Mappings for communications minimization using distribution and alignment

Catherine Mongenet
Université Louis Pasteur de Strasbourg, Laboratoire ICPS
Pôle API, Boulevard Sébastien Brant, 67400 Illkirch, France
E-mail : mongenet@icps.u-strasbg.fr

Abstract

This paper introduces results on communications minimization for systems of affine recurrence equations. We show how the dependences of such a system can be decomposed into two classes : the auto-dependences and the crossed-dependences. We then present how the localization optimization can be achieved in two steps : a distribution of the auto-dependences and an alignment of the crossed-dependences. Since localization of all the dependences of a problem is generally not possible, we finally introduce a heuristic to globally minimize the communications.

Keywords: parallelization techniques, localization optimization, communications minimization, systems of recurrence equations.

Introduction

Among the various and fruitful research works concerned with parallelism, parallelizing techniques and compilers developments have focus a lot of attention over the past decade ([12], [3], [22], [4], [1]). One of the major concern in parallelizing compilers is the development of efficient loop parallelization techniques for which an important issue is to optimize localization and minimize communications ([11],[8], [21], [6], [17]).

This paper focuses on this question. We consider problems defined by systems of affine recurrence equations (called SARE in the following), which are formal versions of single assignment loop nests. The parallelization of such systems relies on a space-time transformation applied to the domain and the dependences of the problem. In this paper the transformations are the classical affine mappings based on an affine-per-variable schedule function and an affine-per-variable allocation function.

When the system of recurrence equations is defined over several variables, one can distinguish two types of dependences : the auto-dependences and the crossed-dependences. Consider a given result variable of the problem, there may exist dependences between different items of this variable (for example $A[z]$ depends on $A[z']$). We call such dependences which are local to a variable the *auto-dependences*. Conversely there may exist dependences between items of different result variables (for example $A[z]$ depends on $B[z']$). We call such dependences the *crossed-dependences*.

Communications minimization can be achieved by optimizing both distribution and alignment. Distribution consists in projecting the different items of a variable in order to minimize the communications due to the auto-dependences. Alignment consists in projecting differently several variables (or several arrays) to the processors in order to minimize the communications due to the crossed-dependences. Other contributions address these problems ([19],[10], [18], [9], [2], [5], [7], [20]).

The aim of this paper is to show how distribution and alignment can both contribute to the determination of efficient affine allocation functions characterized by a minimization of the communications. It is organized as follows. Section 1 presents the method of SARE parallelization, the dependence analysis and the space-time mapping. It especially recalls the formalism used to properly analyze the multiple use of data induced by affine dependences. It is based on the notion of utilization sets and vectors and of emission sets ([16], [15]). Section 2 introduces the ideas related to communications minimization. Section 3 shows how to determine allocation-per-variable directions that distribute each variable and result in localization optimization relatively to auto-dependences. Section 4 focuses on the crossed-dependences and shows how to align together the different variables of a problem in order to optimize the localization according to these crossed-dependences. In most problems it is not possible to remove all the communications. This is why we present in section 5 a heuristic to globally optimize localization of a problem and illustrate it on an example. Finally in section 6 we compare this approach with related works and give some concluding remarks.

1 Parallelization of systems of affine recurrence equations

The problems we consider are characterized by systems of affine recurrence equations (SARE). Each equation of the system is of the form:

$$X[z] = f\big(\ldots, Y\big[g(z)\big], \ldots\big), \qquad z \in D, \qquad (E)$$

where X and Y are variable names indexed over a convex bounded polyhedron $D \subset \mathbf{Z}^n$, called the *index domain* or the *iteration space* [1]. It is defined by a set of linear constraints expressed by a system of linear inequalities $D = \{z \in \mathbf{Z}^n \mid P \cdot z \leq Q\}$ where P is a $m \times n$ integer matrix and Q a integer vector of dimension m. The computation function f is any O(1) function. Function g is an affine function of the form $g(z) = R \cdot z + r$ from $\mathbf{Z}^n$ to $\mathbf{Z}^{n'}$ $(n' \leq n)$, called the *index function*. R is a $n' \times n$ integer matrix called the *index matrix* and r is a vector of dimension n'.

1.1 Dependence definitions

In order to parallelize such a SARE, the dependence analysis has first to be realized. It relies on the notions of *utilization set* and *emission set*.

For each argument Y appearing in the right-hand side of any equation (E), we characterize the set of points using $Y[z_0]$, $z_0 \in \mathbf{Z}^{n'}$. This set is called its *utilization set* and defined by :

Definition 1 *The utilization set associated with a given item $Y[z_0]$ relative to equation (E) is:*

$$Util_E(Y, z_0) = \{z \in D_p \mid g(z) = z_0\}.$$

We show in [15] that a utilization set is a convex bounded polyhedron whose linear space is $Ker(R)$ where R is the index matrix. Its dimension is therefore $n - Rank(R)$ where n is the dimension of the domain. The basis vectors of $Ker(R)$ characterize the utilization set and are called the *utilization vectors*. They are denoted by $u_{E,Y,i}$ $(i = 1, \ldots, \ell)$. Notice that when R is a full row rank matrix, the utilization set has dimension 0 and does not require any utilization vector.

In a SARE the causal dependences imposed by the problem are related to result variables defined on the same domain $(n' = n)$ and expressed for any result item $Y[z_0]$ by the *dependence vectors* $d_{Y,z_0} = z - z_0$, for any $z \in Util_E(Y, z_0)$ [2].

All dependence vectors d_{Y,z_0} of a given set have the same origin z_0 and their extremities belong to the convex bounded polyhedron $Util_E(Y, z_0)$. Therefore they define a *cone* which is characterized by its finite set of *extremal vectors*. These extremal dependence vectors are denoted by d_{Y,z_0,ex_i} for $i \in \mathbf{N}$.

In order to distinguish between distribution and alignment, we classify the dependences [3] into two groups.

Definition 2 *Let (E) be an equation of the form $X[z] = f\big(\ldots, Y\big[g(z)\big], \ldots\big)$ where Y is a result variable. The dependence related to argument Y is called an auto-dependence if $Y = X$. It is called a crossed-dependence from Y to X if $Y \neq X$.*

All points $z_0 = g(z)$ which are origin of a cone according to one dependence form a set called the *emission set*. This set is particularly important when the problem is mapped on a parallel solution. Hence its points originate the corresponding communications. It is defined by:

Definition 3 *The emission set associated with result variable Y relative to equation (E) is*

$$\begin{aligned} Emit_E(Y) &= \{z_0 \in D \mid \exists z \in D \text{ such that } g(z) = z_0\} \\ &= \{R \cdot z + r \mid z \in D\} = g(D) \end{aligned}$$

1.2 Scheduling and allocation functions

The determination of a parallel program from a problem defined by a system of recurrence equations consists in the determination of a space-time mapping. Such a mapping is characterized by a schedule function and by an allocation function.

The *schedule functions* we consider are affine-per-variable functions. They are defined by :

$$\begin{array}{llll} t_X : & D \subset \mathbf{Z}^n & \longrightarrow & \mathbf{N} \\ & z & \longmapsto & t_X(z) = \lambda_X \cdot z + \alpha_X \end{array}$$

where λ_X is an integral vector and α_X are integer constant. The schedule function defines on the domain a set of hyperlanes orthogonal to the *schedule vector* λ_X called the *scheduling surfaces*. Notice that $Ker(\lambda_X)$ has dimension $n-1$ and is the linear space of any scheduling surface.

Recall that any affine schedule function must satisfy a set of causal constraints expressed by a system of inequalities. For a SARE the causal constraints related to the auto-dependences of variable X are expressed by $\lambda_X \cdot d_{ex} > 0$ for all the extremal dependence vectors d_{ex} associated with variable X. Moreover for any crossed-dependence from variable Y to variable X characterized by dependence vector(s) d_{YX} the causal constraint is defined by $t_Y(z) < t_X(z + d_{YX})$ for all the corresponding utilization points z.

The allocation functions we consider are affine-per-variable functions which project all the points of the domain $D \subset \mathbf{Z}^n$ onto a set of virtual processors in $\mathbf{Z}^{n-1}$ along a direction called *allocation or projection direction*. The allocation direction associated to any variable of the problem is defined by a unimodular vector $\xi_X \in \mathbf{Z}^n$. The allocation functions are defined by :

$$\begin{array}{llll} alloc_X : & \mathbf{Z}^n & \longrightarrow & \mathbf{Z}^{n-1} \\ & z & \longmapsto & \sigma_X \cdot z + \gamma_X \end{array}$$

where σ_X is a $(n-1) \times n$ integer matrix called the *allocation matrix* and γ_X is an integer vector of dimension $n-1$. The row vectors of σ_X form the basis vectors of the processors space associated with variable X. They are determined from the allocation direction ξ_X using an Hermite normal form decomposition.

Recall that the schedule and allocation functions associated with variable X must guaranty that a processor executes at most one computation associated with X at any given time step. This requires the condition $\lambda_X \cdot \xi_X \neq 0$ to hold.

Once a schedule and an allocation are determined, the parallel solutions of a problem in terms of a program and communications can be synthesized by applying the schedule and allocation functions to the domain, the utilization

[1] Notice that all the equations of the system are defined over the same domain, which corresponds in the context of loop parallelization to perfect loop nests.

[2] When the index matrix R is equal to the identity matrix Id, the dependence is uniform and is simply characterized by one constant dependence vector.

[3] Recall that a SARE defines a set of single-assignment statements. Therefore all the dependences defined in the system are true dependences.

and extremal dependence vectors. The projection of the domain results in the shape of the processor arrays while the projection of the utilization and extremal vectors defines the communications between the processors.

2 Communications minimization

The objective of this paper is to show how auto-dependences and crossed-dependences have to be analyzed in order to globally minimize the communications related to a given problem. The number of communications obviously depends on the choice of allocations. We will therefore focus on the determination of the allocation functions.

Best results will be obtained if the different variables of a problem are allocated separately. Hence communications result either from auto-dependences or from crossed-dependences. If we consider separately each result variable and if we determine one (or possibly several) optimal allocation directions that minimize the number of communications due to its auto-dependences then one can globally minimize the number of communications due to all the auto-dependences of a problem. This process is called the **distribution**. It successively applies to the different result variables of the problem.

Once the distribution process is completed we have one or several optimal allocation directions for each variable X. If we apply these directions to the problem, we get a processor array [4] for each variable X. Each processor computes several different items of variable X, at most one at each time step. The objective is now to map all these arrays on a single array of processors. We call this process the **alignment**. Each processor of the final array computes at each time step an item of each result variable (at most). The order between these elementary computations may be imposed by the crossed-dependences.

Since the objective in this paper is to minimize the communications, we first focus on the determination of allocations. The schedule will be determined at a second stage, according to the causal constraints associated with both the auto-dependences and the crossed-dependences. This step is not detailed in this paper. We simply illustrate it on the example of section 5.

3 Distributing auto-dependences

An auto-dependence occurs between two different items of the same result variable (i.e. different elements of the same array).

Localization will be optimized according to the auto-dependences of one variable if the allocation direction projects the corresponding utilization points onto a minimal number of processors. The determination of the efficient allocation directions depends on the nature of the dependences: uniform or affine.

3.1 Uniform auto-dependences

Let us consider a result variable $Y[z]$ whose auto-dependences are all uniform (this case has been fully presented in [14]). In order to minimize the communications local to Y we introduce the following notion.

Definition 4 *For a given emission point z_0 related to a result variable Y, we denote by $\overline{Util}(Y, z_0)$ the set defined by z_0 and the union of all the utilization sets related to $Y[z_0]$:*

$$\overline{Util}(Y, z_0) = \{z_0\} \cup_E Util_E(Y, z_0)^5$$

As shown in [14] the number of communications local to Y is minimized if the allocation direction ξ_Y projects the set $\overline{Util}(Y, z_0)$ onto a minimal number of processors. The number of communications is equal to $nb - 1$ where nb is the number of lines parallel to ξ_Y cutting $\overline{Util}(Y, z_0)$ in at least one point.

Example 1. Let us consider the problem defined by the following uniform recurrence equation:

$$a(i,j) = f\big(a(i+1, j-2), a(i, j-1), a(i-2, j+1)\big) \quad (1)$$

where f is any O(1) computation function and where the points (i, j) are defined over any convex polyhedron domain of $\mathbf{Z}^2$.

This problem is characterized by three constant dependence vectors respectively associated with the first, second and third occurrence of variable a in the right hand side of the equation : $d_1 = (-1, 2), d_2 = (0, 1), d_3 = (2, -1)$. We have $\overline{Util}\big(a, (i_0, j_0)\big) = \{(i_0, j_0), (i_0 - 1, j_0 + 2), (i_0, j_0 + 1), (i_0 + 2, j_0 - 1)\}$ and the only direction that projects these four points optimally is $\xi = (1, -1)$. It defines two parallel lines intersecting $\overline{Util}\big(a, (i_0, j_0)\big)$. It is therefore characterized by only one communication of any element of variable a. □

Notice that a variable element which is sent only once to a processor, will be used for several computations on that processor. For the above example, each element is used for three different computations on the same processor. Therefore the element must be properly bufferized. We show in [14] how to compile this bufferization using FIFO structures.

3.2 Affine auto-dependences

Let us consider a result variable $Y[z]$ characterized by affine auto-dependences (this case has been fully presented in [13]). An affine dependence is defined by a utilization set of dimension $\delta \geq 1$ and by δ utilization vectors $u_{Y_i}, i = 1 \ldots, \delta$.

If broadcast communications are available on the target architecture, then we can determine efficient schedules enforcing the use of these broadcast primitives. This is achieved by scheduling at the same time step several elements of a utilization set. In this case the allocation associated with any variable must be determined after the schedule by trying to maximize localization for the remaining dependences. This alternative has been fully presented in [13] and is not discussed in this paper. We restrict ourselves to parallel solutions without broadcast and we therefore have to first determine the allocation. This determination is guided by the necessity to localize as many affine dependences as

[4] Depending on the shape of the domain, these arrays may not have the same shape.

[5] Since the dependences are uniform, each utilization set is reduced to one point : the extremity of the corresponding uniform dependence vector.

possible. In this case the schedule will be determined after the allocation.

In order to minimize the number of communications according to the affine auto-dependences related to result variable Y, one must try to localize as many utilization sets as possible. This can be achieved by choosing as allocation direction a vector projecting a utilization set on a minimal number of processors.

For a one-dimensional utilization set characterized by a unique utilization vector u_Y the communication is optimally reduced with allocation direction $\xi_Y = u_Y$. Hence all the corresponding utilization points are projected on the same processor (called the *utilization processor*) and this dependency requires at most one communication : from the *emission processor* (the processor where the corresponding emission point is implemented) to the utilization processor[6].

For a two-dimensional utilization set characterized by two utilization vectors u_{Y_1} and u_{Y_2}, one must choose as allocation direction a vector parallel to the plane defined by u_{Y_1} and u_{Y_2} in order to project the plane on a single line of utilization processors. The dependence requires one communication from the emission processor to one utilization processor (the processor associated with one of the extremal dependence vector), followed by communications along the line of utilization processors. Notice that in this case the remaining communications are local because each utilization processor sends the data to its neighboring processor according to the utilization vectors. This determination extends to utilization sets of larger dimension.

When a result variable is characterized by several affine auto-dependences, the allocation direction is determined using the above remarks. When two affine auto-dependences related to the same variable are characterized by non parallel utilization vectors, we will be able to maximize the localization according to one dependence but not to all the other ones. The number of communications related to these dependences is proportional to the number of utilization points (cf. [13]). In such a case, the different allocation directions minimizing respectively the different utilization sets have to be considered in the full process of parallelization.

Example 2. Let us consider the problem defined by the following affine recurrence equation:

$$\begin{aligned} a(i,j) &= f\big(a(i-1,1), a(0,j)\big) \qquad (2)\\ & \qquad 1 \leq i \leq N,\ 1 \leq j \leq N-i+1 \end{aligned}$$

where f is any O(1) computation function and where $a(0,j)$, $1 \leq j \leq N$ are given data.

This problem is characterized by two one-dimensional affine dependences described in figure 1 where the crosses represent the initial values. It is respectively characterized by utilization vectors $u_1 = (0,1)$ and $u_2 = (1,0)$.

The first dependence is localized if the allocation direction is equal to $\xi_a = u_1 = (0,1)$. All the utilization points associated with one element of a are projected on the same processor and there is only one communication from the emission processor associated with any emission point to the corresponding utilization processor. With this choice of allocation the second dependence is not localized.

[6] If the emission point belongs to the linear space containing the utilization set, then the emission processor coincides with the utilization processor and no communication is needed.

In order to localize the second dependence, we must choose $\xi_a = u_2 = (1,0)$ as allocation direction. In this case it is completely localized with no communication at all because the emission processor coincides with the corresponding utilization one. In this situation, the first dependence of course requires communications.

Since we restrict ourselves to affine allocations, there is no way to simultaneously localize these two affine dependences which are characterized by non parallel utilization vectors. The two above allocation directions are interesting. None it better than the other in terms of communications. In both cases, there will be one communication directed by the non localized utilization vector from any processor P_i to P_{i+1}. Therefore in the process of parallelization, these two solutions have to be studied from which different efficient parallel solutions will be synthesized. Notice that both allocation directions are compatible with the most efficient broadcast-free schedule $\lambda_a = (1,1)$. □

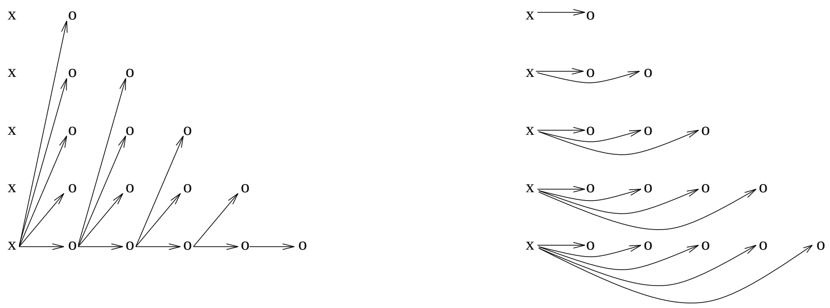
4 Aligning crossed-dependences

When a problem is characterized by several variables one can allocate separately these different variables to the processors in order to minimize communications due to auto-dependences and to crossed-dependences. Once distribution has been realized, one has for each variable the allocation direction that minimizes its local communications (There may be several allocation directions realizing such a minimization). From such a direction, one can determine the allocation matrix.

The alignment process applies then to each crossed dependence and tries to minimize the number of communications due to crossed-dependences. In order to keep as much freedom as possible, the alignment process applies to the allocation matrices obtained from the distribution process unimodular transformations from which the allocation functions will be determined. The use of unimodular matrices allows simple geometrical transformations such as symmetry on the parallel solutions. The result can simplify the communications.

Example. Let us consider two variables A and B that are allocated to a square array of processors using allocation matrices σ_A and σ_B. Let us suppose that these two allocations remove all the communications due to the auto-dependences of both variables, but that however if we map the two processor arrays associated respectively with A and B onto the final array using the identity function, then the crossed-dependence between A and B results in a communication from any processor $P_{x,y}$ to processor $P_{y,x}$. This communication will be localized if we apply unimodular matrix $U = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ to σ_B. Hence it will map processor $P_{x,y}$ associated with variable a and processor $P_{y,x}$ associated with b onto the same final processor. In this case the allocation functions will be : $alloc_A(z) = \sigma_A \cdot z$ and $alloc_B(z) = U\ \sigma_B \cdot z$. □

The alignment process is realized in the following way. We fix the allocation function for one result variable (let us say it is variable X) as simply equal to $alloc_X(z) = \sigma_X \cdot z$. The other allocation functions related to the other result variables Y are defined from the $(n-1) \times n$ allocation matrices σ_Y by : $alloc_Y(z) = U_Y\ \sigma_Y \cdot z + c_Y$ where U_Y is a



First argument Second argument

Figure 1: Dependences for example 2

$(n-1) \times (n-1)$ unimodular matrix and c_Y is a constant integer vector of dimension $(n-1)$.

The values of matrices U_Y and of vectors c_Y are determined in order to align as many variables as possible, i.e. in order the avoid communications for as many crossed-dependences as possible.

For each crossed-dependence there are two possible ways to avoid a communication. We can either localize the crossed dependence or align it with a non-localized auto-dependence.

Let us consider a crossed-dependence from variable Y to variable X defined on a convex domain D. From now on, to simplify the notations we denote its utilization sets by $Util_{YX}(Y, z_0)$, its dependence vectors by d_{YX}, the corresponding emission set by $Emit(Y)$ and its index matrix and vector by R_Y and r_Y. Moreover we denote by ξ_X and σ_X (respectively ξ_Y and σ_Y) any allocation vector and matrix minimizing the number of communications due to the auto-dependences of X (respectively Y).

4.1 Localization of a crossed-dependence

This case signifies that one should try to allocate variables Y and X in such a way that the two extremities of the dependence vectors d_{YX} are projected on the same processor. Using the previous notations, the following proposition gives the necessary and sufficient condition for localization to be successful.

Proposition 1 *A crossed dependence from Y to X characterized by dependence vectors d_{YX} is **communication-free** if there exist a unimodular matrix U_Y and a constant integer vector c_Y satisfying :*

$$\begin{cases} U_Y\ \sigma_Y\ R_Y = \sigma_X\ R_Y & (L1) \\ U_Y\ \sigma_Y \cdot r_Y + c_Y = \sigma_X \cdot r_Y + \sigma_X \cdot d_{YX} & (L2) \end{cases}$$

Proof. The two extremities of the dependence vectors d_{YX} are projected on the same processor

$$\text{iff } alloc_Y(z_{emit}) = alloc_X(z_{emit} + d_{YX}) \quad \forall z_{emit} \in Emit(Y),$$

$$\text{iff } alloc_Y(R_Y \cdot z + r_Y) = alloc_X(R_Y \cdot z + r_Y + d_{YX}) \quad \forall z \in \bigcup_{z_{emit}} Util_{YX}(Y, z_{emit}),$$

$$\text{iff } U_Y\ \sigma_Y\ R_Y \cdot z + U_Y\ \sigma_Y \cdot r_Y + c_Y = \sigma_X\ R_Y \cdot z + \sigma_X \cdot r_Y + \sigma_X \cdot d_{YX} \quad \forall z \in \bigcup_{z_{emit}} Util_{YX}(Y, z_{emit}).$$

This implies conditions (L1) and (L2). □

4.2 Alignment of a crossed-dependence with an auto-dependence

This case consists in aligning the crossed-dependence with an auto-dependence related to variable Y that already induces one communication. This is possible when the emission sets related to these two dependences intersect. Since the auto-dependence requires a communication link, if both dependences are aligned then this link can to used for the crossed-dependence as well. The following proposition gives the necessary and sufficient condition for such an alignment to be successful.

Proposition 2 *A crossed dependence from Y to X characterized by dependence vectors d_{YX} is **communication-free** if there exist a unimodular matrix U_Y and a constant integer vector c_Y satisfying :*

$$\begin{cases} U_Y\ \sigma_Y\ R_Y = \sigma_X\ R_Y & (A1) \\ U_Y\ \sigma_Y \cdot r_Y + U_Y\ \sigma_Y \cdot d_Y + c_Y = \\ \qquad \sigma_X \cdot r_Y + \sigma_X \cdot d_{YX} & (A2) \end{cases}$$

where d_Y is a dependence vector related to a non localized auto-dependence of variable Y.

Proof. The crossed-dependence vector d_{YX} is aligned with the auto-dependence vector d_Y

$$\text{iff } alloc_Y(z_{emit} + d_Y) = alloc_X(z_{emit} + d_{YX}) \quad \forall z_{emit} \in Emit(Y)$$

$$\text{iff } alloc_Y(R_Y \cdot z + r_Y + d_Y) = alloc_X(R_Y \cdot z + r_Y + d_{YX}) \quad \forall z \in \bigcup_{z_{emit}} Util_{YX}(Y, z_{emit})$$

$$\text{iff } U_Y\ \sigma_Y\ R_Y \cdot z + U_Y\ \sigma_Y \cdot r_Y + U_Y\ \sigma_Y \cdot d_Y + c_Y = \sigma_X\ R_Y \cdot z + \sigma_X \cdot r_Y + \sigma_X \cdot d_{YX} \quad \forall z \in \bigcup_{z_{emit}} Util_{YX}(Y, z_{emit})$$

This implies conditions (A1) and (A2). □

These two propositions can be synthesized in a theorem giving the necessary and sufficient condition for a crossed-dependence to be communication-free.

Theorem 1 *A crossed-dependence from Y to X characterized by dependence vectors d_{YX} is **communication-free** if and only there exist a unimodular matrix U_Y and a constant integer vector c_Y satisfying :*

$$\left\{\begin{array}{ll} U_Y\ \sigma_Y\ R_Y = \sigma_X\ R_Y & (L1) \\ U_Y\ \sigma_Y \cdot r_Y + c_Y = \sigma_X \cdot r_Y + \sigma_X \cdot d_{YX} & (L2) \end{array}\right.$$

or

$$\left\{\begin{array}{ll} U_Y\ \sigma_Y\ R_Y = \sigma_X\ R_Y & (A1) \\ U_Y\ \sigma_Y \cdot r_Y + U_Y\ \sigma_Y \cdot d_Y + c_Y = & \\ \qquad \sigma_X \cdot r_Y + \sigma_X \cdot d_{YX} & (A2) \end{array}\right.$$

When the index matrix R_Y is a full-row rank matrix, conditions (L1) and (A1) simplify to $U_Y\ \sigma_Y = \sigma_X$. It yields the following result [7].

Corollary 1 *Let R_Y be a full-row rank index matrix. The crossed dependence associated with vectors d_{YX} may be **communication-free** only if the allocation directions ξ_X and ξ_Y are parallel vectors.*

Proof. By definition of an allocation matrix, its row vectors are orthogonal to the corresponding allocation direction. Since U_Y is a unimodular matrix the row vectors of $U_Y\sigma_Y$ are also orthogonal to ξ_Y. Therefore the condition $U_Y\sigma_Y = \sigma_X$ holds if and only if ξ_X and ξ_Y are parallel. □

If the allocation directions ξ_X and ξ_Y are parallel but if neither condition (L2) nor condition (A2) can be satisfied, then the crossed-dependence will impose communications. This communication occurs from the processor defined by $alloc_Y(z_{emit})$ to any processor defined by $alloc_X(z_{emit} + d_{YX})$, i.e. from the processor indexed by $U_Y\ \sigma_Y \cdot z_{emit} + c_Y$ to any processor indexed by $\sigma_X \cdot z_{emit} + \sigma_X \cdot d_{YX}$. Since $U_Y\ \sigma_Y = \sigma_X$ these processors are at a constant distance. Conversely if $U_Y\ \sigma_Y \neq \sigma_X$ (i.e. if ξ_X and ξ_Y are not parallel), then the distance between these processors depends on the emission point. This result is expressed by the following corollary.

Corollary 2 *If ξ_X and ξ_Y are not parallel, then the crossed dependence associated with vectors d_{YX} **requires non constant communications.***

We now examine alignment of affine crossed-dependences and we show how to use theorem 1 in this context.

4.3 Alignment of affine crossed-dependences

Let us first recall that if an affine dependence is characterized by a utilization set of dimension larger than 1, then there is no way to completely localize it. For a bidimensional utilization set, the best we can get is a set of local communications along a line of processors, as already mentioned in section 3.2. Therefore in the following we will restrict ourselves to one-dimensional utilization sets and show how to avoid communications.

Let us consider a one-dimensional affine crossed-dependence from variable Y to variable X. We denote its utilization vector by u_{YX} and its two extremal dependence vectors by $d_{YX_{ex1}}$ and $d_{YX_{ex2}}$.

[7] This result applies in particular to any uniform crossed-dependence since R_Y is the identity matrix.

To avoid communications related to such a crossed dependence we must first guaranty that all the utilization points associated with a given emission point are projected on the same processor. The following theorem gives a necessary condition for such a projection to be possible.

Theorem 2 *The one-dimensional affine crossed dependence from Y to X characterized by utilization vector u_{YX} **may be communication-free** only if ξ_X is parallel to u_{YX}.*

Proof. The projection of each one-dimensional utilization set on a single processor implies :

$$alloc_X(z_{emit} + d_{YX}) = alloc_X(z_{emit} + d'_{YX})$$

for any dependence vectors d_{YX} and d'_{YX}. Since any dependence vector is equal to $d_{YX} = d_{YX_{ex1}} + x\ u_{YX}, x \in \mathbf{Z}$, we have :

$$\begin{array}{l} alloc_X(z_{emit} + d_{YX_{ex1}} + x\ u_{YX}) = \\ \qquad alloc_X(z_{emit} + d_{YX_{ex1}} + x'\ u_{YX}) \end{array}$$

for any integer constants x and x'.

We therefore have $alloc_X(u_{YX}) = \sigma_X \cdot u_{YX} = 0$ where 0 is the null vector of $\mathbf{Z}^{n-1}$. From the definition of σ_X, the allocation direction ξ_X is orthogonal to the row vectors of σ_X. Therefore ξ_X and u_{YX} are parallel. □

If such an allocation direction can be obtained, we still have to avoid a new communication from the emission processor of Y to the unique utilization processor. This can be done either by aligning the emission point on that processor, or by using a communication link that already exists for an auto-dependence of Y. It is realized by applying theorem 1 to one of the extremal dependence vectors $d_{YX_{ex}}$.

5 A heuristic to globally optimize localization

A problem is characterized by a set of dependences that can be both uniform and affine, and both auto and crossed dependences In most cases it is impossible to localize all the dependences. Localization of one dependence may result in communication for another one. Examples 1 and 2 illustrate this situation when the problem has respectively several uniform or affine auto-dependences on the same variable.

To globally optimize localization for a whole problem we must determine the largest possible sets of pairs of result variables such that there exist matrices U_Y and vectors c_Y satisfying theorem 1.

Generally there are several possible solutions to this problem. Therefore it is important to get several sets of efficient allocations and to compare their communications performances.

These solutions can be obtained using the following heuristic.

(1) Consider separately each variable, and determine for each one the allocation direction(s) that optimize their **distribution**, i.e. that minimize the number of communications related to their auto-dependences. Recall that there may be several efficient allocation directions for one variable. Consider for example a variable characterized only by uniform auto-dependences and suppose that the corresponding set $\overline{Util}$ contains three points defining a triangle.

The three directions parallel to the edges of this triangle are then optimal allocation directions resulting in only one communication.

(2) Consider successively all the **affine crossed-dependences** from any variable Y to any variable X. If the utilization set is one-dimensional characterized by utilization vector u_{YX} then choose as allocation direction ξ_X parallel to u_{YX} according to theorem 2. If the utilization set is multi dimensional, then a linear combination of the utilization vectors will partly localize the utilization points (along one dimension).

(3) Consider successively the variables characterized by affine crossed-dependences. From (2) each crossed-dependence imposes one (or several) optimal allocation directions. If this direction is also optimal for the auto-dependences (i.e. given by (1)), then keep it (or them) as the best choice(s). Otherwise keep all the solutions given by (1) and (2) and determine for each of them the number of communications they globally induce for the variable.

(4) From (3) we get a set of possible allocation directions for each variable. Consider all the possible combinations with one allocation direction for each variable. Try to find a unimodular matrix U_Y and a unimodular vector c_Y satisfying either conditions (L) or conditions (A) given in theorem 1 for any pair of variables that have crossed-dependences. If none exists, then the dependences according to these two variables can not be all localized. Determine the allocation functions directly from the initial allocation directions and compute the number of resulting communications. Otherwise determine the allocation functions from U_Y and c_Y.

From this process we will get, in the best cases, one optimal allocation function for each variable. However in most cases we will get a set of solutions, each characterized by remaining communications. We will then have to compare them and compile the corresponding code, after the determination of schedule function for each variable.

Let us now illustrate this approach on an example.

Example 3. Consider the following SARE.

$$a(i,j) = f_1\big(a(i,j-1), a(i-1,j-1), b(j,j)\big) \quad (3)$$
$$b(i,j) = f_2\big(b(i-1,j), b(i-1,1)\big) \quad (4)$$

over the convex domain $D = \{(i,j) \mid 1 \le i \le N,\ 1 \le j \le i\}$.

Variable a is characterized by two uniform auto-dependences respectively defined by uniform dependence vectors $d_{a_1} = (0,1)$ and $d_{a_2} = (1,1)$ (cf. figure 2). From the set $\overline{Util}\big(a,(i_0,j_0)\big) = \{(i_0,j_0),(i_0,j_0+1),(i_0+1,j_0+1)\}$ we deduce that the communications due to the auto-dependences of a are minimized by choosing as allocation direction ξ_a any of the three vectors $(0,1), (1,1)$ or $(1,0)$.

The auto-dependences related to variable b are both uniform and affine. The first argument in equation (4) corresponds to a uniform dependence whose dependence vector is $d_{b_1} = (1,0)$. The second argument defines an affine dependence shown in figure 2 and characterized by :

$$\begin{aligned} &Util_2\big(b,(i_0,j_0)\big) = \{(i_0+1,j) \mid 1 \le j \le i_0+1\}\\ &Emit_2(b) = \{(i,1) \mid 0 \le i \le N-1\}\\ &u_{b_2} = (0,1)\\ &d_{b_2,(i_0,1)} = (1,j),\ 0 \le j \le i_0 \end{aligned}$$

The first dependence is localized if the allocation direction is $\xi_b = d_{b_1} = (1,0)$, while the second argument is localized if $\xi_b = u_{b_2} = (0,1)$ (cf. section 3). Any of these two choices will therefore result in one set of communications.

There is one affine crossed-dependence from variable b to variable a shown in figure 2 and characterized by :

$$\begin{aligned} &Util_{ba}\big(b,(i_0,j_0)\big) = \{(i,j_0) \mid j_0 \le i \le N\}\\ &Emit_{ba}(b) = \{(j,j) \mid 1 \le j \le N\}\\ &u_{ba} = (1,0)\\ &d_{ba,(i_0,i_0)} = (i,0),\ 0 \le i \le N-i_0 \end{aligned}$$

From theorem 2 we deduce that this crossed-dependence will be localized if $\xi_a = u_{ba} = (1,0)$. Since $\xi_a = (1,0)$ is one of the three allocation directions optimizing localization for the auto-dependences of a, we restrict ourselves to this unique most efficient choice for variable a which results in one communication for variable a on each processor.

Let us now consider the combination of $\xi_a = (1,0)$ with one of the two best allocation directions for variable b : $\xi_b = (0,1)$ [8]. The allocation matrices respectively associated with ξ_a and ξ_b are $\sigma_a = (0\ 1)$ and $\sigma_b = (1\ 0)$. Let $alloc_a(i,j) = \sigma_a \cdot (i,j)^t$ and $alloc_b(i,j) = U\ \sigma_b \cdot (i,j)^t + c$. Let us try to find U and c satisfying either conditions (L) or conditions (A) of theorem 1. Notice that since the problem is defined in $\mathbf{Z}^2$, the unimodular matrix U is reduced to an integer (i.e. 1 or -1).

In this example the index matrix and vector related to the crossed-dependence are respectively $R = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$ and $r = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

Therefore conditions (L) simplify to :

$$\begin{cases} U\ \sigma_b\ R = \sigma_a\ R & (\ell 1)\\ c = \sigma_a \cdot d_{ba} & (\ell 2) \end{cases}$$

and conditions (A) simplify to :

$$\begin{cases} U\ \sigma_b\ R = \sigma_a\ R & (a1)\\ U\ \sigma_b \cdot d_b + c = \sigma_a \cdot d_{ba} & (a2) \end{cases}$$

where $d_{b1} = (1,0)$ is the non-localized auto-dependence vector related to variable b.

From conditions $(\ell 1)$ and $(a1)$ we deduce $U = 1$. Condition $(\ell 2)$ implies $c = 0$ while condition $(c2)$ yields $c = -1$.

It is therefore possible to allocate variables a and b to a final array of processors in such a way that the crossed-dependence is communication-free. Moreover this can be achieved either by localizing it (solution obtained from (ℓ_2)) or by aligning it to a non-localized auto-dependence of b (solution obtained from $(a2)$). Both solutions require two links of communications on each processor, one for an auto-dependence of a, the other for the uniform auto-dependence of b.

Once the allocations functions are determined, we must compute the schedule functions associated with each variable. These functions are determined from the auto and the crossed dependences.

[8] The other choice results in the same allocation matrices for both variables and is not discussed here.

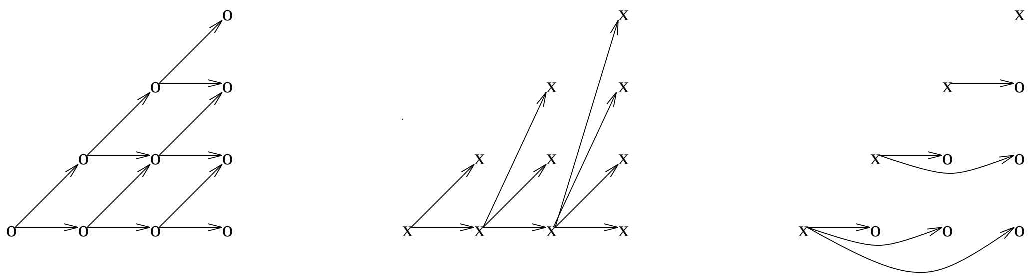


Figure 2: Dependences for example 3

From the causal constraints associated with the auto-dependences of variable a, we deduce that the schedule vector $\lambda_a = (\lambda_{a_1}, \lambda_{a_2})$ must verify : $\lambda_{a_2} > 0$ and $\lambda_{a_1} + \lambda_{a_2} > 0$. Moreover it must guaranty that $\lambda_a \cdot \xi_a \neq 0$. This condition implies $\lambda_{a_1} \neq 0$. The most efficient schedule vector for variable a is therefore $\lambda_a = (1, 1)$.

The causal constraints related to the auto-dependences of variable b imply that the schedule vector $\lambda_b = (\lambda_{b_1}, \lambda_{b_2})$ must verify : $\lambda_{b_1} > 0, \lambda_{b_1} + x\lambda_{b_2} > 0, \forall x \geq 0$. Moreover the condition $\lambda_b \cdot \xi_b \neq 0$ yields $\lambda_{b_2} \neq 0$. The most efficient schedule vector for variable b is therefore $\lambda_b = (1, 1)$.

The causal constraints related to the crossed-dependence from b to a are defined by $t_b(z) < t_a(z+d_{ba})$, i.e. $\lambda_b \cdot z + \alpha_b < \lambda_a \cdot z + \lambda_a \cdot d_{ba} + \alpha_a$ for all the dependence vectors d_{ba}. Since $\lambda_a = \lambda_b$ it simplifies to $\alpha_b < \lambda_a \cdot d_{ba} + \alpha_a$ for any crossed-dependence vector $d_{ba} = (i, 0), 0 \leq i \leq N - i_0$. This implies $\alpha_b < \alpha_a$. We choose $\alpha_b = 0$ and $\alpha_a = 1$.

To conclude, the two solutions are scheduled by the same functions :

$$\begin{aligned} t_a(i,j) &= i+j+1 \\ t_b(i,j) &= i+j \end{aligned}$$

and have the following allocation functions :

Solution 1 (localization) : $\begin{aligned} alloc_a(i,j) &= j \\ alloc_b(i,j) &= i \end{aligned}$

and

Solution 2 (alignment) : $\begin{aligned} alloc_a(i,j) &= j \\ alloc_b(i,j) &= i-1 \end{aligned}$

The resulting architectures are linear arrays of processors. They have respectively N processors for the first solution and $N+1$ processors for the second one. $\square$

This heuristic still leaves several "ad hoc" decisions to be made and cannot directly be implemented in a practical compiler. Further work needs to be done to automate this process and extend it from loop nests to real programs.

6 Conclusion

The alignment and communication minimization problems have focused a lot of attention over the past few years. In [2] J.M. Anderson and M. Lam deal the problem of aligning data and computations in order to avoid nonlocal communications. They propose a decomposition algorithm based on three components. In [20] W. Shang and Z. Shu adress this problem in the context of single loop nests and propose necessary and sufficient conditions for the existence of mappings without nonlocal communications.

In this paper we deal with affine dependences and show how the dependence analysis based on the notion of utilization sets and vectors allows us to avoid non local communications.

In [5] A. Darte and Y. Robert introduce the notion of communication graph to synthesize the dependence information characterizing affine loop nests. They show that the problem of communication minimization is NP-complete in the simpler case of uniform loop nests and they propose in this case a heuristic to minimize communications. This approach is extended in [7] where M. Dion and Y. Robert introduce the notion of access graph which differs from the communication graph by labelling the edges by the corresponding index matrices. They show that the communication minimization problem of affine loop nests is NP-complete and give a heuristic for communication optimization in this case.

In our work, the localization optimization results and the heuristic are based on two types of dependences : the auto-dependences and the crossed-dependences. The process of communications minimization is based on two phases : the distribution of the result variables according to their auto-dependences, followed by the alignment of the different arrays to a single array of processors. This alignment is realized according to the crossed-dependences and is based on the use of unimodular matrices applied to the allocation matrices : this explicitely defines the allocation functions.

This paper focuses on communications minimization for result variables. The ideas presented here extend to the alignment of data variables using the information synthesized in the utilization sets and vectors.

The localization optimization presented here applies with broadcast-free schedules. However a SARE contains affine dependences that define multiple-use of data. If these (or some of these) multiple-uses of a given variable may be scheduled at the same time step using broadcast communications primitives, then the global schedule of the SARE is generally faster than the most efficient broadcast-free schedule. It is therefore interesting to tackle the problem of communications minimization in the general context of point-to-point and broadcast communications. The ideas introduced in this paper can be extended to this more general study as it has already been introduced in [13].

References

[1] S.P. Amarasinghe, J.M. Anderson, M.S. Lam, and C.W. Tseng. The suif compiler for scalable parallel machines. In *Proceedings of the seventh SIAM Conference on Parallel Processing for Scientific Computing*, Feb 1995.

[2] J.M. Anderson and M.S. Lam. Global optimizations for parallelism and locality on scalable parallel machines. *ACM Sigplan Notices*, 28(6):112–125, 1993.

[3] U. Banerjee. An introduction to a formal theory of dependence analysis. *Journal of Supercomputing*, 2:133–149, 1988.

[4] U. Banerjee, R. Eigenman, A. Nicolau, and A. Padua. Automatic program parallelization. *Proceedings of the IEEE*, 81(2):211–243, 1993.

[5] A. Darte and Y. Robert. A graph-theoretic approach to the alignment problem. Technical Report 93-20, LIP-ENS Lyon, France, 1993.

[6] E. D'Hollander. Partitioning and labeling of loops by unimodular transformations. *IEEE Transactions on parallel and distributed systems*, 3(4):465–476, 1992.

[7] M. Dion and Y. Robert. A graph-theoretic approach to the alignment problem. Technical report, LIP-ENS Lyon, France, 1994.

[8] M. Dowling. Optimal code parallelization using unimodular transformations. *Parallel Computing*, 16:157–171, 1990.

[9] M. Gupta and P. Banerjee. A methodology for high-level synthesis of communication for multicomputers. In *6th ACM Int. Conf. on Supercomputing*, 1992.

[10] C.H. Huang and P. Sadayappan. Communication-free hyperplane partitioning of nested loops. In Banerjee, Gelernter, Nicolau, and Padua, editors, *Languages and Compilers for Parallel Computing*, pages 186–200. Springer Verlag, 1991. LNCS No 589.

[11] F. Irigoin and R. Triolet. Dependence approximation and global parallel code generation for nested loops. In M. Cosnard et al, editor, *Int. Workshop on parallel and distributed algorithms*, pages 297–308. Elsevier, 1989.

[12] D.J. Kuck, R.H. Kuhn, B.R. Leasure, and M.J. Wolfe. The structure of an advanced retargetable vectorizer. In *Supercomputers : design and applications tutorial*, pages 967–974. 1984.

[13] C. Mongenet. Data compiling for systems of affine recurrence equations. In *IEEE Int. Conference on Application Specific Array Processors, ASAP'94*, pages 212–223. IEEE Computer Society Press, 1994.

[14] C. Mongenet. Data compiling for systems of uniform recurrence equations. *Parallel Processing Letters*, 4(3):245–257, 1994.

[15] C. Mongenet, Ph. Clauss, and G.R. Perrin. Geometrical tools to map systems of affine recurrence equations on regular arrays. *Acta Informatica*, 31:137–160, 1994.

[16] C. Mongenet and G.R. Perrin. Synthesis of systolic arrays for inductive problems. In *Conference on Parallel Architectures and Languages Europe, PARLE'87*, pages 260–277. Springer Verlag, 1987. LNCS 259.

[17] J. Ramanujam. Non-unimodular transformations of nested loops. In *Supercomputing 92*, pages 214–223, 1992.

[18] J. Ramanujam and S. Sadayappan. Compile-time techniques for data distribution in distributed memory machines. *IEEE Transactions on parallel and distributed systems*, 2(4):472–482, 1991.

[19] V. Roychowdhury, L. Thiele, S.K. Rao, and T. Kailath. On the localization of algorithms for vlsi processor arrays. In *IEEE VLSI signal processing III*, 1988.

[20] W. Shang and Z. Shu. Data alignment of loop nests without nonlocal communications. In *IEEE Int. Conference on Application Specific Array Processors, ASAP'94*, pages 439–450. IEEE Computer Society Press, 1994.

[21] M. Wolf and M. Lam. A loop transformation theory and an algorithm to maximize parallelism. *IEEE Transactions on parallel and distributed systems*, 2(4):452–471, 1991.

[22] H.P. Zima, H.J. Bast, and H.M. Gerndt. Superb ; a tool for semi-automatic mimd/simd parallelization. *Parallel Computing*, 6:1–18, 1988.