LOAD BALANCING IN GRID COMPUTING USING ANT COLONY ALGORITHM AND MAX-MIN TECHNIQUE

Rose Karimpour[1], Mohammad Reza Khayyambashi[2], Naser Movahhedinia[3]

[1, 2, 3] Department of Computer Architecture, Faculty of Computer Engineering
University of Isfahan
Isfahan, Iran

Email: R.K@eng.ui.ac.ir[1], M.R.Khayyambashi@eng.ui.ac.ir[2], Naserm@eng.ui.ac.ir[3]

ABSTRACT

Stagnation is one of the complicated issues in Grid computing systems, which is caused by random arrival of tasks and heterogeneous resources. Stagnation occurs when a large number of submitted tasks are assigned to a specific resource and make it overflow. To prevent this scenario, a load balancing algorithm based on Ant Colony algorithm and Max-min technique is proposed in this paper. In the proposed algorithm, the resource manager of the system finds the best resource for a submitted task according to a matrix that indicates the characteristics of all resources as pheromone values. By choosing the best resource for the submitted task, a local pheromone update is applied to the selected one to reduce the tendency of being selected by onward new tasks. After this assigned task is executed properly, a global pheromone update is performed to renew the status of all resources for the next submitted tasks. To avoid stagnation, a comparison between a predefined threshold and the pheromone value of each resource is performed to keep the number of assigned tasks below this threshold. Due to harmonizing the resources' characteristics and tasks, the proposed algorithm is able to reduce the response time of the submitted tasks while it is simple to be implemented.

Keywords: Grid computing, Ant colony algorithm, Stagnation, Load balancing.

1.0. INTRODUCTION

Distributed computing is used for running computing applications on several computers which are connected via a network. The motivation of building this system is sharing of resources [1].

The best ways to construct a distributed system are cloud and grid computing. A cluster system includes several computers that are connected by a local network. Due to the fixed area of the cluster system, a computing application would be limited and flexible. To overcome this problem, grid computing is increasingly applied in which diverse resources in different regions are connected via a comprehensive network like the internet [2].

To utilize all available resources and minimize the execution time of the input tasks, a resource management with an effective scheduling algorithm is required to distribute the input load equally between the resources of the grid computing system. This algorithm should act in a way that the difference between the simplest resource and the complex one becomes minimal. Because of the dynamic inclusion and exclusion of resources in the grid environment, the scheduling algorithm must consider these circumstances and adapt to the dynamic changes of the system [3].

In general a scheduling technique consists of three phases [4]:

- Gathering information: In this phase, the resource manager gathers the information about the status of the computing resources and recognizes whether there is an unbalanced load on the environment.
- Decision making: The resource manager focuses on an optimal distribution by choosing the waiting tasks in resources' queues and decides to move them to the idle ones.
- Data migration: Tasks are moved from an overloaded resource to an underloaded one.

Load balancing algorithms can be divided into many categories from different points of view. These algorithms can be divided into static and dynamic algorithms in which the time of making a decision is considered. In static scheduling, all information about the tasks and resources is available. In contrast, in dynamic scheduling the information about the status of the resources is obtained during the execution time [5]. From another point of view, these algorithms can be divided to centralized and decentralized algorithms. In a centralized algorithm, one central scheduler is responsible for making balancing decision. Some of the algorithms categorized in this field are as follows [6]:

- Minimum Execution Time (MET): The execution time of the submitted task is estimated in each resource. The resource with the least execution time is selected regardless of its availability.
- Minimum Completion Time (MCT): The completion time of the submitted task is estimated in each resource. This time includes the execution time and the time needed to wait in the resource's queue. Each task is allocated to each resource with the least completion time.
- Min-Min: The completion time of each unmapped task in each resource is estimated and the minimum task/resource time is assigned. After mapping this task, it would be removed from the unmapped queue.
- Max-Min: This approach is similar to Min-Min approach except the maximum completion time of task/resource is selected.
- Opportunistic Load Balancing: Each task is allocated to the resource with the least current load. The main idea of this approach is to occupy all resources as much as possible.

Max-min algorithm dedicates large tasks as priority ones in comparison with small tasks which provides the opportunity of executing more small tasks during large ones simultaneously regarding to determining the make-span by the duration of large task. But this approach would bring a noticeable problem in case of entering tasks with diverse completion time and execution one. In this scenario, the distributed environment would experience performance declination with excessive waiting time for small tasks [7].

In [8] a new approach based on Max-min algorithm is presented that covers the above mentioned scenario by executing big tasks with slower resources that minimizes waiting times of the smaller tasks by allocating them to fastest resources concurrently which facilitates the execution of smaller tasks during large tasks execution. But this approach presents a scenario with the assumption of resources without current loads which is not suitable for real scenarios.

Nowadays, the ant colony technique is a good candidate to solve the optimization problems. This technique is a subset of artificial intelligence techniques based on the behavior of independent agents. In these communities, each agent has a simple structure, but their environmental behavior are complicated. These agents communicate with each other directly (signals, symptoms) or indirectly (via impacting on the environment) and solve a problem together. These agents communicate with the environment via their sensors and have influences on it. Ant colony algorithm is a good candidate for solving load balancing in the Grid computing systems because it is conformed to solve static and dynamic optimization problems [9].

In this paper, a new approach based on Ant colony algorithm and Max-min is presented in which a matrix is used to keep the information about the status of all resources/tasks. By submitting a new task to the system, a column presenting the completion time of this task on each resource is added to the matrix as pheromone value with regard to considering the current loads of available resources. By using the Max-Min technique, a task with largest execution time is allocated to a resource with minimum completion time. By sending this task to the selected resource, a local pheromone update function is performed to make this resource less desirable for the next submitted tasks. After the task is executed on the resource, a global pheromone update is applied to all resources/tasks in the matrix to update the status of them. This iteration would be executed again by entering a new task to the system.

The remainder of this paper is organized as follows: Section 2 is dedicated to related works. In this section, previous load balancing approaches based on ant colony algorithm and Max-min technique are mentioned. In section 3, the proposed algorithm is described in detail and section 4 focuses on the setup of the simulation

results followed by comparing the proposed approach with previous algorithms in section 5 and 6. Finally section 7 is dedicated to conclusion.

2.0. RELATED WORK

With regard to grid computing that plays impressive role in executing complicated tasks which need resources with sufficient requirements, load balancing is considered as one of the dominant issues in this field, which assists increase in performance in distributed system, encouraging many researchers to release new ideas for task scheduling. However, most of the existing approaches do not consider real scenarios and evaluate their approaches with predetermined assumption that prevent applying these proposals to real distributed systems. In this section, some of the related approaches are expanded regarding to their disadvantages.

In [10], a scheduling algorithm based on ACO for assigning tasks in Grid computing is proposed. This approach called BACO tries to optimize the completion time of executing tasks in Taiwan UniGrid by selecting the best resources according to local and global pheromone update techniques. Local pheromone update is done after a task is assigned to a resource and updates the status of the resource. This update decreases the attraction for selecting this resource for another submitted task. Global pheromone update is done after this task is executed in the resource and makes this resource a choice for executing the next incoming task. But this approach balances the system without considering the size of the incoming tasks. Also it is only applied to Taiwan UniGrid environment with limited number of resources and imported tasks and is not tested in a wider environment.

Another approach using ant colony technique for dynamic load balancing is proposed in [11]. This approach tries to minimize the completion time of the submitted tasks based on local and global pheromone update. When a task is submitted to the system, the execution time of this task in each resource is estimated. The estimation time is set as the initial pheromone value for each resource. The one that has the least value is selected to execute the submitted task. However, this approach is tested for tasks with small size (1000 to 5000 MI).

In [12], an idea based on ant colony for load balancing in Grid computing system is proposed. In this approach, the pheromone update process is conducted by applying encouragement and punishment. If a resource executes the assigned task successfully, more pheromone is allocated to that resource by an encouragement function and raises its chance to be selected for the next submitted tasks. If a resource cannot process a task completely, the punishment function assigns a lower pheromone value to this resource. However, a comparison has not been made between this idea and other approaches.

The Max-Min Ant system (MMAS) which is proposed in [13], is one of the most successful algorithms in ant colony algorithm. The main advantage of this approach is the pheromone trail threshold, which keeps all of them between a predefined lower and upper bound. This threshold is designed to avoid the stagnation problem. This approach has the favorable property of robustness, which means when a failure occurs in the system, the algorithm can handle it and continue to perform. But tuning the predefined parameters so that the algorithm performs well is difficult.

In [14] a new task scheduling strategy is proposed which is called RASA which exerts the combination of Max-min and Min-min. Max-min is used to neglect extra delays for large tasks by executing them with small ones simultaneously and Min-min is applied to execute small tasks prior to large ones. But this approach is not expanded for real scenarios in Grid environment.

A Qos load balancing algorithm is presented in [15] in which a formula with the combination of various parameters and predetermined factors is applied to a particular environment call Grid-JQA which is experienced the disadvantage of not presenting mathematical solution for practical situations.

UllahMunir presents a new load balancing algorithm named Qossufferage in [16] by considering Grid bandwidth and task bandwidth requirement. Although this approach leads to more reasonable make-span in

comparison with distinct Max-min and Min-min, the proposed algorithm does not consider CPU rates needed to execute tasks as a noticeable factor.

In [17] a static broadcast algorithm (SBA) is presented where the status of a resource is changed by the arrival or migration of a task, this resource sends a broadcast message to all of the available resources to inform them about its new status. In this approach, each resource has a table including the status of all available resources and helps it to find the best one. But the drawback of this approach is the time needed for executing its code for the Grid environment.

Other approaches that are used for comparing with the proposed algorithm are Random ones [18]. In these approaches, when a task is submitted to the system, it is assigned to a resource in a random manner. These scheduling algorithms distribute loads unequal between the resources without considering impressive factors which influence on the performance of load balancing algorithms.

In [8] a new approach based on Max-min algorithm is presented which covers the problem of Max-min approach in case of entering tasks with diverse completion and execution time. The Proposed approach solve this problem by executing large tasks by slower resources which minimizes short tasks' waiting time and allocating them to fastest resources concurrently which facilitates the execution of more small tasks during large one execution. But this approach presents a scenario with the assumption of resources without current loads which is not suitable for real scenarios.

3.0. PROPOSED ALGORITHM

In this paper a new algorithm based on ant colony algorithm and Max-min is presented. The purpose of this algorithm is to minimize the computational time of tasks that are submitted to the Grid environment. This environment consists of a server that contains information about the Grid's resources and a resource broker that is assumed to schedule the submitted tasks and selects the best resources to execute them. To simplify the choice of these resources, this broker uses a matrix that comprises the pheromone value of each resource/task. This value demonstrates the status of the resource/task and is obtained by the characteristics of imported tasks, the amount of current resource's load and its relative position to the broker.

The proposed algorithm consists of the following phases:

1. A task is submitted to the system by user. This request also includes the task's size and the number of CPU cycles it needs. This request is sent to the resource broker.
2. The resource broker determines the requirement for the new submitted task. This broker demands the information of each resource from the information server.
3. According to the information obtained from the information server, the broker calculates the initial pheromone value of each resource for each task. This value is determined by the required time for transmitting the task from the broker to each resource and the estimated time for executing this task on each of them. The transmission time is computed by dividing the size of the task ($Size_{task}$) by the available bandwidth between the broker and the corresponding resource (B_R). The completion time for the task in that resource depends on the numbers of CPU cycles needed for the task (C_{TASK}), the speed of the processor of that resource in Million Instruction per Second ($MIPS_{resource}$) and the current load on it. This initial value is formulated as Eq. 1.

$$PheromoneValue_{resource,task} = \frac{1}{\left(\frac{Size_{task}}{B_R} + \frac{C_{TASK}}{MIPS_{resource} * (1 - load_{resource})} \right)} \quad (1)$$

The pheromone values of each task in each resource are calculated according to Eq. 1 and are used to construct a matrix as follows:

$$PheromoneValue = \begin{array}{c} \\ resource_1 \\ \vdots \\ resource_m \end{array} \overbrace{\begin{bmatrix} PhromoneValue_{11} & PhromoneValue_{12} & .....PhromoneValue_{1n} \\ .... & ..... & ..... \\ PhromoneValue_{m1} & PhromoneValue_{m2} & .....PhromoneValue_{mn} \end{bmatrix}}^{tasks}$$

4. According to this matrix, the task with maximum execution time is assigned to a resource with minimum completion time.
5. After this assignment, the local pheromone update function is applied to the selected resource. This is designed to reduce the tendency to choose the selected resource for new tasks. This function is done by the following equation (Eq. 2) where $\xi \in (0,1)$ is the pheromone decay rate and θ_0 is the initial value for the pheromone:

$$PheromoneValue_{resource,task}(t+1) = (1-\xi) * PheromoneValue_{resource,task}(t) + \xi\theta_0 \quad (2)$$

6. When the submitted task is assigned to the resource and is completely executed on it, the whole Pheromone Value for each resource is recalculated. This global pheromone update is performed by Eq. 3 where ρ is the convergence speed of the algorithm to the best solution:

$$PV_{resource,task}(t+1) = (1-\rho) * PV_{resource,task}(t) + \rho * \Delta PV^{Best}{}_{resource,task} \quad (3)$$

According to this equation, for updating the pheromone value of a resource, if the executed task in iteration (t) is used the corresponding resource to reach to the best resource, $\Delta PV^{Best}{}_{resource,task} = \frac{1}{L_{Best}}$ where L_{Best} is the length of the path that the executed task traverses from the broker to reach out to the best resource. Otherwise this value will be zero. After the pheromone values of all resources are updated, they are verified to be in predefined bound ($PV_{Min} \le PV_{resource,task} \le PV_{Max}$). If the $PV_{resource,task} \le PV_{Min}$, it is set to PV_{Min} and if it is bigger than PV_{Max}, it is set to this value. These upper and lower threshold are chosen experimentally by the designer. This threshold causes to decrease the tendency to choose a specific resource that is often the best one and avoid stagnation on it.

7. By submitting a new task, this algorithm would be performed from the step 1.

To clarify major steps which provide an appropriate load balancing algorithm in Grid environment, the pseudo code of the proposed approach is mentioned in Table 1.

Table 1: The pseudo code of the proposed algorithm

For all submitted tasks($Task_i$)
  For all resources
    Calculate $PheromoneValue_{resource_j, task_i}$
    Add $PheromoneValue_{resource_j, task_i}$ to $PheromoneValue$ (j,i)
While $PheromoneValue$ is not empty
    Find $Task_i$ with maximum execution time
    Designate $Task_i$ to $resource_j$ with the least $PheromoneValue$ (j,i)
    local pheromone update for $resource_j$
After executing $Task_i$ to $resource_j$
    Global pheromone update

4.0. PARAMETERS DITERMINATION

This section determines the parameters of the proposed algorithm. These parameters must be framed in such a way that the completion time is placed in its best position. For determining these parameters, the characteristics of tasks and resources should be the same as Table 2.

Table 2: The characteristics of the Grid environment

Number of resources	100
Number of processing elements in each resource	1-5
Processing elements speed	10 to 50 MIPS
Bandwidth	1000 to 5000
Length of tasks	0-50000 MI

For simulation, it is assumed that the grid environment includes several resources connected through different communication links with various speeds. The bandwidth used between grid resources is assumed to be 1000 to 5000 Byte per second.

The length of the tasks used in this simulation is introduced as Millions of Instructions (MI). The submitted tasks can be data intensive and computationally intensive. In this paper, the tasks are computationally intensive, which is more likely in the real world and the loss of resource computational power is more expensive than loss of memory.

As the topology of the Grid environment and the tasks entered to the system is changing in each simulation, for determining the completion time of the submitted tasks the average completion time of 30 runs of the proposed algorithm is considered. In this simulation, the number of submitted tasks is assumed to be 1000 and the number of resources is 100.

In Figure 1, the effect of ξ factor on the performance of the proposed algorithm is shown. According to this experiment, 0.1 is the best ξ for achieving the best completion time.

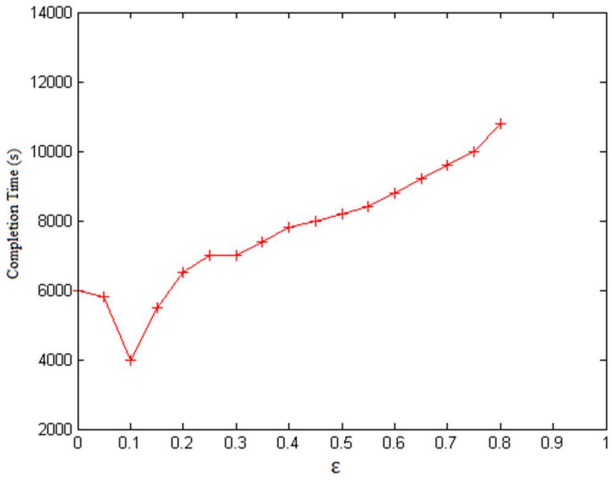

Fig. 1: The effect of ξ factor on performance.

According to Eq. 3, which is performed to all resources for global update, ρ is the trail persistence of the algorithm to the best solution which concludes $(1-\rho)$ as evaporation rate. This mechanism helps to restrict the resource selections by gradually neglecting the bad choices.

For small Grid environment, the more $\rho$is determined, the more extensive resources would be searched by the algorithm to find an appropriate one. When the algorithm encounters a large number of resources in Grid, this factor would restrict the search space by omitting inappropriate resources. Thus, the value of this factor is limited in large environment to increase the performance of the Grid more practically.

This factor is examined in various values in Figure 2 to determine the best one for enhancing the performance of the proposed algorithm regarding to Table 2 which depicts the characteristics of the Grid environment which is considered as a large environment.

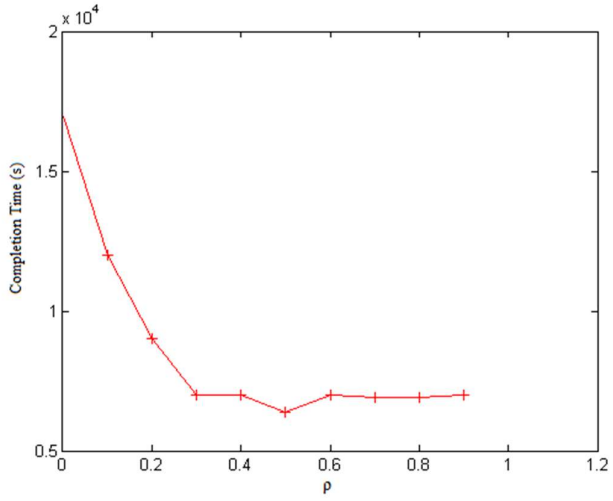

Fig. 2: The effect of ρfactor on performance.

The overview of predetermined factors which assist the proposed algorithm to present the best performance in Grid environment is illustrated in Table 3 [13].

Table 3: The parameters of the proposed algorithm

Parameters	Value
ξ	0.1
θ_0	$\frac{1}{n * L_{Best}}$ n= The number of resources L_{Best} =the length of the path that the executed task traverses from the broker to reach out to the best resource.
ρ	0.5

5.0. SIMULATION RESULTS

This section is dedicated to the comparison of the proposed algorithm with other prior load balancing approaches with considering 100 resources and 1000 imported tasks and their characteristics based on Table 2. As the topology of the Grid environment and the tasks entered to the system is changing in each simulation, for determining the completion time of the submitted tasks, the average completion time of 30 runs of the proposed algorithm is considered with dedicating 65% of tasks with large size (40000 to 50000) and the rest of them with small tasks to determine the performance of the proposed algorithm.

As the proposed algorithm is derived from three main approaches (Max-min, improved Max-min [8] and RASA), they have the same time complexity $O(nm^2)$ in which n is the number of available resources and m is considered as the number of imported tasks. Despite of the same complexity, the presented algorithm is experienced better completion time with more reliable load balancing algorithm with the ability of executing small tasks and large ones concurrently. This new characteristic helps small tasks not wait for large ones execution and assists them with the ability of concurrency.

The completion time of each approach is illustrated in Fig. 3 with regard to the Grid environment characteristics in Table 2.

As Random approaches dedicate imported tasks to resources without any specific algorithm and not considering impressive factors on enhancing the Grid performance, they experience the most completion time in comparison with other mentioned approaches.

Max-Min algorithm as one of the most common load balancing algorithm, have the disability of declining make-span in scenarios, in which the number of large tasks are much more than small ones that leads them to wait extra time for large tasks execution. This scenario which has been implemented in this paper, depicts the disadvantage of this algorithm emphasizing on not belonging the concurrent execution to speed up small tasks.

Regarding SBA, this approach needs extra time spending to search for appropriate resources in each resource table that contains all the resource characteristics without particular strategies to omit undesirable resources, causing this approach to face with remarkable completion time.

Improved Max-min approach [8] similar to the proposed algorithm has higher completion time as this approach does not consider current loads of the resources as a significant factor to assign new tasks which makes this algorithm not be realistic.

Considering Fig. 3, the proposed algorithm has about 19% improvement in decreasing the required completion time to balance the Grid environment compared with Improved Max-min, which is placed in second degree in the predetermined conditions.

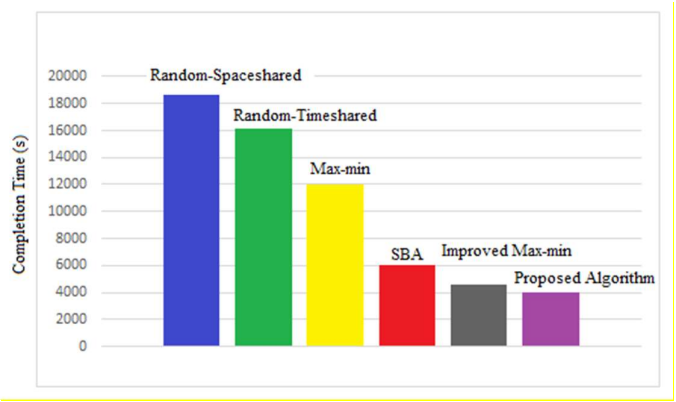

Fig. 3: Comparing the completion time of different algorithms

6.0. VARIOUS JOB CHARACTERISTICS EFFECT ON THE PERFORMANCE OF THE PROPOSED ALGORITHM

In the two proposed scenarios, the characteristics are determined based on Table 2 and 3 by varying the number of jobs entered to the Grid environment and their length respectively.

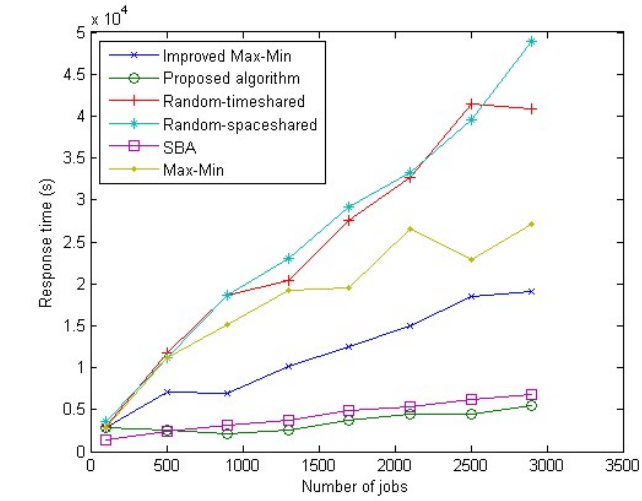

Fig. 4: The Effect of Increasing Number of jobs on response time

Fig. 4. Compares the make-span of six load balancing algorithm when the number of jobs is increased. As can be illustrated in the figure, all algorithm response approximately linear growth to the number of jobs increment except the proposed algorithm with slight slope. This result is due to the specific characteristic of the proposed algorithm in which the ants carry the ID of the entered jobs rather than the whole jobs to find the appropriate

resource which reduces the amount of bandwidth needed and make the algorithm to perform faster in large number of jobs. When the desired node is found, the whole job transfers was performed.

In the second scenario, the length of entered jobs is increased to 50000 MI each step by dedicating 65% of jobs with 10000 to 50000 and the rest of them with small sizes. As mentioned before, by having the concurrency characteristics of the proposed algorithm, growing in the length of jobs does not affect its performance consciously. This scenario is depicted in Fig.5.

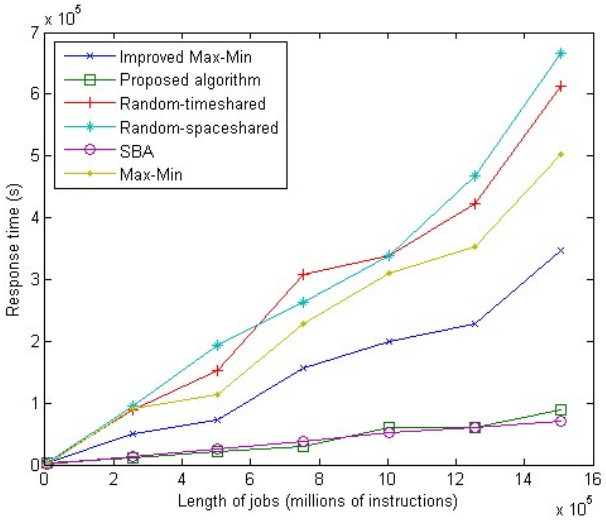

Fig. 5. The Effect of Increasing the length of jobs on response time

7.0. CONCLUSION

In this paper, a scheduling approach based on ant colony algorithm and the Max-Min technique is proposed to solve the stagnation problem in grid computing. For the proposed algorithm, a pheromone value matrix is used to select the best resource/task. The initial pheromone value of each resource/task is assigned based on the time needed to transfer this task from the resource broker to the corresponding resource and the estimation time for executing ~~it on~~ this resource. The local pheromone update function is performed to the selected resource to make it less desirable for others during its execution. After the assigned task is executed, a global pheromone update is applied to the matrix to update the status of all resources for the next submitted tasks. This update makes all the pheromone values below a limited bound to prevent stagnation. This algorithm is simple due to the existence of information of resources and tasks in the information server. The experimental results show that by implementing the proposed algorithm in the Grid environment, the completion time of submitted tasks has about 19% improvement compared to the previous algorithms.

REFERENCES

[1] R. Baldoni, M. Bertier, M. Raynal, and S. Tucci-Piergiovanni, *Parallel Computing Technologies*, vol. 4671. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1–14.

[2] S. M. Hashemi and A. K. Bardsiri, "Cloud Computing Vs. Grid Computing," *ARPN J. Syst. Softw.*, vol. 2, no. 5, pp. 188–194, 2012.

[3] J. Nabrzyski, J. M. Schopf, and J. Węglarz, Eds., *Grid Resource Management*, vol. 64. Boston, MA: Springer US, 2004.

[4] P. Srivastava, S. Gupta, and D. S. Yadav, "Improving Performance in Load Balancing Problem on the Grid Computing System," *Int. J. Comput. Appl.*, vol. 16, no. 1, pp. 6–10, 2011.

[5] U. K. Kumar, "A Dynamic Load Balancing Algorithm in Computational Grid Using Fair Scheduling," *Int. J. Comput. Sci. Issues*, vol. 8, no. 5, pp. 123–129, 2011.

[6] M. Hemamalini, "Review on Grid Task Scheduling in Distributed Heterogeneous Environment," *Int. J. Comput. Appl.*, vol. 40, no. 2, pp. 24–30, 2012.

[7] A. Qazi, R. G. Raj, M. Tahir, M. Waheed, S. U. R. Khan, and A. Abraham, "A Preliminary Investigation of User Perception and Behavioral Intention for Different Review Types: Customers and Designers Perspective," The Scientific World Journal, vol. 2014, Article ID 872929, 8 pages, 2014. doi:10.1155/2014/872929.

[8] O. M. Elzeki, M. Z. Reshad and M. A. Elsoud, "Improved Max-Min Algorithm in Cloud Computing, " *International Journal of Computer Applications*,vol. 50, no. 12, pp. 1-6, 2012.

[9] S. A. Thakare, "Comparison of Swarn Inteligence Technique," *Int. J. Comput. Sci. Bus. INFORMATICS*, vol. 1, no. 1, pp. 1–11, 2013.

[10] R. Chang, J. Chang, and P. Lin, "Balanced job assignment based on ant algorithm for computing grids," in *Proceedings of the 2nd IEEE Asia-Pacific Service Computing Conference*, 2007, pp. 291–295.

[11] S. Lorpunmanee, M. Sap, A. Abdullah, and C. Chompoo-inwai, "An ant colony optimization for dynamic job scheduling in grid environment," *Int. J. Comput. Inf. Sci. Eng.*, vol. 1, no. 4, pp. 207–214, 2007.

[12] H. Yan, X. Shen, X. Li, and M. Wu, ""An improved ant algorithm for job scheduling in grid computing," in *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, 2005, pp. 2957–2961.

[13] T. Stutzle and H. Hoos, "MAX-MIN ant system," *Futur. Gener. Comput. Syst.*, vol. 16, pp. 889–914, 2000.

[14] S. Parsa and R. Entezari-Maleki, "RASA: A New Grid Task Scheduling Algorithm," *International Journal of Digital Content Technology and its Applications*,vol. 3, pp. 91-99, 2009.

[15] L. Mohammad Khanli, and M. Analoui,"Grid_JQA: A QoS Guided Scheduling Algorithm for Grid Computing," The Sixth International Symposium on Parallel and Distributed Computing (ISPDC'07), IEEE, 2007.

[16] E. UllahMunir, J. Li, and Sh. Shi,"QoSSufferage Heuristic for Independent Task Scheduling in Grid," *Information Technology Journal*, vol. 6, no. 8, pp. 1166-1170, 2007.

[17] S. A. Ludwig and A. Moallem, "Swarm Intelligence Approaches for grid Load Balancing." *Journal of Grid Computing, vol. 9, no. 3, pp.* 279-301, 2011.

[18] A. Rekaby and M. Abo Rizka, "A Comparative Study in dynamic job Scheduling Approaches in Grid Environment," *International Journal of Grid Computing & Applications (IJGCA)*, vol.4, no.3, pp. 1-10, 2013.