Parallel Processing Letters

Revisiting the decomposition of Karp, Miller and Winograd *

ALAIN DARTE and FRÉDÉRIC VIVIEN
CNRS URA 1398, Ecole Normale Supérieure de Lyon
69364 LYON Cedex 07
e-mail: [darte,fvivien]@lip.ens-lyon.fr

Received (February 1995)
Revised (August 1995)
Communicated by (Gerasoulis and Yang)

ABSTRACT

This paper is devoted to the construction of multi-dimensional schedules for a system of uniform recurrence equations. We show that this problem is dual to the problem of computability of a system of uniform recurrence equations. We propose a new study of the decomposition algorithm first proposed by Karp, Miller and Winograd: we base our implementation on linear programming resolutions whose duals give exactly the desired multi-dimensional schedules. Furthermore, we prove that the schedules built this way are optimal up to a constant factor.

Keywords: system of uniform recurrence equations, null-weight cycles detection, multi-dimensional scheduling, linear programming methods

1 Introduction

This paper is concerned with the problem of building multi-dimensional schedules for system of uniform recurrence equations, especially when there exists no linear schedule. **Systems of uniform recurrence equations** were first defined and studied, by Karp, Miller and Winograd, in the now famous paper: "The organization of computations for uniform recurrence equations" [6]. This model happened to be powerful enough to describe a large class of algorithms, and simple enough to allow a precise description of the organization of computations. These two properties are certainly responsible for the success of uniform recurrence equations over the last decade, especially in the field of systolic array design methodologies [8,11], for which this model provides a sound mathematical framework.

Karp, Miller and Winograd described the structure of a system of uniform recurrence equations with a dependence graph, whose edges are labelled with multi-dimensional vectors. They studied the calculability of such systems and showed that

*Supported by the French Council for Research CNRS, and by the ESPRIT Basic Research Action 6632 "NANA2" of the European Economic Community.

this problem is equivalent to the problem of detecting in the dependence graph, the cycles whose total weight is a non positive vector.

When the system of equations is reduced to a single equation, the problem of computability can be solved by a simple linear programming resolution. Furthermore, considering the dual linear program shows that the computability of the equation is equivalent to the existence of a linear schedule. This result is the base of most systolic array design methodologies, such as space-time mapping (first introduced in [10] and developed in many other papers), and some parallelization techniques such as the hyperplane method [9,3].

In the general case, thus the case of several equations, Karp, Miller and Winograd showed that the problem of calculability is decidable and they gave an algorithm that solves it. This algorithm is based on a recursive decomposition of the dependence graph. Deeper studies and variants of this algorithm have been proposed by Rao [12], Kosaraju and Sullivan [7], Cohen and Meggido [2], Backes [1]. All these works deal with cycle detection or longest dependence paths construction. However, the dual aspect of this decomposition has never been really studied. We show in this paper that the dual problem leads to the construction of multi-dimensional schedules. More precisely, we show that:

- Any computable system of uniform recurrence equations admits an explicit order of computations, given by a multi-dimensional schedule.
- Such a schedule can be efficiently built by a dual representation of Karp, Miller and Winograd's decomposition.
- The schedule that we build is nearly optimal: it expresses the maximal parallelism implicitly contained in the system of uniform recurrence equations.

The paper is organized as follows. In Section 2, we introduce the notations and definitions that we need. In Section 3, we recall Karp, Miller and Winograd's decomposition that detects null weight cycles in a dependence graph. In Section 4, we explain how a step of the decomposition can be efficiently implemented by a single linear programming resolution and we study the dual linear program. In Section 5, we show how this dual resolution allows us to build nearly latency optimal multi-dimensional schedules. Finally, in Section 6, we give some concluding remarks: we summarize the paper and we show briefly how this work is linked to our first motivation: the automatic parallelization of nested loops. However, due to the lack of space, we refer to [4] for complete proofs and for details on the applications and interests of our approach, especially how it applies to the parallelization of arbitrary nested loops and how to rewrite the code to handle "shifted-linear" schedules.

2 Definitions and notations

$\mathbb{Z}$ and $\mathbb{Q}$ denote, respectively, the set of integral and rational numbers, and $\mathbb{Z}^n$ and $\mathbb{Q}^n$ denote the set of n-dimensional integral and rational vectors. $[z]_i$ denotes the i^{th} component of vector z, and $x.y$ the dot product of vectors x and y. For any finite set S, we denote by $|S|$ the number of elements in S.

2.1 Systems of uniform recurrence equations (SURE)

2.1.1 Definition by equations

A **system of uniform recurrence equations** (SURE) is a finite set of equations of the form $V_i(z) = f_i(V_{i_1}(z - d_{i_1,i}), \ldots, V_{i_{m_i}}(z - d_{i_{m_i},i}))$, where:

- V_i is a data array (or **variable**) to be computed for all integral points z included in a subset $\mathcal{P}$ of $\mathbb{Z}^n$. $\mathcal{P}$ is called the **iteration domain**. Boundary values are assumed to be given at points $z \notin \mathcal{P}$ wherever required for variable evaluation at points $z \in \mathcal{P}$.
- z is a vector in $\mathbb{Z}^n$ called **iteration vector**.
- $d_{i,j}$ are vectors in $\mathbb{Z}^n$ called **dependence vectors**.
- f_i is a strict function with m_i arguments whose properties (if any) will not be considered.

One says that $V_i(z)$ depends on $V_{i_1}(z-d_{i_1,i})$, ..., $V_{i_{m_i}}(z-d_{i_{m_i},i})$. These dependence relations define a graph in $\mathcal{P} \subset \mathbb{Z}^n$ called the **expanded dependence graph**.

Example: Throughout the paper, we will work on the following system of uniform equation, with $\mathcal{P} = \{(i,j) \mid 1 \leq i,j \leq N\}$:

$$\left\{ \begin{array}{l} a(i,\ j,\ k) = b(i+1,\ j-1,\ k) + a(i,\ j,\ k-1) \\ b(i,\ j,\ k) = a(i-1,\ j,\ k) + b(i,\ j,\ k+1) \end{array} \right.$$

2.1.2 Definition by dependence graph

A SURE can be equivalently defined in terms of a multigraph G called a **reduced dependence graph** (or briefly, dependence graph) that gives a condensed representation of the expanded dependence graph. G is naturally defined as follows:

- For each variable V_i, there is a vertex v_i in G.
- For each V_i such that $V_i(z)$ depends on $V_j(z - d_{j,i})$, there is an edge in G, from v_j to v_i labelled with vector $d_{j,i}$ (called the **weight** of the edge).

A SURE is then simply defined by a dependence graph G and an iteration domain $\mathcal{P}$. In the following, V denotes the set of vertices of G, E the set of edges of G. $w(e)$ represents the weight of an edge e (i.e. the dependence vector associated to it), $t(e)$ the vertex from which edge e is directed (the tail) and $h(e)$ the vertex to which edge e is directed (the head).

Figure 1 in Section 5.3 shows the dependence graph associated to our example.

2.1.3 Difference between SUREs and nested loops

A SURE may seem identical to perfectly nested loops with uniform dependences, since both can be defined by a dependence graph with uniform dependences and an iteration domain. However, the way these graphs are built makes a huge difference.

In nested loops, there is an **explicit** order of computations: the lexicographic order on iteration vectors, since innermost loops are scanned for each scan of the outermost loops. Dependence vectors can be flow, anti or output dependences but they are always lexicographically positive: therefore, the expanded dependence graph is acyclic and the reduced dependence graph has no cycle of null weight.

In a SURE, the order of computations is an **implicit** order: the left-hand side of each equation can be computed only if all arguments of the right-hand side have been computed. All dependence vectors can be seen as flow dependences, since values are used after being computed, but they do not have to be lexicographically positive. The main consequence is that a SURE is not always computable.

In this sense, the framework of SURE is a more general framework than perfectly nested loops with uniform dependences. The simple fact that the dependence vectors do not have to be lexicographically positive leads to two difficult problems:

(i) A SURE is not always computable: this problem is linked to the problem of detecting null weight cycles in the reduced dependence graph G [a].

(ii) The order of computations is implicit: building an explicit order is linked to the construction of multi-dimensional schedules as defined in Section 2.2.

The goal of this paper is to show that these two problems are actually dual problems and that they can be solved efficiently by an algorithm based on a recursive decomposition of the dependence graph G.

2.2 Schedules

A schedule gives an explicit order of the computations described by a SURE. Among all schedules, schedules that correspond to loops execution are particularly interesting. A linear schedule (wave-front or hyperplane method [9,3]), defined below, corresponds in term of loops to the possibility of rewritting a SURE with one sequential outermost loop and all inner loops parallel. However, such a schedule does not exist for all SUREs. When the degree of parallelism contained in the SURE is not sufficient, one has to consider an extension of the notion of schedule, called multi-dimensional schedule.

2.2.1 One-dimensional schedules

A **one-dimensional schedule** is a mapping from $V \times \mathbb{Z}^n$ to $\mathbb{Z}$ that gives an execution order to the computation of each value $V_i(z)$ while preserving dependence relations. Thus, a function T, with values in $\mathbb{Z}$, is a schedule if and only if for all iteration vectors p and q, for all variables v and w:

$$T(v,p) \geq T(w,q) + 1 \text{ whenever } (w,q) \rightarrow (v,p) \tag{1}$$

where $(w,q) \rightarrow (v,p)$ means that $V(p)$ depends on $W(q)$ (and thus $p, q \in \mathcal{P}$).

Note that if T satisfies inequations 1 but takes its values in $\mathbb{Q}$ instead of $\mathbb{Z}$, then $\lfloor T \rfloor$ is a schedule since

$$T(v,p) \geq T(w,q) + 1 \Rightarrow T(v,p) \geq \lfloor T(w,q) \rfloor + 1 \Rightarrow \lfloor T(v,p) \rfloor \geq \lfloor T(w,q) \rfloor + 1$$

[a] This is Karp, Miller and Winograd's main result: the computability of a SURE is decidable and can be checked in poynomial time. However, for all known extensions of the model of SURE, the problem is undecidable (see [14]).

For this reason, in the rest of the paper, we will consider rational functions that satisfy inequations 1, since they are easier to build. We will call such functions rational schedules.

A schedule is an **affine schedule** if it is a mapping T of the form:

$$\begin{array}{rcll} V \times \mathbb{Z}^n & \longrightarrow & \mathbb{Q} & \\ T:\ (v,\ p) & \rightarrow & X_v.p + \rho_v & \qquad \text{where } X_v \in \mathbb{Q}^n \text{ and } \rho_v \in \mathbb{Q} \end{array}$$

If all X_v are equal, T is called a **shifted-linear schedule**, and if, moreover, all ρ_v are null, then T is called a **linear schedule**. Note that for shifted-linear schedules, inequations 1 reduce to:

$$\forall e \in G,\ X.w(e) + \rho_{h(e)} - \rho_{t(e)} \geq 1$$

Such a vector X is called a **strictly separating hyperplane**. When we just have the relation $X.w(e) + \rho_{h(e)} - \rho_{t(e)} \geq 0$, X is said to be a **weakly separating hyperplane** for edge e.

2.2.2 Multi-dimensional schedules

We are now ready to define multi-dimensional schedules introduced by Karp, Miller and Winograd [6], Rao [12] in the context of SUREs and Feautrier [5] in the context of nested loops.

A **multi-dimensional schedule** (of dimension d) is a mapping from $V \times \mathbb{Z}^n$ to $\mathbb{Z}^d$ that gives an execution order to the computation of each value $V_i(z)$, while preserving dependence relations.

The execution order is now a d-dimensional order, that we take to be the strict lexicographic order $\gg_l$ (so as to correspond to the execution of d nested loops): you can visualize this by saying that the successive components given by the schedule correspond to days, hours, minutes, seconds, and so on ... The compatibility with dependence relations is not anymore an inequality on values but on d-dimensional vectors and can be expressed with $\gg_l$. The constraints to be satisfied are now:

$$T(v,p) \gg_l T(w,q) \text{ whenever } (w,q) \rightarrow (v,p) \tag{2}$$

that is to say:

$$\exists k,\ 1 \leq k \leq d, \forall i,\ i < k,\ [T(v,p)]_i = [T(w,q)]_i \text{ and } [T(v,p)]_k \geq [T(w,q)]_k + 1 \tag{3}$$

As previously, integral schedules are extended to rational functions that satisfy constraints 3 and once again, things get simpler when considering particular schedules such as affine multi-dimensional schedules. A d-dimensional schedule is an **affine schedule** if it is a mapping T of the form:

$$\begin{array}{rcll} V \times \mathbb{Z}^n & \longrightarrow & \mathbb{Q}^d & \\ T:\ (v,\ p) & \rightarrow & (X_v^1.p + \rho_v^1, \ldots, X_v^d.p + \rho_v^d) & \qquad \text{where } X_v^i \in \mathbb{Q}^n \text{ and } \rho_v^i \in \mathbb{Q} \end{array}$$

When the vectors X_v^i are equal (for all v), inequations 3 reduce to:

$$\begin{cases} \forall e \in E, \exists k_e,\ 1 \le k_e \le d \text{ such that } X^{k_e}.w(e) + \rho_{h(e)}^{k_e} - \rho_{t(e)}^{k_e} \ge 1 \\ \text{and } \forall i,\ 1 \le i < k_e,\ X^i.w(e) + \rho_{h(e)}^i - \rho_{t(e)}^i = 0 \end{cases}$$

Thus, the existence of such multi-dimensional schedules is strongly related to the notion of weakly and strictly separating hyperplanes. This paper aims to build such schedules and to show the link between their dimension and the parallelism contained in the SURE.

3 Detection of null weight cycles in a SURE

Consider a SURE defined by its iteration domain $\mathcal{P}$ and its reduced dependence graph G. When $\mathcal{P}$ is bounded, the system is computable if and only if the expanded dependence graph has no cycle. Furthermore, when $\mathcal{P}$ is bounded but sufficiently large, the system is computable if and only if G has no cycle of null weight. In this section, we present a variant of the decomposition of Karp, Miller and Winograd that solves this last problem: the detection of null-weight cycles in the graph G.

3.1 G': subgraph of null weight multi-cycles

We denote by C the **connection matrix** of the reduced dependence graph G defined as follows: C is the $|V| \times |E|$ matrix such that $C_{ve} = -1$ if edge e leaves vertex v, $C_{ve} = 1$ if edge e enters vertex v, and $C_{ve} = 0$ otherwise. We denote by D the **dependence matrix** whose columns are the dependence vectors. Finally, we denote by B, the block matrix whose first rows are the rows of C and whose last rows are the rows of D.

We call a multi-cycle a union of cycles, not necessarily connected. We denote by G' the **subgraph of the multi-cycles of null weight** of G, i.e. the subgraph of G generated by the edges that belong to at least one multi-cycle of null weight.

A non negative vector q with $|E|$ integral components corresponds to a null weight multi-cycle which uses q_e times edge e if and only if $Bq = 0$ (i.e. $Cq = 0$ and $Dq = 0$). An edge e belongs to the subgraph G' if and only if there exists $q \ge 0$, such that $Bq = 0$ and $q_e > 0$ (indeed, if there exists a rational solution, then by scaling its components, there exists an integral solution). We will show in Section 4 how to build directly G' by a single linear programming resolution.

3.2 Decomposition algorithm: KMW

bool KMW(G)

(i) Build G' the subgraph of G generated by all edges that belong to a null weight multi-cycle of G.

(ii) Compute G'_1, G'_2, $\ldots$, G'_s, the s strongly connected components of G'.

- If $s = 0$, G' is empty, return TRUE.
- If $s = 1$, G' is strongly connected, return FALSE.
- Otherwise return $\wedge_i \mathrm{KMW}(G'_i)$ (logical AND).

Assume that G is strongly connected and that all vertices belong to at least one edge (otherwise first decompose G into strongly connected components and remove all "lonely" vertices, if any). Then, G has no cycle of null weight (i.e. G is computable) if and only if $\text{KMW}(G) = \text{TRUE}$. This has been proved in [6].

In the following, we will denote by d the **depth** of the decomposition algorithm, i.e. the longest sequence of recursive calls to the decomposition algorithm KMW.

3.3 Longest dependence path in a SURE

Once we know whether a SURE is computable, it is interesting to give an idea of the length of the longest path in the expanded dependence graph. This length gives a lower bound on the sequentiality of the SURE and thus gives an upper bound on the parallelism it contains.

In [3], it is shown that for **a single** uniform recurrence equation, the latency of the optimal linear schedule is linear in the domain size N and is equivalent to the length of the longest dependence path, on full dimensional domains whose size tends to infinity. Here, in a SURE, when linear schedules do not always exist, the length of the longest path is not necessarily linear in N anymore, it can be equivalent to kN^p for some constants k and p. However, we have the following result:

Theorem 1 (Rao, Vivien) *If d is the depth of the decomposition algorithm 3.2 applied to the dependence graph G, there exists a dependence path in the expanded dependence graph of length at least kN^d for some positive constant k.*

This has been shown the first time by Sailesh Rao [12], even if its proof does not seem to be complete. A complete proof can be found in [4], with the following hypothesis: the SURE is defined in $\mathbb{Z}^n$ by a dependence graph G and an iteration domain $\mathcal{P}$ that contains a n-dimensional cube of size N (thus $\mathcal{P}$ is full dimensional) and that is contained in a n-dimensional cube of size λN, for some constant $\lambda \geq 1$.

4 Construction and properties of G'

We come back in this section to the construction of the subgraph G'. First, we show how to build G' efficiently (by only one linear programming problem instead of one per edge as in [6]). Second, we show that this construction is closely related to the construction of weakly and strictly separating hyperplanes and thus to affine multi-dimensional schedules.

4.1 Linear program for building G'

Step (1) of the decomposition algorithm, which consists in the construction of G', can be done by solving only one linear program. We will show indeed that the edges of G' are exactly the edges e for which $v_e = 0$ in any optimal solution of the following linear program:

$$\min \left\{ \textstyle\sum_e v_e \quad | \quad q \geq 0, \quad v \geq 0, \quad q + v \geq 1, \quad Bq = 0 \right\} \tag{4}$$

Note first that linear program 4 has a finite solution: $q = 0$ with $v = 1$ is indeed a solution. The construction of G' is a consequence of the following lemma:

Lemma 1 *For any optimal solution (q,v) of program 4:*

- $q_e \neq 0 \Leftrightarrow v_e = 0 \Leftrightarrow e \in G'$.
- $q_e = 0 \Leftrightarrow v_e = 1 \Leftrightarrow e \notin G'$.

Therefore, solving program 4 gives directly G': the edges of G' are the edges e such that $v_e = 0$.

4.2 Interpretation of the dual

Now, to better understand what is behind this linear program, let us consider its dual. Program 4 can be written in a canonical form as:

$$\min \left\{ \ \textstyle\sum_e v_e \quad | \quad q \geq 0, \quad v \geq 0, \quad w \geq 0, \quad q + v = 1 + w, \quad Bq = 0 \ \right\} \tag{5}$$

Its dual can be written (after some manipulations) as:

$$\max \left\{ \ \textstyle\sum_e z_e \quad | \quad 0 \leq z \leq 1, \quad X.w(e) + \rho_{h(e)} - \rho_{t(e)} \geq z_e \ \right\} \tag{6}$$

where inequality $z_e \geq 0$ corresponds to variable w_e, inequality $z_e \leq 1$ to variable v_e while inequality $X.w(e) + \rho_{h(e)} - \rho_{t(e)} \geq z_e$ corresponds to variable q_e. The dual solution has an interesting property as shown by lemma 2.

Lemma 2 *For any optimal solution (z, X, ρ) of the dual program 6:*

$$e \in G' \Leftrightarrow X.w(e) + \rho_{h(e)} - \rho_{t(e)} = 0 \tag{7}$$

$$e \notin G' \Leftrightarrow X.w(e) + \rho_{h(e)} - \rho_{t(e)} \geq 1 \tag{8}$$

Proof. The proof is based on the complementary slackness theorem [13]. Details can be found in [4]. □.

Lemma 2 shows that considering the dual provides separating hyperplanes, which are strictly separating hyperplanes for the edges not in G' and weakly separating hyperplanes for edges in G'. Furthermore, for each subgraph G that appears in the decomposition, these hyperplanes are those that are the "most often strict": the number of edges, for which such an hyperplane is strict, is maximal.

During the decomposition algorithm, one can associate to each vertex v of G, a sequence of vectors $X_v^1, \ldots, X_v^{d_v}$, obtained by considering the dual problem 6. d_v is the depth of the decomposition algorithm at which vertex v is removed. This sequence of vectors has the following property:

Theorem 2 *The $d_v - 1$ first separating hyperplanes $X_v^1, \ldots, X_v^{(d_v-1)}$ associated to a vertex v of G are linearly independent. Furthermore, when G is computable, all separating hyperplanes $X_v^1, \ldots, X_v^{d_v}$ associated to v are linearly independent.*

Proof. This result is a consequence of lemma 2. See [4] for a detailed proof. □.

Note that this implies that the depth of the decomposition is bounded by $n+1$ (and even by n when G is computable).

5 Optimal multi-dimensional schedule for a computable SURE

We are now ready to merge all previous results and propose nearly optimal multi-dimensional schedules for computable SUREs.

5.1 Construction of a d-dimensional schedule

Consider a computable dependence graph G and assume first that G is strongly connected. We apply to G the decomposition algorithm, focusing this time on the dual program 6.

We build a multi-dimensional schedule for G, by building for each vertex v of G, the sequence of vectors $X_v^1, \ldots, X_v^{d_v}$ and the sequence of constants $\rho_v^1, \ldots, \rho_v^{d_v}$, obtained by considering the dual program 6 during the decomposition algorithm. Note that such separating hyperplanes X are not unique: once we know which edge corresponds to an equality like 7 and which one corresponds to an inequality like 8, one can choose another objective function than $\sum_e z_e$. One can for example try to minimize the latency corresponding to the vector X, i.e. $\max_{p\in\mathcal{P},\ q\in\mathcal{P}} X.(p-q)$. See [3] for this kind of optimization.

For each vertex v, we complete the sequences of vectors X_v^i and constants ρ_v^i with zeros so as to obtain sequences of length d.

Lemma 3 *The multi-dimensional function T:*

$$\begin{array}{lcl} V \times \mathcal{P} & \longrightarrow & \mathbb{Q}^d \\ (v,\ p) & \rightarrow & (X_v^1.p+\rho_v^1,\ \ldots,\ X_v^{d_v}.p+\rho_v^{d_v},\ 0,\ \ldots,\ 0) \end{array}$$

defines a multi-dimensional schedule.

Proof. This is a direct consequence of lemma 2. $\Box$.

When the dependence graph has more then one strongly connected component, we first schedule independently each of them with a multi-dimensional schedule built as above. Then, we schedule them with respect to each other by a topological sort on the acyclic graph constitued by the strongly connected components.

5.2 Latency of the multi-dimensional schedules

We assume the same properties on the iteration domain $\mathcal{P}$ than in theorem 1. We can then show that the multi-dimensional schedules obtained by this method are nearly optimal in the following sense:

Theorem 3 *The multi-dimensional schedule built in Section 5 is nearly optimal: if d is the depth of the decomposition algorithm, the latency of the schedule is $O(N^d)$ and the length of the longest dependence path is $\Omega(N^d)$.*

Proof. This is a consequence of theorem 1 and a consequence of the fact that there exists a vertex at depth d in the decomposition, thus at least d linearly independent separating hyperplanes. See once again [4] for more details. $\Box$.

5.3 Example

We now come back to our example (see Section 2.1.1). First of all, we have to check the computability of the SURE. The algorithm shows that G' is generated by the two self-dependence edges. G' has two strongly connected components, none of them has a null-weight multi-cycle, therefore G is computable. Furthermore, since $d = 2$, theorem 3 shows that there exists a multi-dimensional schedule whose latency is $O(N^2)$ and a dependence path of length $\Omega(N^2)$. The decomposition is illustrated by the following figure:

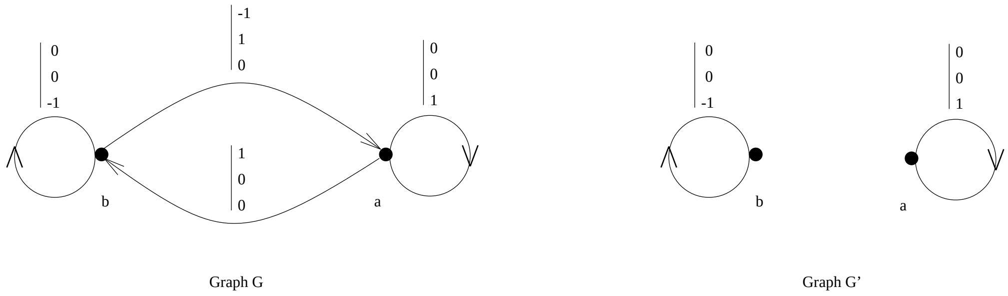

Figure 1: Decomposition of the dependence graph

Hence, we are assured that the SURE is computable. We now have to build **a** 2-dimensional schedule. Lemma 2 summarizes the constraints such schedules have to satisfy. To choose one solution, we minimize, for example, $\sum_i |X_i|$, the norm of the vector X [b]. Note that if the iteration domain is a cube, this is equivalent to minimizing the latency. At level 1 of the decomposition, we thus have to solve the following linear program (we wrote $X = (x, y, z)$):

$$\min \left\{ |x| + |y| + |z| \;\middle|\; \begin{array}{r} -x + y + \rho_a - \rho_b \geq 1 \\ x + \rho_b - \rho_a \geq 1 \\ -z = 0 \\ z = 0 \end{array} \right\}$$

whose optimal solution is $X = [0,\ 2,\ 0]$, $\rho_a = 0$, $\rho_b = 1$.

Then, we start again on each strongly connected component of G' so as to build the second dimension of our schedule. In this example, the second level is straightforward since each component contains only one edge.

The final result is thus the following:

variable	schedule	first level	second level
a			
	linear parts	$[0,\ 2,\ 0]$	$[0,\ 0,\ 1]$
	constant parts	0	0
b			
	linear parts	$[0,\ 2,\ 0]$	$[0,\ 0,\ -1]$
	constant parts	1	0

and, if $\mathcal{P}$ is the 3-dimensional cube of size N, this corresponds to the following nested loops description:

```
for j = 1 to N            /* Vector (0,2,0) */
   for k = 1 to N           /* Vector (0,0,1) */
      forall i = 1 to N
```

[b]Removing absolute values can be done simply by minimizing $\sum_i \gamma_i$ instead of $\sum_i |X_i|$ with $\gamma_i \geq X_i$ and $\gamma_i \geq -X_i$. One can also write $X_i = Y_i - Z_i$, $Y_i \geq 0$, $Z_i \geq 0$ and minimize $\sum_i Y_i + Z_i$.

```
        a(i, j, k) = b(i+1, j-1, k) + a(i, j, k-1)
      endforall
    endfor
    for k = N to 1 step -1   /* Vector (0, 0, -1) */
      forall i = 1 to N
        b(i, j, k) = a(i-1, j, k) + b(i, j,  k+1)
      endforall
    endfor
endfor
```

6 Conclusion

The results presented in this paper are mainly based on Karp, Miller and Winograd's decomposition algorithm which decides if a SURE is computable or not. We first showed how this algorithm can be efficiently implemented by linear programming resolutions. Then, by an interpretation of the dual linear programs, we explained how to build multi-dimensional schedules that express explicit orders of computations for the SURE.

We believe that our approach has two main interests.

First, we are able to guarantee the quality of our schedules: they are indeed nearly optimal in the sense that their latency and the length of the longest dependence paths are of the same order. The schedules that we obtain reveal the maximal degree of parallelism contained in the SURE, while the longest dependence paths reveal the maximal degree of sequentiality in the SURE.

Second, our resolution gives directly the simpliest form of schedules that can be considered to keep optimal latency: at each level of the decomposition, they are affine schedules whose linear part is the same on the current strongly connected component; we call such schedules **locally** shifted-linear schedules. There is thus no need to look for complicated schedules such as arbitrary affine multi-dimensional schedules. This is very important in practice, since the final goal is to use such schedules to generate code. The simpler the schedule, the easier the implementation, the better the chance of further optimizations.

Because of space limitation, we did not address in this paper the possible applications of such schedules constructions. Actually, our initial motivation was to find a way to parallelize arbitrary nested loops, especially when dependences are not uniform but are expressed as direction vectors or dependence cones: our idea was to try to extend linear scheduling methods. Now that we are able to optimally schedule any SURE, we can propose a new and optimal parallelization technique for nested loops. The main idea is to transform the dependence graph associated to the nested loops into an equivalent dependence graph where all dependences are uniform but not necessarily lexicographically positive, i.e. nothing but the dependence graph of a SURE on which we apply the techniques presented in this paper. We refer to [4] for more details.

We believe that this approach is a very promising approach since it can be proven optimal with respect to the quality of the dependence analysis.

Dedication

We would like to dedicate this work to Hervé Le Verge who died a few months ago. Hervé was a bright researcher in the field of systolic methodology design.

References

1. Wolfgang Backes. *The structure of longest paths in periodic graphs.* PhD thesis, Universität des Saarlandes, Sarbrücken, July 1993.
2. Edith Cohen and Nimrod Megiddo. Strongly polynomial-time and NC algorithms for detecting cycles in dynamic graphs. In *Prodceedings of 21st Annual ACM Symposium on Theory of Computing*, pages 523–534, 1989.
3. Alain Darte, Leonid Khachiyan, and Yves Robert. Linear scheduling is nearly optimal. *Parallel Processing Letters*, 1(2):73–81, 1991.
4. Alain Darte and Frédéric Vivien. Automatic parallelization based on multi-dimensional scheduling. Technical Report 94-24, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, France, September 1994.
5. Paul Feautrier. Some efficient solutions to the affine scheduling problem, part II, multi-dimensional time. *Int. J. Parallel Programming*, 21(6):389–420, December 1992. Available as Technical Report 92-78, Laboratoire MASI, Université Pierre et Marie Curie, Paris, October 1992.
6. R.M. Karp, R.E. Miller, and S. Winograd. The organization of computations for uniform recurrence equations. *Journal of the ACM*, 14(3):563–590, July 1967.
7. S. Rao Kosaraju and Gregory F. Sullivan. Detecting cycles in dynamic graphs in polynomial time (preliminary version). In ACM Press, editor, *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 398–406, May 1988.
8. S.Y. Kung. *VLSI array processors.* Prentice-Hall, 1988.
9. Leslie Lamport. The parallel execution of DO loops. *Communications of the ACM*, 17(2):83–93, February 1974.
10. Patrice Quinton. Automatic synthesis of systolic arrays from uniform recurrent equations. In *The 11th Annual International Symposium on Computer Architecture*, Ann Arbor, Michigan, June 1984. IEEE Computer Society Press.
11. Patrice Quinton and Yves Robert. *Systolic Algorithms and Architectures.* Prentice Hall, 1991. Translated from French, Masson (1989).
12. Sailesh K. Rao. *Regular Iterative Algorithms and their Implementations on Processor Arrays.* PhD thesis, Stanford University, October 1985.
13. Alexander Schrijver. *Theory of Linear and Integer Programming.* John Wiley and Sons, New York, 1986.
14. Hervé Le Verge and Yannick Saouter. New results on computability of recurrence equations. Technical Report 380-93, LaBRI, Bordeaux (France), September 1993. Submitted to Foundations of Computer Science.