Limit Value Task Scheduling (LVTS): an Efficient Task Scheduling Algorithm for Distributed Computing Environment

G.K.Kamalam, K.Sentamilselvan

Abstract**: *The grid computational environment suits to meet the computational demands of large, diverse groups of tasks. Assigning tasks to heterogeneous wide spread resources seems complex and is termed as an NP-Complete problem. A new task scheduling algorithm, called Limit Value Task Scheduling Algorithm (LVTS) is presented to efficiently identify the appropriate resources, which is responsible for the scheduling process. The proposed algorithm (LVTS) schedules the tasks to the appropriate resources by calculating the limit value of the tasks and the ceil value of the tasks which represents the completion time of the last tasks scheduled in the resource with highest processing capacity. The efficiency of the (LVTS) measured based on makespan and resource utilization. Experimental results indicates LVTS algorithm sounds good than the Min-min on both makespan and resource utilization.***

Keywords: Grid, Heuristic scheduling, Job Scheduling, Task Scheduling

I. INTRODUCTION

Grid consists of distributed large scale heterogeneous computing environment and it comprises the various resources are connected in the high speed networks [1]. This enables scientific or complex problems to solve using available computing machine. Frequently, computing resource and the tasks are dynamically changing. Hence, effective utilization of the resource and task scheduling becomes a serious issue [2].

Computational grid is a big task can be divided into small task and this can be allocate in different individual machine and execute. Finally it will be compiled and make a results. The task is allocated to the resource which completes the execution of the task at less time. Resource utilization is found based on resource idle time. Performance of task scheduling determined based on makespan. makespan value is the maximum value among 'n' tasks completion time.

For task scheduling, Min-Min, Max-Min, MCT, Heuristic scheduling algorithm, Simulated Annealing algorithm and Genetic algorithm had achieved good results for NP-Complete problems [3].

The heuristic scheduling algorithm is categorized in two different ways, such as batch or online or mode. In the batch mode, the tasks are independent, this will execute in any order and the tasks will be scheduled by batch by batch. In an online mode, the task is scheduled in based on the available resources and as soon as possible it arrives.

II. LITERATURE REVIEW

Min-min algorithm, is very simple, runs fast and provides better performance. Min-min algorithm schedules all simple tasks that are not yet mapped. For each task, waiting time and execution time on every machine is calculated. Algorithm spends less time to complete all tasks. But, this leads to load imbalance [4].

Max-min scheduling algorithm is similar to Min-min, task with maximum among 'n' minimum completion time on all machines is found out and assigned to the particular resource [5].

In On-line mode heuristic algorithm, tasks are scheduled when they arrive at earliest (as soon as) to the system. This is more suitable to grid environment.

Opportunistic Load Balancing (OLB) algorithm will select the task at random manner for the next available machine. But, this will not consider the task completion and execution time. So, it gives the poor makespan time. The main objective is utilizing the resource effectively. MET algorithm assigns every task to available resource with minimum execution time. The drawback of MET is, the availability of the machine is not considered and leads to imbalance of load. Advantages of OLB, MET algorithms are merged and developed a new algorithm called Minimum Completion Time algorithm. At one time, MCT schedules a single task. For every task on every machine task waiting and execution time value computed. Task with minimum value allocated to the corresponding machine. This causes that particular machine may have the better result for any other task [1].

Min-mean schedules in phases. First phase, the minimum among 'n' minimum completion time of a task on every available resource is found out, and the task is allocated to the corresponding resource. The process is repeated for all 'n'tasks.

The mean completion time of all the tasks is computed. Second phase, the tasks whose completion time greater than mean completion time is rescheduled [2].

Manuscript published on November 30, 2019.
***** Correspondence Author

Dr.G.K.Kamalam*, Assistant Professor(SLG) in Kongu Engineering College Erode
Mr.K.Sentamilselvan, Assistant Professor in Kongu Engineering College Erode

DMMCWTSA schedules tasks in stages. First stage, TW (Task Worth) value computed. Tasks are ordered in WS (Worth Set) based on minimum TW value. The tasks ordered in WS scheduled to resource with CT value (Completion Time).

Second stage, tasks rescheduled if rescheduling leads to reduced makespan value [8].

In Resource Fitness Task Scheduling Algorithm (RFTSA) allocates scheduled tasks based on the calculated fitness value to the available machine [7].

The heuristic scheduling algorithm aims to minimize the makespan, i.e., completion time of the metatask as soon as possible [9.10]. The proposed algorithm (LVTS) schedules the tasks to the appropriate resources by calculating the limit value of the tasks and the ceil value of the tasks which represents the completion time of the last tasks scheduled in the resource with highest processing capacity. The proposed algorithm performs the mapping of the tasks the resources efficiently and achieves reduced makespan and better resource utilization.

III. MAPPING APPROACH

A. Assumptions and Notations are as follows [1]:

Assumptions:
meta-task is a set of independent and non-communicating tasks is considered for scheduling.
Static mapping is carried out for scheduling tasks to resources
Each resource executes a single independent task at a time.
The sizes of the tasks and the number of resources are static and known a priori.
ETC matrix of size n*m, represents the accurate estimate of the expected execution time of each job on each resource where n-represents the number of tasks and m represents the number of resources.

Notations	Definitions
$Length_i$	Length of the task T_i in MI
$Power_j$	Processing capacity of resource R_j in MIPS
$ETC(Ti,Rj) = Lengthi/Powerj$	expected execution time of the task T_i, on resource R_j
$T= \{T_1, T_2, \dots, T_n\}$	Task set
$R= \{R_1, .., R_m\}$	Resource set
CTT_i	expected completion time of task T_i
RT_j	ready time of resource R_j
CT_j	expected completion time of all task T_i scheduled on resource R_j.
$makespan = \max(CTT_i)$	overall completion time of all tasks $1 \le i \le n$

B. Limit Value Task Scheduling (LVTS) Algorithm

(LVTS) algorithm aims to solve the task-scheduling problem efficiently. The main idea of the proposed algorithm is inspired from the traffic rules in the highways, where the width of the road has an important effect in increase in the number of vehicles in the traffic. This leads to the under utilization of the other path exists for the same destination. By considering the two concepts, the proposed algorithm is designed. Thus the proposed algorithm (LVTS) allocate a submitted group of tasks by using this rule, where the number of tasks that are allocated to resources proportion to the processing capacity of the resources. The highest number of tasks should be scheduled into the resource that has highest capacity. This leads to the under utilization of the other available resources. The proposed algorithm (LVTS) schedules the tasks to the suitable resources by calculating the effective finish time of the tasks and completion time of the resources which has the highest processing capacity. That is, the algorithm schedules each resource the best fit group of tasks to achieve approximately finish time equal to LV.

The LVTS algorithm consists of two stages: grouping stage and an allocating stage.

1. Grouping stage: Firstly, the information about resource availability and task requirements is considered. Secondly, the algorithm sorts the arrived tasks into descending order of the length of the task and the order is maintained in OL (Task order list). Thirdly, the algorithm calculates the Effective Finish Time (EFT) of the scheduling model.

Equation (1) computes the Total Lengths of the tasks

$$TL = \sum_{i=1}^{n} \mathrm{MI}_i \qquad (1)$$

Equation (2) calculates the Total Processing Capacity of the resources

$$TPC = \sum_{j=1}^{m} \mathrm{MIPS}_j \qquad (2)$$

Equation (3) calculates the Effective Finish Time of the tasks

$$EFT = \frac{TL}{TPC} \qquad (3)$$

Equation (4) specifies the number of tasks to be scheduled on eash resource

$$LV = \frac{EFT}{Number\ of\ Resources} \qquad (4)$$

Task ordered list (OL) indicates the task order to be mapped to the resource until the completion time of the tasks is less than the effective finish time of the tasks. Otherwise, the remaining tasks are scheduled to the next resource and so on. It is stated in the other way, each resource_j should execute only a certain number of tasks, so that the number of tasks where the total completion time $TCT(\text{resource}_j)$ of the tasks submitted to the resource_j is approximately equal to the LV value, where the TCT (resource_j) is the summation of completion time of all tasks that are assigned to resource_j. The limit value LV is used as a reference for all resources to identify the time that each resource should be spent in executing the tasks.

2. Allocating stage: In this stage, the proposed algorithm (LVTS) assigns tasks to resources according to the limit value LV. The resource that has the highest processing power compared to that of the other available resource is selected and its completion time value is considered as ceil value (CV) for selecting the tasks for rescheduling. The tasks whose completion time greater than the ceil value (CV) is selected. The selected tasks are rescheduled to the resources whose completion time is minimum compared to that of the completion time of the previously allocated resource.

C. Limit Value Task Scheduling (LVTS) Algorithm:

The proposed algorithm works in two phases.

```
First Phase:
The length of the tasks for 'n' tasks is stored inTL[n]
Input for m resources, the processing power of the resources as PP[m]
    //sort the tasks list in the descending order of the length of the tasks

$for\ i = 1\ to\ n\ do$
 $for\ j = i + 1\ to\ n\ do$
   $if\ (TL[i] < TL[j])$
      $temp = TL[i]$
      $TL[i] = TL[j]$
      $TL[j] = temp$
//sum of the length of the tasks
$for\ i = 1\ to\ n\ do$
   $TotalLength += TL[i]$

//sum of the processing power of the resources
$for\ i = 1\ to\ m\ do$
   $TPC += PP[i]$
//Effective Finish Time of the tasks is calculated as
$$EFT = \frac{TL}{TPC}$$
//Limit Value of the tasks is calculated as,
$$LV = \frac{EFT}{Number\ of\ Resources}$$
$for\ i = 1\ to\ m\ do$
    $CT_i = 0$
   $RT_i = 0$

$for\ i = 1\ to\ n\ do$
   $selected[i] = -1$

$for\ i = 1\ to\ n\ do$
   $if(selected[i] == -1)$
   $for\ j = 1\ to\ m\ do$
      $k = 0$
      $if(CT_j < PFT)$
         $CTj = ETCij + RTj$
            $selected[i] = 1$
           $RT[j] = CT[j]$
           $CTT[i] = CT[j]$
          $scheduled[i] = j$
Second Phase:

//finding the ceil value
$highpower = PP[1]$
$j = 1$
$for\ i = 2\ to\ n\ do$
    $if(PP[i] > highpower)$
       $highpower = PP[i]$
       $j = i$
$CV = CT[j]$

$for\ i = 1\ to\ n\ do$
   $if(CTT[i] > CV)$
      $min = CT[scheduled[i]]$
      $for\ j = 1\ to\ m\ do$
         $t = CT[j] + ETC[i][j]$
         $if\ (t < (min)$
            $min = t$
   $CTT[i] = min$
   $CT[j] = min$
   $CT[scheduled[i]] = min$

$makespan = CTT[1]$

$for\ i = 2\ to\ n\ do$
   $if(CTT[i] > makespan)$
      $makespan = CTT[i]$
```

IV. RESULTS AND DISCUSSION

A. Example

The data considered for illustration is eight tasks and three resources. The length of eight tasks is shown in Table I. The processing capacity of three resources is shown in Table II. The ETC matrix is constructed and is given in Table III.

Table - I: Tasks length with millions of instructions (MI).

Tasks	Length
T1	25100
T2	30800
T3	242700
T4	68000
T5	6400
T6	175900
T7	116800
T8	36700

Table – II: Processing Capacity of Resources

Resources	Processing Capacity
R1	5000
R2	3000
R3	1000

Table–III: Consistent ETC Matrix Model- high task/resource heterogeneity

Tasks	R1	R2	R3
T1	5.02	8.37	25.1
T2	6.16	10.27	30.8
T3	48.54	80.9	242.7
T4	13.6	22.67	68
T5	1.28	2.13	6.4
T6	35.18	58.63	175.9
T7	23.36	38.93	116.8
T8	7.34	12.23	36.7

TL=2884400
TPC=9000
EFT = 320.49
LV = 106.83
Ordered list OL = {t3, t6, t7, t4, t8, t2, t1, t5}.

During first phase of the algorithm, the ordered list (OL) gives the order of the tasks to be scheduled to the resource and the scheduling is continued until the completion time of the tasks is less than the Limit Value (LV) of the tasks. Otherwise, the remaining tasks are scheduled to the next resource and so on. Resource Task Pair selected during first phase of algorithm is shown in Table IV. Completion Time of tasks on the Resources during first phase of algorithm is shown in Table V.

Table – IV: Resource Task Pair selected during first phase of algorithm (LVTS)

R1	R2	R3
T3, T6	T7, T4, T8, T2, T1, T5	--

Table–V: Completion Time of tasks on the Resources during first phase of (LVTS) Algorithm

Tasks	Completion Time		
	R1	R2	R3
T3	48.54	--	--
T6	83.72	--	--
T7	--	38.93	--
T4	--	61.6	--
T8	--	73.83	--
T2	--	84.1	--
T1	--	92.47	--
T5	--	94.6	--

Table VI shows makespan produced by Min-min algorithm, proposed algorithm (LVTS) and the task, resource selected pair in each algorithm at the end of first phase.

Table-VI: makespan comparison after first phase

Algorithm	R1	R2	R3	makespan
Proposed Algorithm	T3, T6	T7, T4, T8, T2, T1, T5	--	94.6
Min-min Algorithm	T5, T1, T8, T4, T6, T3	T2, T7	--	110.96

From Table VI, its evident that LVTS performs better compared to Min-min. When compared with resource utilization, improvement has to be done based on the outcome of the first phase.

In second phase, the algorithm works as follows:

CV = 83.72

To achieve efficient scheduling, the proposed algorithm reschedules the tasks which are scheduled on this resource whose CT > CV to a better resource new CT value less than that of the old CT value.

The tasks selected for rescheduling and the resource to which the tasks are rescheduled are shown in the Table VII.

Table –VII: Tasks Selected for Rescheduling

Tasks	Scheduled on resource	Rescheduled on resource
T2	R2	R3
T1	R2	R3
T5	R2	R3

The makespan value, Resource task selected combination for the existing Min-min algorithm and proposed (LVTS) algorithm is shown in Table VIII. From the result it is evident that the proposed (LVTS) algorithm reaches reduced makespan and better utilization of resources.

Table-VIII Comparisons between algorithms in makespan

Algorithm	R1	R2	R3	makespan
Proposed Algorithm (after phase 1)	T3, T6	T7, T4, T8, T2, T1, T5	--	94.6
Proposed Algorithm (after phase 2)	T3, T6	T7, T4, T8	T2, T1, T5	83.72
Min-min Algorithm	T5, T1, T8, T4, T6, T3	T2, T7	--	110.96

Benchmark Model Descriptions

Benchmark Model	Descriptions
Size of the ETC matrix	512X16
Number of tasks	512
Number of resources	16
Different types of ETC matrix	12
In each type number of instances	100
Instances generated based on the three metrics	task heterogeneity, resource heterogeneity, and consistency
Instances represented as	u-x-yyzz.k
u	uniform distribution (in generating ETC matrix)
x	the type of consistency (c-consistent, i-inconsistent, s-semi-consistent or partially consistent).
consistent ETC matrix	whenever a resource r_j executes any tasks t_i faster than resource r_k, then resource r_j executes all tasks faster than r_k
Inconsistent ETC matrix	resource r_j may be faster than resource r_k for executing some tasks and slower for others.
partially-consistent ETC matrix	includes a consistent sub-matrix
Task heterogeneity	Variation in the execution time for a task t_i on all resources
yy	Task heterogeneity (hi-high, lo-low)
Resource heterogeneity	Variation in the execution time of all the tasks on a resource r_j.
zz	resource heterogeneity (hi-high, lo-low)

The twelve various instances classified as three groups each composing of four instances is listed in Table IX.

Table-IX: Matrix ETC Model classification

Consistency	Heterogeneity			
	Task (High)		Task (Low)	
	Resource (High)	Resource (Low)	Resource (High)	Resource (Low)
Consistent	u-c-hihi-0	u-c-hilo-0	u-c-lohi-0	u-c-lolo-0
Inconsistent	u-i-hihi-0	u-i-hilo-0	u-i-lohi-0	u-i-lolo-0
Partially-consistent	u-pc-hihi-0	u-pc-hilo-0	u-pc-lohi-0	u-pc-lolo-0

B. Criteria for Evaluation

Fig. 1 shows that the Limit Value Task Scheduling (LVTS) Algorithm provides better makespan than Min-min Heuristic Scheduling Algorithm. Fig. 2, 3, 4, 5 compares the makespan of Min-min, LVTS for four different cases of consistency, task/resource heterogeneity.

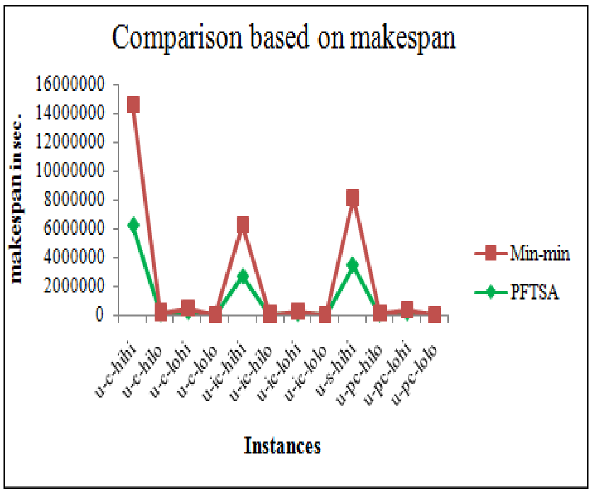

Fig.1. makespan value

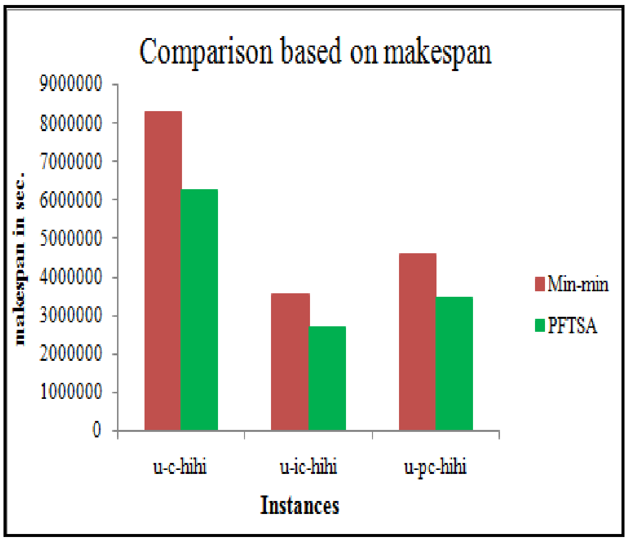

Fig. 2. makespan value for task resource heterogeneity-High

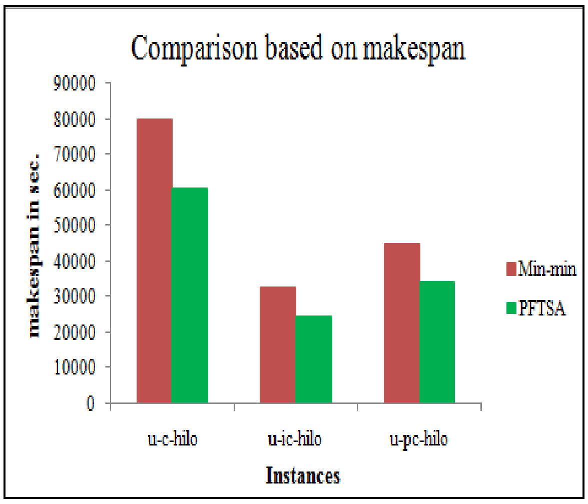

Fig. **3. makespan value for high task, low resource heterogeneity**

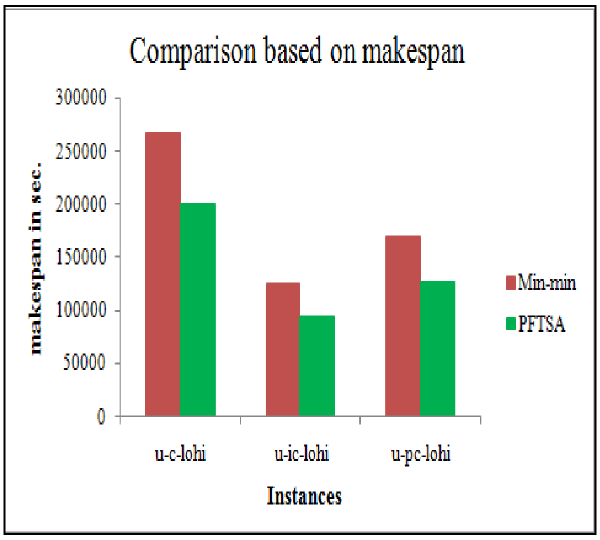

Fig. 4. makespan value for low task, high resource heterogeneity

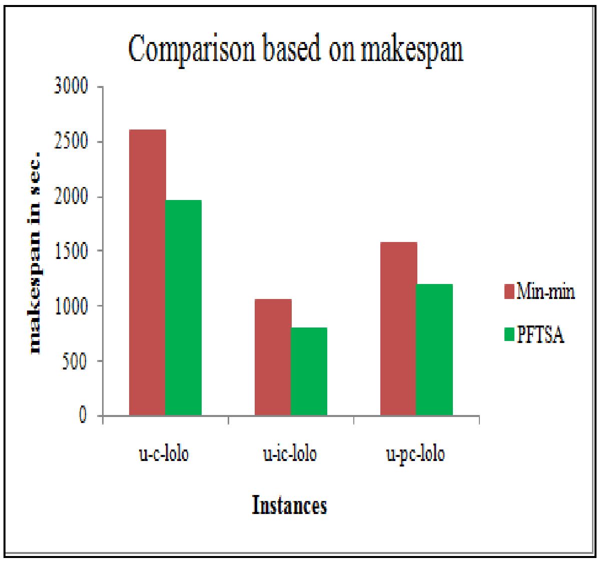

Fig. 5. makespan value for task resource heterogeneity-Low

V. CONCLUSION AND FUTURE WORK

Evaluation of the LVTS algorithm and existing Min-min algorithm is carried out using Braun et. al.[1] benchmark simulation model. Results clearly depict (LVTS) produces good makespan and resource utilization than Min-min. The proposed LVTS algorithm is limited with computational intensive tasks and future scope is to deal with data-intensive tasks.

REFERENCES

1. T.D. Braun, H.J.Siegel, and N.Beck., "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems", Journal of Parallel and Distributed Computing 61, 2001, pp.810-837.
2. G.K.Kamalam, and Dr.V.Murali Bhaskaran., "A New Heuristic Approach:Min-mean Algorithm For Scheduling Meta-Tasks On Heterogeneous Computing Systems", IJCSNS International Journal of Computer Science and Network Security, Vol.10 No.1, 2010, pp. 24-31.
3. G.K.Kamalam, and Dr.V.Murali Bhaskaran., "An Improved Min-Mean Heuristic Scheduling Algorithm for Mapping Independent Tasks on Heterogeneous Computing Environment", International Journal of Computational Cognition, Vol. 8, N0. 4, 2010, pp. 85-91.
4. G.K.Kamalam, and Dr.V.Murali Bhaskaran., "An Efficient Hybrid Job Scheduling for Computational Grids", International Journal of Computer Applications, 2011.
5. G.K.Kamalam, and Dr.V.Murali Bhaskaran., "New Enhanced Heuristic Min-Mean Scheduling Algorithm for Scheduling Meta-Tasks on Heterogeneous Grid Environment", European Journal of Scientific Research, Vol.70 No.3, 2012, pp. 423-430.
6. G.K.Kamalam, and Dr.V.Murali Bhaskaran., "Novel Adaptive Job Scheduling algorithm on Heterogeneous Grid Resources", American Journal of applied Sciences, 2012, pp. 1294-1299.
7. G.K.Kamalam.,"Resource Fitness Task Scheduling Algorithm for Scheduling Tasks on Heterogeneous Grid Environment", Australian Journal of Basic and Applied Sciences, Vol.8 No.18, 2014, pp. 128-135.
8. G.K.Kamalam, and Ms.B.Anitha., "Double Min-Min Credit Worth Task Scheduling Algorithm for Mapping Meta-Tasks on Heterogeneous Grid Environment", International Journal for Modern Trends in Science and Technology, Vol.3 No.2, 2017, pp. 18-24.
9. Muhanad Tahrir Younis, Shengxiang Yang., "Hybrid meta-heuristic algorithms for independent job scheduling in grid computing", Vol. 72, 2018, pp.498-517.
10. Paolo Bellavista, et. al., "GAMESH: A grid architecture for scalable monitoring and enhanced dependable job scheduling", Vol. 71, 2017, pp.192-201.

AUTHORS PROFILE

Dr.G.K.Kamalam is working as an Assistant Professor (SLG) in the Department of Information Technology, Kongu Engineering College, Tamil Nadu, and India. Her research area is Grid and Cloud Computing. She has presented 35 papers in national and international conferences and published 25 papers in international journals in her research and other technical areas. Her Area of Interest includes Algorithm Analysis and Optimization Techniques.

K.Sentamilselvan received BE Degree in Computer Science and Engineering from Anna University, Chennai, TN, India in 2010. He received M.Tech degree with specialization of Information Security from Pondicherry Engineering College, Pondicherry, India in 2013. He is currently working an Assistant professor in the Department of Information Technology at Kongu Engineering College of Perundurai, Erode, TN, India. He is life member of Computer Society of India (CSI). His research interest is Hacking, Web application Security, Information Security, Cloud and Grid computing.