IEEE Region 10 Conference on Computer and Communication Systems, September 1990, Hong Kong

17

✓

PARALLEL GAUSSIAN ELIMINATION FOR BANDED MATRIX - A COMPUTATIONAL MODEL

V. Mani, B. Dattaguru, N. Balakrishnan and T.S. Ramamurthy
Department of Aerospace Engineering
Indian Institute of Science
Banaalore 560 012, INDIA

Abstract - Parallel Gaussian elimination technique for the solution of a system of equations $Ax = C$ where A is a banded matrix, is modeled as a acyclic directed graph. This graph is useful in the identification of parallel operations, the minimum absolute completion time for the solution process and the minimum number of processors required to solve it in minimum time. Hu's level scheduling strategy is used for scheduling operations to processors. The absolute minimum completion time sets a limit on the speed-up. The absolute minimum completion time is dependent on the order of A matrix and is independent of the bandwidth. The minimum number of processors required to complete the solution process is fixed by the bandwidth and is independent of the order of A matrix. A method of incorporating communication aspects in between processors in four kinds of interconnections is also presented.

1. INTRODUCTION

Reviewing the progressive advances and development in computer architectures and scientific computing, we observe an increasing demand for parallel and distributed computing. This trend has led the researchers from various disciplines to focus their attention towards development of parallel computational techniques. There is also a great potential for using parallel computers in structural mechanics [1]. The developments of parallelism in numerical algorithms can be found in references [2,3,4]. Ortega and Voigt [5] presents the development of parallel algorithms for engineering problems. Recently structural engineers have developed parallel implementation of finite element technique. Law [6] presents a parallel finite element solution. In this method, the finite element displacements are directly computed without forming the global stiffness matrix on an MIMD multiprocessing system. A parallel finite element method and its implementation on a hypercube computer is discussed in [7] and parallel element oriented conjugate gradient procedure is used to compute the displacements. Chein and Sun [8] proposed two parallel forming procedures and a parallel Gaussian elimination technique [9] for the solution of large system of equations. Two practical parallel algorithms for solving systems of dense linear equations on an MIMD computer is given in [10].

There are three well defined modules in finite element program: (i) pre-processing, (ii) solution, (iii) post-processing. In this paper, we deal with the solution module. We propose a parallel Gaussian elimination technique for the solution of linear equations. We consider the direct solution of $Ax = C$, where A is a banded matrix with half bandwidth b. We model the situation as a acyclic directed graph. In this graph, the nodes represent arithmetic operations applied to the elements of A and the arcs represent the precedence relation that exists among the operations in the solution process. This graph gives the clear picture to the user in identifying the operations that can be done in parallel. This graph is also useful in scheduling operations to the processors. The absolute minimum completion time and the lower bound on the minimum number of processors required to solve the equations in minimum time can be found from it. Speed-up approaches a limit using parallel processors, set by the absolute minimum time, is also brought out from this graph.

For scheduling operations to processors Hu's [11] level scheduling strategy is used. The minimum number of processors is obtained from the directed graph in [12]. The task graph is also useful in finding the number of communications needed in between the processors for the solution in a multi-processor architecture.

2. PARALLELISM IN GAUSSIAN ELIMINATION FOR BANDED MATRIX

We consider the direct solution of

$$Ax = C \tag{1}$$

using Gaussian elimination, where

$$A = [a_{ij}] \quad i = 1,2,\dots,n; \quad j = 1,2,\dots,n+1 \tag{2}$$

$$\text{and } a_{i,n+1} = C_i \quad i = 1,2,\dots,n \tag{3}$$

Also A matrix is banded with half bandwidth b as shown in Fig. 1. In general Gaussian elimination has two steps [13]: (i) triangulation and forward elimination TF, (ii) Back substitution BK. They are described by the following procedures.

```
Procedure TF/Triangulation and Forward Elimination
  begin for k = 1 step 1 until n-1 do
    begin
      for all $a_{ij} \neq 0$, $k+1 \leq j \leq n+1$ do
              $a_{ij} = a_{kj}/a_{kk}$                          (4)
      for all $a_{ik}.a_{ij} \neq 0$, $k+1 \leq i \leq n$;
              $k+1 \leq j \leq n+1$, do
              $a_{ij} = a_{ij} - a_{ik}.a_{ij}$                 (5)
      end
              $a_{n,n+1} = a_{n,n+1}/a_{nn}$                    (6)
      end

Procedure BK/Back substitution
   begin for k = n step-1 until 2, do
      begin if $a_{k,n+1} \neq 0$ then
      for all $a_{ik} \neq 0$, $1 \leq i \leq k-1$, do
          $a_{i,n+1} = a_{i,n+1} - a_{ik}.a_{k,n+1}$            (7)
      end
      for all k, $1 \leq k \leq n$
              $x_k = a_{k,n+1}$                                 (8)
```

CH2866-2/90/0000-170 $1.00 © 1990 IEEE

For ease of understanding, we will show the operations that can be done in parallel [13]

It is to be noted that the operations (4), (5) and (7) can be done in parallel and they are denoted below.

```
     TF:  for k = 1,2,...,n
  for $k+1 \le j \le n+1$
      $a_{ij} = a_{ij}/a_{kk}$   divide in parallel
  for $k+1 \le i \le n$,  $k+1 \le j \le n+1$
      $a_{ij} = a_{ij} - a_{ij} \cdot a_{ij}$   update in parallel      (9)
     BK:  for k = n, n-1, ..., 2
  for $1 \le i \le k-1$
      $a_{i,n+1} = a_{i,n+1} - a_{ik} \cdot a_{k,n+1}$   update in
                                                           parallel.
```

We shall refer to each operation specified by (4) and (6) as divide operation and each specified by (5) and (7) as update operation. From (9) it can be seen that, if there are sufficient processors the operations can be executed simultaneously. Each outer do loop in triangulation or back substitution is referred to as a pivoting step. There are n pivoting steps in forward elimination and n-1 in back substitution process.

We assume that each divide or update operation takes one unit of time. It is well known for a full matrix, that the parallel solution of equation (1) takes a minimum of (3n-2) time units, if an unlimited number of processors are provided. Since A is banded, the minimum completion time is expected to be smaller. We show that the minimum completion time for a banded matrix is also (3n-2) and it is independent of bandwidth. This minimum completion time sets a limit on the speed-up.

3. TASK GRAPH

As can be seen in (4), at a given step k, the divide operation on an element a_{kj} cannot be executed until the latest operations on a_{kj} and a_{kk} have been completed at some previous step. Similarly, it is evident from (5), the execution of an update operation on a_{ij} can only begin after the latest operations on a_{ij}, a_{ik}, and a_{kj} have been completed. Wing and Hwang [11], used the acyclic directed graph for parallel solution of a sparse matrix and derived some interesting properties of the graph. There exists a set of precedence relation among the operations. It is convenient to represent these relations in a multi task system as a directed graph [14], G(V,E) consisting of a set of nodes V and a set of arcs E defined as follows [13]:

$V = \{v_i \mid v_i$ is a node if and only if it represents a divide or update operation in the solution process$\}$

$E = \{(v_i, v_j) \mid (v_i, v_j)$ is an arc if and only if v_j requires the results of v_i and (i) v_i, v_j are operations in forward elimination process or (ii) v_i, v_j are operations on elements $a_{k,n+1}$ in the back substitution process$\}$.

We shall refer to G(V,E) as a "task graph" and the three terms "operation", "task", and "node" will be used interchangeably. With the assumption that the execution of each task takes one unit of time, the task graph is a "unit execution time" system.

For the A matrix shown in Fig. 1, the task graph for the forward elimination process is shown in Fig. 2. Task graph represents the solution process of the equations in Fig. 1. Zero elements of A matrix are not shown. The solution process consists of the following operations:

Node number	Operation
1	$a_{12} = a_{12}/a_{11}$
2	$a_{13} = a_{13}/a_{11}$
3	$a_{17} = a_{17}/a_{11}$
4	$a_{22} = a_{22} - a_{21} \cdot a_{12}$
5	$a_{23} = a_{23} - a_{21} \cdot a_{13}$
6	$a_{27} = a_{27} - a_{21} \cdot a_{17}$
7	$a_{32} = a_{32} - a_{31} \cdot a_{12}$
8	$a_{33} = a_{33} - a_{31} \cdot a_{13}$
9	$a_{37} = a_{37} - a_{31} \cdot a_{17}$
10	$a_{23} = a_{23}/a_{22}$
11	$a_{24} = a_{24}/a_{22}$
12	$a_{27} = a_{27}/a_{22}$
13	$a_{33} = a_{33} - a_{32} \cdot a_{23}$
14	$a_{34} = a_{34} - a_{32} \cdot a_{24}$
15	$a_{37} = a_{37} - a_{32} \cdot a_{27}$
16	$a_{43} = a_{43} - a_{42} \cdot a_{23}$
17	$a_{44} = a_{44} - a_{42} \cdot a_{24}$
18	$a_{47} = a_{47} - a_{42} \cdot a_{27}$
19	$a_{34} = a_{34}/a_{33}$
20	$a_{35} = a_{35}/a_{33}$
21	$a_{37} = a_{37}/a_{33}$
22	$a_{44} = a_{44} - a_{43} \cdot a_{34}$
23	$a_{45} = a_{45} - a_{43} \cdot a_{35}$
24	$a_{47} = a_{47} - a_{43} \cdot a_{37}$
25	$a_{54} = a_{54} - a_{53} \cdot a_{35}$
26	$a_{55} = a_{53} - a_{54} \cdot a_{45}$
27	$a_{57} = a_{57} - a_{53} \cdot a_{37}$
28	$a_{45} = a_{45}/a_{44}$
29	$a_{46} = a_{46}/a_{44}$
30	$a_{47} = a_{47}/a_{44}$
31	$a_{55} = a_{55} - a_{54} \cdot a_{45}$
32	$a_{56} = a_{56} - a_{54} \cdot a_{46}$
33	$a_{57} = a_{57} - a_{54} \cdot a_{47}$
34	$a_{65} = a_{65} - a_{64} \cdot a_{45}$
35	$a_{66} = a_{66} - a_{64} \cdot a_{46}$
36	$a_{67} = a_{67} - a_{64} \cdot a_{47}$
37	$a_{56} = a_{56}/a_{55}$
38	$a_{57} = a_{57}/a_{55}$
39	$a_{66} = a_{66} - a_{65} \cdot a_{56}$
40	$a_{67} = a_{67} - a_{65} \cdot a_{57}$
41	$a_{67} = a_{67}/a_{66}$
Back Substitution	
42	$a_{47} = a_{47} - a_{46} \cdot a_{67}$
43	$a_{57} = a_{57} - a_{56} \cdot a_{67}$
44	$a_{37} = a_{37} - a_{35} \cdot a_{57}$
45	$a_{47} = a_{47} - a_{45} \cdot a_{57}$
46	$a_{27} = a_{27} - a_{24} \cdot a_{47}$
47	$a_{37} = a_{37} - a_{34} \cdot a_{47}$
48	$a_{17} = a_{17} - a_{13} \cdot a_{37}$
49	$a_{27} = a_{27} - a_{23} \cdot a_{37}$
50	$a_{17} = a_{17} - a_{12} \cdot a_{27}$

In the graph the divide operations are indicated by squares and update operations by circles. Note that (by definition) there are no arcs from the forward elimination steps to back substitution steps except the one through the node 41, which is a distinct node.

4. PROPERTIES OF TASK GRAPH

Referring to the algorithm in Section 2, we see that the task graph can be constructed in stages. Let $G_k(V_k, E_k)$ be a subgraph constructed after the completion of pivoting step k. Then G_n is the subgraph representing forward elimination. In this, there are 2n-1 steps involved and there are n-1 steps in the back substitution process. It is proved in [13] that this graph is acyclic. The definition in [13] states that "A path of length k from node v_i to node v_j is a sequence of k nodes $v_{r1}, v_{r2}, \ldots, v_{rk}$ such that $v_{r1} = v_i$ and $v_{rk} = v_j$ and $(v_{rm}, v_{rm+1}) \in E$ for all $i \le m \le k-1$".

An arc (v_i, v_j) is redundant (transitive [14]) if there exists a path of length greater than two from v_i to v_j. The reason for the emphasis on redundant arcs is that they illustrate the existence of an optimal scheduling strategy using two processors for unit execution time. This strategy is optimal only if all redundant arcs in the graph are eliminated by appropriate task scheduling to multiple processors [15]. The depth of a node v_i [11-13] is the length of the longest path from the initial node to v_i. In fact it is the earliest time at which v_i can be executed. The depth of the task graph G(V,E), denoted by D, is the longest path in G. It is the minimum completion time of the solution process even if an unlimited number of processors are used. The level number u_i of a node v_i is defined as

$$u_i = D+1 - \max_k L_k \qquad L_k \text{ is the length of a path from } v_i \text{ to a terminal node}$$

The level number is simply the latest time by which node v_i must be processed in order to complete the task graph in minimum time D.

Lemma 1: The longest path D, for the task graph G(V,E) of a banded matrix is always 3n-2.

Proof: For a given value of k in the forward elimination, the divide nodes are at level p(p = 2k-1); the update nodes for i = k+1 are at level p+1 and the update nodes for i > k+1 are at level p+2. The total number of levels in the forward elimination is (2n-1). There are (n-1) steps (also n-1 levels) in the back substitution. So there are 3n-2 levels in the solution process. There are arcs only from a node at level p to nodes at level p+1 and p+2. We also know that there is atleast one node at every level. Therefore the longest path is the one which is passing through the nodes at all levels and hence it is 3n-2. It is also known that for a full matrix, the solution takes a minimum of 3n-2 time units [13].

Lemma 2: The longest path is independent of bandwidth b.

Proof: In the forward elimination process, we see that for every value of k, there are b divide operations and b*(b-1) update operations. If the value of bandwidth b is increased, it only increases the number of nodes at a given level and it does not increase the number of levels. In the banded matrix case the number of levels is the same as D. The longest path pass through at least one node at every level. Hence the longest path D (which is 3n-2) is independent of bandwidth. Also this is the reason that minimum time for the solution process is 3n-2 for both banded and full matrix, if provided with an unlimited number of processors.

The task graph for the banded matrix case is also acyclic, and the properties derived for sparse matrix in reference [13] are applicable to banded matrix also. The total number of nodes in the Gaussian elimination process are:

$$(n-b)^2 b + \sum_{l=0}^{b}(b-1)^2 + (n-b)(b-1) + \sum_{l=1}^{b-1}(b-1) \qquad (10)$$

In [12,16], lower bounds on the minimum number of processors required to complete a general task graph and lower bounds on the completion time if a fixed number of processors are used are derived. These results are very useful and we apply these results to our banded matrix case. Let p* be the minimum number of processors required to complete the task graph in D time units. Then from [12]

$$p^* \geq \max_i \left\lceil \sum_{k=1}^{i} n_k / i \right\rceil \qquad (11)$$

where n_k is the number of nodes at level k. We know at any level the number of nodes is a function of half bandwidth b. Increasing b, increases the number of nodes at any level and thus results in an increase in the value of p* in equation (11). Hence, we can say that the optimal number of processors required to complete the task graph in D time units is fixed by the value of half bandwidth b. Also increase in n only increases the value of D and not the number of nodes at any level. Let t* be the minimum completion time to process a task graph with p processors. Then from [12],

$$t^* \geq \max_i \left\lceil \frac{\sum_{i=1}^{i} n_k}{p} + D - 1 \right\rceil \qquad (12)$$

The speed-up [8] is defined in most parallel algorithms as

$$\text{speed-up} = \frac{\text{user time for one processor}}{\text{user time for N processors}}$$

From the task graph (without considering the communication time in between processors), we can see that the user time with one processor is the total number of nodes in the task graph (unit execution time). We have also proved that D which is (3n-2) is the longest path in the graph, is the minimum completion time of the solution process if an unlimited number of processors are used. Hence the maximum speed-up that can achieved is

$$\text{maximum speed-up} = \frac{\text{user time for one processor}}{\text{D, the longest path in the graph}}$$

Also the efficiency of a parallel algorithm is defined

$$\text{Efficiency} = \frac{\text{speed-up}}{\text{number of processors}}$$

Now we will show Amdhal's law [17,18] and its significance in our banded matrix case. Amdhal's law states: "A small number of sequential operations can effectively limit the speed-up of a parallel algorithm". Let f be the fraction of operations in a computation that must be performed sequentially, where $0 \leq f \leq 1$. It is easy to see that the maximum speed-up achievable by a parallel computer with p processors is

$$S \leq \frac{1}{f + (1-f)/p} \qquad (13)$$

For example, if the sequential operations is 10% of the total computation, then the maximum achievable speed-up is 10. We can see from equation (13) that with an infinite number of processors, the maximum achievable speed-up is 1/f. The same result is also obtained in our analysis. From our analysis, we can see the relation of f as

$$f = \frac{\text{D the longest path in the graph}}{\text{total number of nodes in the graph}}$$

and the maximum speed-up achievable is 1/f, no matter how many processors a computer has.

5. ILLUSTRATIVE EXAMPLE

Example 1: A 10x10 matrix with half bandwidth 3 is considered. There are 77 nodes in the forward elimination and 17 nodes in the back substitution. The minimum completion time (D) is found to be 28. The minimum number of processors (p^*) required to complete the task graph in 28 time units is obtained as 5. Table 1 shows the minimum completion time, speed-up and efficiency with the increase in number of processors. Beyond 5 processors the speed-up is the same. This can also be obtained from Amdhal's law as the maximum speed-up achievable. Table 2 shows the effect of bandwidth on the maximum achievable speed-up. A 10x10 matrix is considered and the bandwidth is changed from 2 to 10 and the results are shown in Table 2. In Table 3, for a given bandwidth the values of n is changed and the results are shown.

From this, we can say that for an increase in bandwidth, there is a substantial increase in maximum achievable speed-up. On the other hand, for a given bandwidth, for an increase in n, the increase in maximum achievable speed-up is very marginal. This fact is also observed in truss problem considered in [8]. In practice, the total user time increases, beyond the optimum number of processors, because of the communication time in between processors.

Scheduling: References [11,13], gives the optimal scheduling strategies to process a task graph. We use the modified scheduling strategy of [11]. We define a ready-task [13] to be one whose immediate predecessors have all been processed. The modified level scheduling [13] is as follows:

) among all the ready tasks, schedule the one with the smallest level number and
if there is a tie, schedule the one with the largest number of immediate successors. This algorithm is applied to the above example 1, and the schedule with 3, processors is shown in Fig. 3. In Fig. 3, a '*' in the schedule indicates that the processor is idle and the number indicates the operation to be done.

6. COMMUNICATION ASPECTS

In the previous sections, the task graph is used to compute the absolute minimum time and the optimal number of processors to complete it in minimum time are discussed. In this section, we will incorporate the communication time in between processors. This analysis is important because in a parallel processing environment, the overall execution time is the sum of computation time and the interprocessor communication time. Our studies are with respect to loosely coupled system, where communication in between processors occurs by exchange of messages through a network and not through a shared main memory as in the case of a tightly coupled system.

Let t_{comp} be the computation time for one node (division or update) in the task graph. Let t_{comm} be the transfer (communication) time to transfer one value from one processor to its neighbor. Because of our scheduling algorithm, each computation done by a processor has to be communicated to all the remaining processors. The time taken to complete the task graph with p processor is t^*. Hence there are (t^*-1) parallel data transfers in the algorithm. Hence the total time for the parallel algorithm with p processor is defined as

$$\text{total time} = t_{comp}*t^* + t_{comm}(t^*-1)*d \qquad (14)$$

where d is a factor that depends on the interconnection network. d is the maximum number of steps needed in the interconnection network to send a data from one processor to all the other processors. We consider four types of interconnection networks with a limited number of processors. The four type of networks considered for time complexities are: mesh array, 3-D mesh array, barrel shifter and cube connection array.

For mesh connected network (Illiac IV), with p processors, it takes d steps to route the data from the processor to any processor, where d is upper bounded by $d \leq p-1$. For cube connected network with p processors the value is: $d \leq \log_2 p$. For barrel shifter networks $d \leq \log_2 p/2$ and for three dimensional mesh $d \leq 1.5\, p^{1/3}$ [18]. Now with the value of d corresponding to the network type, we can exactly compute the total time. We can also see from equation (14) that t^* reaches a maximum value D with the increase in number of processors. The value of d increases with the increase in number of processors. Hence the total execution time decreases with the increase in number of processors only up to a certain number of processors.

7. CONCLUSIONS

The usefulness of acyclic directed graph in identifying parallel operations, computing the minimum completion time, the minimum number of processors to complete the graph in minimum time and the maximum achievable speed-up are presented. The usefulness of this graph in improvement of program efficiency is discussed in [19]. The absolute minimum completion time is dependent on the number of equations and independent of the bandwidth. On the other hand, maximum achievable speed-up and the optimal number of processors required to complete the job in minimum time are dependent on the half bandwidth and is independent of the number of equations. A method of incorporating the interprocessor communication time and its effect on the overall computation time is also brought out. This study is useful for engineers working with large system of equations on a multiprocessor system.

REFERENCES

[1] A.K. Noor and S.N. Atluri, 'Advances and trends in computational structural mechanics', AIAA Jnl. 25, 1987, pp. 977-995.

[2] D. Heller, 'A survey of parallel algorithms in numerical linear algebra', SIAM Review, 20, 1978, pp. 740-777.

[3] W.L.Miranker,'A survey of parallelism in numerical analysis',SIAM Review,13,1971, pp.524-547.

[4] K.A. Gallivan, R.J. Plemmons and A.H. Sameh, "Parallel algorithms for dense linear algebra computations", SIAM Review, Vol. 32, No. 1, pp. 54-134, March 1990.

[5] J.M. Ortega and R.G. Voigt, 'Solution of parallel differential equations on vector and parallel computers', NASA Report CR.172500 or ICASE Report No. 85-1, 1985.

[6] K.H. Law, 'A parallel finite element solution method', Computers and Structures, 23, 1986, pp. 845-858.

[7] W.T. Carter Jr., T.L. Sham, K.H. Law, 'A parallel finite element method and its prototype implementation on a hypercube', computers and Structures, 31,1989, pp. 921-934.

[8] L.S. Chien and C.T. Sun, 'Parallel processing techniques for finite element analysis of nonlinear large truss structures', Computers and Structures, 31, 1989, pp. 1023-1029.

[9] L.S. Chien and C.T. Sun, 'A parallel Gaussian elimination procedure for finite element analysis', NASA Technical Report, NASA Langley Research Center, July, 198).

[10] R.E. Lord, J.S. Kowalik and S.P. Kumar, 'Solving linear algebraic equations on an MIMD

computer', Journal of the Association for Computing Machinery, 30, 1983, pp. 103-117.

[11] T.C. Hu, 'Parallel sequencing and assembly line problems', Operations Research, 9, 1961, pp. 844-848.

[12] C.V. Ramamurthy, K.M. Chandy and M.J. Gowzalez, Jr., 'Optimal scheduling strategies in a multiprocessor system', IEEE .Trans. on Computers, C-21, 1972, pp. 137-146.

[13] O.Wing and J.W. Huang, 'A computation model of parallel solution of linear equations', IEEE Trans.on Computers, C-29,1980, pp.632-638.

[14] E.G.Coffman, Jr.(Ed.), 'Computer and Job Shop Scheduling Theory', New York, Wiley 1976, Chap I, pp. 1-50.

schedule', in Computer and Job Shop Scheduling Theory, E.G. Coffman Jr (Ed.) New York, Wiley, 1976, pp. 51-99.

[16] E.B. Fernandez and B. Bussel, 'Bounds on the number of processors and time for multiprocessor optimal schedule', IEEE Trans. on Computers, C-22, 1973, pp. 745-751.

[17] M.J. Quinon, 'Designing Efficient Algorithms for Parallel Computers', McGraw-Hill, New York, 1988.

[18] Hwang, K. and Briggs, F.A., 'Computer Architecture and Parallel Processing', McGraw-Hill, New York, 1984.

[19] C.Q. Yang and B.P. Miller, "Performance measurement for parallel and distributed programs: A structured and automatic approach", IEEE Transactions on Software Engineering, Vol. 15, No. 12, Dec. 1989, pp. 1615-1629.

Table 1: Performance measures of interest

Number of processors, p	Minimum completion time	Speed-up S	Efficiency%
1	94	1.00	100
2	48	1.9583	97.92
3	36	2.6111	87.04
4	30	3.1333	78.33
5	28	3.3571	67.14
6	28	3.3571	55.95
7	28	3.3571	47.95
8	28	3.3571	41.96
9	28	3.3571	37.30
10	28	3.3571	33.57

Table 2: Effect of bandwidth on speed-up; n = 10

bandwidth	total number of nodes	D	f	max.speed-up S
2	46	28	0.6087	1.6429
3	94	28	0.2929	3.3571
4	150	28	0.1867	5.3571
5	210	28	0.1333	7.5000
6	270	28	0.1037	9.6429
7	326	28	0.0859	11.6229
8	374	28	0.0749	13.3571
9	410	28	0.0683	14.6429
10	430	28	0.0651	15.3572

Table 3: Effect of n on speed-up; b = 4

n	D	total number of nodes	f	max. speed-up S
10	28	150	0.1867	5.3571
15	43	245	0.1755	5.6977
20	58	340	0.1706	5.8621
25	73	435	0.1678	5.9389
30	88	530	0.1660	6.0227
35	103	625	0.1648	6.0680
40	118	720	0.1639	6.1017
45	133	815	0.1632	6.1278
50	148	910	0.1626	6.1486

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & & & & a_{17} \\ a_{21} & a_{22} & a_{23} & a_{24} & & & a_{27} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & & a_{37} \\ & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & a_{47} \\ & & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} \\ & & & a_{64} & a_{65} & a_{66} & a_{67} \end{bmatrix}$$

Fig. 1 Matrix under consideration

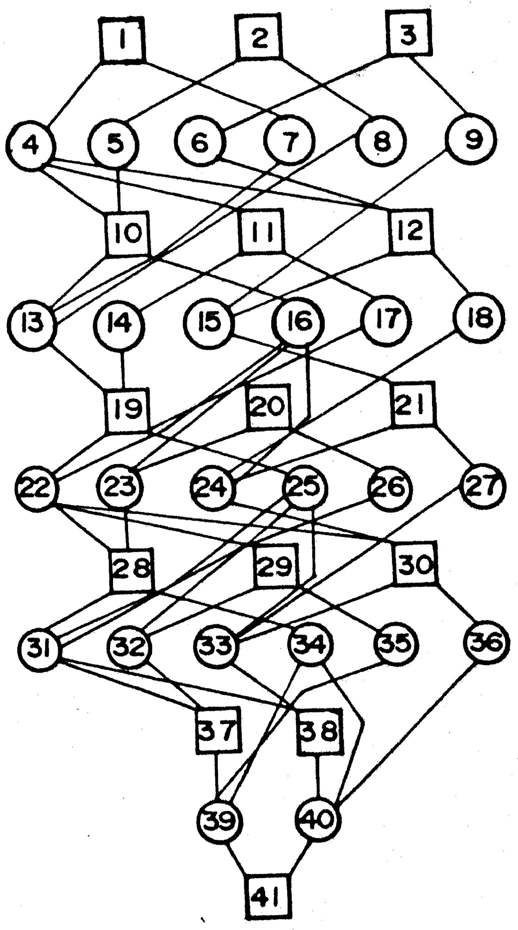
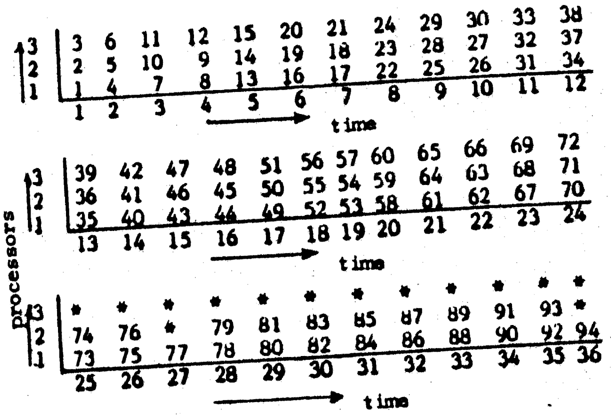


Fig. 2. Task Graph


processors
time
time
time

processors	1	2	3	4	5	6	7	8	9	10	11	12
3	3	6	11	12	15	20	21	24	29	30	33	38
2	2	5	10	9	14	19	18	23	28	27	32	37
1	1	4	7	8	13	16	17	22	25	26	31	34

processors	13	14	15	16	17	18	19	20	21	22	23	24
3	39	42	47	48	51	56	57	60	65	66	69	72
2	36	41	46	45	50	55	54	59	64	63	68	71
1	35	40	43	44	49	52	53	58	61	62	67	70

processors	25	26	27	28	29	30	31	32	33	34	35	36
3	*	*	*	*	*	*	*	*	*	*	93	*
2	74	76	*	79	81	83	85	87	89	91	92	94
1	73	75	77	78	80	82	84	86	88	90	[illegible]	[illegible]

Fig. 3 Schedule with 3 processors