_1 \text{ and}$$
$$d_j(i_1, j_1) = g_4(i_1, j_1) - j_1 \quad (5)$$

For DCH2, we have

$$d_i(i_2, j_2) = i_2 - g_1(i_2, j_2) \text{ and}$$
$$d_j(i_2, j_2) = j_2 - g_2(i_2, j_2) \quad (6)$$

The properties of DCH1 and DCH2 can be found in [4].

3. IMPROVED TILING METHOD

Cho and Lee [2] present a more general and powerful loop splitting method to enhance all parallelism on a single loop. The method uses more information from the loop such as increment factors, and the difference between the distance of a dependence, and that of the next dependence. Cho and Lee [3] derive an efficient method for nested loops with simple scripts from enhancing [2].

The minimum dependence distance tiling method [6] presents an algorithm to convert the extreme points with real coordinates to the extreme points with integer coordinates. The method obtains an **IDCH** (Integer Dependence Convex Hull) from a DCH. It can compute d_{imin}, that is the minimum value of the dependence distance function $d_i(i_1, j_1)$, and d_{jmin}, that is the minimum value of the dependence distance function $d_j(i_1, j_1)$, from the extreme points of the IDCH. The first minimum dependence distances d_{imin} and d_{jmin} are used to determine the uniform tile size in the iteration space.

3.1. Data Dependence Analysis in Non-uniform and Flow Dependence Loops

From our doubly nested loop model shown in Figure 1, the array subscripts, $f_1(I, J)$, $f_2(I, J)$, $f_3(I, J)$, and $f_4(I, J)$, can be written as

$$\begin{aligned} f_1(I, J) &= a_{11}i + b_1 j + c_{11} \\ f_2(I, J) &= a_{12}i + b_1 j + c_{12} \\ f_3(I, J) &= a_{21}i + b_2 j + c_{21} \\ f_4(I, J) &= a_{22}i + b_2 j + c_{22} \end{aligned} \quad (7)$$

By the system of linear Diophantine equations given by Eq. (7), Figure 1 can be written as follows.

```
do i = l1, u1
do j = l2, u2
A(a11i + b1 j + c11, a12i + b1 j + c12) = ...
   ... = A(a21i + b2 j + c21, a22i + b2 j + c22)
enddo
enddo
```

Figure 2. Another form of Figure 1

The system of linear Diophantine equations given by Eq. (1) is written as follows

$$\begin{aligned} a_{11}i + b_1 j + c_{11} &= a_{21}i + b_2 j + c_{21} \\ a_{12}i + b_1 j + c_{12} &= a_{22}i + b_2 j + c_{22} \end{aligned} \quad (8)$$

The dependence distance function $d(i_1, j_1)$ in flow dependence loops gives the dependence distances $d_i(i_1, j_1)$ and $d_j(i_1, j_1)$ in dimensions i and j, respectively. For uniform dependence vector sets these distances are constant. But, for the non-uniform dependence sets these distances are linear functions of the loop indices. We can write these dependence distance functions in a general form as

$$\begin{aligned} d(i_1, j_1) &= (d_i(i_1, j_1), d_j(i_1, j_1)) \\ d_i(i_1, j_1) &= p_1 * i_1 + q_1 * j_1 + r_1 \end{aligned}$$

$$\epsilon_j(i_1, j_1) = p_2 * i_1 + q_2 * j_1 + r_2 \quad (9)$$

where p_i, q_i, and r_i are real values and i_1, j_1, i_2, and j_2 are integer variables of the iteration space.

The properties and theorems for tiling of nested loops with flow dependence can be described as follows.

Theorem 3.1 *If there is only flow dependence in the loop, DCH1 contains flow dependence tails and DCH2 contains flow dependence heads.*

Proof: The system of inequalities in Eq. (3) defines DCH1 and $i_1 = x_1$, $j_1 = y_1$, $i_2 = \alpha_{11}x_1 + \beta_1 y_1 + \gamma_{11}$, and $j_2 = \alpha_{12}x_1 + \beta_1 y_1 + \gamma_{12}$. If there exists flow dependence, we can assume that (i_1, j_1, i_2, j_2) is a solution to the flow dependence. From the definition of flow dependence, (i_1, j_1) should be written somewhere in the iteration space before (i_2, j_2) is referenced. So the dependence and execution order in the iteration space is from (i_1, j_1) to (i_2, j_2), which is equivalent to (x_1, y_1) to $(\alpha_{11}x_1 + \beta_1 y_1 + \gamma_{11}, \alpha_{12}x_1 + \beta_1 y_1 + \gamma_{12})$. Here, (x_1, y_1) is the flow dependence tail. Since (x_1, y_1) satisfies Eq. (3) and we have assumed that (i_1, j_1, i_2, j_2) is a solution, DCH1 must contain flow dependence tails.

The proof that DCH2 contains flow dependence heads is similar to the proof for DCH1.

Theorem 3.2 *If there is only flow dependence in the loop, then $a_i(x, y) = 0$ or $\epsilon_j(x, y) = 0$ does not pass through any DCH.*

Proof: $a_i(x_1, y_1)$ and $\epsilon_j(x_1, y_1)$ corresponds to DCH1 in dimensions i and j, respectively, and $a_i(x_2, y_2)$ and $\epsilon_j(x_2, y_2)$ corresponds to DCH2. From the definition of flow dependence, we have either $i_1 < i_2$ or $i_2 = i_1$ and $j_2 > j_1$. Let us consider the case that $i_1 < i_2$. Since $a_i(x_1, y_1)$ is $i_2 - i_1$, DCH1 is on the side of $a_i(x_1, y_1) > 0$. So $a_i(x_1, y_1) = 0$ does not pass through DCH1. In the case that $i_2 = i_1$ and $j_2 > j_1$, DCH1 is on the side of $\epsilon_j(x_1, y_1) > 0$ because $\epsilon_j(x_1, y_1)$ is $j_2 - j_1$. Thus, $\epsilon_j(x_1, y_1) = 0$ does not pass through DCH1.

The proof for DCH2 follows similarly.

Theorem 3.3 *If there is only flow dependence in the loop, the minimum and maximum values of the dependence distance function $d(x_1, y_1)$ appear on the extreme points.*

Proof: From theorem 3.2, when there exists only flow dependence in the loop, $a_i(x_1, y_1) = 0$ or $\epsilon_j(x_1, y_1) = 0$ does not pass through any IDCH, and IDCH is on the side of $a_i(x_1, y_1) > 0$ or $\epsilon_j(x_1, y_1) > 0$. From property in [5], we know that the minimum and maximum values of the dependence distance function $d(x_1, y_1)$ appear on the extreme points. The lines $a_i(x_1, y_1) = a_{imin}$ and $\epsilon_j(x_1, y_1) = \epsilon_{jmin}$ are tangential to the IDCH. Since both a_{imin} and ϵ_{jmin} are positive, the minimum and maximum values are the minimum and maximum values of the dependence distance function $d(x_1, y_1)$.

Theorem 3.4 *If there is only flow dependence in the loop, the minimum dependence distance value a_{imin} is equal or greater than zero.*

Proof: Let (i_1, j_1) and (i_2, j_2) be the iterations that cause any flow dependence. Then, (i_1, j_1) and (i_2, j_2) satisfy Eq. (8). From the definition of flow dependence, we have either $i_1 < i_2$, or $i_2 = i_1$ and $j_2 > j_1$. We can now solve Eq. (8) with $i_1 = x_1$, $j_1 = y_1$, $i_2 = \alpha_{11}x_1 + \beta_1 y_1 + \gamma_{11}$, $j_2 = \alpha_{12}x_1 + \beta_1 y_1 + \gamma_{12}$.

Let us consider the case that $i_1 < i_2$. Since $x_1 < \alpha_{11}x_1 + \beta_1 y_1 + \gamma_{11}$, we have $(\alpha_{11}-1)x_1 + \beta_1 y_1 + \gamma_{11} = d_i(x_1, y_1) > 0$. So $d_{imin} > 0$.

In the case that $i_2 = i_1$ and $j_2 > j_1$, since $x_1 = \alpha_{11}x_1 + \beta_1 y_1 + \gamma_{11}$ and $y_1 < \alpha_{12}x_1 + \beta_1 y_1 + \gamma_{12}$, we also have $d_i(x_1, y_1) = 0$ and $d_j(x_1, y_1) > 0$. So $d_{imin} = 0$ and $d_{jmin} > 0$.

From theorem 3.4, we know that when there is only flow dependence in the loop and d_{imin} is zero, d_{jmin} is greater than zero. In this case, since $d_j(x_1, y_1) = 0$ does not pass through the IDCH, the minimum value of $d_j(x_1, y_1)$, d_{jmin}, occurs at one of the extreme points.

Theorem 3.5 *If there is only flow dependence in the loop, the difference between the distance of a dependence and that of the next dependence, d_{inc}, is equal to or greater than zero.*

Proof: Let i, i+1 be values for the source of a dependence, and that of the next dependence, respectively. Let Dist(i, j), Dist(i+1, j) be the distances of a dependence and that of the next dependence, respectively. Then, from the dependence distance function given in Eq. (9), we can know that Dist(i, j) = p_1*i + q_1*j + r_1 and Dist(i+1, j) = p_1*(i+1) + q_1*j+ r_1. Thus, the difference d_{inc} = Dist(i+1, j) - Dist(i, j) = p_1.

Case 1: when ($p_1 \geq 1$), $d_{inc} \geq 1$.

Case 2: when ($0 < p_1 < 1$),

Let Dist(i, j) = l+ceil(n_1/m) = l+1

where $l \geq 0$ and $m > n_1$.

Let $d_{inc} = p_1 = n_2/m$ where $m > n_2$.

If $(n_1+n_2)/m > 1$, then Dist(i+1, j) = l+2;

So, $d_{inc} = 1$.

If $(n_1+n_2)/m \leq 1$, then Dist(i+1, j) = l+1;

So, $d_{inc} = 0$.

In this case, $d_{inc} = 1$ or $d_{inc} = 0$.

Case 3: when ($p_1 = 0$), $d_{inc} = 0$.

Thus, d_{inc} is equal to or greater than zero when there is only flow dependence in the loop.

3.2. Loop Tiling and Transformation for Non-uniform and Flow Dependence Loops

From theorem 3.5, when $p_1 > 0$ and $q_1 \geq 0$, we know that the difference between the distance of a dependence and that of the next dependence in loop with flow dependence, d_{inc}, is equal to or greater than zero.

For each i_1, d_{imin} is incremented as the value of i_1 is incremented. So, the second d_{imin} is equal to or greater than the first one, and the third one is greater than the second one, and so on.

The improved tiling method for doubly nested loops with non-uniform and flow dependence is described as **Algorithm Tiling_Method**, which is the algorithm of tiling loop by the incrementing minimum dependence distance as shown in Figure 3. This algorithm computes the incrementing

minimum dependence distance, tiles the iteration space efficiently according to the incrementing minimum dependence distance, and transforms it into parallel loops.

Algorithm Tiling_Method (i_1, j_1, l_1, l_2, u_1, u_2, $d_i(i_1, j_1)$)

i_1, j_1 : i and j value for the source of the first minimum dependence in the loop computed by the extreme points of the IDCH

l_1, l_2, u_1, u_2 : the lower and upper bounds of outer loop and inner loop, respectively

$d_i(i_1, j_1)$: the dependence distance function of the IDCH

St_n : i value for the first iteration in the nth tile

Sr_n : i value for the source of the first dependence in the nth tile

Tg_n : j value for the target of the first dependence in the n-1th tile.

$Dist_n$: the minimum dependence distance value in the nth tile

S_d : the number of i iterations between a dependence source and next source,

S_d is given by $|a_{21}|/g$ iterations where $g = gcd(a_{11}, a_{21})$ or $|a_{22}|/g$ iterations where $g = gcd(a_{12}, a_{22})$ if (a_{11}=0 or a_{21} =0)

Step 1: $n = 1$; $St_1 = l_1$; $Sr_1 = i_1$;
  $Dist_1 = ceil(p_1*Sr_1 + q_1* j_1 + r_1)$; /*first d_{imin} */
  $St_2 = Sr_1 + Dist_1$;
  If ($b_{21} == 0$) then
    $Tg_2 = (a_{12}*Sr_1 + b_{12}*j_1 + c_{12} - a_{22}*St_2 - c_{22}) / b_{22}$;
  Else $Tg_2 = (a_{11}*Sr_1 + b_{11}*j_1 + c_{11} - a_{21}*St_2 - c_{21}) / b_{21}$;
  If ($St_2 \geq u_1$ or $Tg_2 \geq u_2$) then
    { $St_2 = u_1 + 1$; goto Step 4}
  Else {goto Step 3};

Step 2: $Dist_n = ceil(p_1*Sr_n + q_1*l_2 + r_1)$; /* d_{imin} */
  $St_{n+1} = Sr_n + Dist_n$;
  If ($b_{21} == 0$) then
    $Tg_{n+1} = (a_{12}*Sr_n + b_{12}*l_2 + c_{12} - a_{22}*St_{n+1} - c_{22}) / b_{22}$;
  Else $Tg_{n+1} = (a_{11}*Sr_n + b_{11}*l_2 + c_{11} - a_{21}*St_{n+1} - c_{21}) / b_{21}$;
  If ($St_{n+1} \geq u_1$ or $Tg_{n+1} \geq u_2$) then
    { $St_{n+1} = u_1 + 1$; goto Step 4};

Step 3: $Sr_{n+1} = St_{n+1} + q$,
  where $0 \leq q < S_d$ and $q = (St_{n+1} - Sr_1) \bmod S_d$.
  n=n+1; Goto Step 2;
  /*Transforming the original loop into following parallel loop*/

Step 4: n = 1; $i = 1$;
  While $i <= u_1$ DO
  Inc = $St_{n+1} - St_n$;
  DOALL $i = i$, i+Inc-1
  DOALL $j = l_2$, u_2
  $A(a_{11}i + b_{1}j + c_{11}, a_{12}i + b_{1}j + c_{12}) = ...$
    ... = $A(a_{21}i + b_{2}j + c_{21}, a_{22}i + b_{2}j + c_{22})$
    ENDDOALL
    ENDDOALL
    $i = i$+Inc; n = n+1;
  ENDWhile

Figure 3. Algorithm of tiling loop by the incrementing minimum dependence distance

Example 1.

```
do i = 1, 50
do j = 1, 50
A(3*i+1, 4*i+2*j+1) = . . .
          . . .=A(2*i-1, i+j-4)
enddo
enddo
```

An example given in Example 1 illustrates the case that there is non-uniform and flow dependence. Figure 4(a) shows CDCH of Example 1. As the example, we can obtain the following results using the improved tiling method proposed in this section.

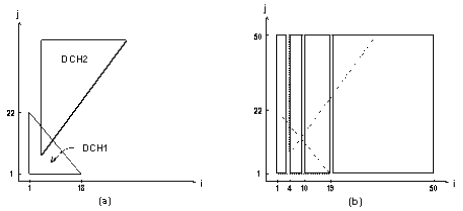

Figure 4. (a) CDCH, (b) Tiling by the incrementing minimum dependence distance in Example 1

From the algorithm to compute a two-dimensional IDCH in [5], we can obtain the extreme points such as (1, 1), (1, 22), and (18, 1) as shown in Figure 4(a). The first minimum value of $d_i(i_1, j_1)$ occurs at one of the extreme points. The i value for the source of the first dependence in the second tile is 4, The i value in the third tile is 10, and next one is 19. But, the last i value become 51 because the last i value (= 52) is greater than the upper boundary of inner loop u_2. Then, we can divide the iteration space by four tiles as shown in Figure 4(b). The result of this loop transformed by Algorithm Tiling_Method is shown in Figure 5.

```
DOALL i = 1, 3
DOALL j = 1, 50
A(3*i+1, 4*i+2*j+1) = . . .
   . . .=A(2*i-1, i+j-4)
ENDDOALL
ENDDOALL

DOALL i = 4, 9
DOALL j = 1, 50
A(3*i+1, 4*i+2*j+1) = . . .
. . .=A(2*i-1, i+j-4)
ENDDOALL
ENDDOALL

DOALL i = 10, 18
DOALL j = 1, 50
A(3*i+1, 4*i+2*j+1) = . . .
 . . .=A(2*i-1, i+j-4)
ENDDOALL
ENDDOALL

DOALL i = 19, 50
DOALL j = 1, 50
A(3*i+1, 4*i+2*j+1) = . . .
   . . .=A(2*i-1, i+j-4)
ENDDOALL
ENDDOALL
```

Figure 5. The result of loop transformed by Algorithm Tiling_Method

4. PERFORMANCE ANALYSIS

Theoretical speedup for performance analysis

can be computed as follows. Ignoring the synchronization, and scheduling overheads, and assuming an unlimited number of processors, each partition can be executed in one time step. Hence, the total time of execution is equal to the number of parallel regions, N_p, plus the number of sequential iterations, N_s. Generally, speedup is represented by the ratio of total sequential execution time to the execution time on parallel computer system as follows:

$Speedup = (N_i * N_j)/(N_p + N_s)$

where N_i, N_j are the size of loop i, j, respectively

Let's consider the loop shown in Example 1. Figure 4(a) shows original partitioning of Example 1. This example is the case that there is only flow dependence and DCH1 overlaps DCH2. Applying the unique set oriented partitioning method to this loop illustrates case 2 of [4]. This method can divide the iteration space into four regions: three parallel regions, AREA1, AREA2 and AREA4, and one serial region, AREA3, as shown in Figure 6. The speedup for this method is (50*50)/(3+44) = 53.2.

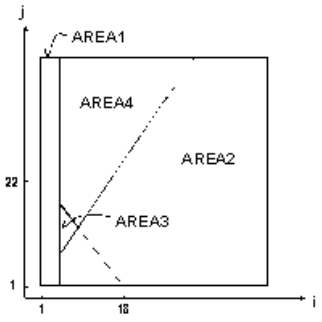

Figure 6. Regions of the loop partitioned by the unique sets oriented partitioning in Example 1.

Applying the minimum dependence distance tiling method to this loop illustrates case 1 of this technique [5], which is the case that line $d_i(i, j) = 0$ does not pass through the IDCH. The minimum value of $d_i(i, j)$, d_{imin}, occurs at the extreme point (1, 1) and $d_{imin} = 3$. The space can be tiled with width = 3, thus 17 tiles are obtained. The speedup for this method is (50*50)/17 =147.

This example is the case that there is non-uniform and flow dependence loop. Applying our proposed method to this loop, the loop is tiled by four areas as shown in Figure 4(b). The iterations within each area can be fully executed in parallel. So, the speedup for this method is (50*50)/4 = 625.

5. CONCLUSIONS

In this paper, we have studied the problem of transforming nested loops with non-uniform and flow dependences to maximize parallelism. The minimum dependence distance tiling method tiles the iteration space by the first minimum dependence distance uniformly. Our proposed method, however, tiles the iteration space by minimum dependence distance values that are incremented as the value for the source of minimum dependence is incremented.

In comparison with some previous methods, our proposed tiling method gives much better speedup than the minimum dependence distance tiling method and the unique set

oriented partitioning method.

Our future research work is to develop a method for improving parallelization of higher dimensional nested loops.

REFERENCES

[1] U. Banerjee, *Dependence Analysis for Supercomputing*, Kluwer Academic Publishers, 1988.

[2] C. K. Cho, J. C. Shim, and M. H. Lee, "A loop transformation for maximizing parallelism from single loops with non-uniform dependences," in *Proceedings of High Performance Computing Asia '97*, pp. 696-699, April 1997.

[3] C. K. Cho and M. H. Lee, "A loop parallization method for nested loops with non-uniform dependences", in *Proceedings of the International Conference on Parallel and Distributed Systems,* pp. 314-321, December 10-13, 1997.

[4] J. Ju and V. Chaudhary, "Unique sets oriented partitioning of nested loops with non-uniform dependences," in *Proceedings of International Conference on Parallel Processing*, vol. III, pp. 45-52, 1996.

[5] S. Punyamurtula and V. Chaudhary, "Minimum dependence distance tiling of nested loops with non-uniform dependences," in *Proceedings of Symposium on Parallel and Distributed Processing*, pp. 74-81, 1994.

[6] S. Punyamurtula, V. Chaudhary, J. Ju, and S. Roy, "Compile time partitioning of nested loop iteration spaces with non-uniform dependences," *Journal of Parallel Algorithms and Applications*, October 1996.

[7] T. Tzen and L. Ni, "Dependence uniformization: A loop parallelization technique," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 5, pp. 547-558. May 1993.

[8] M. Wolfe and C. W. Tseng, "The power test for data dependence," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 5, pp. 591-601, September 1992.

[9] A. Zaafrani and M. R. Ito, "Parallel region execution of loops with irregular dependences," in *Proceedings of the International Conference on Parallel Processing*, vol. II, pp. 11-19, 1994.

정 삼 진

1974년~1979년 경북대학교 고분자공학과(학사).
1985년~1986년 Indiana Univ. 전산학(석사).
1993년~2000년 충남대학교 전산학(박사).
1992년~1997년 혜천대학 전산정보처리과 조교수, 전자계산연구소장
1997~현재 천안대학교 정보통신학부 부교수
관심분야 : 병렬 컴파일러, 객체지향 프로그래밍 언어