Generalized Unimodular Loop Transformations for Distributed Memory Multiprocessors

K G Kumar*, D Kulkarni, A Basu
Center for Development of Advanced Computing
2/1 Brunton Road, Bangalore 560 025, India

Abstract

In this paper, we present a generalized unimodular loop transformation as a simple, systematic and elegant method for partitioning the iteration spaces of nested loops for execution on distributed memory multiprocessors. We present a methodology for deriving the transformations that internalize multiple dependences in a multidimensional iteration space without resulting in a deadlocking situation. We then derive the general expression for the bounds of the transformed loops in terms of the bounds of the original space and the transformation matrix elements.

Keywords: Parallelizing Compilers, Restructuring Transformations, Loop Partitioning, Iteration Spaces, Dependence Vectors.

1 Introduction

The great difficulty of programming distributed memory multiprocessors has motivated a spate of research activity in the automatic source level restructuring of sequential programs for execution on such machines. Many among these efforts have addressed the problem of partitioning loops since, nested loops form the core of most compute intensive programs. Efficient execution of these programs on distributed memory systems demands a methodology for partitioning the iteration spaces of these nested loops, so that different partitions can be mapped to different processors.

Schemes such as [4, 9, 12] look for independent iteration partitions, such that iterations within a partition are dependent whereas those belonging to different partitions are totally independent. These schemes however have limited applicability because in many cases even simple combinations of dependences cause strict sequentialization and the schemes result in very little parallelism.

It is our view point, especially in the context of distributed memory multiprocessors, that a lot of potential parallelism would be lost if one insists on total independence of partitions instead of looking for minimal dependence. Thus we view the problem of loop partitioning as one of partitioning the iteration space in a way that all the iterations in a partition are dependent and there is *minimal* data dependence among different partitions. Further, unlike techniques in some earlier work [10] that are based on the *hyperplane method* [8], and which are more appropriate for inner loop parallelization, we regard coarse grain or outer loop parallelization as a more natural choice for distributed memory multiprocessors.

*kumar@cdacb.ernet.in

Unimodular transformations introduced in [3] are an innovative and unified way of viewing a host of different loop transformations. It is shown in [3] that the general hyperplane method, loop reversal, and loop interchange can be viewed in the uniform frame work of unimodular transformations. The optimism has it that all possible loop transformations can be formulated as unimodular transformations [3].

In our earlier work [6, 7] we have presented the problem of partitioning the iteration space of a nested double loop for parallel execution on a distributed memory machine as one of finding a unimodular transformation called *rotation* which changes a nested loop into a new loop with outer loop parallelization. Rotation is simple, elegant and systematic. The points in the iteration space can correspond to individual iterations or groups of iterations, as long as the grouping is a valid one. We introduced the notion of a *weight* associated with each dependence vector to characterize the net *volume of communication* in the iteration space. The rotation of interest is the best among possible unimodular transformations, evaluated on the basis of three parameters, namely - the *parallelism factor*, the *load imbalance*, and the *volume of communication* due to dependence vectors across the boundaries of iteration space partitions. We also provided a combined metric that helps choose the best direction among the available dependence distance vectors.

In this paper, we present a generalization of the results of [6, 7] for arbitrary loop nest levels and to internalize several dependences. We present a systematic method for obtaining the unimodular matrix defining the transformation which is also guaranteed not to give rise to a deadlocking situation. We then derive the generalized expressions for the bounds of the transformed loops for arbitrary iteration space dimensions.

The rest of the paper is organized as follows. Section 2 introduces the problem of iteration space partitioning and summarizes the results in [6, 7]. Section 3 presents our methodology of obtaining a generalized transformation for n-dimensional iteration spaces and for internalizing several dependences. Section 4 presents the derivation of the general expressions for loop bounds of the transformed space. Finally, Section 5 contains some concluding remarks and directions for future work.

2 Background

In this section we briefly describe the problem of iteration space partitioning and summarize related work by other researchers. We then provide a brief introduction to our approach to the problem in the context of perfectly nested double loops. The details can be found in [6, 7].

2.1 Iteration Space Partitioning

We consider only perfectly nested loops with constant loop limits and arbitrary nesting levels.

The set of iterations defined by a nested loop of level n can be denoted as a subset of $\mathbf{R}^n$, the set of all real n-vectors. Rectangular, triangular and trapezoidal shaped spaces are routinely found in scientific programs, though the rectangular shape is more frequent and more regular.

Definition 1 (Rectangular Space) $\Re \subset \mathbf{R}^n$ *such that*

$$\Re = \{(i_1, ..., i_n) \mid m_1 \leq i_1 \leq M_1, ..., m_n \leq i_n \leq M_n\},$$

where i_1, ..., i_n *are the iteration indices, and* $(m_1, M_1), ..., (m_n, M_n)$ *are the respective constant loop limits, is a rectangular iteration space.* □

$$
\begin{array}{ll}
L_1: & \textbf{do } I_1 = m_1, M_1, 1 \\
L_2: & \quad \textbf{do } I_2 = m_2, M_2, 1 \\
 & \qquad H(I_1, I_2) \\
 & \quad \textbf{enddo} \\
 & \textbf{enddo}
\end{array}
$$

Figure 1: Typical double loop

$$
\begin{array}{ll}
L_1: & \textbf{do } I_1 = m_1, M_1, 1 \\
L_2: & \quad \textbf{do } I_2 = m_2, M_2, 1 \\
 & \qquad A(I_1, I_2) = A(I_1 - 2, I_2 - 1) + A(I_1 - 3, I_2 + 2) \\
 & \quad \textbf{enddo} \\
 & \textbf{enddo}
\end{array}
$$

Figure 2: Example dependence vectors

The iterations are denoted by n-vectors in $\Re$. The dependence between two iterations, $\mathbf{i} = (i_1, ..., i_n)$ and $\mathbf{j} = (j_1, ..., j_n)$, can be represented by the *dependence distance vector* $d = (i_1 - i_1, ..., j_n - i_n)$, and the corresponding direction vector is denoted as $(sign(j_1 - i_1), ..., sign(j_n - i_n))$. A dependence distance vector is positive if corresponding direction vector has a nonzero leading element as $+1$ [1]. A distance vector is negative otherwise.

For example, in Figure 2 the distance vectors are (2,1) and (3,-2), and the corresponding direction vectors (1,1) and (1,-1). Both the vectors are positive.

The problem of partitioning the iteration space of any set of n-nested loop is one of splitting up the collection of iterations in the space among mutually exclusive sets constituting the various partitions. The fundamental objective of partitioning would be to either

- Maximize the size of each partition which are such that no two elements of the same partition are dependent,

  OR

- Maximize the number partitions which are such that elements of different partitions are not dependent.

The *hyperplane* or *wavefront* method suggested by Lamport [8] for parallelizing **do** loops is the most prominent among the ones which achieve the former objective. The attempt here is to find the most optimum family of planes of parallelism in the iteration space from the point of view of maximizing the number of iteration points on each plane and minimizing the interplane dependences. The resulting transformation is essentially inner loop parallelization with the planes, corresponding to each of the outer loop instances, executing strictly sequentially. This however is not particularly suited for distributed memory multiprocessors.

The wavefront method has served as the basis for several later work. A recent work [10] that is aimed at distributed memory systems attempts to partition the nested loops using both iteration and data space graphs. The method, however is limited to the cases

$$
\begin{array}{ll}
L_1^U: & \textbf{do } K_1 = m_1', M_1', 1 \\
L_2^U: & \quad \textbf{do } K_2 = m_2'(K_1), M_2'(K_1), 1 \\
 & \qquad H(K_1, K_2) \\
 & \quad \textbf{enddo} \\
 & \textbf{enddo}
\end{array}
$$

Figure 3: Transformed double loop

where a single subscripted variable causes the dependences. It is not clear how one can handle the situations that involve many subscripted variables in the dependences as is usually the case. It is our view that flow dependence taken abstractly, without reference to the individual variables involved, should form the basis for partitioning.

Examples of those that achieve the latter objective are [9, 12, 4]. In [9] and [12] the maximum cardinality of partitions of the iteration spaces into independent groups is estimated. They however do not provide a method of identifying the elements of the groups. In [4] the above two methods are extended. It also provides a polynomial time algorithm to compute the elements in each partition. Since these methods look for complete independence of the partitions they result in very little parallelism.

A related area, which has received considerable attention lately, is that of grouping together iterations to obtain reduced iteration spaces. These techniques, which have been termed tiling [13, 11] or grouping [5] serve the purpose of combining the communications associated with each group in order to save on the relatively high setup time for individual communications. They also result in smaller and less complex iteration space for further partitioning. In [13], tiling is achieved by multidimensional strip mining and in [11] by identifying suitable tiling hyperplanes. Both works describe the choice of optimum shapes and sizes of the tiles and conditions for deadlock free tiling. [5] investigates the formation of groups of fixed size along the direction of any dependence vector. The investigation however concentrates on guaranteeing absence of deadlock and does not address the issues of optimizing directions of grouping or group sizes.

2.2 The Unimodular Transformation *Rotation*

Our objective for partitioning is to maximize the number partitions while aiming for *minimal* interpartition dependences as against *no* interpartition dependences. We achieve the desired partitioning through unimodular transformations.

The concept of unimodular transformations, introduced by Banerjee [3], provides an elegant generalization of various transformations of a nested loop. It is a new perspective of the whole list of transformations: loop interchange, loop reversal, loop skewing and inner/outer loop parallelization. The task is to find a unimodular matrix U ($\mid det(U) \mid = 1$) that characterizes the form of the loop after transformations. The new loop indices in the transformed loop are obtained in terms of the old loop indices by a matrix multiplication. Here we restrict ourselves to doubly nested loops. For example, we transform a double loop (L_1, L_2) (Figure 1) into a new double loop (L_1^U, L_2^U) through a transformation defined by the unimodular matrix U (Figure 3). The new iteration space looks as in Figure 4.

Intuitively, the first column of U gives the orientation of the wavefront, and the second column makes U unimodular. The transformation $rotation(d)$ is a unimodular transformation that parallelizes the outer loop such that the distance vector d is internalized to

Figure 4: Transformed iteration space

the inner loop. Let $d = (a, b)$, and

$$U = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix}$$

be the unimodular matrix that characterizes the transformation. Let the $d' = (a', b')$ be the transformed d.

$$d' = d.U = (au_{11} + bu_{21}, au_{12} + bu_{22})$$

The requirement that there be no dependence at level one in the transformed loop implies that $a' = 0$. That is, $u_{11} = -b$ and $u_{21} = a$. And u_{12}, u_{22} any pair of integers such that $au_{12} + bu_{22} \geq 0$ and U is unimodular. In the shared memory case, according to the *principle of program transformations* [3], the transformation is valid iff each distance vector $d'' = (a'', b'')$ remains positive after the transformation. Analytically, $a''u_{11} + b''u_{21} > 0$, or $a''u_{11} + b''u_{21} = 0$ and $a''u_{12} + b''u_{22} > 0$. However, it is not necessary to ensure that the resulting direction vectors are all positive as we are essentially attempting outer loop parallelization for execution on distributed memory machines. Hence the matrix U always exists.

The choice of distance vector d depends on the parallelism that *rotation*(d) produces. The amount of parallelism is intuitively equal to the size of the outer loop after transformation. The choice of d depends also on the extent to which the load is balanced. Intuitively this is the variation in the size of the inner loop for different values of the outer loop index. The choice also depends on the communication overhead that is necessitated by the distance vectors across the partitions in the transformed loop. A good choice for d is the one that is appropriate from above three criteria. These parameters are termed the *parallelism factor*, the *load imbalance*, and the *volume of communication* respectively. We obtain the parallelism factor **P** for a given d as,

$$P = \mid b \mid (M_1 - m_1) + \mid a \mid (M_2 - m_2) + 1 - max(\mid a \mid -1, 0) * max(\mid b \mid -1, 0) \qquad (1)$$

where (m_1, M_1) and (m_2, M_2) are the bounds of the two original loops.

The measure of how well the load is balanced in the transformed loop corresponds to how little the inner loop size varies from one instance of outer loop to another. The load imbalance V is the variance of the inner loop size for different values of outer loop index and is obtained as,

$$V = \left(\sum_{i=1}^{\alpha} i^2 + \alpha^2 (A/h - 1)(\alpha + P(A/h - 1)/2) - A\alpha^2/h \right) 2h^2/P\alpha^2 \tag{2}$$

where

$$h = min \left(\lfloor \frac{M_1 - m_1}{a} \rfloor, \lfloor \frac{M_2 - m_2}{b} \rfloor \right) + 1$$

$$\alpha = min\,(\mid b \mid (M_1 - m_1), \mid a \mid (M_2 - m_2)) - max(\mid a \mid -1, 0) * max(\mid b \mid -1, 0)/2.$$

and

$$A = \frac{(M_1 - m_1 + 1)(M_2 - m_2 + 1)}{P}$$

When a rotation is performed, the dependences that are not internalized to a partition are reflected as communications between partitions. It is not adequate to consider only the number of such communications. Instead we need a measure of volume of communication in the iteration space of the transformed loop. We introduced the notion *weight* with each dependence vector in terms of which we measured the volume of communication. Then the net inter-partition communication volume in the transformed is,

$$C(d_i) = \sum_{i=1}^{k} N(d_i) w_i - N(d_i) w_i. \tag{3}$$

where the first term on the right hand side represents the total volume of communication and the second term represents the volume of internalized communications. We can form a metric that combines the parameters $P, V and C$ in an architecture dependent way [6, 7].

3 Extension to N–D Iteration Spaces

Discussion in the previous section was restricted to a doubly nested loop and involved the internalization of exactly one dependence vector.In this section we extend the framework introduced in [6, 7] to N-dimensional (N–D) iteration spaces and for internalizing more than one dependence vector.

Definition 2 *We say that an N–D iteration space $\Re$ internalizes a dependence vector d at level $r \leq N$, if the outermost $N - r$ loops of $\Re$ are independent of each other with respect to d.*

Internalization of vector d at level $r \leq N$ in other words means that d is confined to the innermost r loops.

Problem: Given an N–D iteration space $\Re$ with m dependences $D = \{d_1, d_2, ..., d_m\}$, the problem here is to find a unimodular matrix U to transform $\Re$ to $\Re'$ which would internalize all elements of a subset of the dependences, say $D_s = \{d_1, d_2, ..., d_r\}$ at a level $k \leq N$.

This would be tantamount to saying that the $N - k$ outer loops of $\Re'$ are independent as far as the subset of vectors D_s are concerned. Unless $N - k$ is non-zero, we have no parallelism. Thus only when $k < N$ do we have a meaningful internalization.

Theorem 1 *Given an N–D iteration space $\Re$ and a dependence vector d in $\Re$, there exists a unimodular matrix U which transforms $\Re$ into $\Re'$ such that d is internalized at level 1 in $\Re'$.*

Proof: The proof is constructive and is based on Algorithm 5.4.1 suggested in [3]. For any dependence vector d of length N, the algorithm yields a N by N unimodular matrix U such that $U.d = (g, 0, 0, ..., 0)$. With a slight modification of the algorithm, we could obtain the N by N unimodular matrix U such that $d.U = (0, 0, ..., 0, g)$. Clearly, the matrix U transforms any $\Re$ into an $\Re'$ which internalizes d at level 1. The transformed algorithm, called A1 is presented below.

Algorithm A1:

Step 1: We form a $N+1$ by N matrix U_0 by adding the elements of vector d as a new row above the rows of an N by N identity matrix. Set $i = 1$.

Step 2: If $i = N$, go to step 6.

Step 3: If $U_0[1, i] = 0$, then set i to $i+1$ and go to step 2.

Step 4: If $U_0[1, i] \geq U_0[1, i+1]$ then set $q = \lfloor (|U_0[1, i+1]| / |U_0[1, i]|) \rfloor$; set $s = sign(U_0[1, i+1]/U_0[1, i])$; and subtract $s * q$ times column i from column $i+1$.

Step 5: Interchange columns i and $i+1$. Go to step 2.

Step 6: Now each of $U_0[1, 1]$ to $U_0[1, N-1]$ is zero. If $U_0[1, N]$ is negative, multiply column N of U_0 by -1. Stop.

The matrix U resulting from the removal of the first row of U_0 at this stage is the desired unimodular matrix.

□

Example 1: Consider a dependence vector (3,1,0) in a 3–D iteration space. The unimodular matrix U to internalize this vector at level 1 may be obtained using the above algorithm to be,

$$U = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -3 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Vector (3,1,0) gets transformed into (0,0,1) in the new iteration space.

The above theorem provides us a transformation for internalizing any single dependence at level 1. We are however interested in finding a transformation for internalizing more than one dependence vector. The following theorem provides an important step in obtaining such a generalized transformation.

Theorem 2 *Given a unimodular matrix U which internalizes dependences $d_1, d_2, ..., d_{r-1}$ in an N–D iteration space at level $k-1$, $k \leq N$, we can find a unimodular matrix U' from U such that U' internalizes $d_1, d_2, ..., d_r$ at level k.*

Proof: The proof here again is constructive. Let U be a unimodular matrix such that $d_i.U = (0, 0, ..., 0, g_1^i, g_2^i, ..., g_{k-1}^i)$ for i varying from 1 to $r-1$. We now construct the required matrix U' using Algorithm A2 below.

Algorithm A2: We first form a $N+1$ by $N-r+1$ matrix V by adding an element each to the top of the first $N-r+1$ columns of U. This top element c_i added to any column

i would be the dot product of d_r and column i of U. Using Algorithm A1, we may now obtain a transformed matrix V' whose first row is of the form $(0, 0, ...0, g)$, where g is the g.c.d of $\{c_1, c_2, ..., c_{N-r+1}\}$. Let V'' be the N by $N-r+1$ matrix formed by deleting the first row of V'. Now we take U' to be the N by N matrix formed by appending $(N-r+2)$th to the Nth columns of U to V''.

Clearly, $d_r.U'$ is of the form $(0, 0, ...0, g_1^r, g_2^r, ..., g_k^r)$ implying that d_r is internalized at level k. Also, the products $d_i.U$, for i varying between 1 and $r-1$, are invariant after the transformation. This is so since the dot product of any of these vectors with the first $N-r+1$ columns of U are all zeros. Any linear combination of these $N-r+1$ columns would therefore still yield a zero dot product with these vectors, and the steps above involve only such linear combinations of the first $N-r+1$ columns. □

We now provide the generalization of Theorem 1 concerning the internalization of an arbitrary number, say r, $r \leq N$, of dependence vectors.

Theorem 3 *Given any N–D iteration space $\Re$ and a set of dependence vectors $\{d_1, d_2, ..., d_r\}$ such that $r \leq N$, there exists a unimodular matrix U which transforms $\Re$ into $\Re'$ such that all these r vectors are internalized at level r in $\Re'$.*

Proof: The proof is by induction.

Base Step: There exists a unimodular matrix U such that dependence vector d_1 is internalized at level 1.

Induction Step: Given a unimodular matrix U such that $d_1, d_2, ..., d_{i-1}$ are internalized at level $k-1$, a unimodular matrix U' may be derived which internalizes $d_1, d_2, ..., d_i$ at level k.

These steps are clearly validated by Theorems 1 and 2 respectively. Thus the required unimodular matrix not only exists but can also be constructed using the above steps.
□

Example 2: If we choose to internalize the vector (0,1,2) following the internalization of (3,1,0) in example 1, then the unimodular matrix U' for this may be obtained from the above methodology as the following

$$\begin{bmatrix} 2 & -1 & 0 \\ -6 & 3 & 1 \\ 3 & -1 & 0 \end{bmatrix}$$

We note that the product of the vector (3,1,0) with U' remains the same as its product with U.

Theorem 4 *A unimodular matrix U which internalizes r dependences in an N–D iteration space transforms all the internalized dependence vectors to positive dependence vectors.*

Proof: We prove this by induction over the steps involved in constructing U. Let $\{d_1, d_2, ..., d_r\}$ be the set of dependence vectors we are interested in internalizing. The first step in this process involves the application of Algorithm A1 to obtain a transformation that internalizes d_1 at level 1. All subsequent steps involve the application of Algorithm A2 to systematically internalize $d_2, d_3, ..., d_r$ at levels 2, 3,..., r respectively.

The basis of our inductive proof is to establish that the first step results in only transformed vectors that are positive. If U_1 is the transformation matrix obtained after internalizing d_1, then we have seen that $d_1.U_1 = (0, 0, ..., g)$, where g is the g.c.d of the elements of d_1. Therefore g is positive, and so all dependence vectors that are internalized at level 1 are transformed to positive vectors.

Let us assume U_i is a transformation matrix that internalizes $d_1, d_2, ..., d_{i-1}$ and also transforms these dependence vectors to positive vectors. The inductive step of our proof involves proving that a transformation matrix U_i obtained from the transformation matrix U_{i-1} using Algorithm A2, transforms $d_1, d_2 ..., d_i$ to positive vectors.

By virtue of the method of Algorithm A1 adopted in deriving the first $N-i+1$ columns of U_i, the dot product of d_i with each of the columns 1 to $N-i$ would be zero. Also, its dot product with column $N-i+1$ would be the g.c.d of $\{c_1, c_2, ..., c_{N-i+1}\}$, the set of dot products of d_i with columns 1 to $N-i+1$ respectively of $U_i - 1$.

We note that Algorithm A2 works with columns 1 to $N-i+1$ of U_{i-1} to construct U_i. The columns $N-i+2$ to N of U_i are in other words identical to the columns $N-i+2$ to N of U_{i-1}. We also note that the dot product of $d_1, d_2, ..., d_{i-1}$ with columns 1 to $N-i+1$ of U_i are still zero. This is so since these columns have been obtained through column operations using columns 1 to $N-i+1$ only of U_{i-1} and the dot product of these with $d_1, d_2, ..., d_{i-1}$ was originally all zero. Thus, the transformations U_i and U_{i-1} have identical effects on the vectors $d_1, d_2, ..., d_{i-1}$. The inductive step is therefore valid.
□

Theorem 5 *Given an N-D iteration space and r dependence vectors the internalization of all r vectors at some level $k \leq N$ requires that all r vectors be confined to a single k-dimensional hyperplane.*

Proof: Let U be a unimodular matrix such that all r dependence vectors are internalized at level $k \leq N$. This implies that $d_i.\overline{u}_j = 0$ for all $1 \leq i \leq r$ and $1 \leq j \leq N-k$. In other words, $d_1, ..., d_r$ should all lie in the hyperplane which is orthogonal to all $\overline{u}_1, ..., \overline{u}_{N-k}$. That is to say that they should lie on a single hyperplane of dimension k.
□

Corollary 1 *Internalization of three dependence vectors at level three would require that they all belong to a 3-D space; at level two would require that they all lie on the same plane; and at level one would require that they lie along the same line.*

From Theorem 5 and Corollary 1 above it is clear that in the general case a minimum of r levels are required to internalize r dependence vectors with no constraints on the vectors regarding relative orientations. Theorem 3 provides us a methodology of obtaining a unimodular matrix U which internalizes r arbitrary dependence vectors in exactly r levels.

It is essential to ensure that the transformation does not give rise to a deadlock. A deadlocking situation is one wherein it would be possible to find some combinations of the transformed vectors which demand an order of execution of the iterations that are in conflict with the order of sequentialization. Formally, we define a deadlocking situation as follows.

Definition 3 *Given a set of dependence vectors* $\{d_1, d_2, ..., d_m\}$ *in an N–D iteration space, a transformation which internalizes any r of them at level k leads to a deadlocking situation iff there exist a set of non-negative integers* $k_1, k_2, ..., k_m$*, not all zeros, such that*

$$k_1.d'_1 + k_2.d'_2 + ... + k_m.d'_m = (0, 0, ...0, s_1, s_2, ..., s_k)$$

where $d'_1, d'_2, ..., d'_m$ *are the results of transforming* $d_1, d_2, ..., d_m$ *respectively, and* $s_1, s_2, ..., s_k$ *are either all zeros or the first non-zero element among them is negative.*

Corollary 2 *Given a set of dependence vectors* $d_1, d_2, ..., d_m$*, a transformation that transforms all of them into non-negative dependence vectors does not lead to a deadlocking situation.*

Proof: This follows from the fact that a linear combination of positive vectors $d'_{r+1}, d'_{r+2}, ..., d'_m$ involving non-negative coefficients $k_1, k_2, ..., k_{m-r}$ cannot be zero or negative when the coefficients are not all zeros. □

Theorem 6 *Given a set of dependences* $\{d_1, d_2, ..., d_m\}$ *in an N–D iteration space, a transformation which internalizes any r of them and which is non-deadlocking can certainly be found if* $m \leq N$.

Proof: We have shown that the methodology of Theorem 3 can be used to internalize a maximum of N dependence vectors in an N–D iteration space. From Theorem 4 we also know that the transformations derived through our methodology always transform the internalized dependence vectors to positive vectors. Thus, we can certainly find a unimodular transformation U which will transform all the m vectors to positive vectors when $m \leq N$. Clearly, we note that U is one among the transformations which internalize r dependences. Further, U has the property that the non-internalized dependence vectors $d_{r+1}, d_{r+2}, ..., d_m$ are all transformed to positive vectors by it. Therefore by Corollary 2 above, U is non-deadlocking and is the desired transformation.
□

When we have m dependence vectors in a N–D iteration space where $m > N$, a transformation U which internalizes $r \leq N$ of the vectors could in general lead to a deadlocking situation. If it could however be guaranteed that all the transformed vectors would be positive, then it is equivalent to guaranteeing the absence of a deadlocking situation. When the number of dependence vectors m is greater than the iteration space dimension N, the following theorem shows that we can choose N out of the m vectors to obtain a unimodular transformation U that non-deadlocking.

Theorem 7 *Given* $m > N$ *positive dependence vectors* $D = \{d_1, d_2, ..., d_m\}$*, in a N–D iteration space where* $m > N$*, there exists a choice,* $\{d_{a_1}, d_{a_2}, ..., d_{a_N}\}$ *of N vectors from D such that, any transformation U that internalizes the chosen N vectors transforms all the vectors in D to positive vectors.*

Proof: Consider the set of points $S = \{(a^j_1, a^j_2, ..., a^j_N) | d_j = a^j_1.i_1 + a^j_2.i_2 + ... + a^j_N.i_N, 1 \leq j \leq m\}$. All of them lie on the non-negative side of the hyperplane $i_1 = 0$, and therefore it is obvious that the origin $(0, 0, ..., 0)$ would be point on an outer face of a convex hull constructed for the set of points $S' = S \cup (0, 0, ..., 0)$. For example, in a 3–D case of Figure 5, right side of the shaded surface is the region of confinement of dependence vectors. This implies that there does exist a face of this convex hull which passes through the

Figure 5: Convex region of vectors

origin. Clearly, by virtue of the property of a convex hull boundary, all points of S that do not lie on this face H, lie on the same side of it. Knowing that the origin already belongs to it, we require $N-1$ other points out of S to define H. Let these points be $p_1, p_2, ..., p_{N-1}$ and let the corresponding dependence vectors be $d_{a_1}, d_{a_2}, ..., d_{a_{N-1}}$. If we have U' as a transformation that internalizes all these $N-1$ vectors, then the first column of U', say $\overline{u}_1$, is normal to all these $N-1$ vectors. In other words, $\overline{u}_1$ is normal to the hyperplane H passing through the origin as well as through $p_1, p_2, ..., p_{N-1}$. Clearly, the vectors corresponding to the remaining points, by virtue of their being on the same side of H, have dot products with $\overline{u}_1$ which are either all non-negative or all non-positive. These correspond to the two choices of the normal $\overline{u}_1$ to the hyperplane H. The first choice is towards the positive i_1 direction and the second towards the negative i_1 direction. We can now obtain a transformation U from U' that would internalize any one of the remaining vectors, say d_{a_N} at level N. The first column of U, say $\overline{u}'_1$, would surely therefore have a positive dot product with d_{a_N}. Effectively, $\overline{u}'_1$ would be the same as $\overline{u}_1$ if $d_{a_N}.\overline{u}_1$ is positive; it would be $-\overline{u}_1$ otherwise. Thus, we can always find a subset of dependences $d_{a_1}, d_{a_2}, ..., d_{a_N}$ from D such that any transformation U that internalizes all of them transforms all of D to positive vectors.

□

We now present an algorithm which determines one set of N dependence vectors from the total collection of m, $(m > N)$ vectors such that internalizing this set yields a non-deadlocking transformation. Our problem is a special and simplified case of the general convex hull determination problem. We here are interested in only one face of the hull that contains the origin.

Algorithm A3:

Step 1: Order the ${}^mC_{N-1}$ combinations[1] of $N-1$ dependence vectors from the total of m vectors. Let $i=1$.

Step 2: Choose the i^{th} combination. Check if every remaining vector lies on the same side of the hyperplane H_i containing all the vectors in the combination. If so go to step 3, else increment i by 1 and repeat step 2.

We form a $N-1$ by N matrix M whose rows are the $N-1$ vectors in the chosen combination. Then, for each of vectors d' other than the ones in the combination, we form a N by N matrix M' by appending d' as the N^{th} row to M. It can be easily shown that the determinant of M' represents the constant term in the equation of the hyperplane parallel to H_i and passing through the point d'. If the determinants of all M' thus formed, are all non-negative or all non-postive, then it implies that all the other vectors d' lie on the same side of H_i.
Step 3: Choose any one of the remaining vectors to be appended to the set of vectors in the i^{th} combination to form the required set.
□
Example 3: Consider the set of dependences {(1,1,7), (1,5,-5), (3,1,-8), (1,1,-7), (1,1,8), (1,5,5), (3,3,2), (4,1,10)}. It can be shown that the pair {(1,1,8), (1,5,5)} is a candidate for determining a face of the convex hull. Therefore internalizing a triple made of these two and any other vector will result in a matrix that transforms every vector to a positive one.

We have thus far discussed the internalization of fewer than N vectors. Internalizing N or more vectors which cannot be all confined to a hyperplane of dimension less than N would lead to no parallelism. It would be necessary to strip mine the outer most loop in this case to obtain some parallelism. The well known method for deadlock free strip mining of an 1–D loop is the g.c.d method. According to this method the iterations at intervals determined by the g.c.d of all the dependence distances are grouped together to be executed in sequence.

4 Bounds of the Transformed Loops

Let U be an N by N unimodular matrix.

$$U = \begin{bmatrix} u_{11} & \cdots & u_{1N} \\ \vdots & & \\ u_{N1} & \cdots & u_{NN} \end{bmatrix}$$

U transforms the N–D iteration space $\Re$ in $(I_1, \ldots, I_N)$ co-ordinate system to an N–D iteration space $\Re'$ in $(K_1, \ldots, K_N)$ co-ordinate system with a new dependence structure. The new co-ordinate system is obtained by

$$(K_1, \ldots, K_N) = (I_1, \ldots, I_N).U$$

Thus,

$$\left.\begin{array}{rcl} K_1 & = & u_{11}I_1 + \cdots + u_{N1}I_N \\ & \vdots & \\ K_N & = & u_{1N}I_1 + \cdots + u_{NN}I_N \end{array}\right\} \qquad (4)$$

[1] A good heuristic here is to put the combinations that contain the vectors making the largest angle to i_1 first.

The limits of K_1, m_1 and M_1, can be obtained by the substitution of limits for $(I_1, \ldots, I_N)$ which are constants — n_1 to N_1, ... , n_N to N_N respectively, based on the signs of $u_{11}, \ldots, u_{N1}$.

$$m_1 = min(u_{11}I_1 + \cdots + u_{N1}I_N)$$

$$M_1 = max(u_{11}I_1 + \cdots + u_{N1}I_N)$$

Therefore,

$$m_1 = u_{11}^{+}n_1 - u_{11}^{-}N_1 + \cdots + u_{N1}^{+}n_N - u_{N1}^{-}N_N$$
$$M_1 = u_{11}^{+}N_1 - u_{11}^{-}n_1 + \cdots + u_{N1}^{+}N_N - u_{N1}^{-}n_N$$

However, the limits for $K_2, \ldots K_N$ can not be obtained in a similar way. A similar approach would implicitly assume that the transformed iteration space is rectangular, whereas, the transformed space in general is not rectangular. This is due to the inherent difficulty in obtaining an integer unimodular matrix that can perform an orthogonal transformation. Such a matrix would have preserved the rectangular shape of the original iteration space. Thus, this simple approach, though elegant, would result in extra points in $(K_1, \ldots, K_N)$ that do not correspond to actual iterations in the original loopnest. In general a particular loop index in the transformed loop depends on rest of the loop indices. The semantics of the loop suggests that the situation is not that bad after all! It can be observed that K_2 depends on K_1, K_3 depends on K_1 and K_2 so on.
That is, K_2 is from $f_1(K_1)$ to $g_1(K_2)$, K_3 is from $f_2(K_1, K_2)$ to $g_2(K_1, K_2)$, so on, and K_N is from $f_{N-1}(K_1, \ldots, K_{N-1})$ to $g_{N-1}(K_1, \ldots, K_{N-1})$, where f and g's are functions that are linear in K's with maxima and minima operators, and floors and ceilings to retain integer values.

In this section we shall derive expressions for the loop limits in a 3–D case. The loop limits for N–D case can be obtained by a straight forward generalization.

Let U_3 be a 3 by 3 unimodular matrix.

$$U_3 = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix}$$

U_3 transforms the 3–D iteration space $\Re^3$ in (I_1, I_2, I_3) co-ordinate system to an 3–D iteration space $\Re^{3\prime}$ in (K_1, K_2, K_3) co-ordinate system with a new dependence structure. The new co-ordinate system is obtained by

$$(K_1, K_2, K_3) = (I_1, K_2, I_3).U_3$$

4.1 Limits for K_1

As above, the limits of K_1, m_1 and M_1 are

$$m_1 = min(u_{11}I_1 + u_{21}I_2 + u_{31}I_3)$$

$$M_1 = max(u_{11}I_1 + u_{21}I_2 + u_{31}I_3)$$

Therefore,

$$\left.\begin{aligned} m_1 &= u_{11}^{+}n_1 - u_{11}^{-}N_1 + u_{21}^{+}n_2 - u_{21}^{-}N_2 + u_{31}^{+}n_3 - u_{31}^{-}N_3 \\ M_1 &= u_{11}^{+}N_1 - u_{11}^{-}n_1 + u_{21}^{+}N_2 - u_{21}^{-}n_2 + u_{31}^{+}N_3 - u_{31}^{-}n_3 \end{aligned}\right\} \quad (5)$$

4.2 Limits for K_2

The original iteration space is bounded by the axes I_1, I_2 and I_3, corresponding to vectors (1,0,0), (0,1,0) and (0,0,1). These vectors when transformed give the limits for K_2. Consider the vector (1,0,0):

$$(1,0,0).U_3 = (u_{11}, u_{12}, u_{13})$$

We know that K_2 depends only on K_1. In the K_1K_2 plane, the above vector corresponds to the line

$$K_2 = \frac{u_{12}}{u_{11}}K_1 + C \tag{6}$$

where is C is a constant. A particular point (p_1, p_2, p_3) in the original iteration space corresponds to

$$(p_1, p_2, p_3).U_3 = (u_{11}p_1 + u_{21}p_2 + u_{31}p_3, u_{12}p_1 + u_{22}p_2 + u_{32}p_3, u_{13}p_1 + u_{23}p_2 + u_{33}p_3) \tag{7}$$

The K_2 intercept C, corresponding to (p_1, p_2, p_3) is

$$C = \frac{1}{u_{11}}(u_{22}u_{11} - u_{12}u_{21})p_2 + \frac{1}{u_{11}}(u_{32}u_{11}) - u_{12}u_{31})p_3$$

$$C = \frac{\Delta_{1,2}^{1,2}}{\Delta_{1,1}^{1,1}}p_2 - \frac{\Delta_{1,3;2}^{1,2}}{\Delta_{1,1}^{1,1}}p_3$$

The Δ notation is an extension of the cofactor of a matrix, defined for a set of selected rows and columns of the matrix. Cofactors can be defined in terms of Δ. Magnitude of $\Delta_{i,j;k,l}^{a,b;c,d}$ is the value of the determinant formed by columns from a to b leaving out columns from c to d, and rows from i to j leaving out rows from k to l from transformation matrix U_3. Sign is positive if the summation of numbers of left out columns and rows is even, negative otherwise. If k,l,c,d are not mentioned it is all those rows and columns from U that do not appear in the ranges of i to j and a to b. For example, $(u_{32}u_{11} - u_{12}u_{31})$ is $-\Delta_{1,3;2}^{1,2}$.

Thus Equation 6 can be written as,

$$K_2 = \frac{\Delta_{1,1}^{2,2}}{\Delta_{1,1}^{1,1}}K_1 + \frac{\Delta_{1,2}^{1,2}}{\Delta_{1,1}^{1,1}}p_2 - \frac{\Delta_{1,3;2}^{1,2}}{\Delta_{1,1}^{1,1}}p_3 \tag{8}$$

Similarly, considering other two vectors, viz. (0,1,0) and (0,0,1) we have other two Equations for K_2.

$$K_2 = \frac{\Delta_{2,2}^{2,2}}{\Delta_{2,2}^{1,1}}K_1 + \frac{\Delta_{2,3}^{1,2}}{\Delta_{2,2}^{1,1}}p_3 - \frac{\Delta_{1,3;2}^{1,2}}{\Delta_{2,2}^{1,1}}p_1 \tag{9}$$

$$K_2 = \frac{\Delta_{3,3}^{2,2}}{\Delta_{3,3}^{1,1}}K_1 + \frac{\Delta_{1,3;2}^{1,2}}{\Delta_{3,3}^{1,1}}p_1 - \frac{\Delta_{2,3}^{1,2}}{\Delta_{3,3}^{1,1}}p_2 \tag{10}$$

The lower and upper bounds for K_2 are obtained by substituting for the bounding values for p_1, p_2 and p_3 in each of the equations, and taking a maxima or minima over the three possible values for lower bounds and upper bounds.

The three lower bounds $lb_1^{K_2}, lb_2^{K_2}, lb_3^{K_2}$ from Equations 8, 9, 10 are,

$$
\begin{aligned}
lb_1^{K_2} &= \frac{\Delta_{1,1}^{2,2}}{\Delta_{1,1}^{1,1}}K_1 + \left[\frac{\Delta_{1,2}^{1,2}}{\Delta_{1,1}^{1,1}}\right]^+ n_2 - \left[\frac{\Delta_{1,2}^{1,2}}{\Delta_{1,1}^{1,1}}\right]^- N_2 + \left[-\frac{\Delta_{1,3;2}^{1,2}}{\Delta_{1,1}^{1,1}}\right]^+ n_3 - \left[-\frac{\Delta_{1,3;2}^{1,2}}{\Delta_{1,1}^{1,1}}\right]^- N_3 \\
lb_2^{K_2} &= \frac{\Delta_{2,2}^{2,2}}{\Delta_{2,2}^{1,1}}K_1 + \left[\frac{\Delta_{2,3}^{1,2}}{\Delta_{2,2}^{1,1}}\right]^+ n_3 - \left[\frac{\Delta_{2,3}^{1,2}}{\Delta_{2,2}^{1,1}}\right]^- N_3 + \left[-\frac{\Delta_{1,3;2}^{1,2}}{\Delta_{2,2}^{1,1}}\right]^+ n_1 - \left[-\frac{\Delta_{1,3;2}^{1,2}}{\Delta_{2,2}^{1,1}}\right]^- N_1 \qquad (11) \\
lb_3^{K_2} &= \frac{\Delta_{3,3}^{2,2}}{\Delta_{3,3}^{1,1}}K_1 + \left[\frac{\Delta_{1,3;2}^{1,2}}{\Delta_{3,3}^{1,1}}\right]^+ n_1 - \left[\frac{\Delta_{1,3;2}^{1,2}}{\Delta_{3,3}^{1,1}}\right]^- N_1 + \left[-\frac{\Delta_{2,3}^{1,2}}{\Delta_{3,3}^{1,1}}\right]^+ n_2 - \left[-\frac{\Delta_{2,3}^{1,2}}{\Delta_{3,3}^{1,1}}\right]^- N_2
\end{aligned}
$$

Similarly, the three upper bounds $ub_1^{K_2}, ub_2^{K_2}, ub_3^{K_2}$ from Equations 8, 9, 10 are,

$$
\begin{aligned}
ub_1^{K_2} &= \frac{\Delta_{1,1}^{2,2}}{\Delta_{1,1}^{1,1}}K_1 + \left[\frac{\Delta_{1,2}^{1,2}}{\Delta_{1,1}^{1,1}}\right]^+ N_2 - \left[\frac{\Delta_{1,2}^{1,2}}{\Delta_{1,1}^{1,1}}\right]^- n_2 + \left[-\frac{\Delta_{1,3;2}^{1,2}}{\Delta_{1,1}^{1,1}}\right]^+ N_3 - \left[-\frac{\Delta_{1,3;2}^{1,2}}{\Delta_{1,1}^{1,1}}\right]^- n_3 \\
ub_2^{K_2} &= \frac{\Delta_{2,2}^{2,2}}{\Delta_{2,2}^{1,1}}K_1 + \left[\frac{\Delta_{2,3}^{1,2}}{\Delta_{2,2}^{1,1}}\right]^+ N_3 - \left[\frac{\Delta_{2,3}^{1,2}}{\Delta_{2,2}^{1,1}}\right]^- n_3 + \left[-\frac{\Delta_{1,3;2}^{1,2}}{\Delta_{2,2}^{1,1}}\right]^+ N_1 - \left[-\frac{\Delta_{1,3;2}^{1,2}}{\Delta_{2,2}^{1,1}}\right]^- n_1 \qquad (12) \\
ub_3^{K_2} &= \frac{\Delta_{3,3}^{2,2}}{\Delta_{3,3}^{1,1}}K_1 + \left[\frac{\Delta_{1,3;2}^{1,2}}{\Delta_{3,3}^{1,1}}\right]^+ N_1 - \left[\frac{\Delta_{1,3;2}^{1,2}}{\Delta_{3,3}^{1,1}}\right]^- n_1 + \left[-\frac{\Delta_{2,3}^{1,2}}{\Delta_{3,3}^{1,1}}\right]^+ N_2 - \left[-\frac{\Delta_{2,3}^{1,2}}{\Delta_{3,3}^{1,1}}\right]^- n_2
\end{aligned}
$$

Now the limits $m_2(K_1)$ and $M_2(K_1)$ for the K_2 loop are

$$
\left.\begin{aligned}
m_2(K_1) &= \lceil max(lb_1^{K_2}, lb_2^{K_2}, lb_3^{K_2})\rceil \\
M_2(K_1) &= \lfloor min(ub_1^{K_2}, ub_2^{K_2}, ub_3^{K_2})\rfloor
\end{aligned}\right\} \qquad (13)
$$

4.3 Limits for K_3

K_3 is a function of both K_1 and K_2 and the limits for K_3 can be found by working on the bounding planes corresponding to (0,0,0),(0,1,0),(0,0,1); (0,0,0),(1,0,0),(0,1,0); and (0,0,0),(1,0,0),(0,0,1). These planes are bounding the original iteration space and the transformed planes bound the transformed space.

Consider the plane, (0,0,0),(0,1,0),(0,0,1). This when transformed by U_3 corresponds to a plane passing through $(0,0,0),(u_{21},u_{22},u_{23}),(u_{31},u_{32},u_{33})$. If the equation for the plane passing through these three points is,

$$K_3 = s_1K_1 + s_2K_2 + C \qquad (14)$$

then by geometry,

$$s_1 = -\frac{\begin{bmatrix} 0 & 0 & 1 \\ u_{22} & u_{23} & 1 \\ u_{32} & u_{33} & 1 \end{bmatrix}}{\begin{bmatrix} 0 & 0 & 1 \\ u_{21} & u_{22} & 1 \\ u_{31} & u_{32} & 1 \end{bmatrix}} = -\frac{(u_{22}u_{33} - u_{32}u_{23})}{(u_{21}u_{32} - u_{31}u_{22})} = -\frac{\Delta_{2,3}^{2,3}}{\Delta_{2,3}^{1,2}} \qquad (15)$$

$$s_2 = -\frac{\begin{bmatrix} 0 & 0 & 1 \\ u_{23} & u_{21} & 1 \\ u_{33} & u_{31} & 1 \end{bmatrix}}{\begin{bmatrix} 0 & 0 & 1 \\ u_{21} & u_{22} & 1 \\ u_{31} & u_{32} & 1 \end{bmatrix}} = -\frac{(u_{23}u_{31} - u_{33}u_{21})}{(u_{21}u_{32} - u_{31}u_{22})} = -\frac{\Delta_{2,3}^{1,3;2}}{\Delta_{2,3}^{1,2}} \tag{16}$$

Consider an arbitrary point (p_1, p_2, p_3) that will be transformed as in Equation 7 and substitute in the Equation 14 for K_3 to find C. It can be shown that

$$C = \frac{\Delta_{1,4}^{1,4}}{\Delta_{2,3}^{1,2}} p_1 \tag{17}$$

and thus Equation 14 can be written as,

$$K_3 = -\frac{\Delta_{2,3}^{2,3}}{\Delta_{2,3}^{1,2}} K_1 - \frac{\Delta_{2,3}^{1,3;2}}{\Delta_{2,3}^{1,2}} K_2 + \frac{\Delta_{1,4}^{1,4}}{\Delta_{2,3}^{1,2}} p_1 \tag{18}$$

Similarly, considering other two possible planes, we can obtain two other equations for K_3.

$$K_3 = -\frac{\Delta_{1,3}^{2,3}}{\Delta_{1,2}^{1,2}} K_1 - \frac{\Delta_{1,2}^{1,3;2}}{\Delta_{1,2}^{1,2}} K_2 + \frac{\Delta_{1,4}^{1,4}}{\Delta_{1,2}^{1,2}} p_3 \tag{19}$$

$$K_3 = -\frac{\Delta_{1,3;2}^{2,3}}{\Delta_{1,3;2}^{1,2}} K_1 - \frac{\Delta_{1,3;2}^{1,3;2}}{\Delta_{1,3;2}^{1,2}} K_2 + \frac{\Delta_{1,4}^{1,4}}{\Delta_{1,3;2}^{1,2}} p_2 \tag{20}$$

The lower bounds for K_3 are obtained from Equations 18, 19, 20 by substituting the bounds in the original space.

$$\begin{aligned}
lb_1^{K_3} &= -\frac{\Delta_{2,3}^{2,3}}{\Delta_{2,3}^{1,2}} K_1 - \frac{\Delta_{2,3}^{1,3;2}}{\Delta_{2,3}^{1,2}} K_2 + \left[\frac{\Delta_{1,4}^{1,4}}{\Delta_{2,3}^{1,2}}\right]^+ n_1 - \left[\frac{\Delta_{1,4}^{1,4}}{\Delta_{2,3}^{1,2}}\right]^- N_1 \\
lb_2^{K_3} &= -\frac{\Delta_{1,3}^{2,3}}{\Delta_{1,2}^{1,2}} K_1 - \frac{\Delta_{1,2}^{1,3;2}}{\Delta_{1,2}^{1,2}} K_2 + \left[\frac{\Delta_{1,4}^{1,4}}{\Delta_{1,2}^{1,2}}\right]^+ n_3 - \left[\frac{\Delta_{1,4}^{1,4}}{\Delta_{1,2}^{1,2}}\right]^- N_3 \\
lb_3^{K_3} &= -\frac{\Delta_{1,3;2}^{2,3}}{\Delta_{1,3;2}^{1,2}} K_1 - \frac{\Delta_{1,3;2}^{1,3;2}}{\Delta_{1,3;2}^{1,2}} K_2 + \left[\frac{\Delta_{1,4}^{1,4}}{\Delta_{1,3;2}^{1,2}}\right]^+ n_2 - \left[\frac{\Delta_{1,4}^{1,4}}{\Delta_{1,3;2}^{1,2}}\right]^- N_2
\end{aligned} \tag{21}$$

Similarly, the upper bounds are,

$$\begin{aligned}
ub_1^{K_3} &= -\frac{\Delta_{2,3}^{2,3}}{\Delta_{2,3}^{1,2}} K_1 - \frac{\Delta_{2,3}^{1,3;2}}{\Delta_{2,3}^{1,2}} K_2 + \left[\frac{\Delta_{1,4}^{1,4}}{\Delta_{2,3}^{1,2}}\right]^+ N_1 - \left[\frac{\Delta_{1,4}^{1,4}}{\Delta_{2,3}^{1,2}}\right]^- n_1 \\
ub_2^{K_3} &= -\frac{\Delta_{1,3}^{2,3}}{\Delta_{1,2}^{1,2}} K_1 - \frac{\Delta_{1,2}^{1,3;2}}{\Delta_{1,2}^{1,2}} K_2 + \left[\frac{\Delta_{1,4}^{1,4}}{\Delta_{1,2}^{1,2}}\right]^+ N_3 - \left[\frac{\Delta_{1,4}^{1,4}}{\Delta_{1,2}^{1,2}}\right]^- n_3
\end{aligned} \tag{22}$$

$$ub_3^{K_3} = -\frac{\Delta_{1,3;2}^{2,3}}{\Delta_{1,3;2}^{1,2}} K_1 - \frac{\Delta_{1,3;2}^{1,3;2}}{\Delta_{1,3;2}^{1,2}} K_2 + \left[\frac{\Delta_{1,4}^{1,4}}{\Delta_{1,3;2}^{1,2}}\right]^+ N_2 - \left[\frac{\Delta_{1,4}^{1,4}}{\Delta_{1,3;2}^{1,2}}\right]^- n_2$$

Now the limits $m_3(K_1, K_2)$ and $M_3(K_1, K_2)$ for the K_3 loop are

$$\left.\begin{array}{r} m_3(K_1,K_2) = \lceil max(lb_1^{K_3}, lb_2^{K_3}, lb_3^{K_3})\rceil \\ M_3(K_1,K_2) = \lfloor min(ub_1^{K_3}, ub_2^{K_3}, ub_3^{K_3})\rfloor \end{array}\right\} \tag{23}$$

In general for an N–D loop, loop index K_i will have ${}^N C_{i-1}$ upper bounds and lower bounds, and the limits are obtained by taking the floors and ceilings as we did just now. An expression for K_i will be in terms of $K_1, \ldots, K_{i-1}$ and constant terms will be permutations of N upper and and lower bounds taken (N-i+1) at a time.

5 Conclusions

We have presented a generalized unimodular loop transformation as a simple, elegant and systematic way of transforming nested loops with uniform dependences. These give rise to new nested loops wherein several chosen dependences are internalized. That is to say these dependences do not give rise to any communications if the outer loops are executed concurrently. Our methodology ensures that the resulting loops have no deadlocking dependences when the inner loops of the transformed iteration space are executed in sequence. We have also derived the general expression for the bounds of the transformed loops in terms of the bounds of the original loops and the elements of the transformation matrix.

We are currently looking for transformations which permute the initial iteration space before applying a unimodular loop transformation. These unified transformations can then internalize any subset of the dependences without giving rise to deadlocks. We are also deriving the expressions for the parameters of the transformations suggested in [6, 7] for the generalized transformation presented in this paper. These parameters are - *parallelism factor*, the *load imbalance* and the *volume of communication*. We are also interested in investigating the correlation of these parameters with a chosen machine model and the choice of an optimum transformation for any given model. Finally, we would like to look into the optimization of the derivation of the transformation themselves.

Acknowledgment

We are indebted to Dr. U Banerjee for discussions and suggestions that introduced us to this problem. We are grateful to Dr. Venkaiah for several valuable discussions. We also would like to thank Srinivas Kumar and Asha for their great help.

References

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

[2] U. Banerjee, A Theory of Loop Permutations, proceedings of the second Workshop on Programming Languages and Compilers for Parallel Computing, 1989.

[3] U. Banerjee, Unimodular Transformations of Double Loops, proceedings of the third Workshop on Programming Languages and Compilers for Parallel Computing, Irvine, California, August 1-3, 1990.

[4] E.H. D'Hollander, Partitioning and Labeling of Index Sets in Do Loops with Constant Dependence Vectors, proceedings of International Conference on Parallel Processing, Vol. 2, pp 139- 144, 1989.

[5] C.T. King and L.M. Ni, Grouping in Nested Loops for Parallel Execution on Multicomputers, proceedings of International Conference on Parallel Processing, Vol. 2, pp 31-38, 1989.

[6] D Kulkarni, K.G. Kumar, A. Basu, and A. Paulraj, Loop Partitioning Unimodular Transformations for Distributed Memory Multiprocessors, to be presented at the Fifth International Parallel Processing Symposium, March, 1991.

[7] D Kulkarni, K.G. Kumar, A. Basu, and A. Paulraj, Loop Partitioning for Distributed Memory Multiprocessors as Unimodular Transformations, submitted to the ACM International Conference on Supercomputing, June, 1991.

[8] L. Lamport, The Parallel Execution of DO Loops, communications of the ACM, 17(2), 1974.

[9] J.K. Peir and Ron Cytron, Minimum Distance - A Method for Partitioning Recurrences for Multiprocessors, proceedings of International Conference on Parallel Processing, pp 217-224, 1987.

[10] J. Ramanujam and P. Sadayappan, A Methodology for Parallelizing Programs for Multicomputers and Complex Memory Multiprocessors, ACM Conference on Supercomputing, 1989.

[11] J. Ramanujam and P. Sadayappan, Tiling of Iteration Spaces for Multicomputers, proceedings of International Conference on Parallel Processing, Vol. 2, pp 179-186, 1990.

[12] W. Shang and J.A.B. Fortes, Independent Partitioning of Algorithms with Uniform Dependencies, proceedings of International Conference on Parallel Processing, Vol. 2, pp 26-33, 1988.

[13] M.J. Wolfe, More Iteration Space Tiling, ACM Conference on Supercomputing, pp. 655-664, 1989.