

A Lowerbound on the Schedule Time for Scheduling Tasks on Distributed Memory Systems

Sekhar Darbha
ECE Department
Rutgers University
Piscataway, NJ 08855-0909
darbha@ece.rutgers.edu

Extended Abstract

Obtaining an optimal schedule for assigning tasks onto distributed memory machines (DMMs) has been proven to be NP-Complete. Several heuristic solutions to the scheduling problem have been proposed which are based on certain assumptions that allow the algorithm to be executed in a polynomial time bound. Even though the heuristics do not guarantee an optimal solution, they have been shown to perform reasonably well for many applications. It is difficult to judge the performance of these heuristics because for a given directed acyclic graph (DAG), it is practically infeasible to obtain the optimal solution.

This paper proposes a method to compute the lowerbound in a polynomial time bound. The idea of the lowerbound is to obtain a solution as close to optimal as possible. Also, the lowerbound should be less than or equal to the optimal solution.

The lowerbound with the inclusion of the inter-processor communication costs has been proposed in [1]. We have observed that for some cases this algorithm does not give a sharp lowerbound. Our scheme provides a sharper lowerbound than the one proposed in [1]. Another scheme is presented in [4] which partitions the graph into certain parts and compute the bound for each part to arrive at a sharper bound. To compute the bound of each partition, they use the same method as proposed in [1]. Thus, our scheme can be used to improve upon the bound proposed in [4] also.

We assume that the task graph is available in the form of a DAG defined by the tuple (V, E, τ, c) , where V is the set of task nodes, E is the set of communication edges, τ is the set of node computation costs and c is the set of edge communication costs. The computation cost of any node i is depicted as $\tau(i)$ and the communication cost of an edge from task i to task j is depicted as $c_{i,j}$. Without loss of generality it can be assumed that there is one entry node and one exit node. If there are multiple entry or exit nodes, then the multiple nodes can always be connected through

a dummy node which has zero computation cost and zero communication cost edges.

The algorithm to compute the lowerbound starts from the entry node and terminates at the exit node. At each stage, the earliest start time (est) and the earliest completion times (ect) of a node are computed. These times are computed using the equations and definitions shown below.

$PRED(i)$: Set of predecessors of any task i , i.e. tasks on which task i is data dependent.

$est(i) = 0$, for the entry node

$$est(i) = \min_{j \in PRED(i)} \max_{k \in PRED(i), k \neq j} (ect(j), ect(k) + c_{k,i}) - \Delta(i)$$

$$ect(i) = est(i) + \tau(i)$$

The value of $\Delta(i)$ is computed in the algorithm for each node i and is deducted from the earliest start time of the task i . The value of $\Delta(i)$ for a task i is zero if it satisfies one of the two conditions; (i) the number of predecessors of task i is less than or equal to one or (ii) The task i satisfies the cost-relationship condition (CRC) given below.

To examine if the CRC is satisfied at node i , we need to compute the array ecc for each node i . The array ecc is defined as follows:

$$ecc_i(j) = (ect(j) + c_{j,i}) \quad j \in PRED(i) \quad (1)$$

Let m and n be the tasks that have the highest and second highest values of $ecc(j)$ for all $j \in PRED(i)$ respectively. Then CRC is satisfied at node i if,

- $\tau(m) \geq c_{n,i}$ if $est(m) \geq est(n)$ or,
- $\tau(m) \geq (c_{n,i} + est(n) - est(m))$ if $est(m) < est(n)$

There can be three possible cases for m and n . In the first case only one value of m and n possible. If only one distinct value of m and n are possible then the CRC given above needs to be satisfied for $\Delta(i)$ to be zero. In the second case, there is only one m , but multiple n are existent. In this case $\Delta(i)$ will be zero if there is at least one pair of m and n which satisfy the CRC . In the third case, there are multiple values

of m . In this case $\Delta(i)$ will be zero if there is at least one pair of m and n which satisfy the CRC . In this case, the value of n is taken from the pool of values for m .

If the CRC is satisfied for a node i , then the value of $\Delta(i)$ will be zero because we cannot lower the start time of task i [3]. If CRC is not satisfied, we need to compute the value of $\Delta(i)$. This value of $\Delta(i)$ is needed to obtain the lowest possible value of the starting time of task i or $est(i)$. Due to space constraints the details of the procedure to compute $\Delta(i)$, in case CRC is not satisfied, is not provided here. The details of the procedure to compute $\Delta(i)$ are given in [2].

The performance of the algorithm to compute the lowerbound in cases where CRC is not satisfied is illustrated for two cases. In the first case, we take the example DAG shown in Figure 1 and compute the lowerbound obtained by our algorithm and by the algorithm presented in [1]. The start and completion times for all the tasks of the DAG obtained by the two algorithms are shown in Table 1. The lowerbound obtained by our algorithm is 29 whereas the lowerbound obtained by the earlier algorithm is 24. The optimal schedule for this DAG is 29. Thus, our algorithm gives a better bound than the earlier algorithm.

For the second case, we have taken the DAG shown in Figure 2. For this DAG, we varied the costs shown on the figure as a-h in the range of 1 to 4. This gives us 65536 (4^8) data sets. For each of these data sets we computed three quantities; (i) lowerbound with our algorithm, (ii) lowerbound with earlier algorithm, and (iii) optimal Schedule. The DAG shown in Figure 2 has five tasks. These five tasks can be scheduled on one to five processors in 52 different ways. For each data set, we compute the schedule time obtained by these 52 allocations and assign the lowest schedule time to the optimal schedule.

From these results, we have observed the following: (i) The lowerbound obtained by our algorithm is always higher than the lowerbound obtained by the earlier algorithm. For the cases in which our lowerbound is higher than the earlier lowerbound, our lowerbound is 16% higher than the earlier lowerbound. (ii) The lowerbound obtained by our algorithm is less than or equal to the optimal schedule for all the data sets. The lowerbound is within 98% of the optimal schedule on the average.

References

[1] M.A. Al-Mouhamed, "Lower Bound on the Number of Processors and Time for Scheduling Precedence Graphs with Communication Costs", *IEEE Transactions on Software Engineering*, vol. 16, no.12, December 1990, pp. 1390-1401.

[2] S. Darbha, "A Lowerbound on the Schedule Time for Scheduling Tasks on Distributed Memory Machines", *Tech. Report, ECE Department, Rutgers University*.

[3] S. Darbha and D.P. Agrawal, "Scalable Scheduling Algorithm For Distributed Memory Machines", *To appear in Eighth IEEE Symposium on Parallel & Distributed Processing*, October 23-27 1996, New Orleans, LA.

[4] K.K. Jain and V. Rajaraman, "Improved Lower Bounds on Time and Processors for Scheduling Precedence Graphs on Multicomputer Systems", *Journal of Parallel and Distributed Computing*, vol. 28, 1995, pp. 101-108.

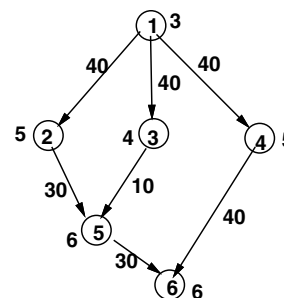


Figure 1: Example Directed Acyclic Graph

Table 1: Start and Completion Times

Node	Earlier Algorithm		Our Algorithm	
	EST	ECT	EST	ECT
1	0	3	0	3
2	3	8	3	8
3	3	7	3	8
4	3	8	3	8
5	12	18	12	18
6	18	24	23	29

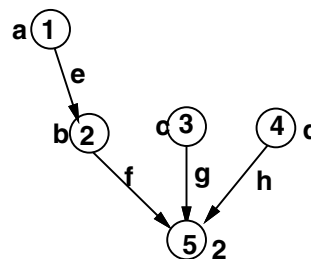


Figure 2: Directed Acyclic Graph