# Enhanced Region Partitioning Method of Non-perfect nested Loops with Non-uniform Dependences

**Sam-Jin Jeong***
Division of Information and Communication Engineering
Cheonan University, Cheonan, Korea

*ABSTRACT*

*This paper introduces region partitioning method of non-perfect nested loops with non-uniform dependences. This kind of loop normally can't be parallelized by existing parallelizing compilers and transformations. Even when parallelized in rare instances, the performance is very poor. Based on the Convex Hull theory which has adequate information to handle non-uniform dependences, this paper proposes an enhanced region partitioning method which divides the iteration space into minimum parallel regions where all the iterations inside each parallel region can be executed in parallel by using variable renaming after copying.*

***Keywords****: Parallelizing Compiler, Non-perfected Loops, Non-uniform Dependences, Region Partition.*

## 1. INTRODUCTION

[1]A lot of work has been done in parallelizing loops with uniform dependences, from dependence analysis to loop transformation. Many techniques have been proposed for parallelizing these kinds of loops, such as loop interchange, loop permutation, skew, reversal, wavefront, tiling, etc [3,4,5,6].

Example :

```
      do I = l1, u1
S1 :     A(2i + 4, i + 5) = . . .
         do J = l2, u2
S2 :              . . .=A(i + 2j + 2, i + j)
         enddo
      enddo
```

According to an empirical study[7], nearly 45% of two dimensional array references are coupled and most of these lead to non-uniform dependences. This paper focuses on parallelizing non-perfect nested loops with non-uniform dependences. Our approach is based on the Convex Hull theory which has been proved [2] to have enough information to handle non-uniform dependences. Based on Unique set theory [1], we will divide the iteration space into several parallel regions where all the iterations inside each parallel region can be executed in parallel. Our technique subsumes the above techniques. And for maximize parallelism, we minimize the size of the parallel region by variable renaming.

The rest of this paper is organized as follows. Section two describes our program model, reviews some fundamental concepts and introduces the concept of Complete Dependence Convex Hull. Section three presents three cases starting from simpler cases and leading upto the more complicated ones of doubly nested non-perfect loops with non-uniform dependences. Section four extends this technique to the general program model, and shows the enhanced region partitioning method which divides the iteration space into minimum parallel regions by using variable renaming. Finally, we conclude in section five.

*Corresponding author. E-mail : sjjeong@cheonan.ac.kr
*Manuscript received Jan 26, 2005; accepted Feb 17, 2005*

## 2. PROGRAM MODEL AND DEPENDENCE ANALYSIS

We consider doubly nested loop program of the form shown in Fig. 1. For the given loop, $l_1$ ($l_2$) and $u_1$ ($u_2$) indicate the lower and upper bounds respectively, and should be known at compile time. We also assume that the program statements inside these nested loops are simple assignment statements of arrays. The dimensionality of these arrays is assumed to be equal to the nested loop depth. To characterize the coupled array subscripts, the array subscripts, $f_1(I, J)$, $f_3(I, J)$, and $f_4(I, J)$, are linear functions of the loop index variables.

```
      do I = l1, u1
S1 :     A(f1(I, J), f2(I, J)) = . . .
         do J = l2, u2
S2 :              . . .  =A(f3(I, J), f4(I, J))
         enddo
      enddo
```

Fig. 1. Doubly nested Non-perfect Loop Program Model

The most common method to compute data dependences involves solving a set of linear diophantine equations with a set of constraints formed by the iteration boundaries.

The loop in Fig. 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 (2.1) and the system of inequalities given by (2.2). The general solution to these equations can be computed by the extended GCD [10] or the power test algorithm [11] and forms a DCH.

$$f_1(i_1, j_1) = f_3(i_2, j_2) \text{ and } f_2(i_1, j_1) = f_4(i_2, j_2) \quad (2.1)$$
$$l_1 \leq i_1, i_2 \leq u_1 \text{ and } j_1 = l_2 - 1, l_2 \leq j_2 \leq u_2 \quad (2.2)$$

There are two approaches to solve the system of Diophantine equations of (2.1). One way is to set $i_1$ to $x_1$ and $j_1$ to $y_1$ and get the solution of $i_2$ and $j_2$.

$$a_{21}i_2 + b_{21}j_2 + c_{21} = a_{11}x_1 + c_{11}$$
$$a_{22}i_2 + b_{22}j_2 + c_{22} = a_{12}x_1 + c_{12}$$

We have the solution as

$$i_2 = \alpha_{11}x_1 + \gamma_{11}$$
$$j_2 = \alpha_{12}x_1 + \gamma_{12}$$

where

$$\alpha_{11} = (a_{11}b_{22} - a_{12}b_{21})/(a_{21}b_{22} - a_{22}b_{21})$$
$$\gamma_{11} = (b_{22}c_{11} + b_{21}c_{22} - b_{22}c_{21} - b_{21}c_{12})/(a_{21}b_{22} - a_{22}b_{21})$$
$$\alpha_{12} = (a_{21}a_{12} - a_{11}a_{22})/(a_{21}b_{22} - a_{22}b_{21})$$
$$\gamma_{12} = (a_{21}c_{12} + a_{22}c_{21} - a_{21}c_{22} - a_{22}c_{11})/(a_{21}b_{22} - a_{22}b_{21})$$

The solution space is the set of points $(x, y)$ satisfying the equations given above. The set of inequalities can be written as

$$\begin{aligned} l_1 \leq \quad x_1 \quad &\leq u_1 \quad (2.3) \\ y_1 &= l_2 - 1 \\ l_1 \leq \alpha_{11}x_1 + \gamma_{11} &\leq u_1 \\ l_2 \leq \alpha_{12}x_1 + \gamma_{12} &\leq u_2 \end{aligned}$$

where Eq. (2.3) defines a DCH denoted by **DCH1**.
Another approach is to set $i_2$ to $x_2$ and $j_2$ to $y_2$ and solve for the solution of $i_1$ and $j_1$.

$$a_{11}i_1 + c_{11} = a_{21}x_2 + b_{21}y_2 + c_{21}$$
$$a_{12}i_1 + c_{12} = a_{22}x_2 + b_{22}y_2 + c_{22}$$

We have the solution as

$$i_1 = \alpha_{21}x_2 + \beta_{21}y_2 + \gamma_{21}$$
$$j_1 = \alpha_{22}x_2 + \beta_{22}y_2 + \gamma_{22}$$

where

$$\alpha_{21} = a_{21}/a_{11}$$
$$\beta_{21} = b_{21}/a_{11}$$
$$\gamma_{21} = (c_{21}-c_{11})/a_{11}$$
$$\alpha_{22} = a_{22}/a_{12}$$
$$\beta_{22} = b_{22}/a_{12}$$
$$\gamma_{22} = (c_{22}-c_{12})/a_{12}$$

The solution space is the set of points $(x, y)$ satisfying the solution given above. In this case the set of inequalities can be written as

$$\begin{aligned} l_1 \leq \alpha_{21}x_2 + \beta_{21}y_2 + \gamma_{21} &\leq u_1 \quad (2.4) \\ l_2 \leq \alpha_{22}x_2 + \beta_{22}y_2 + \gamma_{22} &\leq u_2 \\ l_1 \leq \quad x_2 \quad &\leq u_1 \\ l_2 \leq \quad y_2 \quad &\leq u_2 \end{aligned}$$

where Eq. (2.4) defines another DCH, denoted by **DCH2**. The above two sets of solutions are both valid. Each of them has the dependence information on one extreme. For some simple cases, for instance, since there is only one kind of dependence, either flow or anti dependence, one set of solution (*i.e.* DCH) should be enough. Punyamurtula and Chaudhary used Eq. (2.3) for their technique [8], while Zaafrani and Ito used Eq. (2.4) for their technique [9]. For those more complicated cases, where both flow and anti dependences are involved and dependence patterns are irregular, we need to use both sets of solutions.
If iteration $(i_2, j_2)$ is dependent on iteration $(i_1, j_1)$, then we have a dependence distance vector $d(x, y)$ with

$$d_i(x, y) = i_2 - i_1 \quad (2.5)$$
$$d_j(x, y) = j_2 - j_1$$

So, for DCH1, we have

$$d_i(x_1, y_1) = (\alpha_{11} - 1)x_1 + \gamma_{11} \quad (2.6)$$
$$d_j(x_1, y_1) = \alpha_{12}x_1 + \gamma_{12} - l_2 + 1$$

For DCH2, we have

$$d_i(x_2, y_2) = (1 - \alpha_{21})x_2 - \beta_{21}y_2 - \gamma_{21} \quad (2.7)$$
$$d_i(x_2, y_2) = (1 - \alpha_{22})x_2 - \beta_{22}y_2 - \gamma_{22}$$
$$d_j(x_2, y_2) = y_2 - l_2 + 1$$

Clearly if we have a solution $(x_1, y_1)$ in DCH1, we must have a solution $(x_2, y_2)$ in DCH2, because they have been solved from the same set of linear Diophantine Eq. (2.1). The union of DCH1 and DCH2 is called Complete Dependence Convex Hull (**CDCH**), and all dependences lie within the CDCH. The properties of DCH1 and DCH2 can be found in [1].

## 3. PARALLELIZATION OF NON-PERFECT LOOPS WITH NON-UNIFORM DEPENDENCES

Based on the unique head and tail sets that we can identify, there are at most four sets, i.e., flow dependence unique tail set, flow dependence head set, anti-dependence unique tail set, and anti-dependence unique head set. We categorize these combinations as three cases starting from simpler cases and leading upto the more complicated ones of doubly nested non-perfect loops with non-uniform dependences as follows:

CASE 1. There is only one kind of dependence which is flow dependence

In this case, DCH1 is flow dependence unique tail set and DCH2 is flow dependence unique head set, as shown in Fig. 2. Clearly the flow dependence unique head set should be run after flow dependence unique tail set. Therefore, the doubly nested loop in Fig. 1 can be transformed as follows.

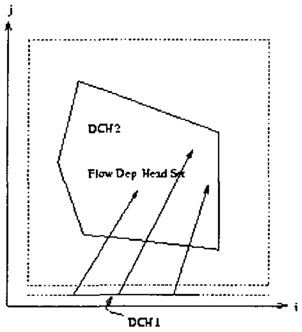


Fig. 2. There are only Flow dependences

```
doall I = l1, u1
S1:     A(a11i1 + c11, a12i1 + c12)) = . . .
    enddo
    doall I = l1, u1
        doall J = l2, u2
S2:             . . . = A(a21i2+b21j2+c21, a22i2+b22j2+c22)
        enddo
    enddo
```

CASE 2. there is only one kind of dependence which is anti dependence

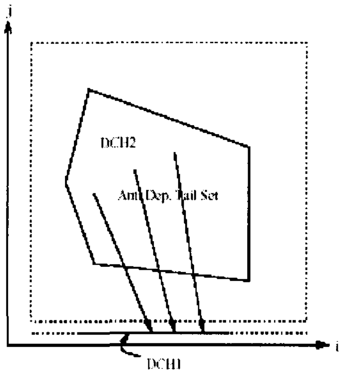


Fig. 3. There are only Anti dependences

In this case, DCH1 is anti dependence unique head set and DCH2 is anti dependence unique tail set, as shown in Fig. 3. Clearly the anti dependence unique head set should be run after anti dependence unique tail set. Therefore, the doubly nested loop in Fig. 1 can be transformed as follows.

```
      doall I = l1, u1
          doall J = l2, u2
S1:           ... = A(a21i2+b21j2+c21, a22i2+b22j2+c22)
          enddo
      enddo
      doall I = l1, u1
S2:           A(a11i1 + c11, a12i1 + c12)) = . . .
      enddo
```

CASE 3. there are both flow and anti dependences

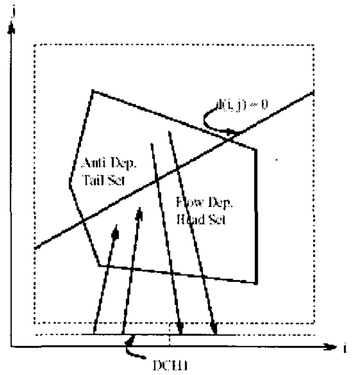


Fig. 4. There are both Flow and Anti dependences

In this case, we can divide the DCH inside the iteration space (in this program model, it is DCH2.) and run in the following order (*anti dependence unique tail set*) → DCH1 → (*flow dependence unique head set*). $d_i(x_2, y_2) = 0$, which is $(1 - \alpha_{21})x_2 - \beta_{21}y_2 - \gamma_{21}$ , will be used as the dividing line.
The converted parallel code of doubly nested loop in Fig. 1 is as follows:

```
     doall I = l1, u1
         doall J = l2, u2
             if ((1 - α21)x2 - β21y2 - γ21) < 0
S2:                . . .= A(a21i2 + b21j2 + c21, a22i2 + b22j2 + c22)
             endif
         enddo
     enddo
     doall I = l1, u1
S1:      A(a11i1 + c11, a12i1 + c12)) = . . .
     enddo
     doall I = l1, u1
         doall J = l2, u2
             if ((1 - α21)x2 - β21y2 - γ21) > 0
S2:                . . .= A(a21i2 + b21j2 + c21, a22i2 + b22j2 + c22)
             endif
         enddo
     enddo
```

But, By variable renaming after copying, execution order is DCH1 → (*flow dependence unique head set*) because anti dependence unique sets can be fully executed concurrently. So, in this case, CASE 3 is changed to CASE 1.

### 4. ENHANCED REGION PARTITIONING METHOD

The system of Diophantine equations and system of linear inequalities can be derived as

$$2i_1 + 4 = j_1 + 2j_2 + 2$$

$$i_1 + 5 = j_1$$

$$l_1 \le i_1 \le u_1,\ i_2 = l_1 - 1$$
$$l_2 \le j_1 \le u_2,\ l_2 \le j_2 \le u_2$$

Now we solve for $j_1$ and $j_2$ by setting $i_1 = x$ and $i_2 = y$.

$$j_1 + 2j_2 + 2 = 2x + 4$$
$$j_1 + j_2 = x + 5$$

$$j_1 = 8,\ j_2 = x - 3$$

So we get DCH1 which is

$$l_1 \le x \le u,\ _1y = l_1 - 1$$
$$l_2 \le 8 \le u_2,\ l_2 \le x - 3 \le u_2$$

The dependence vector is

$$d_i(x, y) = j_2 - i_1 = 8 - x$$
$$d_j(x, y) = j_2 - i_2 = x - 3$$

For DCH2, set $j_1 = x$ and $j_2 = x$.

$$2i_1 + 4 = x + 2y + 2$$
$$i_1 + 5 = x + y$$

$$i_1 = 1/2x + y - 1$$
$$i_1 = x + 2y - 5$$

DCH2 is

$$l_1 \le 1/2x + y - 1 \le u_1$$
$$l_1 \le x + y - 5 \le u_1$$
$$l_2 \le x \le u_2$$
$$l_2 \le y \le u_2$$

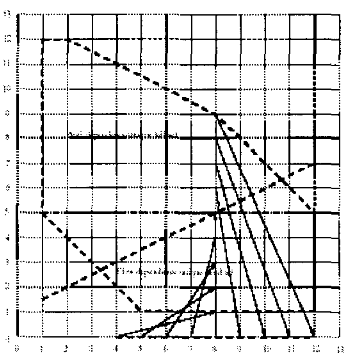

Fig. 5. Example of unique sets in iteration space

Fig. 5 shows the unique sets in the iteration space when $l_1 = 1$, $u_1 = 12$, $l_2 = 1$, and $u_2 = 12$.
The parallelized code is as following:

```
      doall I = l1, u1
         doall J = max(l2, 6), u2
S2:               . . .=A(i + 2j + 2, i + j)
         enddo
      enddo
```

```
      doall I = l1, u1
S1:      A(2i + 4, i + 5) = . . .
      enddo
      doall I = l1, u1
         doall J = l2,max(u2, 5)
S2:         . . .=A(a21i2 + b21j2 + c21, a22i2 + b22j2 + c22)
         enddo
      enddo
```

The parallelized code after variable renaming is as following:

```
      doall I = l1, u1
S1:      A(2i + 4, i + 5) = . . .
      enddo
      doall I = l1, u1
         doall J = l2,max(u2, 5)
S2:         . . .=A(a21i2 + b21j2 + c21, a22i2 + b22j2 + c22)
         enddo
      enddo
```

## 5. CONCLUSIONS

This paper introduced unique sets oriented technique to parallelize non-perfect nested loops with non-uniform dependences. Current parallelizing compilers can't handle this kind of loop, either leaving it to run sequentially, or giving poor performance even when parallelized. The research about non-uniform dependence has been restricted and most of these techniques assume a perfect nested loop model. This paper used the concept of Complete Dependence Convex Hull, Unique Head and Tail Sets. This paper addressed the issues of how to analyze non-perfect nested loops and how to partition the iteration space according to the information collected using unique sets. And we proposed an enhanced region partitioning method which divides the iteration space into minimum parallel regions by using variable renaming. Our future research work is to improve the other techniques for higher dimensional nested loops and/or loops with multiple dependences.

## REFERENCES

[1] J. Ju and V. Chaudhary, "Unique sets oriented partitioning of nested loops with non-uniform dependences," in **Proceedings of the International Conference on Parallel Processing**, pp. III45-III52, 1996.

[2] T. H. Tzen and L. M. Ni, "Dependence uniformization: A loop parallelization tehnique," **IEEE transactions on Parallel and Distributed Systems**, vol. 4, pp. 547-558, May 1993.

[3] M. Wolfe, "Loop skewing: The wavefront method revisited," **International Journal of Parallel Programming**, pp. 279-293, Aug. 1986.

[4] U. Banerjee, **Loop Transformations for Restructuring compilers.** Kluwer Academic Publishers, 1993.

[5] M. E. Wolfe and M. S. Lam, "A loop transformation theory and an algorithm to maximize parallelism," **IEEE transactions on Parallel and Distributed Systems**, vol. 2, pp. 452-471, Oct 1991.
[6] M. J. Wolfe, **Optimizing Supercompilers for supercomputers.** MIT Press, 1989.
[7] Z. Shen, Z. Li, and P. C. Yew, "An empirical study on array subscripts and data dependencies," in **Proceedings of the International Conference on Parallel Processing**, pp. II145-II152, 1989.
[8] 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.
[9] A. Zaafrani and M. Ito, "Parallel region execution of loops with irregular dependences," in **Proceedings of the International Conference on Parallel Processing**, pp. II--11 to II--19, 1994.
[10] U. Banerjee, **Dependence Analysis for Supercomputing.** Kluwer Academic Publishers, 1988.
[11] M. Wolfe and C. W. Tseng, "The power test for data dependence," **IEEE Trans.** ***Parallel and Distributed Systems***, vol. 3, no. 5, pp. 591-601, Sep. 1992.

**Sam-Jin Jeong**
He received the B.S. in polymer science from KyungBuk National university, Korea in 1979, and the M.S. in computer science from Indiana university, USA in 1987, and also received Ph.D. in computer science from ChungNam National university, Korea in 2000. From 1988 to 1991, he was a senior research staff at SamSung Electric Co. From 1992 to 1997, he was an assistant professor at Haecheon University. Since then, he has been with Cheonan University as a professor. His main research interests include parallelizing compiler, parallel systems, general compiler, and programming languages.