www.ioirp.com
International Journal of Innovative Research in Computer Science and Engineering (IJIRCSE)
ISSN: 2394-6364, Volume – 1, Issue – 13. December 2015

Task Unique Credit System Based Scheduling Algorithm for Mapping Meta-Tasks on Heterogeneous Grid Environment

V.Manju Barkavi
PG Scholar
Information Technology
Kongu Engineering College
Perundurai, India
manjuvivek12@gmail.com

Dr.G.K.Kamalam Ph.D.,
Assistant Professor(SLG)
Information Technology
Kongu Engineering College
Perundurai, India
kamalamparames@gmail.com

Ms.B.Anitha M.E
Assistant Professor
Information Technology
Kongu Engineering College
Perundurai, India
anitha@gmail.com

***Abstract*—Grid is an infrastructure that enables the integrated and collaborative use of sharing of geographically distributed resources, can achieve the computing power as a super computer does with a lower cost. Grid computing enables large-scale resource sharing and has emerged for satisfying the increasing demand of huge computing power for solving advanced science and engineering applications. Grid is a heterogeneous distributed system. Scheduling is a key component and has become one of the important research objective, it directly involves in achieving high performance of distributed grid applications. Scheduling task on the heterogeneous resources spread over a distributed grid computing environment is an NP-complete problem. Efficient scheduling algorithm is essential for achieving high performance in heterogeneous grid system. This paper proposes a new heuristic scheduling algorithm Task Unique Credit System Scheduling Algorithm (TUCSA), provides the order in which the tasks are to be scheduled is determined based on the highest credit value of the task. The experimental results reveal that the proposed TUCSA algorithm achieves reduced make span and well suits for the grid environment.**

Keywords—component; formatting; style; styling; insert (key words)

I. INTRODUCTION

Grid Computing is a promising technology that provides consistent, pervasive, and inexpensive access to the geographically distributed heterogeneous computational resources and huge storage space and achieving high throughput in the distributed environment [1]. Grid is classified into two categories, computational grid and data grid. Computational Grid is a hardware and software infrastructure that provides inexpensive access to widely distributed heterogeneous resources to solve the complicated computational problems [7]. The computational grid is a computational framework. The main aim of computational grid is to utilize the idle and scattered resources to splve the growing computational demands. Data Grid deals with the sharing, access and managing of massive data available in the distributed environment. The key technologies that affect the efficiency of grid are resource allocation, load balancing and task scheduling algorithms [14]. Due to the hetereogeneous nature of the grid environment, task scheduling becomes a complicated and challenging problem. The grid computing technology emerges to meet the demands of the scientific and research organizations. Scientific research problems that involves compute-intensive and data-intensive applications which requires huge amount of storage space and large number of computing resources and also the application to be completed within the realistic time frame [2]. In general, tasks can be classified into independent tasks and dependent tasks. Dependent tasks requires communication with other tasks for its completion. The independent tasks does not require communication with other tasks. Metatask is defined as a collection of independent tasks with no inter-task dependencies. The objective of the task scheduling is to reduce the makespan and improve the resource utilisation. Makespan is defined as the total time required to complete the metatask. Over the years, many scheduling algorithms have been developed.Grid scheduling problem is a NP-Complete problem. Computing optimal solution for NP-complete problem is intractable. The reasonable assumptions are difficult to find an optimal solution, due to the dynamic grid nature. Hence, heuristic algorithms provide good solutions. Heuristic approach proves to be the efficient approach for solving the computationally hard problems. The heuristic approach includes Opportunistic Load Balancing (OLB), Minimum Completion Time

(MCT), Min-min, Max-min, and Genetic Algorithm (GA). However, thes approaches do not consider the effects of QoS, which is an important criteria in grid scheduling.

OLB allocates the job to the next available machine in the arbitrary order. This algorithm allocates the task without considering the expected execution time of the task on the machine [3,8]. MCT schedules the task by estimating the expected completion time for a task on all machines and allocates the task to the machine with the minimum completion time. MCT considers only one task at a time for scheduling [5]. Minimum Execution Time (MET) algorithm allocates the task to the machine by considering the minimum expected execution time regardless of the availability of the machine, it causes poor makespan and severe load imbalance.

Min-min algorithm is the best heuristic scheduling algorithm for scheduling the meta-tasks to the machines. Min-min algorithm schedules the tasks in two phases, in first phase, the minimum expected competion time for each task in all machines is calculated. In second phase, the task with overall minimum expected completion time is selected and scheduled to the corresponding machine. The algorithm does not consider the idleness of the machine [3,8,9].

Max-min algorithm is similar to the Min-min algorithm, it differs only in the second phase. It identifies the task with the maximum expected competion time and schedules to the appropriate machine. Max-min algorithm is better only when most of the tasks arriving to the grid environment res shortest.

QoS Sufferage divides the tasks into high Qos and low QoS groups. The tasks in each group can be executed on hosts with the corresponding QoS provision, the algorithm finds ths the sufferage value by obtaining the difference between the best minimum completion time and its second minimum completion time. The task with the greatest sufferage value is selected and allocated to the machine with the minimum completion time.

He et al. [10] propsed an algorithm QoS Guided min-min heuristic scheduling algorithm. The authors considerd network bandwidth as QoS parameter and groups the task into high QoS and low QoS groups. The algorithm schedules high QoS tasks to the resources that provide high QoS and then schedules the low QoS tasks. The scheduling frequency and the method to classify general QoS have to be improved.

Sameer and Joshi [15] presented two algorithms QoS guided weighted mean time-min algorithm and QoS guided weighted mean time Min-min Max-min selective algorithm for QoS based grid task scheduling. QoS guided weighted mean time-min groups the tasks into high QoS and low QoS tasks. The algorithm calculates the weighted mean time for each task in high QoS group and schedules the maximum weighted mean time task to the minimum completion time resource. This process repeats until all the tasks in the high QoS group get scheduled. The same procedure repeats for low QoS group tasks. QoS guided weighted mean time min-min max-min selective heuristic algorithm works as follows. The resources groups into n groups based on QoS services. The algorithm calculates the weight of the resource in each group. It calculates the weighted mean time of each task in the group. It computes the standard deviation of the completion time of the unassigned tasks. The algorithm schedules the task with maximum/minimum weighted mean time by selecting Min-min or Max-min algorithm based on the relative standard deviation value. Dong et al [6] presented a scheduling strategy based on QoS priority grouping. The author considered the deadline of the task, acceptance rate of tasks, and makespan as the scheduling parameter. The limitation of the algorithm is that it does not perform well in the complicated environment.

The previous work [11,12,13]Min-mean heuristic scheduling algorithm works in two phases. In the first phase, Min-mean heuristic scheduling algorithm starts with a set of all unmapped tasks. The algorithm calculates the completion time for each task on each resource and finds the minimum completion time for each task. From that group, the algorithm selects the task with the overall minimum completion time and allocates to the appropriate resource. Removes the task from the task set. This process repeats until all the tasks get mapped. The algorithm calculates the total completion time of all the resources and the mean completion time. In the second phase, the algorithm selects the resource whose completion time is greater than mean completion time. The algorithm arranges the selected resources in the decreasing order of the completion time. The algorithm reschedules the tasks assigned on the selected resources to the resource, whose completion time is less than the mean completion time. The current research problem in task scheduling is to bring out an efficient algorithm to improve the computational efficiency of the grid [4]. The scope of this work is to propose an efficient task scheduling algorithm on the basis of the highest credit value of the task.

II. MATERIALS AND METHODS

A. Problem Definition:

An application consists of 'n' independent meta-task and a set of 'm' heterogeneous resources. The problem of mapping the 'n' meta-tasks to the set of 'm' heterogeneous resources in a grid computing environment is an NP-Complete problem. This paper proposes a new algorithm Task Unique Credit System Scheduling Algorithm for solving the scheduling problem in a grid computing environment. Task scheduling in grid represent two important things: ordering and mapping. Ordering refers to the order in which the task to be executed. Mapping refers to the process of selecting an appropriate resource and scheduling the task to those resources. For every mapping process, the performance potential is determined to identify the best resource for the execution of the task. The order in which the tasks to be mapped to a set of resources determines the efficient scheduling which results in the reduced makespan. The proposed new algorithm Task Unique Credit System Scheduling Algorithm provides an ordered set of tasks, which specifies the order in which the tasks to be scheduled to the set of 'm' resources. The proposed Task Unique Credit System Scheduling Algorithm provides reduced makespan than the existing Min-min heuristic scheduling algorithm.

B. Proposed TUCSSA Algorithm.

The mapping of task to the resource is made based on the following assumptions

- A meta-task (ie. A set of independent, non-communicating tasks) is being mapped.
- Heuristics derive a mapping statically.
- Static mapping of meta-tasks to resources is being performed.
- Each resource executes a single task at a time. The order in which the tasks are scheduled is based on the task credit.
- The number of meta-tasks to execute and the number of resources are static and known a prior.
- The accurate estimate of the expected execution time for each task on each resource is contained within an ETC matrix of size n*m, where n is the number of task and m is the number of resources.
- $ETC(t_i, r_j)$ represents the expected execution time of the task t_i on resource r_j
- Task set is represented as T=$\{T_1, T_2, \ldots \ldots \ldots T_n\}$
- Resource set represented as R=$\{R_1, R_2, \ldots \ldots \ldots R_m\}$
- ET_{ij}- expected execution time of task T_i on resource R_j
- TCT_{ij}- expected completion time of task T_i on resource R_j
- RT_j- ready time of resource R_j
- Makespan =max(TCT_{ij})
- ETC matrix is computed by the formula

$$ETC_{ij} = \frac{Tasklength_i}{Power_j}$$

where $Tasklength_i$ represents the length of the task T_i in MI and $Power_j$ represents the computing power of the resource R_j in MIPS

The ready time of the resource is the time at which the resource R_j completes the executing of the previously assigned tasks and is defined as

$$RT_j = \sum_{i=1}^{n} ET_{ij}$$

The proposed approach considers two criteria for task scheduling: 1) Task Length and 2) Unique identification value. The algorithm schedules the task to the resources based on the credit system. Each task is assigned a credit value based on their task length and unique identification value.

Task Length Credit

The steps involved in calculating the task length credit is as follows:

1. The average of the task length, say Avglen is calculated using the formula,

International Journal of Innovative Research in Computer Science and Engineering (IJIRCSE)
ISSN: 2394-6364, Volume – 1, Issue – 13. December 2015

$$\text{Avglen}=\sum_{i=1}^{n}\frac{Tasklength_i}{n}$$

where $Tasklength\ i$ represent the length of the i^{th} task and n represents the number of task to be scheduled.

2. For each task t_i, the difference in $Tasklength_i$ with respect to average length , is calculated by

$$DT_i = |Avglen - Tasklength_i|$$

where DT_i is the difference in the task length of taskt_i. It is computed by calculating the absolute difference of task length of i^{th} task and average length.

3. Credits are assigned to each task using the following formula

maxTL= max($Tasklength_i$), 1<=i<=n
CValue1= maxTL/5
CValue2= maxTL/4
CValue3=CValue1+CValue2
CValue4= CValue2+CValue3

where maxTL is the maximum value of task length.

The algorithm for finding the Task Length Credit is shown below:
for all submitted tasks in the taskset T,

$$\text{Avglen}=\sum_{i=1}^{n}\frac{Tasklength_i}{n}$$

end for
for all submitted tasks in the taskset T,

$$DT_i = |Avglen - Tasklength_i|$$

if DT_i<=CValue1 then CL_i=5
elseif CValue1<=DT_i<=CValue2 then CL_i=4
elseif CValue2<=DT_i<=CValue3 then CL_i=3
elseif CValue3<=DT_i<=CValue4 then CL_i=2
elseif CL_i=1
endfor

Task Unique Identification Value Credit

Task unique identification value is also an important criterion for scheduling tasks. Each task may have unique identification value. In the proposed approach, the scheduling of the task to the resource is based on total credit which is based on the task length and its unique identification value. The algorithm for finding the unique identification credit value is shown below:

for all submitted tasks in the taskset T,
find out the tasks with highest unique identification value
choose denomination value, dv
for each tasks t_i in the taskset T,
compute $UIVC_i = UIV_i/dv$
end for
end for

The primary step in the above algorithm is to find the unique identification credit value for each task t_i. The next important step is to determine the denominator value dv. If the highest unique identification value is two digit, dv=100, if it is a three digit number, then dv=1000.

The two credits, task length credit and unique identification value credit for each task is calculated. The final step is to find out the total credit for each task t_i using the formula

$$TC_i = CL_i \times UIVC_i$$

Finally, the tasks are arranged in a credit set CS in the descending order of their total credit. The tasks having the highest total credit will be scheduled first.

An Illustrative Example

To illustrate how the proposed algorithm Task Unique Credit System Scheduling Algorithm schedules the tasks efficiently, consider the following example for a grid environment with ten tasks and three resources. The task length for each task is given in Table1 and the computing power of each resource is given in Table2. The ETC matrix is given in Table3.

Table1 Tasklength for each Task in MI

Task	Tasklength
T_1	50000
T_2	55000
T_3	56000
T_4	58000
T_5	65000
T_6	69000
T_7	75000
T_8	78000
T_9	83000
T_{10}	95000

Table2 Computing power for each Resource in MIPS

Resource	Power
R_1	5200
R_2	4800
R_3	6000

Table3 ETC matrix

Task	R_1	R_2	R_3
T_1	9.62	10.42	8.33
T_2	10.58	11.46	9.17
T_3	10.77	11.67	9.33
T_4	11.15	12.08	9.67
T_5	12.5	13.54	10.83
T_6	13.27	14.38	11.5
T_7	14.42	15.63	12.5
T_8	15	16.25	13
T_9	15.96	17.29	13.83
T_{10}	18.27	19.79	15.83

The average length of the task is

Avglen = 68,400

The maximum length of the task MaxTL = 95,000

CValue1= 19000

CValue2= 23750

CValue3= 42750
CValue4= 66500

The task length credit for each task is computed and the result is shown in Table4.

Table 4 Task Length Credit

Task	CL
T_1	5
T_2	5
T_3	5
T_4	5
T_5	5
T_6	5
T_7	5
T_8	5
T_9	5
T_{10}	3

A unique identification value is selected in random in the range 1 to 10 is assigned to each task. The unique identification value credit is computed using the algorithm 2 and is shown in Table5. The total credit for each task t_i is calculated using the given formula and the result is shown in Table5.

$$TC_i = CL_i \times UIVC_i$$

Table 5 Total Credit for each Task

Task	CL	UIV	UIVC	TC
T_1	5	1	0.01	0.05
T_2	5	7	0.07	0.35
T_3	5	9	0.09	0.45
T_4	5	2	0.02	0.1
T_5	5	6	0.06	0.3
T_6	5	10	0.1	0.5
T_7	5	3	0.03	0.15
T_8	5	8	0.08	0.4
T_9	5	5	0.05	0.25
T_{10}	3	4	0.04	0.12

The tasks to be submitted are ordered in the credit set CS in the descending order of TC_i.
CS={$T_6, T_3, T_8, T_2, T_5, T_9, T_7, , T_{10}, T_4, T_1$}
Now, the tasks are scheduled to the resource with the minimum completion time. The makespan for the above example is 43.96 sec.
The makespan produced by Min-min algorithm compared to the result of the proposed Task Unique Credit System Scheduling Algorithm is shown in Table6.

Table 6 A comparisons between algorithms in makespan and tasks scheduling

Algorithms	R1	R2	R3	Makespan
Min-min	T_2 T_5 T_8	T_3 T_6 T_9	T_1 T_4 T_7 T_{10}	46.33
Proposed algorithm	T_3 T_5 T_{10}	T_8 T_9 T_1	T_6 T_2 T_7 T_4	43.96

Task Unique Credit System Scheduling Algorithm (TUCSSA):

For all submitted tasks in the task set T
Calculate task length credit using Algorithm1
Calculate unique identification credit value using Algorithm2
End for
For each task t_i
Compute $TC_i = CL_i \times UIVC_i$
Order the task in the Credit Set CS in the descending order of TC_i
For all tasks T_i in the Credit Set CS
For all resources R_j
Compute $TCT_{ij} = ET_{ij} + RT_j$
End for
Do until all tasks in CS are mapped
For each task in CS find the earliest completion time and the resources that obtains it
Find the task t_K to the resource R_j that gives the earliest completion time
delete task t_K from CS
update RT_j
update TCT_{ij}
End for
End do
Compute makespan = max(TCT_{ij}) for all i,j

III.RESULTS AND DISCUSSION

This section presents the experimental results obtained for the benchmark model by Braun et al[5].

1.Benchmark Descriptions. The instances of the benchmark model are classified into 12 different types of ETC matrix for 512 tasks and 16 resources, with each of them consisting of the 100 instances based on the three metrics: task heterogeneity, resource heterogeneity, and consistency.

Instances are represented as u-x-yyzz.k where,

u – represents uniform distribution (in generating ETC matrix)

x – represents the type of consistency (c-consistent, i-inconsistent, s-semi-consistent or partially consistent).

An ETC matrix is consistent, whenever a resource r_i executes any tasks t_i faster than resource r_j, then resource r_i executes all tasks faster than r_j. An ETC matrix is inconsistent, resource r_i may be faster than resource r_j for executing some tasks and slower for others.

An ETC matrix is semi-consistent if it includes a consistent sub-matrix.

Task heterogeneity: Variation in the execution time for a task t_i on all resources.

yy – heterogeneity of the tasks(hi-high, lo-low)

Resource heterogeneity: Variation in the execution time of all the tasks on a resource r_j.

zz – heterogeneity of the resources(hi-high, lo-low)

The 12 different instances are divided into three groups of four instances each and shown in Table 7.

Table 7 ETC Model

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
Semi-consistent	u-s-hihi-0	u-s-hilo-0	u-s-lohi-0	u-s-lolo-0

2. Evaluation Parameters.

Makespan

Makespan is the important optimization criteria for grid scheduling. Makespan is calculated as makespan=max(CT(t_i))

Figure 1 shows that the Task Unique Credit System Scheduling Algorithm provides better makespan than Min-min Heuristic Scheduling Algorithm.

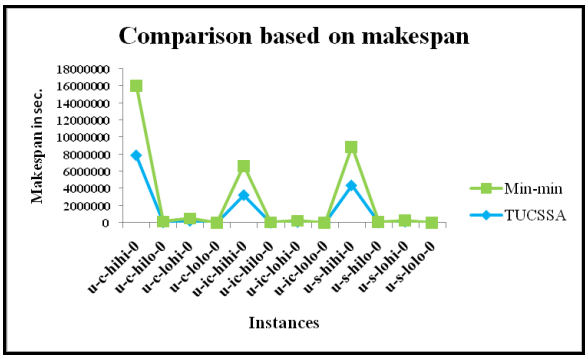

Figure 1: Comparison based on makespan

Figure 2, 3, 4, 5 show the comparison of the makespan values obtained by Min-min and TUCSSA in all the four instances which comprises High Task High Machine, High Task Low Machine, Low Task High Machine, Low Task Low Machine. The four instances are represented for consistent, inconsistent, semi-consistent or partially consistent heterogeneous computing systems.

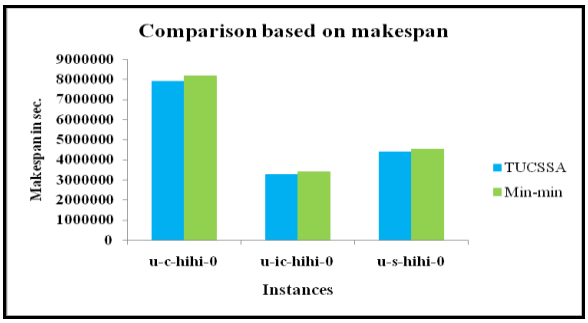

Figure 2: Comparison based on makespan for High Task High Machine Heterogeneity

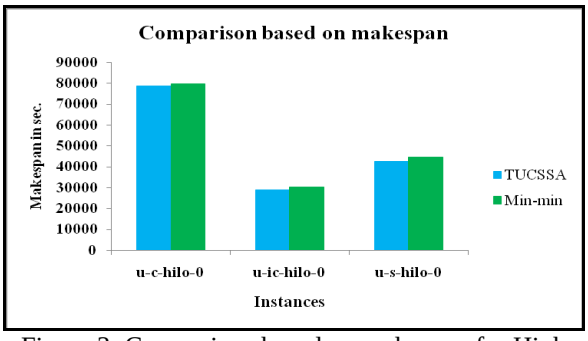

Figure 3: Comparison based on makespan for High Task Low Machine Heterogeneity

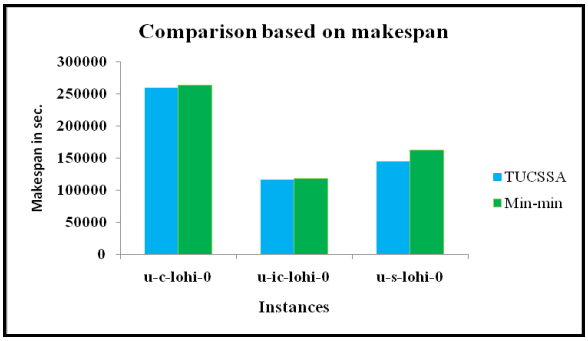

Figure 4: Comparison based on makespan for Low Task High Machine Heterogeneity

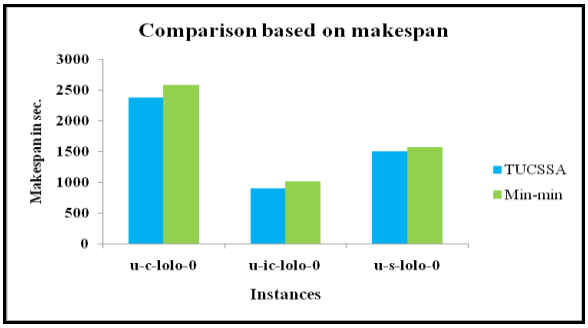

Figure 5: Comparison based on makespan for Low Task Low Machine Heterogeneity

IV.CONCLUSION

Selecting an appropriate resource and scheduling the suitable task to that resource is a challenging work in the computational grid environment. This work proposes a new heuristic task scheduling algorithm called Task Unique Credit System Scheduling Algorithm. The implementation of Task Unique Credit System Scheduling Algorithm and Min-min heuristic scheduling algorithm are tested using the benchmark simulation model [5]. The experimental result show that the Task Unique Credit System Scheduling Algorithm outperforms the Min-min heuristic algorithm and delivers reduced makespan on various instances such as task heterogeneity(high,low), resource heterogeneity(high, low) and consistency(consistent, inconsistent and semi-consistent).

V.REFERENCES

[1] M.Abdullah, M.Othman, H.Ibrahim, and S.Shamala, "Optimal workload allocation model for scheduling divisible data grid applications", Future Generation Computer Systems 26, pp.971-978, 2010.

[2] R.Albodour, A.James, and N.Yaacob, "High level QoS-driven model for Grid applications in a simulated environment", Future Generation Computer Systems 28, pp. 1133-1144, 2012.

[3] R.Armstrong, D.Hensgen, and T.Kidd, "The Relative Performance of Various Mapping Algorithms is Independent of Sizable Variances in Run-time Predictions", In 7th IEEE Heterogeneous Computing Workshop(HCW'98), pp. 79-87, 1998.

[4] H.Baghban, A.M. Rahmani, " A Heuristic on Job Scheduling in Grid Computing Environment", In Proceedings of the seventh IEEE International Conference on Grid and Cooperative Computing, pp. 141-146, 2008.

[5] 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, pp.810-837, 2001.

[6] F.Dong, J.Luo, L.Gao, and L.Ge, "A Grid Task Scheduling Algorithm Based on QoS Priority Grouping", 5th International Conference on Grid and Cooperative Computing, 2006.

[7] I.Foster and C. Kesselman, "The Grid: Blueprint for a Future Computing Infrastructure", Morgan Kaufmann Publishers, USA, 1998.

[8] R.F.Freund and H.J.Siegel,"Heterogeneous Processing", IEEE Computer , 26(6), pp. 13-17, 1993.

[9] R.F.Freund, and M.Gherrity, "Scheduling Resources in Multi-user Heterogeneous Computing Environment with Smart Net", In Proceedings of the 7th IEEE HCW, 1998.

[10] X-S.He, X-H.Sun, and V.L. Gregor, "QoS Guided Min-Min Heuristic for Grid Task Scheduling", Journal of Computer Science and Technology, pp. 442-451, 2003.

[11] 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, pp. 24-31, 2010.

[12] 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, pp. 85-91, 2010.

[13] 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, pp. 423-430, 2012.

[14] E.U.Munir, J.Li, and S.Shi, "QoS Sufferage Heuristic for Independent Task Scheduling in Grid", Information Technology Journal 6(8), pp. 1166-1170, 2007.

[15] S.C.Sameer, and R.C. Joshi, "QoS Guided Heuristic Algorithms for Grid Task Scheduling", International Journal of Computer Applications, Vol. 2, No. 9, pp. 24-31, 2010.