Parallel execution of irregular meshes into a systolic linear array

S. Fidanova * [a] and A. Goldman † [b]

[a]CICT-BAS, Acad.G. Bonchev str. Bl.25A,
1113 Sofia, Bulgaria,
e-mail: stefka@iscbg.acad.bg

[b] LMC - IMAG,
100 Rue des Mathematiques, 38041 Grenoble Cédex 09 - France,
e-mail: goldman@imag.fr

Abstract

We present in this paper a scheduling algorithm for the execution of irregular meshes into a systolic linear array. The data dependence graphs of many problems in computer science are irregular meshes. These include the unbounded knapsack problem, the subset sum problem, the change making problem and the longest common subsequence problem. In this paper a scheduling algorithm for a dependence graph is represented by an allocation function and an initial time for each vertex. The allocation and timing function in this paper are improvements of the previous methods. Our method reaches two lower bounds, the lower bound on the number of processors and the lower bound on time.
Key-words: Irregular mesh, scheduling, systolic algorithm, regular interconnection networks.

1 Introduction

1.1 General context

A parallel algorithm based on communicating tasks is usually represented by an oriented graph where the nodes represent the tasks (or the instructions) and the arcs represent the precedence constraints between the tasks. Similarly, a distributed-memory parallel computer can be represented by a graph where the nodes represent the processors and the edges represent the communication links among the processors.

The parallelization process needs first to determinate a scheduling of the tasks (which consists of assigning to each task an initial execution time and a processor allocation). In the absence of communications and assuming an unbounded number of processors, it is easy to

*Supported by Réseau Formation Recherche, Europe Centrale et Orientale. On leave from Bulgarian Academy of Sciences, Bulgaria. Supported by the BG MSHE grant No-MU02/96.

†Supported by CNPq (Proc. 200590/95-2). On leave from DCC-IME-USP Brazil.

determine the longest path (also called *makespan*) of the graph. In other hand, with a given number of processors, with limited memory, the problem becomes NP-hard [5].

In this paper, we consider processor networks with point to point connections, where each processing element is able to perform any arithmetic operations and has a limited memory capacity. The systolic array works in a SIMD way.

The linear array is very useful for parallel architectures that contains an hamiltonian path (grid, hypercube, de Bruijn), because it can implement the linear array without delay. Several computers of the new generation consist of a systolic array (linear array or grid) for some calculations. The Germany corporation ISATEC makes a grid systolic array *Systola 1024* with 1024 processors.

We will now present the problem and recall some basic definitions needed in the subsequent section. In section 2 we study the execution of an irregular mesh on a systolic linear array. We will first establish a general lower bound for irregular meshes, then, we will present an optimal systolic algorithm, both in time and in number of processors. We give also an algorithm on a ring.

It is worth to observe that a similar work was done by Andonov [2], but in this paper we improve the algorithm, and we propose the lower bounds in the number of processors and in the time.

1.2 Brief presentation of the topologies

The irregular mesh family can represent the execution of many problems, including dynamic programming problems such as the knapsack problem, change making problem and longest common subsequence. Different parallel algorithms based on dynamic programming technique for knapsack problem are presented in [1], [2], [3], [4]. They used different interconnection networks such as linear array, grid or hypercube. Let us now present the precedence graph and interconnection topologies.

- **Irregular mesh** A 2-dimensional irregular mesh $NM(m, c)$ with jumps w_i has as vertex the set of pairs (i, j), where $0 \leq i \leq m-1$, $0 \leq j \leq c-1$. There are arcs from node (i, j) to the nodes $(i+1, j)$ and $(i, j+w_i)$. An example of NM is given on figure 1. The grid is an irregular mesh NM where $w_i = 1$ for all i.

  For sake of simplicity, we will denote the sets $L_i = \{(i, j), 0 \leq j \leq c-1\}$ by rows. We will define also $w_{\min} = \min_i w_i$, $w_{\max} = \max_i w_i$ and $W = \sum_{k=0}^{m-1} w_k$.

  An irregular mesh is the precedence graph of the following class of equations:

  $$\begin{array}{ll} v[i, j] = v[i-1, j] & 0 \leq j < f(i), 0 \leq i < m \\ v[i, j] = F(v[i-1, j], v[i, j-f(i)] + g(i)) & f(i) \leq j < c, 0 \leq i < m, \end{array}$$

  where the values $v[0, j]$ for $j = 0, \ldots, c-1$ are given. The integer functions f and g are known in advance, and the values $v[m-1, j], j = 0, \ldots c-1$ must be computed.

- **Linear array** The vertices of linear network with length n are labeled by the elements of the set $\{0, 1, \ldots, n-1\}$, there is an edge between vertices u and v if $|u - v| = 1$.

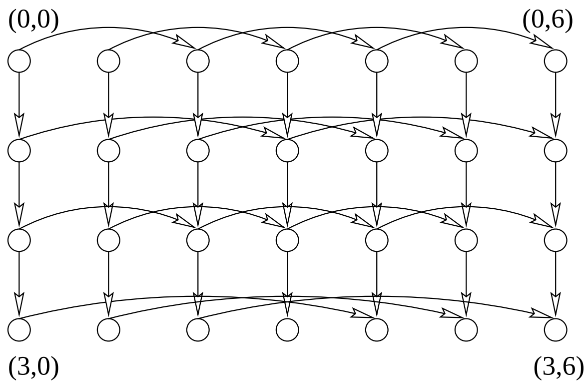

Figure 1: Irregular mesh $NM(4,7)$ with jumps $w_0 = w_2 = 2$, $w_1 = 3$ and $w_3 = 4$.

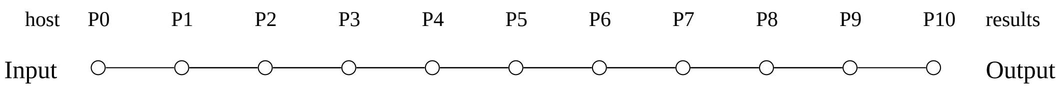

Figure 2: Linear array with length 11.

1.3 General lower bound

It is well-known that the minimum execution time of a precedence graph is greater than or equal to the length of the its longest path. This leads to the following result:

Proposition 1 *The longest path length of the irregular mesh $NM(m,c)$ with jumps $(w_i)_0^{m-1}$ is equal to $m + \left\lceil \frac{c}{w_{\min}} \right\rceil - 1$.*

Proof: Given i, $0 \leq i < m-1$, each task (i,j) of $NM(m,c)$ precedes the task $(i+1,j)$. There are m rows in $NM(m,c)$ with jumps w_i on row i. The length of the paths in each row i is $\left\lfloor \frac{c}{w_i} \right\rfloor$ or $\left\lceil \frac{c}{w_i} \right\rceil$, and the length of the longest path on the rows is $\left\lceil \frac{c}{w_{\min}} \right\rceil$. Hence, the longest path of the $NM(mc)$ has $m + \left\lceil \frac{c}{w_{\min}} \right\rceil - 1$ arcs.

□

2 Systolic algorithm

In this section we study the execution of $NM(m,c)$ on a systolic linear array. This corresponds to a linear array connected to a host computer at one extremity, where the tasks to be performed are sent from the host at periodic times [6].

In our systolic array model, at each step, each processor:

- Receives input from the left neighbor

- Performs the computation (eventually using the local memory)
- Generates outputs to the right neighbor

Time is partitioned into steps and the array operates synchronously.

We give now an algorithm for scheduling $NM(m,c)$ on a systolic linear array. For our algorithm we suppose $w_{\max} \leq \frac{c}{2}$. We consider a linear network of processors of size $n = W\left(= \sum_{k=0}^{m-1} w_k\right)$. Processor 0 is connected to the host, and the rightmost processor is connected to the output. The allocation is such that task (i,j) is allocated on processor $\sum_{k=0}^{i-1} w_k + j \bmod w_i$.

We detail now the algorithm: Each row i is distributed among w_i processors. There are w_i independent chains of tasks in row i, each chain is allocated on the same processor.

Task (i,j) begins to execute on processor $\sum_{k=0}^{i-1} w_k + j \bmod w_i$ at time $j + \sum_{k=0}^{i-1} w_k + j \bmod w_i$. It precedes tasks $(i+1,j)$ and $(i, j+w_i)$ which are respectively executed on processors and at times:

task	processor	time
(i,j)	$p = \sum_{k=0}^{i-1} w_k + j \bmod w_i$	$t = j + \sum_{k=0}^{i-1} w_k + j \bmod w_i$
$(i+1,j)$	$p_1 = \sum_{k=0}^{i} w_k + j \bmod w_{i+1}$	$t_1 = j + \sum_{k=0}^{i} w_k + j \bmod w_{i+1}$
$(i,j+w_i)$	$p_2 = \sum_{k=0}^{i-1} w_k + j \bmod w_i$	$t_2 = j + w_i + \sum_{k=0}^{i-1} w_k + j \bmod w_i$

Observing that $p_1 - p = t_1 - t$, $p_2 = p$ and $t_2 - t = w_i$, we verify that the proposed scheduling is feasible, there is a complete execution example on figure 3.

The last task of the irregular mesh, that is $(m-1, c-1)$, begins its execution at time $t_f = c - 1 + \sum_{k=0}^{m-2} w_k + (c-1) \bmod w_{m-1}$ on processor $p_f = \sum_{k=0}^{m-2} w_k + (c-1) \bmod w_{m-1}$.

We can see that p_f is at distance $w_{m-1} - (c-1) \bmod w_{m-1}$ of the last processor, so the last task will exit the linear array on time $t_f + w_{m-1} - (c-1) \bmod w_{m-1} + 1$, that is $t_f = c + W$.

The case where $w_{\max} > \frac{c}{2}$ is little different, there are chains with only one task, that is, without horizontal precedence. Given a row, we allocate all the tasks without horizontal precedence on the same processor. Let $w'_i = \min\{w_i, c - w_i + 1\}$, we give an irregular mesh scheduling on a linear array with $\sum_{i=0}^{m-1} w'_i$ processors.

Task (i,j) begins to execute on processor $\sum_{k=0}^{i-1} w'_k + \min\{j \bmod w_i, c - w_i + 1\}$ at time $j + \sum_{k=0}^{i-1} w'_k + \min\{j \bmod w_i, c - w_i + 1\}$.

We give below a lower bound on the number of processors required to execute $NM(m,c)$ on a systolic linear array. When the rows are feed to the processors in a column order we can state the following proposition.

Proposition 2 *To execute $NM(m,c)$ with set of jumps $(w_i)_0^{m-1}$ on processors with memory of size 1, at least $\sum_{k=0}^{m-1} w'_k$ processors are needed.*

Proof. For calculating row i from NM, at least w'_i processors are needed.

Assume that for row i we have x processors and $x < w'_i$. If $w_i \leq \frac{c}{2}$, as the memory size is 1, after feed the first w_i tasks from the row i (the initial tasks of the w_i chains). Only

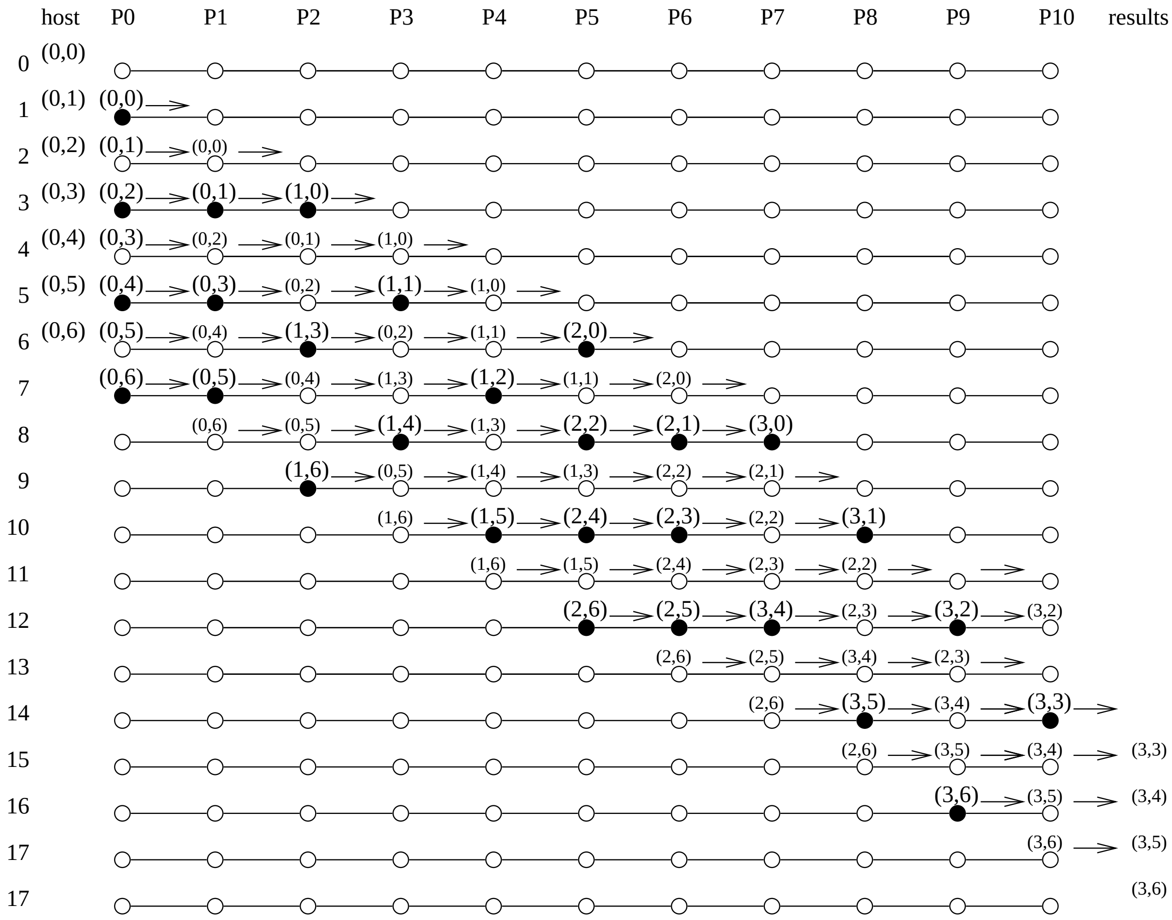

Figure 3: Execution of $NM(4,7)$ with jumps $\{2,3,2,4\}$ on linear network with 10 processors. We have represented the working processors in black and $\rightarrow$ represents the communications.

x_i can be stored in memory, $w_i - x_i$ are stored in the send/receive links. When the second tasks of each chain enter, before w_i entrances, at least one of the first tasks will exit the x_i processors without resolving the precedence relation. Then, memory will be needed for storing the results of these tasks.

If $w_i > \frac{c}{2}$, as above we will need at least $c - w_i$ processors for the chains with more than one task. All the other chains with only one task can be executed on another processor.

Using w'_i processors by row, we will use $\sum_{k=0}^{m-1} w'_k$ processors.

□

Claim 1 *Using processors with memory of size α, the necessary number of processors is $n = \sum_{k=0}^{m-1} \left\lceil \frac{w'_k}{\alpha} \right\rceil$ and the execution time of $NM(m,c)$ is $c + \sum_{k=0}^{m-1} \left\lceil \frac{w'_k}{\alpha} \right\rceil$.*

Proof. As each processor has a memory of size α, we can calculate $\min(\alpha, w'_i)$ chains (we consider the chains with only one task as one chain) of row L_i in each one. Making that we will need $\left\lceil \frac{w_i'}{\alpha} \right\rceil$ processors to calculate row L_i.

If $w_{\max} \leq \frac{c}{2}$, the allocation is such that task (i,j) is allocated on processor $\sum_{k=0}^{i-1} \lceil \frac{w_k}{\alpha} \rceil + \left\lfloor \frac{j \bmod w_i}{\alpha} \right\rfloor$, and begins it execution at $j + \sum_{k=0}^{i-1} \lceil \frac{w_k}{\alpha} \rceil + \left\lceil \frac{j \bmod w_i}{\alpha} \right\rceil$. In the other case, the allocation is similar.

□

Proposition 3 *The lower bound of execution of any algorithm with d initial data on a systolic linear array with p processors is $d + p$.*

Proof. Let we have a linear array with p processors, and let we have any algorithm with d initial data. For executing the algorithm each initial data must cross all the processors, the minimum time one data spend to cross the linear array is p. Hence the minimum execution time of any algorithm with d initial data on a linear array with p processors is $d + p$.

□

Our algorithm, to execute $NM(m,c)$ with jumps $(w_i)_0^{m-1}$, in a systolic linear array reaches the two lower bounds. It uses $\sum_{k=0}^{m-1} w_k'$ processors (lower bound on processors) and has c initial data with execution time $W + c$ (lower bound on time).

2.1 Ring extension

We give also an extension of the previous algorithm to the ring. We will describe only the case where $w_{\max} \leq \frac{c}{2}$, the extension for $w_{\max} > \frac{c}{2}$ is similar. We define as ring, a systolic linear array where the last processor has an arc (unidirectional) to the first processor.

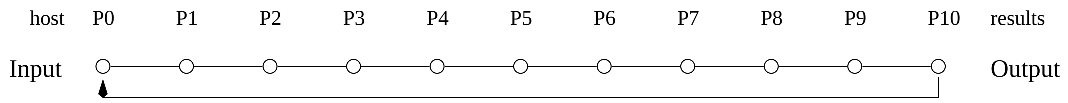

Figure 4: Ring with length 11.

If $W > c$ we can state the following claim:

Claim 2 *It is possible to execute $NM(m,c)$ with set of jumps $(w_i)_0^{m-1}$ on processors with memory of size 1, in a ring with c processors within at most $c + c\left\lceil \frac{W}{c} \right\rceil$ steps.*

Proof: Use the following scheduling function: the task (i,j) is allocated on processor $p = (\sum_{k=0}^{i-1} w_k + j \bmod w_i) \bmod c$ and is executed at time $t = j + \sum_{k=0}^{i-1} w_k + j \bmod w_i$.

□

In other hand, if $W \leq c$ we can execute $NM(m,c)$ with W processors, within $c + W$ steps.

Claim 3 *The previous scheduling is optimal in time and in the number of processors.*

Proof: If we could execute the irregular mesh faster, it would be possible to execute the irregular mesh on the linear array faster.

If $W \leq c$ the number of processors is given by proposition 1, in the other case, with less than c processors, there would be an conflict between the data from the host and the data from the last processor.

□

Now we will study the a more general model. A (α, p)-ring is a systolic linear array with p processors, each one with memory of size α, and a queue which is a FIFO memory that is used to transmit data from the last processor to the first one.

Proposition 4 *The time to execute $NM(m,c)$ with set of jumps $(w_i)_0^{m-1}$ on a (α, p)-ring (with $p \leq c$) is $c + W' + \left\lceil \frac{W'}{p} \right\rceil (c-p)$, where $W' = \sum_{i=0}^{m-1} w'_i$ and $w' = \min\{w_i, c - w_i + 1\}$.*

Proof: The idea is to execute the chains from the irregular mesh by blocks of size p. While a block is being processed the next block can be queued in the queue. When the first processor of the ring is free the queue give it new data to process.

Using an allocation similar to the used in the Claim 1 (if $w_{\max} \leq \frac{c}{2}$ the task (i,j) is allocated on processor $\left(\sum_{k=0}^{i-1} \left\lceil \frac{w_k}{\alpha} \right\rceil + \left\lfloor \frac{j \bmod w_i}{\alpha} \right\rfloor\right) \bmod p$, in the other case the allocation is similar) we can give the execution time for each task.

The tasks from the first block begin its execution at $j + \sum_{k=0}^{i-1} \left\lceil \frac{w_k}{\alpha} \right\rceil + \left\lfloor \frac{j \bmod w_i}{\alpha} \right\rfloor$. For the second block we have to add the waiting time in the queue, $c - p$. If $p = c$ we do not use the queue.

In general for the elements of block I the time is $j + \sum_{k=0}^{i-1} \left\lceil \frac{w_k}{\alpha} \right\rceil + \left\lfloor \frac{j \bmod w_i}{\alpha} \right\rfloor + I(c-p)$. As there are $\left\lceil \frac{W'}{p} \right\rceil$ blocks. The total execution time is:

$$c + W' + \left\lceil \frac{W'}{p} \right\rceil (c-p),$$

that is

$$O\left(c + \frac{W'c}{p}\right).$$

□

3 Conclusion

The purpose of this paper was to study the schedule of the irregular mesh in some regular structures. This problem is important in practice, because it modelises some dynamical programming problems. These problems are very useful in economy (loading ships, financial operations and so on).

We presented a schedule of the irregular mesh for the simplest interconnection network: a linear array, on a very limited computational model (systolic). We obtained optimal results on this very regular structure. We also extended the results to the ring.

We described the linear systolic array, but we can implement this algorithm on any array with an Hamiltonian path. VLSI technologie allow the constructions of cheap and specialized architectures. Our architectures are very regular and are extremly well suited to VLSI implementation.

References

[1] V. Aleksandrov and S. Fidanova. On the expected execution time for a class of non uniform recurrence equations mapped onto 1d array. *Parallel Algorithms and Applications*, 1:303 – 314, 1994.

[2] R. Andonov and P. Quinton. Efficient linear systolic array for knapsack problem. *Technical Report - IRISA - France*, 1(639), 1992.

[3] G.H. Chen, M.S. Chern, and J.H. Jang. Pipeline architectures for dynamic programming algorithms. *Parallel Computing*, 1(13):111 – 117, 1990.

[4] J. Lin and J. Storer. Processor efficient hypercube algorithm for the knapsack problem. *Journal of Parallel and Distributed Computing*, 3(13):332 – 337, 1991.

[5] C. Picouleau. New complexity results on scheduling with small communication delays. *Journal of Discrete Applied Mathematics*, 1-3(60):331 – 342, 1995.

[6] Y. Robert and P. Quinton. *Algorithmes et architectures systoliques.* Masson, 1989.