www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 03 Issue 07 July, 2014 Page No. 6943-6949

An Efficient Job Scheduling Algorithm using Min-Min and Ant Colony Concept for Grid Computing

***Davinder Kaur*[1], *Sarpreet Singh*[2]**

[1]Department of Computer Science Engg. ,Sri Guru Granth Sahib World University,
Fatehgarh Sahib, Punjab, India
davinder.gill301@gmail.com

[2]Department of Computer Science Engg. ,Sri Guru Granth Sahib World University,
Fatehgarh Sahib, Punjab, India
ersarpreetvirk@gmail.com

Abstract: ***Grid computing has emerged as an important field from the distributed and parallel computing where the resources of various computers in the network are used to solve a particular problem, because of high demand of computational power and need of high performance. To achieve the promising potential of grid computing, an effective and efficient job scheduling is required. Job scheduling is used to schedule the user jobs to appropriate resources in grid environment. Min-Min is the most simple and well known scheduling algorithm in grid computing that is used to minimize the makespan. The drawback of min-min algorithm is that the schedule produced by min-min is not optimal with respect of load balancing and it is not effective for resource utilization and one resource can execute one job at a time, the number of resources is known in prior. In this paper, a Min-Min Ant Colony (MMAC) algorithm is proposed that reduces the makespan and maximize the resource utilization using the features of both min-min algorithm and ant colony optimization. It is a two phase algorithm.***

Keywords: Grid Computing, Job Scheduling, Min-Min algorithm, Ant Colony Optimization (ACO).

1. Introduction

Grid is a type of parallel and distributed system that enables the sharing, selection and aggregation of resources distributed across multiple administrative domains based on their availability, capability, performance, cost and user's quality-of-service requirements [1]. Grid computing is recognized as one of the most powerful vehicles for high performance computing for data-intensive scientific, business, engineering applications. A Grid is loosely coupled, geographically distributed and heterogeneous in nature. Foster and Kesselman defined the Grid as follows [2]*: A computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities*.

Grid is distinguished from traditional distributed computing because of its focus on large-scale resource sharing and high performance orientation. Grid computing evolved the great technologies to effectively utilize the resources. To make use of great capabilities of this distributed system, effective and efficient scheduling algorithms are needed. Depending on their goals, these algorithms assign jobs to the best machines which produced better quality of service [3].

Scheduling is considered one of the important issue in grid computing. Job Scheduling is choosing the most suitable resource for a job to complete its execution either in terms of waiting time, turnaround time or cost [4]. The traditional scheduling algorithms does not work efficiently in the grid environment due to its heterogeneous nature. The demand for effective scheduling increases to achieve high performance computing. Typically, it is difficult to find an optimal resource allocation which minimizes the schedule length of jobs and effectively utilize the resources. Scheduling on a grid has three main phases [5]. Phase one is resource discovery [6], which generates a list of potential resources. Phase two involves gathering information about those resources and choosing the best set to match the application requirements. In phase three the job is executed.

Job scheduling is based on the necessity of a user who has a set of jobs to execute. Grid user compose their applications as a distributed applications. Sometime the user's machine is not able to process the jobs either because of resource or hardware constraints, for such cases the user can use the grid system for running the job. The user submits the set of jobs to the job scheduler and the job scheduler splits the job depending on certain factors and gives it to the machines having available resources on the grid. The machines will complete the task and final result will be given to the user[7].

There are relatively a large number of job scheduling algorithms to minimize the total completion time of the jobs in distributed systems. Actually, these algorithms try to minimize the overall completion time of the jobs by finding the most suitable resources to be allocated to the jobs. It should be noticed that minimizing the overall completion time of the jobs does not necessarily result in the minimization of execution time of each individual job [8].

The most simple and well known algorithm that is used in grid are Min-Min and Ant colony optimization. The min-min algorithm estimate the execution and completion time of each job on the each available resources in grid. The ant colony algorithm for job scheduling in grid aims at submitted jobs to resources based on the processing ability of jobs as well as the characteristics of the jobs.

There are two phase in the Min-Min algorithm. In the first phase it finds the minimum execution time of all tasks. Then in the second phase it chooses the task with the least execution time among all the tasks. The algorithm proceeds by assigning the task to the resource that produces the minimum completion time. The same procedure is repeated by Min-Min until all tasks are scheduled. The following are the some limitation of Min-Min algorithm:-

i. It chooses smaller tasks first which makes use of resource with high computational power. As a result, the schedule produced by Min-Min is not optimal when number of smaller tasks exceeds the large ones.
ii. One resource can execute only one job at a time .
iii. Size and number of resources are static and should be know in prior.[7]

ACO has been used by researchers in grid computing for addressing load balancing [9], job scheduling [10] and related problems. Ant colony algorithm is the bio-inspired heuristic algorithm, which is derived from the social behavior of ants. Ants work together to find the shortest path between their nest and food source. When the ants move, each ant will deposit a chemical substance called pheromone. Using this pheromone, the shortest path is found. The same concept is used to assign jobs in grid computing. When a resource is assigned a job and completes its job then its pheromone value will be added each time. If a resource fails to finish a job, it will be punished by reducing pheromone value. The issue here is the stagnation, where there is a possibility of jobs being submitted to same resources having high pheromone value [7].

In the ant colony algorithm [10] , the load balancing method is proposed to solve the issue of stagnation. The algorithm work as follows :-

i. The user will send request to process a job
ii. The grid resource broker will find a resource for the job
iii. The resource broker will select the resource based on the largest value in the pheromone value matrix
iv. The local pheromone update is done when a job is assigned to a resource.
v. The global pheromone update is done when a resource completes a job
vi. The execution result will be sent to the use

When the resource broker select a particular resource for a job j, jth column of the Pheromone Value matrix will be removed and jobs will be assigned to other resources. Thus the load balancing is achieved [7].

In this paper, a new scheduling algorithm is proposed to resolve the above mentioned problems with the Min- min algorithms by using the cons of both min-min algorithm and ant colony optimization. The proposed algorithm applies the Ant Colony Optimization in the first phase and then Min-Min strategy for better result. The new algorithm is implemented in real time environment using java. The result show better resource utilization and minimum total completion time of task.

The remaining parts of this paper are organized as follows: Section 2 presents the related works and several well known scheduling algorithms which are benchmarks of many other works. In Section 3, a new scheduling algorithm is proposed and the prominence of the algorithm is demonstrated through an example. Section 4 compares the scheduling algorithms and presents the results of the comparison. Finally, Section 5 concludes the paper and presents future works.

2. Related Work

Opportunistic Load Balancing (OLB) simply assigns tasks to the next available machine. If more than one machine is available, one machine is chosen arbitrarily. It does not take into account the expected execution time of the task on that machine. Thus it provides a load balanced schedule but it produces a very poor makespan.

Minimum Execution Time (MET) assigns each task in an arbitrary order to the machines that requires least execution time without considering the availability of the resource and its current load. This algorithm improves the makespan to some extent but it can causes a severe load imbalance. The motivation behind MET is to give each task to its best machine. Its only advantage over MCT is its simplicity of implementation.

Minimum Completion Time (MCT) assigns each task, in arbitrary order, to the machine with the minimum expected completion time for that task [11]. This causes some of the tasks to be assigned to the machines that do not have the minimum execution time for them. The intuition behind MCT is to combine the benefits of OLB and MET, while avoiding the circumstances in which OLB and MET perform poorly. But this algorithm considers the job only one at a time.

T. Kokilavani et al., [12] have proposed a new scheduling algorithm named Load Balanced Min-Min (LBMM) Algorithm for Static Meta-Task scheduling in grid environment to overcome the limitations of Min-Min algorithm. It is performed in two-phases. It uses the advantages of Max-Min and Min-Min algorithms and covers their disadvantages. In the first phase the traditional Min-Min algorithm is executed and in the second phase the tasks are rescheduled to use the unutilized resources effectively. The LBMM algorithm reduces the makespan and increases the resource utilization.

Saeed Parsa et al.,[8] have proposed a new task scheduling algorithm in called RASA(Resource Aware Scheduling Algorithm) , considering the distribution and scalability characteristics of grid resources. RASA is composed of two traditional scheduling algorithms: Max-min and Min-min. It uses the Min-min strategy to execute small tasks before the large ones and applies the Max-min strategy to avoid delays in the execution of large tasks and to support concurrency in the execution of large and small tasks. Experimental results show that if the number of available resources is odd it is preferred to apply the Min-min strategy in the first round otherwise is better to apply the max-min strategy the first.

George Amalarethinam. D.I. et al., [11] have proposed a new heuristic technique called Max- min Average algorithm for task scheduling in the heterogeneous grid computing environment. The aim of this proposed algorithm is to minimize the idle time and makespan of the tasks. The proposed algorithm Min-mean works in two phases. In phase 1, the task allocation is done based on the Max- min algorithm. In phase 2, the mean of completion time of all the machines are taken. The tasks allocated to the selected machines are reallocated to the machines whose completion time is less than the mean value.

Salman Meraji et al., [13] have proposed a new algorithm which is called best-min algorithm in order to overcome the disadvantage of min-min algorithm that is schedule produced by min-min is not optimal with respect of load balancing and max-min's relative time to finish assigning tasks is too high. It is a two phase algorithm. The best-min algorithm uses min-min to get the makespan in first step and
reschedule the tasks in the second phase in order to reduce the makespan. There is a condition in algorithm that best-min should consider all the resources in grid environment and this is caused to maximize resource utilization as well. The best-min try to minimize makespan while maximize resource utilization and matching proximity.

HE. X et al., [14] have proposed a new algorithm based on the conventional Min-Min algorithm to achieve high throughput computing in grid environment. This proposed algorithm named as QoS guided Min-Min for heuristic grid task scheduling, schedules tasks requiring high bandwidth before the others. Therefore, if the bandwidth required by different tasks varies highly, the QoS guided Min- Min algorithm provides better results than the traditional Min-Min algorithm. Whenever the bandwidth requirement of all of the tasks is almost the same, the QoS guided Min-Min algorithm acts similar to the traditional Min-Min algorithm. Furthermore, it also tolerates inaccurate execution estimations.

O. M. Elzeki et al.,[15] have proposed a algorithm called Improved Max-Min Algorithm in cloud computing based on comprehensive study of the RASA algorithm in scheduling tasks and the concept of Max-min strategy. Improved Max-min is based on the expected execution time instead of complete time as a selection basis. Improved Max-min supports load balance of available resources and allow concurrent execution of submitted tasks with higher probability rather than original Max-min.

Singh. M et al., [16] present a QoS based predictive Max-Min, Min-Min Switcher algorithm for scheduling jobs in a grid. The algorithm makes an appropriate selection among the QoS based Max-Min or QoS based Min-Min algorithm on the basis of heuristic applied, before scheduling the next job. The effect on the execution time of grid jobs due to non-dedicated property of resources has also been considered. The algorithm uses the history information about the execution of jobs to predict the performance of non-dedicated resources.

Kobra Etminani et al.,[3] have introduce a new scheduling algorithm to achieve high computing throughput in a grid environment, called Min-min Max-min Selective Algorithm based on two conventional scheduling algorithms, Min-Min and Max- Min. This algorithm uses the cons of both the conventional algorithm and at the same time, overcome their pros. The algorithm determines to select one of these two algorithms, dependent on the standard deviation of the expected completion times of the tasks on each of the resources. The experimental results show that the Selective algorithm outperforms the traditional Min-Min and Max-Min heuristics.

Kumar Nishant et al., [17] proposed an algorithm for load distribution of workloads among nodes of a cloud or grid by the use of Ant Colony Optimization (ACO) . The main aim of the algorithm is load balancing of nodes. This algorithm has an edge over the original approach in which each ant build their own individual result set and it is later on built into a complete solution. However, in this algorithm the ants continuously update a single result set rather than updating their own result set. The algorithm uses the both forward and backward movements for overload and underload.

Siriluck Lorpunmanee et al., [18] have developed a general framework of grid scheduling using dynamic information and an ant colony optimization algorithm to improve the decision of scheduling. The performance of various dispatching rules such as First Come First Served (FCFS), Earliest Due Date (EDD), Earliest Release Date (ERD), and an Ant Colony Optimization (ACO) are compared. It is found that the scheduling system using an Ant Colony Optimization algorithm can efficiently and effectively allocate jobs to proper resources.

D.Maruthanayagam et al., [19] proposes an improved ant colony scheduling algorithm combined with the concept of Resource Aware Scheduling Algorithm (RASA).. The RASA algorithm first estimates the completion time of the tasks on each of the available grid resources and then applies the Max-min and Min-min algorithms. Before assigning the tasks to machines, the expected execution time for each task on each machine must be estimated and represented by an Expected Time calculation. And also this algorithm can find an optimal processor and network for each machine to allocate a job that minimizes the tardiness time of a job when the job is scheduled in the system. The proposed scheduling algorithm is designed to achieve high throughput computing in a grid environment.

3. Proposed Work

Our proposed job scheduling algorithm, MMAC, is presented in Figure 1. It is a two phase algorithm. In the first phase ant colony concept is used. The ant in the proposed algorithm will continuously originate from the master node. The ant traverse the width and length of the network in such a way that they know about the location of overloaded and underloaded nodes in a network. The ant along the movement can check the processes of each node in the network and update the process value in the database and also check which node responses fast. The processes on the other nodes can be calculated using the min-min algorithm or in java, processes can be calculated using process thread. In the second phase, the master node assign the job to a node who has a less processes using the min-min strategy.

The proposed algorithm works slightly different from conventional min-min algorithm. Instead of first calculating the expected execution time of each job on the each machine and then select the job who has a least completion time among all the jobs. Assign that job to the resource that produces the minimum completion time. The proposed algorithm assign the

job to resource on the basis of processes run on that resource. RF means which resource responses first(RF).
No._of_Process (NOP): The NOP is the number of processes on each resource in a network means how much machine is utilized.

```
1.  For all tasks $t_i$ in MT & all resources Rj
2.  At starting all resources will be free &
    initialize the parameter NOP
3.  Scanning thread (Ant) scan all resources in
    that network
4.  Check the NOP and RF for every resource
5.  Store the value of NO in database
6.  From database select resource with minimum
    value of NOP
7.  Assign the job to selected resource
8.  If new job arrived
9.  Go to step no. 3
10. Elseif job does not arrived within 20 minute
    than ant again scan the resources for checking
    the processes on each resource
11. Else exit
```

Figure 1. MMAC Algorithm

In the proposed (MMAC) algorithm, select one node as a master node. The selection of master node is not a permanent thing but a new master node can be elected if the previous node stops functioning properly due to some inevitable circumstances. The ant will continuously originate from master node and check the processes of each node and update them in database as mentioned above. Whenever client perform any operation using client socket the request will first move to master node. The master node based on the request check in database for less utilized node. Then master node will send that user request to another node who have less processes using TCP socket. The master node can communicate with the other nodes in a network using TCP socket. The ant can originate after every 15-20 minute however any request from client side arrived or not. This make MMAC more efficient to produce which increase load balancing and resource utilization and reduce makespan.

4. An Illustrative Example

Consider a grid environment with two resources R1 and R2 and a meta-task MT that contain four jobs T1, T2, T3 and T4. The master node is schedule all the tasks within MT on the available resources R1 and R2. The Min-Min algorithm is simple and produces a better makespan than the other algorithms discussed in the related work, the proposed algorithm executes the ant colony concept in first phase for the better load balancing and min-min algorithm is used in the second phase to assign the jobs to resources who have less process according to first phase. In the proposed algorithm the computation time can be calculated by using formula:

$$\textbf{Makespan} = \max(CT(t_i, m_j))$$
$$CT_{ij} = ET - ST$$

where CT $\rightarrow$ Computational time
ET $\rightarrow$ End time of job on the sub server
ST $\rightarrow$ Start time when job is assigned to master server

Table 1 represents the computational time of the tasks on each resource using above formula:

Table 1. Computational Time of Tasks

Tasks	*Resources*	
	R1	*R2*
T1	3	X
T2	5	5
T3	2	X
T4	1	X

X: The machine is not eligible for the task because of the more processes on machine.

The static mapping of jobs to resources according to conventional Min-Min is shown in Figure 2. Min-Min choose the minimum completion time and so assign all the jobs to resource R1and resource R2 remains idle. The makespan produced by Min-Min is 11 sec.

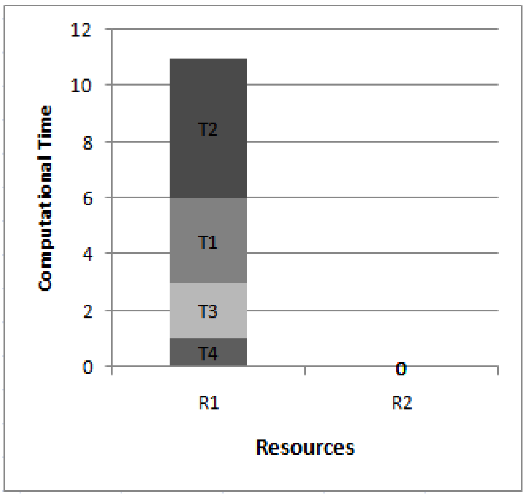

Figure 2. Gantt chart of Min-Min Algorithm

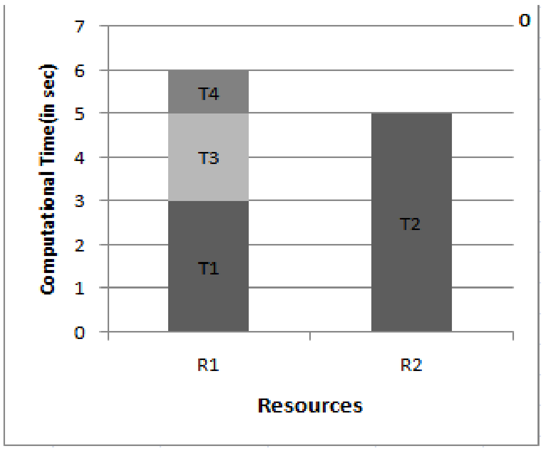

Figure 3. Gantt chart of MMAC Algorithm

According to the proposed MMAC algorithm when request for task T2 arrived then processes on resource R2 is less than the resource R1 so task T2 is schedule to resource R2 instead of resource R1. The makespan produce by MMAC is 6 that is less than Min-Min. Mapping of tasks based on MMAC is shown in figure 3.

5. Results and Comparison

To evaluate and compare the efficiency of the proposed algorithm, problems having machine heterogeneity and task heterogeneity are collected from various literature [3], [12], [14] and execute the proposed MMAC algorithm. The proposed algorithm can be implemented in real environment using Java Development Kit and Wamp server. The results obtained (in sec) for the algorithms are tabulated and shown in Table 2.

Table 2. Comparison of Min-Min and MMAC algorithm

Problem set	*Min-Min(in sec)*	*MMAC(in sec)*
P1	8	5
P2	16	10
P3	33.4	25
P4	33.9	28
P5	11	6

To see how MMAC outperforms Min-Min the results are plotted in a graph and shown in Figure 4. From following figure we can observe that MMAC tries its best to produces less makespan than the Min-Min algorithm for all problems.

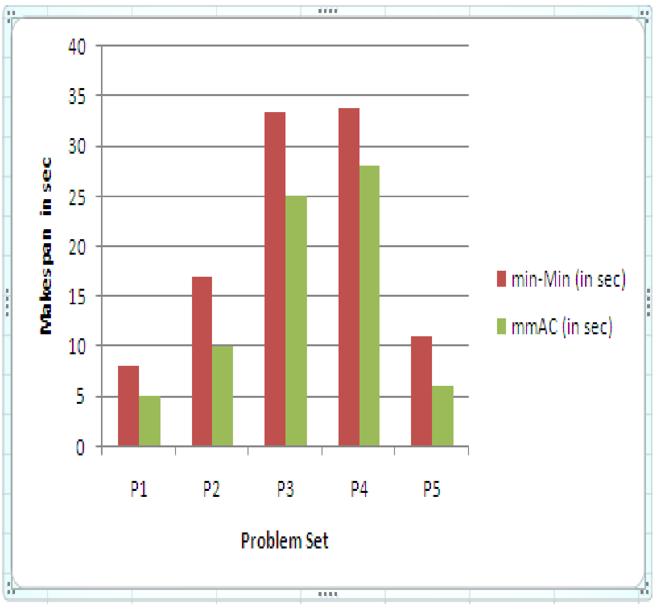

Figure 4. Graphical representation to show improvement of MMAC outperforms Min-Min

Further for load balancing the proposed algorithm uses the concept of ant colony. The MMAC according to the processes on each resource tries its best to use all the available resources in a network. For some problems proposed algorithm uses the same amount of resources, but balances the load in those resources than Min-min. Table 3 shows the resource utilization rate of both algorithms. Resource utilization for a particular problem is calculated using the following formula:

$R= Mi * CT_{ij} *100/TNP$

Here TNP represents Total Number of Processes on Resource to which the job is assigned.

The resource utilization rate is represented as graph in Figure 5. From the following figure we can observe that MMAC uses the maximum amount of resources while reducing the makespan obtained from Min-Min algorithm.

Table 3. Resource Utilization in Percentage

Problem Set	Algorithm Used	M1	M2	M3	M4	M5
P1	Min-Min	100	0	-	-	-
	MMAC	60.72	90	-	-	-
P2	Min-Min	100	0	-	-	-
	MMAC	95	100	-	-	-
P3	Min-Min	100	0	-	-	-
	MMAC	100	0	-	-	-
P4	Min-Min	0	0	0	100	0
	MMAC	0	90	17.45	0	35.89
P5	Min-Min	100	13	72	18	14
	MMAC	92	100	85	78.99	20.02

6. Conclusion

Min-Min algorithm are applicable in small scale distributed systems. When the number of the small tasks is more than the number of the large tasks in a meta-task, the Min-Min algorithm cannot schedule tasks, appropriately, and the makespan of the system gets relatively large. Furthermore it does not provide a load balanced schedule and in this one resource can execute only one job at a time. To overcome the limitations of Min-Min algorithm and to achieve high computing throughput in a grid environment, a new scheduling algorithm, is proposed. It is performed in two-phases. It uses the advantages of Min-Min and ant colony algorithms and cover the disadvantages of min-min algorithm. The proposed algorithm is based on processes on each resource. Evaluation of our new heuristic is done using JDK and wamp server. The experimental results obtained by applying the proposed algorithm for various problems shows that it outperforms the existing scheduling algorithms. This study is only concerned with the number of the resources .number of processes and task execution time. The study can be further extended by considering low and high machine heterogeneity and task heterogeneity. Also considering the cost factor for job execution and cost of communication and some other can be other open problem in this area.

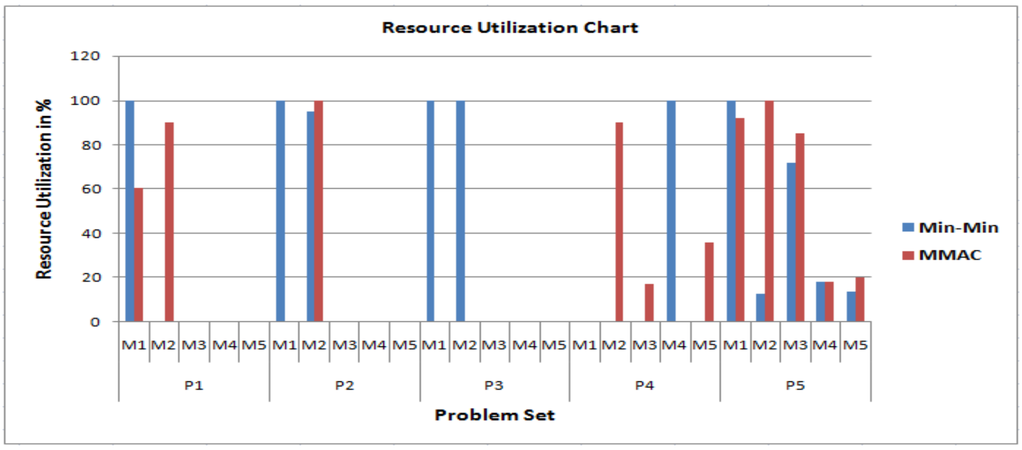

Figure 5. Graphical representation to show more resource utilization of MMAC over Min-Min

References

[1] N.A. Azeez1; A.P. idoye; A.O. Adesina; K.K. Agbele; Iyamu Tiko, and I.M. Venter, “Peer to Peer Computing and Grid Computing: Towards a Better Understanding”, 2011.

[2] I. Foster and C. Kesselman. “The Grid: Blueprint for a New Computing Infrastructure*”.* Morgan Kaufmann, San Francisco, CA, July 1998.

[3] Kobra Etminani , Mahmoud Naghibzadeh , Noorali Raeeji Yanehsari , “A HYBRID MIN-MIN MAX-MIN ALGORITHM WITH IMPROVED PERFORMANCE”, ubicc.

[4] Dipti Sharma, Mr. Pradeep Mittal , “Job Scheduling Algorithm for Computational Grid in Grid Computing Environment”, *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol. 3, Issue 5, May 2013.

[5] J. M. Schopf, “A General Architecture for Scheduling on the Grid”, Special issue of JPDC on Grid Computing, 2002 .

[6] E. Bagheri, M. Naghibzadeh, “A New Approach to Resource Discovery and Dissemination for Pervasive Computing Environments Based on Mobile Agents”, Scientia Iranica Journal, Vol. 14, No. 6, pp. 612-624, 2007.

[7] G. Jaspher W. Kathrine and Mansoor Ilaghi U,“Job Scheduling Algorithms in Grid Computing – Survey*”, International Journal of Engineering Research & Technology (IJERT)* Vol. 1 Issue 7, September - 2012 ISSN: 2278-0181.

[8] Saeed Parsa and Reza Entezari-Maleki, “RASA: A New Task Scheduling Algorithm inGrid Environment,” World Applied Sciences Journal,7(Special Issue of Computer & IT),pp 152-160, 2009

[9] H. Jamal, A. Nasir, K. Ruhana, K. Mahamud and A.M. Din, Load Balancing Using Enhanced Ant Algorithm in Grid Computing, Proceedings of the Second International Conference on Computational Intelligence, Modelling and Simulation, pp. 160-165, 2010.

[10] Ku Ruhana Ku-Mahamud and Husna Jamal Abdul Nasir. “Ant Colony Algorithm for Job Scheduling in Grid Computing”, in 2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation, IEEE Computer Society 2010, pp. 40-45.

[11] George Amalarethinam. D.I, Vaaheedha Kfatheen .S, “Max-min Average Algorithm for Scheduling Tasks in Grid Computing Systems”, *International Journal of Computer Science and Information Technologies*, Vol. 3 (2) , 2012 ,3659-3663.

[12] T. Kokilavani, Dr. D.I. George Amalarethinam, “Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing” *International Journal of Computer Applications* (0975 – 8887) Volume 20– No.2, April 2011

[13] Salman Meraji, M. Reza Salehnamadi, “A Batch Mode Scheduling Algorithm for Grid Computing,” *Journal of Basic and Applied Scientific Research*, Year :2013, Volume: 3, Issue: 4, pp. 173-181, ISSN 2090-4304

[14] HE. X, X-He Sun, and Laszewski. G.V, "QoS Guided Min- Min Heuristic for Grid Task Scheduling," *Journal of*

Computer Science and Technology, Vol. 18, pp. 442-451, 2003.

[15] O. M. Elzeki , M. Z. Reshad , M. A. Elsoud , “Improved Max-Min Algorithm in Cloud Computing”, *International Journal of Computer Applications* (0975 – 8887) Volume 50 – No.12, July 2012.

[16] Singh. M and Suri. P.K, QPS A QoS Based Predictive Max-Min, Min-Min Switcher Algorithm for Job Scheduling in a Grid, Information Technology Journal, Year: 2008, Volume: 7, Issue: 8, Page No.: 1176-1181.

[17] Kumar Nishant, Pratik Sharma, Vishal Krishna, Chhavi Gupta and Kunwar Pratap Singh et al., “Load Balancing of Nodes in Cloud Using Ant Colony Optimization” 14th *International Conference on Modelling and Simulation,* 2012

[18] Siriluck Lorpunmanee, Mohd Noor Sap, Abdul Hanan Abdullah, and Chai Chompoo-inwai, “An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment” *world Academy of Science, Engineering and Technology International Journal of Computer*, Information Science and Engineering Vol:1 No:5, 2007

[19] D.Maruthanayagam, Dr. R.Uma Rani, “Improved Ant Colony Optimization for Grid Scheduling” *IJCSET* |November 2011 | Vol 1, Issue 10, 596-604

.

Sarpreet Singh received his B.Tech degree in Computer Science from Sant Longowal Institute of Engineering and Technology (SLIET) of Punjab in 2006, M.Tech degree in Computer Science from Punjabi University Patiala of Punjab in 2009. He is a Assistant Professor of Computer Engineering at Sri Guru Granth Sahib World University, Fatehgarh Sahib now. His research interests include Grid computing, Distributed databases design concepts and knowledge management..

Author Profile

Davinder Kaur is a M.Tech. student in the Department of computer science and engineering at Sri Guru Granth Sahib World University of Fatehgarh Sahib, Punjab, India. She received the B.Tech. degree in information technology from Banda Singh Bahadur Engineering College of Fatehgarh Sahib in 2012. Her research interests include grid computing, data mining and cloud computing