DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Relatório Técnico

RT-MAC-9605

Cycle shrinking by dependence reduction

Kunio Okuda

Maio 96

Cycle shrinking by dependence reduction*

Kunio Okuda
Universidade de São Paulo
Instituto de Matemática e Estatística
Departamento de Ciência da Computação
Rua do Matão, 1010
CEP 05508-900 São Paulo, SP, Brazil
Tel. + 55 11 818 6135 Fax + 55 11 814 4135
email: kunio@ime.usp.br

Abstract

We present a new simple cycle shrinking technique called *dependence reduction*. It consists of a transformation of the dependence graph to reduce the number of execution steps as well as the communication between processors. Compared to the well-known GSS method (*Generalized Selective Cycle Shrinking*) the proposed method presents the advantage of a simpler analysis and better result. Comparison with other well known methods are also shown through illustrative examples. We gave simple and sufficient conditions for the dependence reduction method to be better than other methods

Key-words: cycle shrinking, dependence reduction, loop parallelization, data dependence analysis, scheduling.

1 Introduction

In parallel computing, nested loop structures offer rich implicit parallelism. Several techniques based on loop transformation, called *cycle shrinking*, to

*Supported by FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) - Proc. No. 94/4544-2, 95/0767-0, and CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) - PROTEM-2-TCPAC Proc. No. 680060/94-4, and Commission of the European Communities through Project ITDC-207.

extract parallelism in loops are known [9, 13]. *Simple cycle shrinking, selective cycle shrinking* and *true dependence cycle shrinking* were introduced by Polychronopoulos [8]. These methods transform sequential loops into parallel loops. A generalization of selective cycle shrinking is *generalized selective cycle shrinking (GSS))* [10, 12]. *Index shift method (ISM)* was introduced by Liu, Ho and Sheu [5]. It can be viewed as a refinement of GSS. Robert and Song proposed a method that combines GSS with ISM [10]. *Affine by Statement* was proposed by Robert and Darte [2, 3, 4]. In this paper we propose a new cycle shrinking technique that transforms the dependence graph, with the goal of reducing the number of communication between the processors and the number of computing steps. It identifies the essential dependencies and allows a simpler scheduling analysis. This paper is organized as follows. In section 2 we define the terminology and discuss GSS. In section 3 we present the new technique through two examples. We compare the results with other methods by using the well-known example from Peir and Cytron [7]. In section 4 we formalize the new technique. Finally we conclude in section 5.

2 Terminology and preliminary results

To facilitate and simplify the following presentation, we make some restriction to the class of perfect nested loop algorithms. Also we adopt simple scheduling and mapping models. For example we will use GSS for scheduling.

2.1 RUN(*Regular Uniform Nest*) algorithm model

for $i_1 = 0$ to N_1 do
    for $i_2 = 0$ to N_2 do
        .
        .
        for $i_n = 0$ to N_n do
            command S_1
            .
            .
            command S_k

where $N_1, \ldots, N_n$ are constant. The set of indices for this algorithm is defined as:

$$Dom = \{I = (i_1, \ldots, i_n) \mid 0 \leq i_j \leq N_j, 1 \leq j \leq n\}$$

We use the definition of dependence and dependence vector according to Banerjee and Polychronopoulos [1, 9, 14]. In RUN the dependence vector between two commands independs on the indices of the particular instance. The importance of uniformity is due to the following two main reasons. 1. Many algorithms for scientific applications have this structure. 2. Its regular structure allows the exploitation of implicit parallelism.

Example 1

for $i = 0$ *to* N *do*
  for $j = 0$ *to* N *do*
    command S_1: $a(i,j) = b(i, j-6) + e(i-1, j+3)$
    command S_2: $b(i+1, j-1) = c(i+2, j+5)$
    command S_3: $c(i+3, j-1) = a(i, j-2)$
    command S_4: $e(i, j-1) = a(i, j-1)$

For this example the set of indices is
$Dom = \{(i,j) \in Z^2 | 0 \leq i, j \leq N\}$.
We have five dependence vectors:

$S_1 \rightarrow S_3 : d_1 = (0,2)$ $S_3 \rightarrow S_2 : d_2 = (1,-6)$ $S_2 \rightarrow S_1 : d_3 = (1,5)$
$S_1 \rightarrow S_4 : d_4 = (0,1)$ $S_4 \rightarrow S_1 : d_5 = (1,-4)$

We have the following dependence matrix:
$D = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 2 & -6 & 5 & 1 & -4 \end{pmatrix}$ We have two dependence cycles as shown by the dependence graph of Figure 1.

2.2 Scheduling

Given a RUN, scheduling is a function $F : Z^n \rightarrow Z$ such that the computation $(i_1, \ldots, i_n)$ is executed at step $F(i_1, \ldots, i_n)$ [2]. For a function $F : Z^n \rightarrow Z$ to be a scheduling, it must satisfy the following condition:
If $S_v(j_1, \ldots, j_n)$ depends on $S_u(i_1, \ldots, i_n)$ then $F(i_1, \ldots, i_n) < F(j_1, \ldots, j_n)$.
The parallel execution of a RUN is the following:

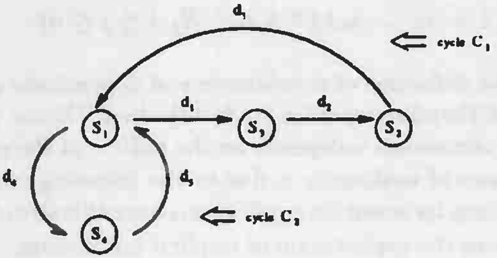

Figure 1: Dependence graph of Example 1

for t = *timemin* to *timemax* do
    execute all $(i_1, \ldots, i_n) \in Dom$ such that $F(i_1, \ldots, i_n) = t$

The total number of steps will be $timemax - timemin + 1$. We use typically the scalar product for scheduling, as in GSS, to be seen later.

2.3 Mapping

Given a RUN, mapping is a function $G : Z^n \rightarrow Z^m$ such that the computation $(i_1, \ldots, i_n)$ is executed at the processor $G(i_1, \ldots, i_n)$. If $F(i_1, \ldots, i_n) = F(j_1, \ldots, j_n)$ then $G(i_1, \ldots, i_n) \neq G(j_1, \ldots, j_n)$ so that the computations scheduled at the same step be mapped to different processors. The typical mapping is a projection along a vector and in this case $m = n - 1$. When *Dom* is projected to Z^{n-1}, $G(Dom)$ will be a network of processors and the projected dependence vectors will represent communication between processors in the network. The exception is when the dependence vectors are parallel to the projection vector. These vectors will not represent communication because the data will be in the same processor.

Let

Comm= unit of time for communication between processors
Comp= unit of time to compute a command
T= total number of steps in parallel execution

Then the total time will be in general $T \times (Comp + Comm)$. However, if all the dependence vectors are parallel to the projection vector, then the total time will be only $T\ Comp$.

2.4 The GSS method

The GSS method is a generalization of the *selective cycle shrinking* method used in a parallelizing compiler [12].

Consider a RUN with dimension n. Let $D = (d_1, \ldots, d_l)$ be a dependence matrix $n \times l$. Let $\pi = (\pi^1, \ldots, \pi^n)$ be a vector such that $\pi \cdot D > 0$ and $\gcd(\pi^1, \ldots, \pi^n) = 1$.

π will be denoted *scheduling vector*. Let the reduction factor be $disp(\pi) = min\{\pi \cdot d_j | 1 \leq j \leq l\}$. All the points I and J in *Dom* that are on the same hyperplane perpendicular to vector π, i.e. $\pi \cdot I = \pi \cdot J$, will be executed simultaneously at step $\lfloor \frac{\pi \cdot I}{disp(\pi)} \rfloor (= \lfloor \frac{\pi \cdot J}{disp(\pi)} \rfloor)$. Such hyperplanes will be called *time hyperplanes*. Find π_0 that minimizes $GSS(\pi) = \frac{max\{\pi \cdot I - \pi \cdot J | I, J \in Dom\}}{disp(\pi)}$.

2.5 Explicit domain

To show the dependences in *Dom* explicitly we will use the following definition:

ED= explicit domain $= \{S_1, \ldots, S_k\} \times Dom$

The definition of the dependence vector will also change:

$S_i \rightarrow S_j : d = (d^1, \ldots, d^n)$ becomes $d = (S_j - S_i, d^1, \ldots, d^n)$

Thus for example 1 we have:

$ED = \{(S_h, i, j) | 1 \leq h \leq 4, 0 \leq i, j \leq N\}$

d_1 becomes $(S_3 - S_1, 0, 2)$.

and d_2 becomes $(S_2 - S_3, 1, -6)$.

For its representation ED will always be identified as a subset of R^{n+1}. Also, each $S_h \times Dom$ will be identified as subset of $\{h\} \times Dom$ and all the points of ED will be connected by dependence vectors explicitly.

This representation is similar to the *Augmented Dependence Graph* (ADG) proposed by Kyriakis-Bitzaros and Goutis [6], but it is simpler. For loops of dimension n with k commands, the dimension of ADG is $n + k + 1$ while the dimension of ED is always $n + 1$, independent of k. This facilitates the visual representation. In the case of a two-dimensional RUN, we can project each $\{S_h\} \times Dom$ to R^2 with each point slightly dislocated in relation to the other $\{S_{\overline{h}}\} \times Dom$, $\overline{h} \neq h$, to avoid superposition. In this way, all the dependences will be shown explicitly in R^2. This representation will illustrate the idea of dependence reduction. Throughout this paper we use *Dom* and ED depending on the convenience.

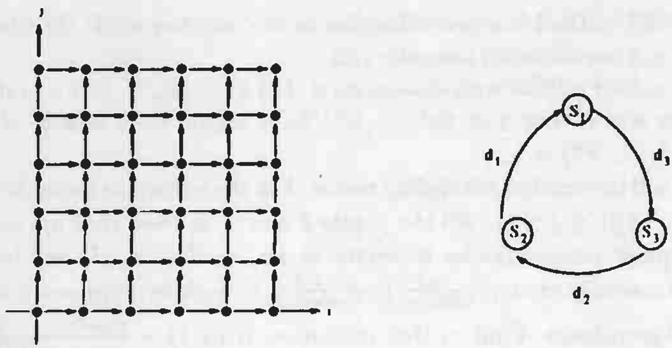

Figure 2: *Dom* and the dependence graph of Example 2

3 Examples

3.1 First case

This first example is very simple and serves to illustrate the usefulness of explicit domain *ED*. (See Figure 2.)

Example 2

for $i = 0$ *to* N *do*
  for $j = 0$ *to* N *do*
    S_1: $a(i,j) = f(b(i-1,j))$
    S_2: $b(i,j) = g(c(i,j-1))$
    S_3: $c(i,j) = h(a(i-1,j))$

Instead of using *Dom* we now consider $ED = \{S_1, S_2, S_3\} \times Dom$. The two-dimensional representation of *ED* with its explicit dependence vectors are shown in Figure 3.

Observe that we have in Figure 3 several "zig-zags" of dependences that do not interfere with one another. This gives us more freedom to do the scheduling.

For example, consider the "zig-zag" constituted by cycle $S_1(2,3) \rightarrow S_3(3,3) \rightarrow S_2(3,4) \rightarrow S_1(4,4)$. The dependences involve different commands. Now let us forget for the moment their location in *ED*, and instead

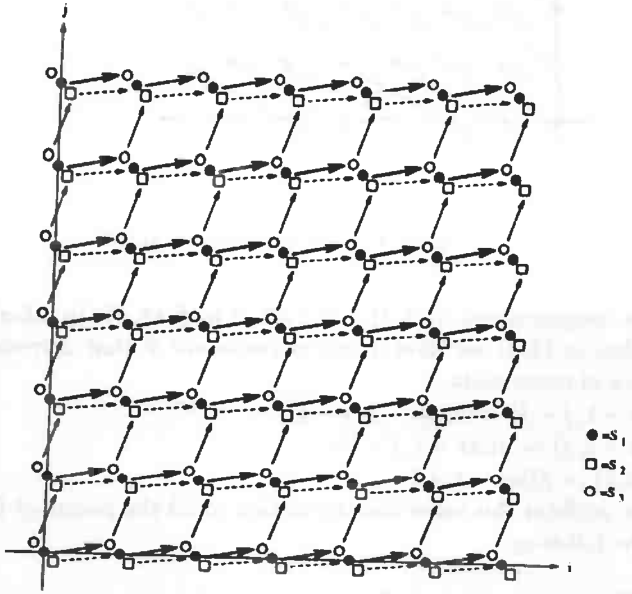

Figure 3: DE for Example 2

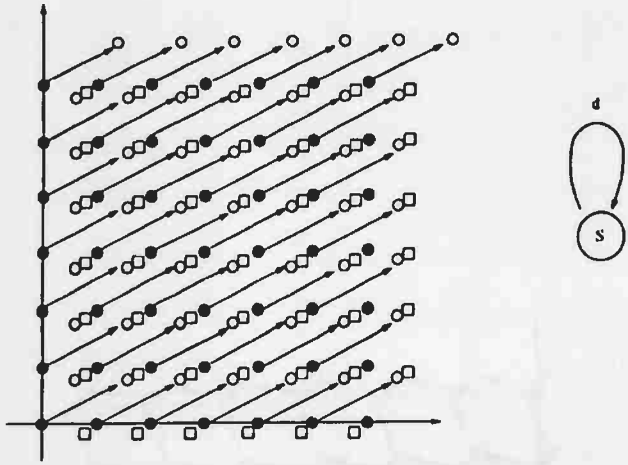

Figure 4: New dependence graph

join the computations $S_3(3,3)$ and $S_2(3,4)$ to $S_1(4,4)$. In other words, consider that in $(4,4)$ we have a *macro command* S that corresponds to the sequence of commands:
$S_3 : c(i-1, j-1) := h(a(i-2, j-1))$
$S_2 : b(i-1, j) := g(c(i-1, j-1))$
$S_1 : a(i, j) := f(b(i-1, j))$
Now we perform this same transformation to all the points of ED. We will have the following.

1. The macro command S will be mapped to a processor.
2. The macro command S will take more time since it now involves in fact the execution of three commands.
3. The dependence vector of the macro command S will be simpler: $d = d_1 + d_2 + d_3 = (1,0) + (0,1) + (1,0) = (2,1)$ (see Figure 4). In other words, the dependence vectors of a cycle are reduced into one single vector, thus the name *dependence reduction*.

From Figure 4 it is easy to observe that the total number of steps required is $N/2$.

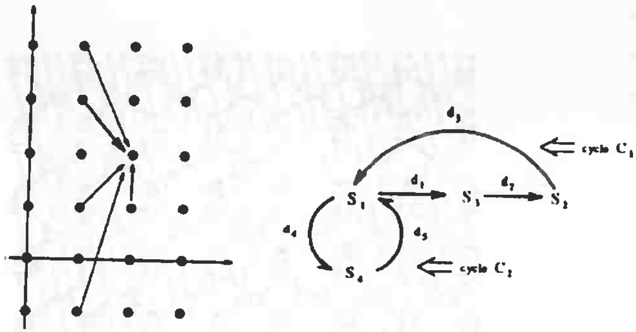

Figure 5: *Dom* and the dependence graph of Example 3

Each step consists of the computation of three functions and one communication (if it exists).

Since we can project along vector d, the time will be $\frac{3N}{2}$ *Comp.*

3.2 Second case

In the second example *ED* gives some intuitive base for the dependence graph transformation to be shown in next section.

Example 3

for $i = 0$ *to* N *do*
    for $j = 0$ *to* N *do*
        $S_1 : a(i,j) = f(b(i-1,j-3) + d(i-1,j+2))$
        $S_2 : b(i,j) = g(c(i-1,j+1))$
        $S_3 : c(i,j) = h(a(i-1,j-1))$
        $S_4 : d(i,j) = k(a(i,j-1))$

Dom and the dependence graph are in Figure 5.

We construct $ED = \{S_1, \ldots, S_4\} \times Dom$ and project the four planes to R^2. We obtain Figure 6.

Figure 6 is decomposed into Figure 7 and 8 that show the dependences of the cycles C_1 and C_2. Figure 9 shows the dependences in relation to $S_1(i,j)$ in particular.

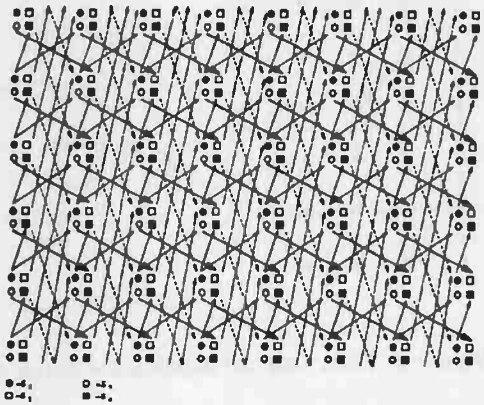

Figure 6: DE for Example 3

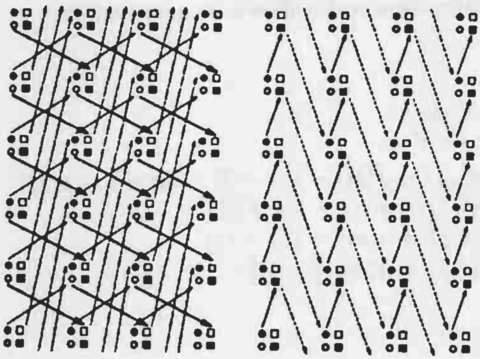

Figure 7: Dependences of cycle C_1

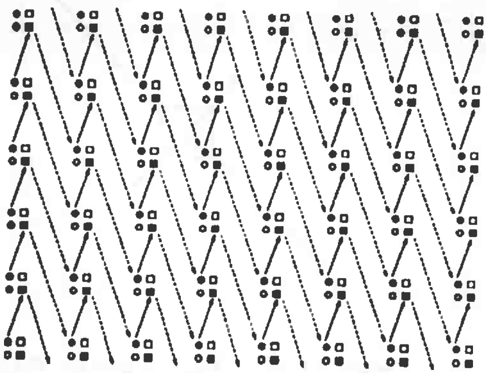

Figure 8: Dependences of cycle C_2

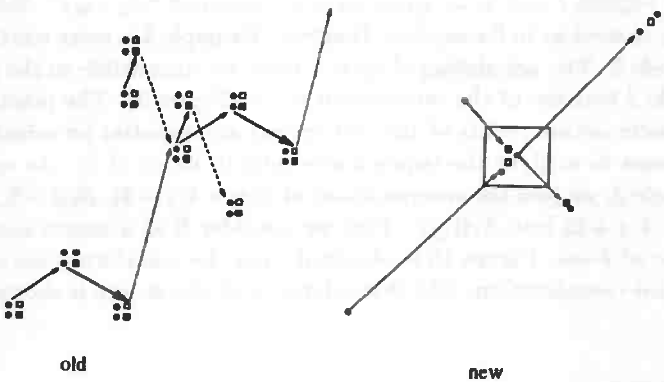

Figure 9: Dependences for $S_1(i,j)$

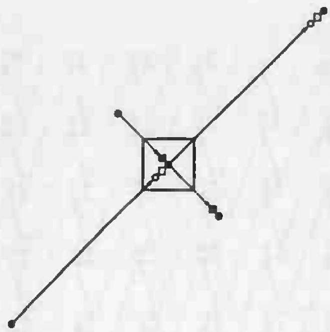

Figure 10: New dependence for $S_1(i,j)$

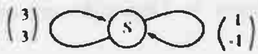

Figure 11: New dependence graph for Example 3

In Figures 7 and 8, we again have independent "zig-zags". Each of these can be treated as in Example 2. However, Example 3 is more restrictive than Example 2. The scheduling of cycle 1 must be compatible to the scheduling of cycle 2 because of the intersection point (Figure 9). The positions of the S_1's (intersection points of the two cycles) are essential for scheduling. We thus want to analyze the dependences only in terms of S_1. As was done in Example 2, we join the computations of $S_2(i-1,j-3)$, $S_3(i-2,j-2)$ and $S_4(i-1,j+2)$ into $S_1(i,j)$. Thus we consider S as a macro command for a point of *Dom*. Figure 10 is obtained from the transformation of Figure 9 with this consideration. The dependence with this macro is shown in Figure 11.

Observations

1. Both in Examples 2 and 3 the macros for the points situated on the border of the domain are incomplete (for example, in Figure 4 for Example 2, the macros on the right border are composed only of S_2 and S_3).

Figure 12: New dependence graph for Example 1

2. For the resultant network of processors, the creation of the macro command means the transfer of some inter-processor communication into the same processor. This contributes to the reduction of the total time spent. In the next section we treat this aspect in more details.

3. The number of dependence vectors is reduced after the transformation of the dependence graph through *dependence reduction*. This results in a reduction of communication between processors. For example, in Example 2, instead of three communications for a, b, and c we now have only one for a. The communication for b and c are now "hidden" in the processors.

3.3 Third case

Let us now go back to Example 1, the well-known example used in many papers (e.g. [2, 3, 10]). Notice that, with the exception of some indices, Example 1 is practically equal to Example 3, They present the same cycles (right side of Figure 5) with different dependences. By applying *dependence reduction* in a similar way to Example 3, we have the graph of Figure 12.

For this graph $\pi_0 = (4, -1)$ is the optimal solution for GSS with the number of steps equal to $\frac{5}{7}N$. Thus the total time will be $\frac{5}{7}(NComm + 4NComp)$.

The time spent by GSS is $8N(Comm + Comp)$. The time by GSS combined with ISM is $2N(Comm + Comp)$ [10]. On the other hand, the time spent by *affine by statement* is $\frac{12}{7}N(Comm + Comp)$ (see [3] which also shows the necessity of at least one communication).

The following table summarizes the results of the several methods for Example 1.

	Computation	Communication
GSS	$8NComp$	$8NComm$
GSS combined with ISM	$2NComp$	$2NComm$
Affine by Statement	$\frac{12}{7}NComp$	$\frac{12}{7}NComm$
Dependence reduction	$\frac{20}{7}NComp$	$\frac{5}{7}NComm$

We conclude that, if $Comm > \frac{8}{7}Comp$, then *dependence reduction* presents the best time of all.

4 Dependence reduction

In this section we formalize the transformations performed in the examples of the previous section. The case of the dependence graph with only one cycle is trivial. We discuss the case of dependence graph with more than one cycle.

4.1 Dependence graph with more than one cycle

Let A be a RUN algorithm. Let $G = (V, E)$ be the dependence graph for A where V and E correspond to the set of commands $S_1, \dots S_m$ and to the dependence between commands, respectively. Each edge of E is labeled by its dependence vector. We use the notation $u \xrightarrow{d} v$ to denote the edge from node u to node v with dependence vector d. We will divide V into two sets: the set of secondary nodes (VS) and the set of principal nodes (VP).

VS={$v \in V|$ v has exactly one edge entering the node and one edge leaving it }

$VP = V - VS$

See Example 3 (Figure 5). For this example, $VS = \{S_2, S_3, S_4\}$ and $VP = \{S_1\}$.

Let $\overline{G} = (\overline{V}, \overline{E})$ be a transformed graph in which $\overline{V} = VP$ and $\overline{E}$ is defined as follows:

$\overline{E}$={ $v \xrightarrow{d} v'|v, v' \in VP$ and in G there exists a path between v and v' whose intermediate nodes all belong to VS and d is the sum of the dependence vectors of this path }.

The commands corresponding to the secondary nodes in this path will be incorporated into a macro command of v'. Clearly if we have another path of this type for v' then the commands corresponding to the secondary nodes in this path will be incorporated too. We denote this transformation by the name of *dependence reduction*. After obtaining $\overline{G}$ by *dependence reduction*, we apply the GSS method to obtain $\overline{\pi_0}$ that minimizes $GSS(\pi)$ using the dependences of $\overline{G}$. Let T be the number of steps for $\overline{G}$. Then the total time for the original algorithm will be $T\ Comm + (l+1)T\ Comp$ where l is the maximum number of secondary nodes that were incorporated into one principal node.

Observations

The process of incorporating secondary nodes into principal nodes reduces the number of dependence vectors and, consequently, after applying mapping, reduces the communications between processors. The basic idea of the transformation is the following. To find a good scheduling, what we really care is the dependence between principal nodes. The dependences between secondary nodes, as well as those between a secondary and a principal node, are not important. In other words, the location of the points P of ED corresponding to these nodes ($P \in S \times Dom$ where $S \in VS$) are not important.

Another advantage of this new method is the following. The reduction of the number of dependence vectors facilitates the application of the GSS method which, in general, is very complex [11]. With this reduction it may even be possible to perform the computation by hand.

Naturally we can question the proposed method: "The proposed method intends to reduce the communication time between processors, increasing however the computation time. The total time can even increase." In a future work, we show a new method, to be called ***partial dependence reduction***, that pursues balance between communication and computation.

4.2 Effect of the transformation on the choice of π

Let G be the original dependence graph and $\overline{G}$ the transformed graph according to section 4.1.

Let π_0 be the vector that minimizes $GSS(\pi) = \frac{max\{\pi \cdot I - \pi \cdot J \mid I, J \in Dom\}}{min\{\pi \cdot d_i \mid d_i \in D\}}$ in G. We want to answer the question whether π_0 can serve as a scheduling vector also for $\overline{G}$.

As π_0 is a scheduling vector for G, we have $\gcd(\pi_0^1, \ldots, \pi_0^n) = 1$ and $\pi_0 \cdot d_i > 0$ for $\forall d_i \in D$. On the other hand, each $\overline{d_j}$ is the sum of some d_i's of D. Thus $\pi_o \cdot \overline{d_j} > 0$, for $\forall \overline{d_j} \in \overline{D}$ and therefore π_0 is a scheduling vector for $\overline{G}$.

Let $\overline{\pi_0}$ be the vector that minimizes $\overline{GSS}(\pi) = \frac{max\{\pi \cdot I - \pi \cdot J \mid I, J \in Dom\}}{min\{\pi \cdot d_i \mid d_i \in \overline{D}\}}$ em $\overline{G}$.

Consider the questions: What is the relation between π_0 and $\overline{\pi_0}$? What is the value of $\overline{GSS}(\pi_0)$?

We show that $\overline{GSS}(\pi_0) \leq GSS(\pi_0)$. That is, the number of steps of the dependence reduction method is *never* greater to that of the GSS method.

Without loss of generality, let d_1 be the dependence vector for which $\pi_0 \cdot d_1$ is minimum between $\pi_0 \cdot d_i$, for $d_i \in D$. Therefore $\pi_0 \cdot d_1 \leq \pi_0 \cdot d_i, \forall d_i \in D$ and $\pi_0 \cdot d_1 \leq \pi_0 \cdot \overline{d_j}$, for $\forall \overline{d_j} \in \overline{D}$. In this way we have $\overline{GSS}(\pi_0) \leq GSS(\pi_0)$ and, *a fortiori*, $\overline{GSS}(\overline{\pi_0}) \leq GSS(\pi_0)$.

On the other hand, the bad case of $\overline{GSS}(\pi_0) = GSS(\pi_0)$ will only occur if $d_1 = \overline{d_1}$ where $\overline{d_1}$ is the vector that minimizes $\pi_0 \cdot \overline{d_j}$ for $\overline{d_j} \in \overline{D}$. Even in this case there is chance of finding $\overline{\pi_0}$ such that $\overline{GSS}(\overline{\pi_0}) > \overline{GSS}(\pi_0)$, since we have less restrictions to search for $\overline{\pi_0}$ than π_0.

4.3 Applicability

To see when it is advantageous to apply the dependence reduction method, we compare a "*floor*" of the total time spent in other methods with a "*ceiling*" of the total time of the dependence reduction method. This "*floor*" is given by the longest path in ED. If the length of this path is T_f, then, except the case with no communication, the total time is always larger or equal than $T_f(Comm + Comp)$. As for the "*ceiling*", the following fact ensures that we have a good chance of computing such a value easily.

Fact

The dependence vectors before reduction are all lexicographically positive. When we perform dependence reduction, we sum lexicographically positive vectors. Therefore we increase the possibility of having a direction along which all the resulting dependence vectors present positive elements. If such a direction exists, then we can parallelize along this direction.

Let us see an example.

Example 4

for $i = 0$ *to* N *do*
  for $j = 0$ *to* N *do*
    for $k = 0$ *to* N *do*
      command S_1: $a(i,j,k) = f_1(d(i-1,j-1,k))$
      command S_2: $b(i,j,k) = f_2(a(i,j-1,k-1))$
      command S_3: $c(i,j,k) = f_3(d(i,j,k-1))$
      command S_4: $d(i,j,k) = f_4(b(i,j,k-1), c(i,j-2,k+1))$

$S_4 \rightarrow S_1$: $d_{11} = (1,1,0)$ $S_1 \rightarrow S_2$: $d_{12} = (0,1,1)$ $S_2 \rightarrow S_4$: $d_{13} = (0,0,1)$
$S_4 \rightarrow S_3$: $d_{21} = (0,0,1)$ $S_3 \rightarrow S_4$: $d_{22} = (0,2,-1)$

The longest path will have $\frac{3N}{2}$ steps. This is the number of steps necessary to complete the cycles $S_4 \rightarrow S_1 \rightarrow S_2 \rightarrow S_4$ along direction j or direction k. Therefore a "*floor*" will be $\frac{3N}{2}(Comm + Comp)$.

On the other hand, when we perform dependence reduction, we have: $\overline{d_1} = d_{11} + d_{12} + d_{13} = (1,1,0) + (0,1,1) + (0,0,1) = (1,2,2)$ and $\overline{d_2} = d_{21} + d_{22} = (0,0,1) + (0,2,-1) = (0,2,0)$.

Both vectors have positive elements along direction j and the minimum of these elements is 2. By parallelizing along this direction, the number of steps will be $\frac{N}{2}$. Thus a "*ceiling*" will be $\frac{N}{2}(Comm + 4Comp)$.

By comparing the "*floor*" and the "*ceiling*", we have $\frac{N}{2}Comm + \frac{4N}{2}Comp < \frac{3N}{2}Comm + \frac{3N}{2}Comp$. We can conclude that the dependence reduction method will be guaranteed to be better if $Comp < 2Comm$.

We emphasize that the comparison gives only a sufficient condition. The fact that the inequality is not satisfied or impossibility to obtain a "*ceiling*" does not imply that the dependence reduction method will be disadvantageous.

5 Conclusion

We have presented a new simple technique for cycle shrinking called dependence reduction. It identifies and distinguishes, in the dependence graph, the nodes corresponding to crucial commands and the nodes corresponding to non crucial commands for scheduling. Based on such information, this technique defines efficiently macro commands in the processors, with the purpose of reducing the number of execution steps and number of communication steps between processors. The new technique also simplifies substantially the application of the GSS method, due to the reduction of the number of dependence vectors. A comparison with other methods has been done by using the same example, showing its efficiency and simplicity. Finally we gave simple and sufficient conditions for the dependence reduction method to be superior to other methods. We are developing a generalized dependence reduction method which is applicable to a dependence graph constituted only of principal vertices.

Acknowledgment

The author wishes to thank Prof. Siang Wun Song for suggestions, Prof. Carlos Eduardo Ferreira for his help and the members of GCPD/DCC/IME/USP for the encouragement.

References

[1] Banerjee, U. An introduction to a formal theory of dependence analysis. *J. Supercomput.* 2(1988) 133-149.

[2] Darte, A. and Robert, Y. Scheduling uniform loop nests. Technical Report, Laboratoire de l'Informatique du Parallélisme-IMAG, Lyon, 1992.

[3] Darte, A. and Robert, Y. Mapping uniform loop nests onto distributed memory architectures. *Parallel Computing* 20(1994) 679-710.

[4] Darte, A. and Robert, Y. Affine-by-statement scheduling of Uniform and affine loop nests over parametric domains. *Journal of Parallel and Distributed Computing* 29, 43-59(1995).

[5] Liu, L. S., Ho, C. W. and Sheu, J. P. On the parallelism of nested for-loops using index shift method. *Proc. Internat. Conf. on Parallel Processing* (Aug. 1990) II-119-II-123.

[6] Kayriakis-Bitzaros, E. D. and Goutis, C. E. An efficient decomposition technique for mapping nested loops with constant dependencies into regular processor array. *Journal of Parallel and Distributed Computing* 16, 258-264(1992).

[7] Peir, J. K. and Cytron, R., Minimum distance: a method for partitioning recurrence for multiprocessors. *IEEE Trans. Comput.* 38(8) (Aug. 1989) 1203-1211.

[8] Polychronopoulos, C. D. Compiler optimization for enhancing parallelism and their impact on architecture design. *IEEE Trans. Comput.* 37(8) (Aug. 1988) 991-1004.

[9] Polychronopoulos, C. D. *Parallel Programming and Compilers.* Kluwer Academic Publishers, 1988.

[10] Robert, Y. and Song, S.W. Revisiting cycle shrinking. *Parallel Computing*, 18(1992) 481-496.

[11] Shang, W. and Fortes, J. A. B. Time optimal linear schedules for algorithms with uniform dependencies. *IEEE Trans. Comput.* 40(6) (Jun. 1991) 723-742.

[12] Shang, W., O'Keefe, M. T. and Fortes, J. A. B. Generalized cycle shrinking. *Parallel Algorithms and VLSI Architecture II.* P. Quinton and Y. Robert (editors), North Holland, 1991.

[13] Wolfe, M. *Optimizing Supercompilers for Supercomputers.* MIT Press, Cambridge, MA, 1989.

[14] Wolfe, M. Data dependence and program restructuring. *J. Supercomput.* 4(1990) 321-344.

RELATÓRIOS TÉCNICOS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Instituto de Matemática e Estatística da USP

A listagem contendo os relatórios técnicos anteriores a 1993 poderá ser consultada ou solicitada à Secretaria do Departamento, pessoalmente, por carta ou e-mail(mac@ime.usp.br).

Imre Simon
THE PRODUCT OF RATIONAL LANGUAGES
RT-MAC-9301, Maio 1993, 18 pp.

Flávio Soares C. da Silva
AUTOMATED REASONING WITH UNCERTAINTIES
RT-MAC-9302, Maio 1993, 25 pp.

Flávio Soares C. da Silva
ON PROOF-AND MODEL-BASED TECHNIQUES FOR REASONING WITH UNCERTAINTY
RT-MAC-9303, Maio 1993, 11 pp.

Carlos Humes Jr., Leônidas de O.Brandão, Manuel Pera Garcia
A MIXED DYNAMICS APPROACH FOR LINEAR CORRIDOR POLICIES
(A REVISITATION OF DYNAMIC SETUP SCHEDULING AND FLOW CONTROL IN MANUFACTURING SYSTEMS)
RT-MAC-9304, Junho 1993, 25 pp.

Ana Flora P.C.Humes e Carlos Humes Jr.
STABILITY OF CLEARING OPEN LOOP POLICIES IN MANUFACTURING SYSTEMS (Revised Version)
RT-MAC-9305, Julho 1993, 31 pp.

Maria Angela M.C. Gurgel e Yoshiko Wakabayashi
THE COMPLETE PRE-ORDER POLYTOPE: FACETS AND SEPARATION PROBLEM
RT-MAC-9306, Julho 1993, 29 pp.

Tito Homem de Mello e Carlos Humes Jr.
SOME STABILITY CONDITIONS FOR FLEXIBLE MANUFACTURING SYSTEMS WITH NO SET-UP TIMES
RT-MAC-9307, Julho de 1993, 26 pp.

Carlos Humes Jr. e Tito Homem de Mello
A NECESSARY AND SUFFICIENT CONDITION FOR THE EXISTENCE OF ANALYTIC CENTERS IN PATH FOLLOWING METHODS FOR LINEAR PROGRAMMING
RT-MAC-9308, Agosto de 1993

Flavio S. Corrêa da Silva
AN ALGEBRAIC VIEW OF COMBINATION RULES
RT-MAC-9401, Janeiro de 1994, 10 pp.

Flavio S. Corrêa da Silva e Junior Barrera
AUTOMATING THE GENERATION OF PROCEDURES TO ANALYSE BINARY IMAGES
RT-MAC-9402, Janeiro de 1994, 13 pp.

Junior Barrera, Gerald Jean Francis Banon e Roberto de Alencar Lotufo
A MATHEMATICAL MORPHOLOGY TOOLBOX FOR THE KHOROS SYSTEM
RT-MAC-9403, Janeiro de 1994, 28 pp.

Flavio S. Corrêa da Silva
ON THE RELATIONS BETWEEN INCIDENCE CALCULUS AND FAGIN-HALPERN STRUCTURES
RT-MAC-9404, abril de 1994, 11 pp.

Junior Barrera; Flávio Soares Corrêa da Silva e Gerald Jean Francis Banon
AUTOMATIC PROGRAMMING OF BINARY MORPHOLOGICAL MACHINES
RT-MAC-9405, abril de 1994, 15 pp.

Valdemar W. Setzer; Cristina G. Fernandes; Wania Gomes Pedrosa e Flavio Hirata
UM GERADOR DE ANALISADORES SINTÁTICOS PARA GRAFOS SINTÁTICOS SIMPLES
RT-MAC-9406, abril de 1994, 16 pp.

Siang W. Song
TOWARDS A SIMPLE CONSTRUCTION METHOD FOR HAMILTONIAN DECOMPOSITION OF THE HYPERCUBE
RT-MAC-9407, maio de 1994, 13 pp.

Julio M. Stern
MODELOS MATEMATICOS PARA FORMAÇÃO DE PORTFÓLIOS
RT-MAC-9408, maio de 1994, 50 pp.

Imre Simon
STRING MATCHING ALGORITHMS AND AUTOMATA
RT-MAC-9409, maio de 1994, 14 pp.

Valdemar W. Setzer e Andrea Zisman
*CONCURRENCY CONTROL FOR ACCESSING AND COMPACTING B-TREES**
RT-MAC-9410, junho de 1994, 21 pp.

Renata Wassermann e Flávio S. Corrêa da Silva
TOWARDS EFFICIENT MODELLING OF DISTRIBUTED KNOWLEDGE USING EQUATIONAL AND ORDER-SORTED LOGIC
RT-MAC-9411, junho de 1994, 15 pp.

Jair M. Abe, Flávio S. Corrêa da Silva e Marcio Rillo
PARACONSISTENT LOGICS IN ARTIFICIAL INTELLIGENCE AND ROBOTICS.
RT-MAC-9412, junho de 1994, 14 pp.

Flávio S. Corrêa da Silva, Daniela V. Carbogim
A SYSTEM FOR REASONING WITH FUZZY PREDICATES
RT-MAC-9413, junho de 1994, 22 pp.

Flávio S. Corrêa da Silva, Jair M. Abe, Marcio Rillo
MODELING PARACONSISTENT KNOWLEDGE IN DISTRIBUTED SYSTEMS
RT-MAC-9414, julho de 1994, 12 pp.

Nami Kobayashi
THE CLOSURE UNDER DIVISION AND A CHARACTERIZATION OF THE RECOGNIZABLE Z-SUBSETS
RT-MAC-9415, julho de 1994, 29pp.

Flávio K. Miyazawa e Yoshiko Wakabayashi
AN ALGORITHM FOR THE THREE-DIMENSIONAL PACKING PROBLEM WITH ASYMPTOTIC PERFORMANCE ANALYSIS
RT-MAC-9416, novembro de 1994, 30 pp.

Thomaz I. Seidman e Carlos Humes Jr.
SOME KANBAN-CONTROLLED MANUFACTURING SYSTEMS: A FIRST STABILITY ANALYSIS
RT-MAC-9501, janeiro de 1995, 19 pp.

C.Humes Jr. and A.F.P.C. Humes
STABILIZATION IN FMS BY QUASI- PERIODIC POLICIES
RT-MAC-9502, março de 1995, 31 pp.

Fabio Kon e Arnaldo Mandel
SODA: A LEASE-BASED CONSISTENT DISTRIBUTED FILE SYSTEM
RT-MAC-9503, março de 1995, 18 pp.

Junior Barrera, Nina Sumiko Tomita, Flávio Soares C. Silva, Routo Terada
AUTOMATIC PROGRAMMING OF BINARY MORPHOLOGICAL MACHINES BY PAC LEARNING
RT-MAC-9504, abril de 1995,16 pp.

Flávio S. Corrêa da Silva e Fabio Kon
CATEGORIAL GRAMMAR AND HARMONIC ANALYSIS
RT-MAC-9505, junho de 1995,17 pp.

Henrique Mongelli e Routo Terada
ALGORITMOS PARALELOS PARA SOLUÇÃO DE SISTEMAS LINEARES
RT-MAC-9506, junho de 1995, 158 pp.

Kunio Okuda
PARALELIZAÇÃO DE LAÇOS UNIFORMES POR REDUCÃO DE DEPENDÊNCIA
RT-MAC-9507, julho de 1995, 27 pp.

Valdemar W. Setzer e Lowell Monke
COMPUTERS IN EDUCATION: WHY, WHEN, HOW
RT-MAC-9508, julho de 1995, 21 pp.

Flávio S. Corrêa da Silva
REASONING WITH LOCAL AND GLOBAL INCONSISTENCIES
RT-MAC-9509, julho de 1995, 16 pp.

Julio M. Stern
MODELOS MATEMÁTICOS PARA FORMAÇÃO DE PORTFÓLIOS
RT-MAC-9510, julho de 1995, 43 pp.

Fernando Iazzetta e Fabio Kon
A DETAILED DESCRIPTION OF MAXANNEALING
RT-MAC-9511, agosto de 1995, 22 pp.

Flávio Keidi Miyazawa e Yoshiko Wakabayashi
POLYNOMIAL APPROXIMATION ALGORITHMS FOR THE ORTHOGONAL Z-ORIENTED 3-D PACKING PROBLEM
RT-MAC-9512, agosto de 1995, pp.

Junior Barrera e Guillermo Pablo Salas
SET OPERATIONS ON COLLECTIONS OF CLOSED INTERVALS AND THEIR APPLICATIONS TO THE AUTOMATIC PROGRAMMINIG OF MORPHOLOGICAL MACHINES
RT-MAC-9513, agosto de 1995, 84 pp.

Marco Dimas Gubitoso e Jörg Cordsen
PERFORMANCE CONSIDERATIONS IN VOTE FOR PEACE
RT-MAC-9514, novembro de 1995, 18pp.

Carlos Eduardo Ferreira e Yoshiko Wakabayashi
ANAIS DA I OFICINA NACIONAL EM PROBLEMAS COMBINATÓRIOS: TEORIA, ALGORITMOS E APLICAÇÕES
RT-MAC-9515, novembro de 1995, 45 pp.

Markus Endler and Anil D'Souza
SUPPORTING DISTRIBUTED APPLICATION MANAGEMENT IN SAMPA
RT-MAC-9516, novembro de 1995, 22 pp.

Junior Barrera, Routo Terada,
Flávio Corrêa da Silva and Nina Sumiko Tomita
*AUTOMATIC PROGRAMMING OF MMACH'S FOR OCR**
RT-MAC-9517, dezembro de 1995, 14 pp.

Junior Barrera, Guillermo Pablo Salas and Ronaldo Fumio Hashimoto
SET OPERATIONS ON CLOSED INTERVALS AND THEIR APPLICATIONS TO THE AUTOMATIC PROGRAMMING OF MMACH'S
RT-MAC-9518, dezembro de 1995, 14 pp.

Daniela V. Carbogim and Flávio S. Corrêa da Silva
FACTS, ANNOTATIONS, ARGUMENTS AND REASONING
RT-MAC-9601, janeiro de 1996, 22 pp.

Kunio Okuda
REDUÇÃO DE DEPENDÊNCIA PARCIAL E REDUÇÃO DE DEPENDÊNCIA GENERALIZADA
RT-MAC-9602, fevereiro de 1996, 20 pp.

Junior Barrera, Edward R. Dougherty and Nina Sumiko Tomita
AUTOMATIC PROGRAMMING OF BINARY MORPHOLOGICAL MACHINES BY DESIGN OF STATISTICALLY OPTIMAL OPERATORS IN THE CONTEXT OF COMPUTATIONAL LEARNING THEORY.
RT-MAC-9603, abril de 1996, 48 pp.

Junior Barrera e Guillermo Pablo Salas
SET OPERATIONS ON CLOSED INTERVALS AND THEIR APPLICATIONS TO THE AUTOMATIC PROGRAMMINIG OF MMACH'S
RT-MAC-9604, abril de 1995, 66 pp.

Kunio Okuda
CYCLE SHRINKING BY DEPENDENCE REDUCTION
RT-MAC-9605, maio de 1996, 25 pp.